Genstat®

Summary

www.vsni.co.uk

VSNi

Genstat® Reference Manual (Release 22)

Part 1: Summary

Genstat Release 22 was developed by VSN International Ltd, in collaboration with practising statisticians at Rothamsted and other organisations in Britain, Australia, New Zealand and The Netherlands.

Published by: VSN International, 2 Amberside, Wood Lane, Hemel Hempstead, Hertfordshire HP2 4TP, UK

E-mail: info@genstat.co.uk

Website: http://www.genstat.co.uk/

First published 1989, as the *Genstat 5 Release 2 Reference Summary*
This edition published 2022, for Genstat Release 22

Citation: VSN International (2022). *Genstat Reference Manual (Release 22), Part 1 Summary*. VSN International, Hemel Hempstead, UK.

Contents

1 The Genstat language 1
1.1 Syntax of the command language 1
1.2 Glossary of terminology 2
1.3 Data structures 11
1.4 Program control 12
2 Data handling 15
2.1 Input and output 15
2.2 Calculations and manipulation 16
2.3 Graphics 21
3 Statistical analyses 23
3.1 Basic and nonparametric statistics 23
3.2 Regression and generalized linear models 24
3.3 Analysis of variance 27
3.4 Design of experiments 30
3.5 REML analysis of linear mixed models 32
3.6 Multivariate and cluster analysis 35
3.7 Time series 37
3.8 Repeated measurements 37
3.9 Survival analysis 38
3.10 Bayesian methods 39
3.11 Spatial statistics 39
3.12 Six sigma 40
3.13 Survey analysis 41
3.14 Data mining 41
3.15 Statistical genetics and QTL estimation 42
3.16 Microarray data 44
3.17 Ecological data 44
4 Syntax summary 46
4.1 Commands 46
4.2 Functions for calculations 492
4.3 Functions for model formulae 507
5 List of commands 509

Conventions

Genstat system words are shown in the Courier typeface e.g. `CALCULATE`. In the syntax summary, elements of the language to be substituted by the user are in italics, e.g. *variate*. New directives, procedures or functions in Release 22, or options and parameters of existing directives or procedures that have been modified in Release 22, are marked by the symbol †.

1 The Genstat language

Genstat has a clear but powerful command language which provides access to the very wide set of facilities summarized in Sections 2 and 3. Alternatively, many standard operations and analyses can be run using the menus in Genstat's Windows interface.

1.1 Syntax of the command language

Input to Genstat is known as a Genstat program. This is made up of statements each of which may use one of the standard Genstat commands (known as directives); alternatively, it may use a Genstat procedure, that is, a subprogram of statements. You can write your own procedures, or use those in the Library distributed with Genstat, or in the library provided at your site.

Whether the statement uses a directive or a procedure, the syntax is identical. First you give the name of the directive (or procedure), then options, and then parameters. Finally, you indicate the end of the statement, either by typing a colon or by ending the line (by typing <RETURN>). Long statements can be continued onto succeeding lines by typing the continuation character (\) before <RETURN>.

Some statements will have neither options nor parameters: for example

```
PAGE
```

to start a new page in output. Others may have no options: for example

```
PRINT STRUCTURE=X,Y; DECIMALS=0,2
```

prints the contents of data structures X and Y with zero and two decimal places respectively. In this statement, there are two parameter settings defining two lists running in parallel. Parameter settings are always in parallel like this, and are separated from one another by semicolons. Options are enclosed in square brackets, and set aspects that apply to all the (parallel) parameter values. They are also separated from one another by semicolons. For example

```
PRINT [CHANNEL=2; INDENTATION=5] STRUCTURE=X,Y; DECIMALS=0,2
```

prints X and Y to output channel 2 with a five-character indentation at the start of each line. Nearly all options, and some parameters, have default values chosen to be those required most often, and so will usually not need to be set.

Settings of options and parameters can be lists (as above), expressions or formulae. Lists may be of numbers (as with DECIMALS above), or identifiers (as with STRUCTURE) or strings. An identifier is the name that you give to a Genstat data structure (for example X or Y), and which you then use to refer to it in the program. They must start with a letter (for Genstat this means the alphabetic characters A to Z, in capitals or lower case, as well as the percent and underline characters) and then contain either letters or digits (the numerical characters 0 to 9); Genstat takes notice of only the first 32 characters. (This is the default in Releases 4.2 onwards, but you can use the SET directive to request that Genstat take notice of only the first eight characters as in earlier releases.) Where a list of identifiers provides *input* to a directive or procedure, you can put an expression instead; this will then be evaluated (to give a list of identifiers containing the results) before the directive or procedure is used. A string is a list of characters. Usually the start and end of the string must be marked by a single quote ('). Strings occur within the Text data structure. Also, the settings of some options and parameters are lists of string "tokens" that can be chosen from a defined list; these do not need to start and end with single quotes. The separator between items in lists is comma; spaces can be included anywhere between items but do *not* act as separators. Formal definitions of expressions, formulae, and all the other concepts of the Genstat language are in the *Guide to the Genstat Command Language*, Part 1, Section 1.2.

Names of directives, procedures, options and parameters are examples of Genstat system words. They can be given in capital or small letters (or in mixtures of both) and, provided you are only using directives and official Genstat Library procedures, they can always be abbreviated to four characters. The same rules apply to string tokens in directives and Library procedures. However, if you or your site have defined your own procedures, you may have chosen names that differ only in the fifth or subsequent characters. If you supply more characters, Genstat will check the name up to the 32nd character, and ignore any characters after that. (You can, however, use the SET directive to request that Genstat also ignores the ninth and subsequent characters, as in releases before 4.2.)

Names of options and parameters can often be abbreviated to fewer than four characters, and there are also rules by which the option or parameter name, with its accompanying equals character, can be omitted altogether. The most useful of these is that, if the first parameter of the directive is the one that comes first in the statement, then the name of the parameter can be omitted: for example

```
PRINT [CHANNEL=2; INDENTATION=5] X,Y; DECIMALS=0,2
```

as STRUCTURE is the first parameter of PRINT. The same rule holds for options:

```
PRINT [2; INDENTATION=5] X,Y; DECIMALS=0,2
```

as CHANNEL is the first option of PRINT. Full details of the rules are in the *Guide to the Genstat Command Language*, Part 1, Section 1.2.

A final point about the first parameter is that its setting determines the length of the parallel lists. The lists for other parameters will be repeated (or recycled) if they are shorter. (If they are longer, Genstat gives an error diagnostic.) For example

```
PRINT A,B,C,D; DECIMALS=0,2
```

prints A with zero decimal places, B with two, and then (recycling the DECIMALS list), C with zero and D with two.

1.2 Glossary of terminology

Backing store is a system provided by Genstat for the convenient storage of data structures and procedures. The OPEN directive allows you to open a backing-store file, the STORE directive stores information and RETRIEVE allows you to access it later on (perhaps in a subsequent run of Genstat). When a data structure is stored, Genstat keeps not only the data values but also all the other associated information (for example level and label definitions of factors, sub-structures of pointers and so on).

Bracket

Round brackets ( ) are used to enclose a list of numbers to be pre- or post-multiplied or to enclose the arguments of a function; they also occur in expressions.

Square brackets [ ] are used to enclose a list of option settings or to enclose the suffix list of a pointer; also, when preceded by $, they enclose lists of unit names or numbers for a qualified identifier.

Curly brackets { } are each synonymous with the corresponding square bracket.

Channel Genstat accesses the files on the computer via *channels*. For each type of file, there is a set of numbered channels that can be used to reference different files in the various input/output directives. For example, there are five input channels, numbered 1 up to 5. Likewise, there are five output channels. Genstat distinguishes between the different types of channel, so you can have one file attached to input channel 3 and a different file simultaneously attached to output channel 3. (See the OPEN directive.)

Character The characters used to form Genstat statements are a subset of those available on most computers. For the Genstat language they are classified as brackets, digits, letters, punctuation symbols, simple operators, or special symbols.

Comment A comment consists of any series of characters that the computer can represent, enclosed by double quotes ("); comments are ignored and can appear anywhere in a Genstat program.

Data structure These are used to store information within Genstat, such as numbers, character strings or even identifiers of other data structures. Directives known as declarations are available to form each of the available types.

Device is a type of plotter selected by the DEVICE directive for use by

Genstat's high-resolution graphics commands.

Diagonal matrix is a data structure that stores the diagonal elements of a square matrix whose other values are all zero. Diagonal matrices can be declared using the DIAGONALMATRIX directive.

Digit The numerical characters 0 to 9 are known as digits in Genstat.

Directive is a standard form of instruction in the Genstat language requesting a particular action or analysis. All Genstat directives have the same syntax.

Directive name is a system word used to request a particular action or analysis from Genstat. Directive names may be abbreviated to four characters; if characters 5-8 are given, they must match the standard form, e.g. TREATMENTSTRUCTURE can be written as TREA, TREAT, TREATM, and so on, but not as TREATS. (Also see *procedure*.)

Expression is an arithmetic expression consisting of lists and functions separated by operators. An expression data structure stores a Genstat expression, and can be declared using the EXPRESSION directive.

Factor is a data structure that specifies an allocation of the units into groups. It is thus a vector that, unlike the variate or the text, takes only a limited set of values, one for each group. The groups are referred to by numbers known as levels; you can also define textual labels. Factors can be declared using the FACTOR directive.

Fixed format In fixed format, data values are arranged in specific *fields* on each line of the file. Each field consists of a fixed number of characters. There is no need for separating spaces. When data are read in fixed format in Genstat (by the READ directive), the tab character is not permitted, nor are comments.

Formula is a model formula of lists and operators defining the list of model terms involved in an analysis. A formula data structure stores a Genstat formula, and can be defined using the FORMULA directive.

Frame In Genstat graphics, *frame* refers to the available plotting area. (See the FRAME and GRAPH directives.)

Free format In *free format*, the data values are separated by one or more spaces (or tabs), and can otherwise be arranged any way you like, on one or more lines, so long as the correct order is maintained. The SEPARATOR option of the READ directive allows separators other than spaces to be requested.

Function denotes a standard operation in an expression or formula, with the form "*function-name* (sequence of *lists* and/or *expressions* separated by ,)". The function-name is a system word and may be abbreviated to four characters; if characters 5-8 are given, they must match the standard form. A wide range of functions are available, for operations ranging from transformations to the

calculation of summary statistics.

Identifier is the name given to a particular data structure within a Genstat program. The first character of an identifier must be a letter; any others can be either letters or digits. Only the first 32 characters are significant; subsequent characters are ignored. The directive `SET` allows you to specify whether or not the case of the letters (small or capital) is to be significant, e.g. whether `LENGTH` is the same as `Length`, or whether only the first eight characters should be significant (as in Releases before 4.2).

Inconsistent structure Genstat data structures that depend on other structures can be left in an inconsistent form if these other structures are deleted. For example, a table depends on its classifying factors.

Item is a number, a string, an identifier, a system word, a missing value, or an operator.

Justification is the process of ensuring that columns of information line up down the left-hand side (left justification) or down the right-hand side (right justification).

Label of a factor is one of the possible textual values that the factor can store.

Letter Letters in Genstat are the upper-case (capital) letters `A` to `Z`, the lower-case letters `a` to `z`, the underline symbol (_), and the percent character (`%`).

Level of a factor is one of the possible (numerical) values that the factor can store.

Line printer is the general term used to denote a character-based graphics device.

List is a sequence of items separated by commas. In an identifier list, each item is an identifier or an unnamed structure, while number or string lists contain numbers or strings respectively. Lists can contain pre- or post-multipliers. Identifier and number lists can contain progressions.

Loop is a series of Genstat commands that is repeated several times, possibly operating on different data structures each time. (See the `FOR` directive.)

LRV structure is a compound data structure storing latent roots and vectors, mainly used in multivariate analysis. They can be declared using the `LRV` directive.

Macro is a Genstat text structure containing a section of a Genstat program. The text must have an unsuffixed identifier. It can be substituted into the program, by giving its identifier, preceded by a contiguous pair of substitution symbols (`##`). The substitution takes place as soon as Genstat reads the pair of hashes. (However, Genstat also has the `EXECUTE` directive, which allows a text containing a list of statements to be executed for example within a loop or procedure.)

Margin The margin of a Genstat table is a section of the table that contains summaries over the values of one or more of the classifying factors. A marginal term of a term *T* in a statistical model is a term composed of factors or variates that are a subset of those of *T*.

Matrix is a data structure that stores a rectangular array of numbers. Matrices can be declared using the `MATRIX` directive.

Missing value is denoted within a Genstat program by one asterisk (`*`). When reading data, a series of contiguous asterisks or an asterisk followed by letters or digits is treated as a missing value too, and other characters can also be defined to represent missing values.

Multiplier allows repetitive lists to be specified concisely. A multiplier may be a number, or the substitution symbol (`#`) followed by a single-valued numerical data structure.

Post-multiplier is given immediately after the second of a pair of round brackets enclosing a list of identifiers, numbers, or strings, and has the effect of repeating the entire list, as a whole, the specified number of times.

Pre-multiplier occurs immediately before the initial (round) bracket of a pair enclosing a list of identifiers, numbers, or strings and has the effect of repeating each item, in turn, the specified number of times.

Number is a sequence of digits, optionally containing a decimal point (`.`). The sequence can be preceded by a sign (`+` or `-`) and can be followed by an exponent: i.e. the letter `E` or `D` (in upper or lower case) optionally followed by spaces, then a sequence of digits optionally preceded by a sign.

Operator is a symbol or symbols denoting an operation in an expression or formula:

Simple `+` (addition), `-` (subtraction), `*` (multiplication or product), `/` (division or nesting), `.` (interaction), `=` (assignment), `<` (less than), `>` (greater than)

Compound `**` (exponentiation), `*+` (matrix multiplication), `-*` (crossed deletion), `-/` (nested deletion), `//` (pseudo-term linkage), `.EQ.` or `==` (equality), `.NE.` or `/=` or `<>` (non-equality), `.LE.` or `<=` (less than or equal to), `.GE.` or `>=` (greater than or equal to), `.LT.` (less than), `.GT.` (greater than), `.EQS.` (string equality), `.NES.` (string non-equality), `.IN.` (set inclusion), `.NI.` (set non-inclusion), `.IS.` (identifier equivalence), `.ISNT.` (identifier non-equivalence), `.AND.` (logical *and*), `.OR.` (logical *or*), `.EOR.` (logical *either or*), `.NOT.` (logical *not*).

Only `+ - * / . -/ -*` and `//` may occur in formulae, while `. -* -/` and `//` cannot occur (as operators) in expressions.

Precedence The list below shows the order in which the operators are evaluated when they are used in expressions, if brackets are not used to make the order explicit:

(1) `.NOT.` Monadic `-`

(2) `.IS.` `.ISNT.` `.IN.` `.NI.` `*+`

(3) `**`
(4) `*` `/`
(5) `+` Dyadic `-`
(6) `<` `>` `==` `<=` `>=` `/=` `<>` `.LT.` `.GT.` `.EQ.` `.LE.` `.GE.` `.NE.` `.NES.`
(7) `.AND.` `.OR.` `.EOR.`
(8) `=`

(*Monadic minus* means the use of the minus sign in a negative number: for example, −1.) Within each class, operations are done from left to right within a expression, unless brackets are used to indicate some other order.

Option Options specify arguments that are global within a Genstat statement: i.e. they apply to all the items in the parameter list(s). Often, but not always, options have default values and so need not be specified.

Option name is a system word that identifies a particular option setting. It can be abbreviated to the minimum number of characters required to distinguish it from the options that precede it in the prescribed order for the directive or procedure concerned; for directives, four characters are always sufficient.

Option sequence is a series of option settings separated by semi-colons (`;`).

Option setting has the form

option-name = *list*, *expression* or *formula*

"*option-name* =" can be omitted if the settings are given in the prescribed order for the directive or procedure concerned: i.e. the name may be omitted for the first setting if this is for the first prescribed option, and for subsequent settings if the previous setting was for the option immediately before the current one in the prescribed order.

Parameter Parameters specify parallel lists of arguments for a statement: i.e. the statement (with its option settings) operates for the first item in each list, then the second, and so on. The number of times that this happens is determined by the length of the parameter list that is first in the prescribed order for the directive or procedure concerned. Subsequent lists are recycled if they are shorter than the first list.

Parameter name is a system word that identifies which parameter is being set. It may be abbreviated to the minimum number of characters required to distinguish it from the parameters that precede it in the prescribed order for the directive or procedure concerned; for directives, four characters are always sufficient.

Parameter sequence is a series of parameter settings separated by semi-colons (`;`).

Parameter setting has the form

parameter-name = *list*, *expression* or `formula"`*parameter-name* =" can be omitted if the settings are given in the prescribed order for the directive or procedure concerned: i.e. the name may be omitted for the first setting if this is for the first prescribed parameter, and for subsequent settings if the previous setting was for the parameter immediately before the current one in the prescribed order. For directives or procedures with only a single parameter, no parameter name is defined.

Pen All the elements of a high-resolution graph, such as symbols, lines, axes, titles, labels, annotation, and filled polygons are drawn by *pens*, which have associated definitions covering various attributes, like colour, font, and symbol type. The pen also indicates the plotting method, that is, what kind of plot is to be drawn. See the `PEN` directive.

Pointer is a data structure that stores a series of identifiers, pointing to other data structures. Pointers can be declared using the `POINTER` directive.

Procedure This is a structure that contains Genstat statements, and fulfils the role of the subroutine in the Genstat language. The use of a procedure looks just like the use of a Genstat directive. All data structures within the procedure are local (i.e. they cannot be referenced, or confused, with data structures outside the procedure); input and output structures for the procedure are defined by option and parameter settings in the procedure call.

Procedure name is a letter followed by letters and/or digits. Procedure names can be defined with up to 32 characters; if more than 32 are given, characters 33 onwards are ignored. The case of the letters (small or capital) is also ignored. When using a procedure, the name can be abbreviated to as few as four characters, provided there is no ambiguity with the names of directives or other procedures. Directives and procedures in the official Genstat library are all defined to have names that are distinct within the first four characters so there should be no problem unless you (or your site) have defined procedures with ambiguous names. If so, Genstat selects the command to use according to the following order of priority: directives, user-defined procedures, procedures in libraries attached by the user (in order of channel number), procedures in the site library, and procedures in the official library.

Procedure Library The Genstat Procedure Library contains procedures contributed not only by the writers of Genstat but also by knowledgeable Genstat users from many application areas and countries. The Library is controlled by an Editorial Board, who check that the procedures are useful and reliable, and maintain standards for the documentation. It is regularly extended and updated, independently to the releases of Genstat itself, and these revised versions are distributed automatically to all supported Genstat sites. Information about the Library is available using procedures in the `help` module of the Library. Other modules cover, for example, manipulation, graphics and various types of statistical analysis. These procedures are all accessed automatically by Genstat, when required. Instructions for authors of procedures can be obtained using procedure `NOTICE`. You can also form your own procedure libraries using the `STORE` directive.

Program is a series of statements, ending with the statement `STOP`.

Progression Lists of numbers ascending or descending with equal increments can be specified succinctly using the form "*number*, *number* . . . *number*" where the first two numbers define the first two elements in the list (and thus the increment) and the list ends with

the value beyond which the third number would be passed. For lists with an increment of plus or minus one, the second number can be omitted, to give the form "*number* . . . *number*".

Punctuation symbol
The Genstat punctuation symbols are:

colon (`:`) indicates the end of a statement;

comma (`,`) separates items;

double quote (`"`) is used to show the beginning and end of a comment;

equals (=) separates an option name or parameter name from its setting;

newline is synonymous with colon, by default, but directive `SET` can request that it be ignored;

semi-colon (`;`) separates lists;

single quote (`'`) is used to show the beginning and end of a string (left single quote (`) is synonymous with single quote);

space can appear between items or can be omitted altogether if the items are already separated by another punctuation symbol, a bracket, an operator, or an ampersand;

tab the tab character is treated as a synonym of space everywhere except within texts and comments or if reading in fixed format (when it is treated as a fault).

Qualified identifier
These may occur in a list of identifiers to define subsets of the values of a data structure (i.e. sub-structures). The form is "*identifier* `$` *qualifier*", where the *qualifier* is a sequence of identifier lists enclosed in square brackets. For factors, variates, and texts, the qualifier has a single list, each element of which defines a subset of the vector concerned. For matrices there are two lists running in parallel, one for each dimension. For a symmetric matrix, there can be either one or two lists, depending on whether or not its two dimensions are to be subset in the same way; one list forms a symmetric matrix, and two lists forms a rectangular matrix, For a diagonal matrix there is a single list. Tables cannot be qualified. The elements of the qualifier lists can be scalars, numbers, variates, quoted strings, or texts. The set of units defined by an element in the qualification list is built up, by taking its values one at a time. Positive numbers (or texts or strings) add units to the set, while negative numbers delete the corresponding units from the set (if already there). A missing value can be used to include all the units, and one of these will be included implicitly at the start of the qualification list if the first element of the list is negative. More details, and examples, are given in Section 4.1.6 of the *Guide to the Genstat Command Language: Part 1 Syntax and Data Management*.

Save structure
is a special-purpose structure defined within Genstat for saving information, for example from an analysis, so that further output can be obtained without repeating all the calculations.

Scalar
is a data structure that stores a single number. Scalars can be declared using the `SCALAR` directive.

Special symbol
The special symbols in Genstat are as follows:

ampersand (`&`) repeats the previous statement name (unless that statement contained a syntax error) and any option settings that are not

explicitly changed;

asterisk (*) denotes a missing value (and is also used as an operator);

backslash (\) is the continuation symbol, typed at the end of a line to indicate that the current statement continues onto the next line (this is unnecessary when directive SET has been used to specify that newline is to be ignored);

dollar ($) precedes a list of unit names or numbers (enclosed in square brackets) that define subsets of a factor, variate, matrix, symmetric matrix, diagonal matrix, or text;

exclamation mark (!) indicates an unnamed structure (vertical bar (|) is synonymous with exclamation mark);

hash (#) is the substitution symbol; when used on its own (i.e. followed just by a punctuation symbol) it represents the default setting of an option; alternatively, it can be followed by the identifier of a data structure whose values are to be inserted at that point in a Genstat statement (the substitution takes place immediately before the statement is executed). A pair of contiguous substitution symbols (##) is used to introduce a macro.

SSPM structure is a compound data structure storing sums of squares and products, means and ancillary information for use in regression and multivariate analysis. SSPMs can be declared using the SSPM directive.

Standard order The values of a set of factors are said to be in *standard order* if their units are arranged so that the levels of the first factor occur in the same order as in its levels vector then, within each level of the first factor, the levels of the second factor are arranged similarly, and so on. (See the GENERATE directive.)

Statement is an instruction in the Genstat language; it has the form

statement-name [*option-sequence*] *parameter-sequence terminator*

If no option settings are given, the square brackets can be omitted. The *terminator* is colon (:), ampersand (&) or newline (unless directive SET has indicated that this is to be ignored).

Statement name is the name of either a directive or a procedure.

String is a sequence of characters forming one unit (or line) of a Genstat text structure. In most contexts, the string must be quoted: i.e. enclosed in single quotes ('). Quoted strings may contain any of the characters available on the computer. However, if single quote ('), double quote ("), or the continuation symbol (\) are required as characters within a quoted string, they must each be typed twice to distinguish this use from their action in, respectively, terminating the string, introducing a comment within the string, or indicating continuation. Newline within a quoted string is taken to terminate the current (quoted) string and begin another one, unless the newline is within a comment or preceded by an (unduplicated) continuation symbol (\), or unless directive SET has specified that newline is to be ignored. Unquoted strings can occur in unnamed texts, or in option or parameter settings where you have to specify a particular string from a prescribed set of alternatives; an unquoted string must have a letter as its first character and contain only letters or digits.

Subfile Backing store files are partitioned into subfiles. These are self-contained, and can be used completely independently of each other.

Subset selection An identifier list can contain qualified identifiers, each defining a list of subsets of the values of the data structure concerned.

Suffix Elements of pointers can be referred to by suffixes. Each suffix takes the form of an identifier list enclosed in square brackets; the list can contain numbers, scalars, or variates to reference an element or elements by number, or texts or quoted strings to reference by label. A null list within the brackets is taken to mean all the elements of the pointer in turn. Where a pointer has other pointers as its elements, their elements can be referred to in the same way, and so the original identifier may be followed by several suffix lists each contained in its own pair of square brackets; these define a list of elements, one for each combination of an element from each suffix list, taking the combinations in an order in which the last list cycles through its elements fastest, then the next to last list, and so on.

Symmetric matrix is a data structure that stores the lower triangle (including the diagonal) of a symmetric square matrix.

System word is a letter followed by letters and/or digits with a special meaning within the Genstat language, e.g. directive, option, parameter, or function names. The case of the letters (small/capital) is not significant; the abbreviation rules vary according to context.

Table is a data structure that stores a multi-dimensional array of numbers, each dimension classified by a factor. Thus a table can be used to hold a summary of data that are classified (by the factors) into groups. Tables can be declared using the `TABLE` directive.

Text is a data structure that stores a series of strings, each one representing a line of textual information. Texts can be declared using the `TEXT` directive.

Tree is a data structure that represents hierarchical structures like classification trees, identification keys and regression trees. Trees can be declared using the `TREE` directive.

TSM structure is a compound data structure storing a model for use in Box-Jenkins modelling of time series. TSMs can be declared using the `TSM` directive.

Unknown cell of a table is used to store the relevant summary of all the observations for which any of the classifying factors of the table has a missing value; these observations cannot be assigned to any cell of the table itself. (See `TABLE`.)

Unnamed structure An identifier list may contain unnamed variates, scalars, texts, pointers, expressions, or formulae. An unnamed structure consists of an exclamation mark, followed by the type code, and then the values contained in round brackets. The type code is `E` for expression, `F` for formula, `P` for pointer, `S` for scalar, `T` for

text, or V for variate. If no code is given, variate is assumed by default.

Variate is a data structure that stores a series of numbers. Variates can be declared using the VARIATE directive.

Vector is a series of values, notionally arranged in a column. Genstat has three different types of vector: factors, texts, and variates.

Window In high-resolution graphics, a window is a rectangular segment of the frame used to plot a particular graph. (See the FRAME directive.)

1.3 Data structures

Data structures store the information on which a Genstat program operates. Structures can be defined, or declared, by a Genstat statement known as a declaration. The directive for declaring each type of structure has the same name as given to that type of structure, for example SCALAR to declare a scalar (or single-valued numerical structure), and so on. These are the directives, with details of their corresponding data structures:

SCALAR	single number
VARIATE	series of numbers
TEXT	series of character strings (or lines of text)
FACTOR	series of group allocations (using a pre-defined set of numbers or strings to indicate the groups)
MATRIX	rectangular matrix
SYMMETRICMATRIX	symmetric matrix
DIAGONALMATRIX	diagonal matrix
TABLE	table (to store tabular summaries like means, totals etc)
DUMMY	single identifier
POINTER	series of identifiers (e.g. to represent a set of structures)
EXPRESSION	arithmetic expression
FORMULA	model formula (to be fitted in a statistical analysis)
LRV	latent roots and vectors
SSPM	sums of squares and products with associated information such as means
TREE	tree (as used to represent classification trees, identification keys and regression trees)
TSM	model for Box-Jenkins modelling of time series

You can rename a data structure, or create a new one with attributes the same as those of an existing structure.

RENAME	renames a data structure, to give it a new identifier
DUPLICATE	forms new data structures with attributes taken from an existing structure
PDUPLICATE	duplicates a pointer, with all its components
SETNAME	sets the identifier of a data structure to be one specified in a text

You can also define data structures whose contents are customized for particular tasks.

STRUCTURE	defines a customized data structure
DECLARE	declares one or more customized data structures

There are commands to access and display the attributes of data structures, and to work out the best formats for printing their values.

DUMP	prints attributes of data structures and other internal information
DECIMALS	sets the number of decimals for a structure, using its round-off
GETATTRIBUTE	accesses attributes of data structures
LIST	lists details of the data structures that currently exist in your program
MINFIELDWIDTH	calculates minimum field widths for printing data structures

1.4 Program control

A Genstat program consists of a sequence of one or more *jobs*. The first job starts automatically at the start of the program. Subsequent jobs can be initialized by the JOB and ENDJOB directives:

JOB	starts a Genstat job (ending the previous one if necessary)
ENDJOB	ends a job

The whole program is terminated by a STOP directive:

STOP	ends a Genstat program

Statements within a program can be repeated using a FOR loop. The loop is introduced by a FOR statement. This is followed by the series of statements that is to repeated (that is, the contents of the loop), and the end of the loop is marked by an ENDFOR statement. Parameters of the FOR directive allow lists of data structures to be specified so that the statements in the loop operate on different structures each time that it is executed.

FOR	indicates the start of a loop
ENDFOR	marks the end of a loop

Genstat has two ways of choosing between sets of statements. The block-if structure consists of one or more alternative sets of statements. The first set is introduced by an IF statement. There may then be further sets introduced by ELSIF statements. Then there may be a final set introduced by an ELSE statement, and the whole structure is terminated by an ENDIF structure. The IF statement, and each ELSIF statement, contains a single-valued logical expression. Genstat evaluates each one in turn and executes the statements following the first TRUE logical found; if none of them is true, Genstat executes the statements following the ELSE statement (if any).

IF	introduces a block-if structure
ELSIF	introduces an alternative set of statements in a block-if structure
ELSE	introduces a default set of statements for a block-if structure
ENDIF	marks the end of a block-if structure

The multiple-selection structure consists of several sets of statements. The first is introduced by a CASE statement. Subsequent sets are introduced by OR statements. There can then be a final, default, set introduced by an ELSE statement, and the end of the structure is indicated by an ENDCASE statement. The parameter of the CASE statement is an expression which must produce a single number. Genstat rounds this to the nearest integer, *n* say, and then executes the *n*th set of statements. If there is no *n*th set, the statements following the ELSE statement are executed (if any).

CASE	introduces a multiple-selection structure
OR	introduces an alternative set of statements for a multiple-selection structure
ELSE	introduces a default set of statements for a multiple-selection structure
ENDCASE	marks the end of a multiple-selection structure

Sequences of statements can be formed into Genstat procedures for convenient future use. The use of a procedure looks just like one of the Genstat directives, with its own options and parameters, which transfer information to and from the procedure. Otherwise the procedure is completely self-contained. The start of a procedure is indicated by a PROCEDURE statement. Then OPTION and PARAMETER statements can be given to define the arguments of the procedure. These are followed by the statements to be executed when the procedure is called, terminated by an ENDPROCEDURE statement.

PROCEDURE	introduces a procedure, and defines its name
OPTION	defines the options of a procedure
PARAMETER	defines the parameters of a procedure
CALLS	lists library procedures called by a procedure
ENDPROCEDURE	indicates the end of a procedure
WORKSPACE	accesses "private" data structures for use in procedures

Any control structure (job, block-if structure, loop, multiple-selection structure or procedure) can be abandoned using an EXIT statement. Also, execution of any of these structures can be interrupted explicitly with a BREAK statement, or implicitly by using DEBUG. Once DEBUG has been entered, Genstat will produce breaks automatically at regular intervals, until it meets an ENDDEBUG statement. You can also issue a faults, warnings or messages.

EXIT	exits from a control structure
BREAK	suspends the execution of a control structure
ENDBREAK	continues execution of a control structure, following a break
DEBUG	can cause a break to take place after the current statement (and at specified intervals thereafter), or immediately after the next fault
ENDDEBUG	cancels DEBUG
FAULT	evaluates a logical expression to decide whether to issue a diagnostic, i.e. a fault, warning or message
DISPLAY	prints, or reprints, diagnostic messages

Macros within a procedure are substituted as soon as they are met during the definition of the procedure. However, it is also possible to execute a set of statements (contained in a text) during execution of the procedure. This can also be useful within loops.

EXECUTE	executes the statements contained within a text

Other commands that may be useful in programs and procedures include the following:

CAPTION	prints captions in standardized formats
COMMANDINFORMATION	provides information about whether (and how) a command has been implemented
COUNTER	increments a multi-digit counter using non base-10 arithmetic
GET	accesses details of the "environment" of a Genstat job
GETTEMPFOLDER	gets the name of the Genstat temporary folder
SET	sets details of the "environment" of a Genstat job
ENQUIRE	provides details about files opened by Genstat
GETATTRIBUTE	accesses attributes of structures
CHECKARGUMENT	checks the arguments of a procedure
LIBEXAMPLE	accesses examples and source code of library procedures
SETCALCULATE	performs Boolean set calculations on the contents of vectors or pointers
SETRELATE	compares the sets of values in two data structures
SPSYNTAX	puts details about the syntax of commands into a spreadsheet
SYNTAX	obtains details of the syntax of a command and the source code of a procedure

ASSIGN	sets elements of pointers and dummies
DELETE	deletes the attributes and values of structures
DUPLICATE	forms new data structures with attributes taken from an existing structure
%LOG	adds text into the Input Log window in the Genstat client
%MESSAGEBOX	displays text in a dialog in the Genstat client
%OPEN	open a binary file for use with %WRITE
%FPOSITION	returns the current position in the binary file opened by %OPEN
%WRITE	writes values of data structures to a binary file opened by %OPEN
%CLOSE	closes the binary file opened by %OPEN
%SLEEP	pauses execution of the server for a time specified in seconds
%TEMPFILE	creates a unique temporary file in the Genstat temporary folder

In some implementations of Genstat, it is possible to suspend the execution of Genstat and return to the operating system of the computer to execute commands, for example to list or edit files on the computer. Likewise, it may be possible to halt the execution of Genstat to execute some other computer program. You can also execute code within an external DLL using the EXTERNAL directive and the OWN function. The OWN directive provides another way of running a user's program from within Genstat. The OWN subroutine, within the Fortran code of Genstat, needs to be modified to call the program. The new code must then be recompiled and linked into a new version of Genstat.

SUSPEND	suspends the execution of Genstat to carry out operating-system commands
PASS	runs another computer program, taking data from Genstat and transferring results back
SHELLEXECUTE	launches executables or opens files in another application using their file extension
EXTERNAL	declares an external function in a DLL for use by the OWN function
OWN	executes the user's own code linked into Genstat
BGXGENSTAT	runs WinBUGS or OpenBUGS from Genstat in batch mode using scripts
RXGENSTAT	submits a set of commands externally to R and reads the output

2 Data handling

2.1 Input and output

Data can be read into Genstat data structures using the READ and SPLOAD directives or the FILEREAD and TX2VARIATE procedures:

READ	reads data from an input file, an unformatted file or a text
FILEREAD	reads data from a text file, assumed to be in a rectangular array
SPLOAD	loads data from a Genstat spreadsheet file
TX2VARIATE	reads values into a variate from a text structure

Files can be connected to input, output or other channels during execution of a Genstat program. Channels can also be closed, terminating the connection, so that they can be attached to other files.

OPEN	opens files and connects them to Genstat input/output channels
CLOSE	closes files, freeing the channels to which they were attached

The channel from which input statements are taken can be changed, as can the channel to which output is sent. It is also possible to send a transcript (or copy) of input and/or output to output files, to skip sections of input or output files, and to obtain information about the files connected to each channel.

INPUT	specifies the channel from which subsequent statements should be read
RETURN	returns to the previous input channel
OUTPUT	specifies the channel to which future output should be sent
COPY	requests a transcript of subsequent input and/or output
SKIP	skips lines of input or output files
ENQUIRE	provides details about files opened by Genstat

The following commands allow you to generate output.

PRINT	prints data in tabular form to an output file or text
CAPTION	prints captions and titles in standardized formats
LIST	lists details of the data structures that currently exist in your program
PAGE	moves to the top of the next page of an output file
PFACLEVELS	prints levels and labels of factors
PLINK	prints a link to a graphics file into an HTML file
DISPLAY	repeats the last Genstat diagnostic
DUMP	prints attributes of data structures and other internal information
DECIMALS	sets the number of decimals for a structure, using its round-off
MINFIELDWIDTH	calculates minimum field widths for printing data structures

You can copy, delete and rename files:

FCOPY	makes copies of files
FDELETE	deletes files
FRENAME	renames files

You can define menus:

QDIALOG	produces a modal dialog box to obtain a response from the user
QUESTION	obtains a response using a Genstat menu (formed using QDIALOG)
QFACTOR	allows the user to decide to convert texts or variates to factors
QLIST	presents a sequence of menus to obtain a response from a list

The values of a data structure, with all its defining information, can be stored in a sub-file of a "backing-

store" file. It can then be retrieved in a later job, without the need to repeat the definitions. The current state of the whole job can also be dumped to an unformatted file, so that it can be picked up and continued on a later occasion.

STORE	stores data structures in a backing-store file
RETRIEVE	retrieves data structures from a backing-store file
CATALOGUE	displays the contents of a backing-store file
MERGE	copies sub-files of backing-store files into a single file
RECORD	dumps the complete details of a job
RESUME	reads and restarts a recorded job

Genstat has several additional commands for accessing data from spreadsheets, databases and other systems. However, these may be unavailable in some implementations.

CSPRO	reads a data set from a CSPro survey data file and dictionary, and loads it into Genstat or puts it into a spreadsheet file
EXPORT	outputs data structures in foreign file formats, or as plain or comma-delimited text
IMPORT	reads data in a foreign file format, and loads it or converts it to a spreadsheet file
DBCOMMAND	runs an SQL command on an ODBC database
DBEXPORT	update an ODBC database table using data from Genstat
DBIMPORT	loads data into Genstat from an ODBC database
DBINFORMATION	loads information on the tables and columns in an ODBC database
DDEEXPORT	sends data or commands to a Dynamic Data Exchange server
DDEIMPORT	gets data from a Dynamic Data Exchange (DDE) server
GRIBIMPORT	reads data from a GRIB2 meteorological data file, and loads it or converts it to a spreadsheet file
SPCOMBINE	combines spreadsheet and data files, without reading them into Genstat
%CD	changes the current directory

2.2 Calculations and manipulation

The directive CALCULATE allows arithmetic calculations on the values of any numeric data structure; logical tests can also be done on numerical and textual values. Functions and operators are available for a very wide range of calculations on matrices and tables. Another general directive is EQUATE, which allows values to be copied from one set of data structures to another; the structures must store values of the same mode (for example, numbers or text), but need not be of the same type. Structure values can be deleted to save space within Genstat; attributes can also be deleted so that the structure can be redefined, for example as another type. Contents of data structures can be compared, to see if they contain the same distinct items, or whether the distinct values in one structure are a subset of those in another. You can also find all the locations where a number, identifier or string occurs within a data structure.

CALCULATE	performs arithmetic and logical calculations
DELETE	allows values and attributes of data structures to be deleted
EQUATE	copies values between sets of data structures
SETRELATE	compares the sets of values in two data structures
GETLOCATIONS	finds locations of an identifier within a pointer, or a string within a factor or text, or a number within any numerical data structure

There are several general directives for manipulating vectors (variates, factors or texts). Units of vectors can be sorted into systematic order or into random order. Boolean arithmetic can be performed on their contents, or you can form all the ways of partitioning them into subsets. A "restriction" can be associated with a vector, so that subsequent statements operate on only a subset of its units. A default length and

labelling can be defined for vectors formed later in the job. Facilities for specific types of vector allow interpolation of values for variates, monotonic regression, calculation of regression quantiles, generation of factor values, and concatenation, editing and searching of text.

SORT	sorts units of vectors into alphabetic or numerical order of an index vector, or forms a factor from a variate or text
SETCALCULATE	performs Boolean set calculations on the contents of vectors and pointers
SETALLOCATIONS	runs through all ways of allocating a set of objects to subsets
RESTRICT	defines a "restriction" on the units of a vector
UNITS	defines default length or labelling for vectors defined subsequently in the job
INTERPOLATE	calculates variates of interpolated values
FRQUANTILES	forms regression quantiles
MONOTONIC	fits an increasing monotonic regression
GROUPS	forms a factor (or grouping variable) from a variate or text, together with the set of distinct values that occur
CONCATENATE	concatenates together lines of text vectors
EDIT	line editor for units of text vectors
TXBREAK	breaks a text structure into individual words
TXCONSTRUCT	forms a text structure by appending or concatenating values of scalars, variates, texts, factors or pointers; allows the case of letters to be changed or values to truncated and reversed
TXFIND	finds a subtext within a text structure
TXINTEGERCODES	converts textual characters to and from their corresponding integer codes
TXPOSITION	locates strings within the lines of a text structure
TXREPLACE	replaces a subtext within a text structure

Another general directive allows you to run many algorithms from the Numerical Algorithms Group Library, for example to build mathematical models.

NAG	calls an algorithm from the NAG Library

Other facilities for vectors are provided by the procedures in the Genstat Procedure Library, including

APPEND	appends a list of vectors of compatible types
FACAMEND	permutes the levels and labels of a factor
FACCOMBINATIONS	forms a factor to indicate observations with identical values of a set of variates, texts or factors
FACDIVIDE	represents a factor by factorial combinations of a set of factors
FACEXCLUDEUNUSED	redefines the levels and labels of a factor to exclude those that are unused
FACGETLABELS	obtains the labels for a factor if it has been defined with labels, or constructs labels from its levels otherwise
FACMERGE	merges levels of factors
FACPRODUCT	forms a factor with a level for every combination of other factors
FACSORT	sorts the levels of a factor according to an index vector
FACLEVSTANDARDIZE	redefines a list of factors so that they have the same levels or labels
FACUNIQUE	redefines a factor so that its levels and labels are unique
FBETWEENGROUPVECTORS	forms variates and classifying factors containing within-group summaries to use e.g. in a between-group analysis
FDISTINCTFACTORS	checks sets of factors to remove any that define duplicate classifications
FMFACTORS	forms a pointer of factors representing a multiple-response

FFREERESPONSEFACTOR	forms multiple-response factors from free-response data
FREGULAR	expands vectors onto a regular two-dimensional grid
FRESTRICTEDSET	forms vectors with the restricted subset of a list of vectors
FROWCANONICALMATRIX	puts a matrix into row canonical, or reduced row echelon, form
FSTRING	forms a single string from a list of strings in a text
FTEXT	forms a text structure from a variate
FUNIQUEVALUES	redefines a variate or text so that its values are unique
FWITHINTERMS	forms factors to define terms representing the effects of one factor within another factor
FVSTRING	forms a string listing the identifiers of a set of data structures
GRANDOM	generates pseudo-random numbers from probability distributions
GRMNOMIAL	generates multinomial pseudo-random numbers
GRMULTINORMAL	generates multivariate normal pseudo-random numbers
JOIN	joins or merges two sets of vectors together, based on classifying keys
MVFILL	replaces missing values in a vector with the previous non-missing value
ORTHPOLYNOMIAL	calculates orthogonal polynomials
QUANTILE	calculates quantiles of the values in a variate
RANK	produces ranks, from the values in a variate, allowing for ties
RESHAPE	reshapes a data set with classifying factors for rows and columns, into a reorganized data set with new identifying factors
SAMPLE	samples from a set of units, possibly stratified by factors
SVSAMPLE	constructs stratified random samples
STACK	combines several data sets by "stacking" the corresponding vectors
STANDARDIZE	standardizes columns of a data matrix to have mean 0 and variance 1
SUBSET	forms vectors containing subsets of the values in other vectors
TXPAD	pads strings of a text structure with extra characters so that their lengths are equal
TXPROGRESSION	forms a text containing a progression of strings
TXSPLIT	splits a text into individual texts, at positions on each line marked by separator character(s)
TX2VARIATE	converts text structures to variates
UNSTACK	splits vectors into individual vectors according to levels of a factor
VEQUATE	equates values across a set of data structures
VINTERPOLATE	performs linear and inverse linear interpolation between variates

There are several procedures for calculating or fitting splines, and for manipulating series of observations of a theoretical curve.

SPLINE	calculates a set of basis functions for M-, B- or I-splines
LSPLINE	calculates design matrices to fit a natural polynomial or trignometric L-spline as a linear mixed model
NCSPLINE	calculates natural cubic spline basis functions (for use e.g. in REML)
PENSPLINE	calculates design matrices to fit a penalized spline as a linear mixed model
PSPLINE	calculates design matrices to fit a P-spline as a linear mixed model
RADIALSPLINE	calculates design matrices to fit a radial-spline surface as a linear mixed model
TENSORSPLINE	calculates design matrices to fit a tensor-spline surface as a linear mixed model

ALIGNCURVE	forms an optimal warping to align an observed series of observations with a standard series
BASELINE	estimates a baseline for a series of numbers whose minimum value is drifting
PEAKFINDER	finds the locations of peaks in an observed series

Directives are available for eigenvalue, QR and singular-value decompositions of matrices, and to form the values of SSPM structures.

FLRV	calculates latent roots and vectors (that is, eigenvalues and eigenvectors)
QRD	calculates QR decompositions of matrices
SVD	calculates singular-value decompositions of matrices
FSSPM	calculates values for SSPM structures (sums of squares and products, means, etc.)

Procedures in the Library for operating on matrices include

FCORRELATION	forms the correlation matrix for a list of variates
PARTIALCORRELATIONS	calculates partial correlations for a list of variates
FHADAMARDMATRIX	forms Hadamard matrices
FPROJECTIONMATRIX	forms a projection matrix for a set of model terms
FRTPRODUCTDESIGNMATRIX	forms summation, or relationship, matrices for model terms
FVCOVARIANCE	forms the variance-covariance matrix for a list of variates
GINVERSE	calculates the generalized inverse of a matrix
LINDEPENDENCE	finds the linear relations associated with matrix singularities
MPOWER	forms integer powers of a square matrix
POSSEMIDEFINITE	calculates a positive semi-definite approximation of a non-positive semi-definite symmetric matrix
VMATRIX	copies values and row/column labels from a matrix to variates and texts

Tables can be formed containing summaries of values in variates: totals, minimum and maximum values, quantiles, numbers of missing and non-missing values, means and variances. Manipulations of multi-way structures include the ability to add various types of marginal summaries to tables, and to combine "slices" of tables, of matrices or of variates.

TABULATE	forms tables of summaries of the values of a variate
MARGIN	calculates or deletes margins of tables
COMBINE	combines or omits "slices" of tables, matrices or variates

Procedures in the Library for operating on tables include

BACKTRANSFORM	calculates back-transformed means with approximate standard errors and confidence intervals
MEDIANTETRAD	gives robust identification of multiple outliers in 2-way tables
MTABULATE	tabulates data classified by multiple-response factors
PERCENT	expresses the body of a table as percentages of one of its margins
SVBOOT	bootstraps data from random surveys
SVCALIBRATE	performs generalized calibration of survey data
SVGLM	fits generalized linear models to survey data
SVREWEIGHT	modifies survey weights adjusting to ensure that their overall sum weights remains unchanged
SVSAMPLE	constructs stratified random samples
SVSTRATIFIED	analyses stratified random surveys by expansion or ratio raising
SVTABULATE	tabulates data from random surveys, including multistage surveys

	and surveys with unequal probabilities of selection
SVWEIGHT	forms survey weights
TABINSERT	inserts the contents of a sub-table into a table
TABMODE	forms summary tables of modes of values
TABSORT	sorts tables so their margins are in ascending or descending order
TCOMBINE	combines several tables into a single table
T%CONTROL	expresses tables as percentages of control cells
VTABLE	forms a variate and set of classifying factors from a table

Directives are available for adding and removing branches of trees, and to assist in the construction and use of trees.

BASSESS	assesses potential splits for regression and classification trees
BCUT	cuts a tree at a defined node, discarding nodes and information below it
BIDENTIFY	identifies specimens using a tree
BJOIN	extends a tree by joining another tree to a terminal node
BGROW	adds new branches to a node of a tree

There are also procedures for displaying and pruning trees. These are provide basic utilities for tree-based analysis, and are used by the existing procedures for classification trees, identification keys and regression trees (BCLASSIFICATION, BKEY and BREGRESSION).

BCONSTRUCT	constructs a tree
BGRAPH	plots a tree
BPRINT	displays a tree
BPRUNE	prunes a tree using minimal cost complexity

Formulae and expressions can be interpreted, revised or constructed automatically from the contents of pointers.

FARGUMENTS	forms lists of arguments involved in an expression
FCLASSIFICATION	forms classification sets for the terms in a formula or breaks a formula up into separate formulae (one for each term)
REFORMULATE	modifies a formula or an expression to operate on a different set of data structures
SET2FORMULA	forms a model formula using structures supplied in a pointer

Values can be assigned to dummies and pointers.

ASSIGN	sets values of dummies and pointers

Aspects of the "environment" of the current job can be modified, such as whether or not Genstat starts output from a statistical analysis at the top of a new page, or whether it should pause during interactive output. New defaults can be set for options and parameters. Details of the environmental settings can be copied into Genstat data structures. Attributes of data structures can also be accessed.

SET	sets details of the "environment" of a Genstat job
SETOPTION	sets or modifies defaults of options of Genstat directives or procedures
SETPARAMETER	sets or modifies defaults of parameters of Genstat directives or procedures
GET	gets details of the "environment" of a Genstat job
GETATTRIBUTE	accesses attributes of data structures
GETNAME	forms the name of a structure according to its IPRINT attribute

There are also various specialist mathematical facilities

BPCONVERT	converts bit patterns between integers, pointers of set bits and textual descriptions
FPARETOSET	forms the Pareto optimal set of non-dominated groups
GALOIS	forms addition and multiplication tables for a Galois finite field
NCONVERT	converts integers between base 10 and other bases
PERMUTE	forms all possible permutations of the integers 1...*n*
PRIMEPOWER	decomposes a positive integer into its constituent prime powers

2.3 Graphics

The following directives for produce the plots in "line-printer" style, i.e. using the characters of ordinary textual output:

LPCONTOUR	produces contour maps of two-way arrays of numbers
LPGRAPH	produces scatter plots and line graphs
LPHISTOGRAM	plots histograms

Genstat can also produce high-resolution plots. The relevant directives have two main purposes. There are those that define the "graphics environment" for subsequent plots, and those that do the plotting. Often the default environment, set up at the start of a program, will be satisfactory. To change the graphics environment, the following directives can be used:

XAXIS	defines the x-axis in each graphical window
YAXIS	defines the y-axis in each graphical window
ZAXIS	defines the z-axis in each graphical window
AXIS	defines an oblique axis for high-resolution graphics
DEVICE	switches between graphics devices
FRAME	defines the positions of the windows within the frame
PEN	defines the properties of the graphics "pens"
DFONT	defines the default font for high-resolution graphics
DKEEP	saves information about the graphics environment in Genstat data structures
DLOAD	loads the graphics environment settings from an external file
DSAVE	saves the current graphics environment settings to an external file
GETRGB	provides a standard sequence of colours (defined by the initial defaults of the Genstat pens)

The directives for plotting high-resolution graphs are:

BARCHART	plots bar charts
DGRAPH	produces scatter plots and line graphs
DHISTOGRAM	plots histograms
DPIE	produces pie charts
DCONTOUR	produces contour maps
DBITMAP	plots a bit map of RGB colours
DSHADE	produces a shade diagram of 3-dimensional data
DSURFACE	draws a perspective plot of a two-way array of numbers
D3GRAPH	plots a 3-dimensional graph
D3HISTOGRAM	produces 3-dimensional histograms
DSTART	starts a sequence of related plots
DFINISH	ends a sequence of related plots
DDISPLAY	redraws the current graphical display
DCLEAR	clears a graphics screen

Other facilities, provided by procedures in the Library include:

BANK	calculates the optimum aspect ratio for a graph
BOXPLOT	draws box-and-whisker diagrams (schematic plots)
DARROW	adds arrows to an existing plot
DBARCHART	plots bar charts for one or two-way tables
DBIPLOT	plots a biplot from an analysis by PCP, CVA or PCO
DCIRCULAR	plots circular data
DCOLOURS	forms a band of graduated colours for graphics
DCOMPOSITIONAL	plots 3-part compositional data within a barycentric triangle
DCORRELATION	plots a correlation matrix
DELLIPSE	draws a 2-dimensional scatter plot with confidence, prediction and/or equal-frequency ellipses superimposed
DERRORBAR	adds error bars to a graph
DKEY	adds a key to a graph
DFRTEXT	adds text to a graphics frame
DFUNCTION	plots a function
DHSCATTERGRAM	plots an h-scattergram
DKSTPLOT	produces diagnostic plots for space-time clustering
DMASS	plots discrete data like mass spectra, discrete probability functions
DMSCATTER	produces a scatter-plot matrix for one or two sets of variables
DOTHISTOGRAM	plots dot histograms
DOTPLOT	produces a dot-plot
DPROBABILITY	creates probability distribution plots
DPSPECTRALPLOT	calculates an estimate of the spectrum of a spatial point pattern
DQMQTLSCAN	plots the results of a genome-wide scan for QTL × E effects in multiple environments
DQSQTLSCAN	plots the results of a genome-wide QTL search in a single environment trial
DREFERENCELINE	adds reference lines to a graph
DRESIDUALS	produces model-checking plots of residuals
DSCATTER	produces a scatter-plot matrix
DSPIDERWEB	displays spider-web and star plots
DTABLE	plots tables
DTEXT	adds text to a graph
DTIMEPLOT	produces horizontal bars displaying a continuous time record
DXDENSITY	produces one-dimensional density (or violin) plots
DXYDENSITY	produces density plots for large data sets
DXYGRAPH	draws two-dimensional graphs with marginal distribution plots alongside the y- and x-axes
DYPOLAR	produces polar plots
FFRAME	forms multiple windows in a plot-matrix for high-resolution graphics
GGEBIPLOT	plots displays to assess genotype+genotype-by-environment variation
INSIDE	determines whether points lie within a specified polygon
LORENZ	plots the Lorenz curve and calculates the Gini and asymmetry coefficients
PLINK	prints a link to a graphics file into an HTML file
RCATENELSON	performs a Cate-Nelson graphical analysis of bivariate data
RUGPLOT	draws "rugplots" to display the distribution of one or more samples
SETDEVICE	opens a graphical file and specifies the device number on basis of its extension
STEM	produces a simple stem-and-leaf chart
TRELLIS	produces trellis plots for each level of one or more factors
WINDROSE	plots rose diagrams of circular data like wind speeds

3 Statistical analyses

3.1 Basic and nonparametric statistics

Many simple statistical operations, including calculation of summary statistics, t-tests, one- and two-way analysis of variance and non-parametric tests are provided by procedures in the Library:

DESCRIBE	calculates summary statistics for variates
TALLY	forms a simple tally table of the distinct values in a vector
VSUMMARY	Summarizes a variate, with classifying factors, into a data matrix of variates and factors
TTEST	performs a one- or two-sample t-test
A2WAY	performs analysis of variance of a balanced or unbalanced design with up to two treatment factors
A2DISPLAY	provides further output following an analysis of variance by A2WAY
A2KEEP	copies information from an A2WAY analysis into Genstat data structures
AONEWAY	provides one-way analysis of variance
BLANDALTMAN	produces Bland-Altman plots to assess the agreement between two variates
CHISQUARE	calculates chi-square statistics for one- and two-way tables
CHIPERMTEST	performs a random permutation test for a two-dimensional contingency table
BNTEST	calculates one- or two-sample binomial tests
PNTEST	calculates one- or two-sample Poisson tests
FCORRELATION	forms the product moment correlation matrix for a list of variates, and tests whether the correlations are zero
PRCORRELATION	calculates probabilities for product moment correlations
CDESCRIBE	calculates summary statistics and tests of circular data
CASSOCIATION	calculates measures of association for circular data
CCOMPARE	tests whether samples from circular distributions have a common mean direction or have identical distributions
FRIEDMAN	performs Friedman's nonparametric analysis of variance
GSTATISTIC	calculates the gamma statistic of agreement for ordinal data
HCOMPAREGROUPINGS	calculates the Rand index, adjusted Rand index or Jaccard index to compare groupings defined by two factors
KAPPA	calculates a kappa coefficient of agreement for nominally scaled data
KCONCORDANCE	calculates Kendall's Coefficient of Concordance (synonym CONCORD)
KOLMOG2	performs a Kolmogorov-Smirnoff two-sample test
KRUSKAL	carries out a Kruskal-Wallis one-way analysis of variance
KTAU	calculates Kendall's rank correlation coefficient τ
LCONCORDANCE	calculates Lin's concordance correlation coefficient
MANNWHITNEY	performs a Mann-Whitney U test
MCNEMAR	performs McNemar's test for the significance of changes
MCOMPARISON	performs pairwise multiple comparison tests within a table of means
QCOCHRAN	performs Cochran's Q test for differences between related-samples
CATRENDTEST	calculates the Cochran-Armitage chi-square test for trend
CMHTEST	performs the Cochran-Mantel-Haenszel test
RUNTEST	performs a test of randomness of a sequence of observations
SIGNTEST	performs a one or two sample sign test
SPEARMAN	calculates Spearman's rank correlation coefficient
STEEL	performs Steel's many-one rank test
TEQUIVALENCE	performs equivalence, non-inferiority and non-superiority tests
WILCOXON	performs a Wilcoxon Matched-Pairs (Signed-Rank) test

STTEST	calculates the sample size for t-tests (including equivalence tests)
SBNTEST	calculates the sample size for binomial tests
SCORRELATION	calculates the sample size to detect specified correlations
SLCONCORDANCE	calculates the sample size for Lin's concordance coefficient
SMANNWHITNEY	calculates sample sizes for the Mann-Whitney test
SMCNEMAR	calculates sample sizes for McNemar's test
SPRECISION	calculates the sample size to obtain a specified precision
SSIGNTEST	calculates the sample size for a sign test

There are also facilities for calculating probabilities, and for fitting or assessing statistical distributions:

DISTRIBUTION	estimates the parameters of continuous and discrete distributions
BBINOMIAL	estimates the parameters of the beta binomial distribution
EDFTEST	performs empirical-distribution-function goodness-of-fit tests
FDRMIXTURE	estimates false discovery rates using mixture distributions
KERNELDENSITY	uses kernel density estimation to estimate a sample density
NORMTEST	performs tests of univariate and/or multivariate Normality
PRCORRELATION	calculates probabilities for product moment correlations
PRDOUBLEPOISSON	calculates the probability density for the double Poisson distribution
PRMANNWHITNEYU	calculates probabilities for the Mann-Whitney U statistic
PRSPEARMAN	calculates probabilities for Spearman's rank correlation statistic
PRWILCOXON	calculates probabilities for the Wilcoxon signed-rank statistic
RFFAMOUNT	fits harmonic models to mean rainfall amounts for a Markov model
RFFPROBABILITY	fits harmonic models to rainfall probabilities for a Markov model
RFSUMMARY	forms summaries for a Markov model from rainfall data
WSTATISTIC	calculates the Shapiro-Wilk test for Normality

3.2 Regression and generalized linear models

Genstat provides directives for carrying out linear and nonlinear regression, also generalized linear, generalized additive and generalized nonlinear models. They are designed to allow easy comparison between models, and comparison between groups of data (specified as factors). The directives for nonlinear regression can also be used for general optimization. There are three preliminary directives for defining the form of model to be fitted, of which the MODEL directive must always be given first:

MODEL	defines the response variate(s) and the type of model to be fitted
TERMS	specifies a maximal model, containing all terms to be used in subsequent regression models
RCYCLE	controls iterative fitting of generalized linear models, generalized additive models and nonlinear models, and specifies parameters and bounds for nonlinear models

Separate directives carry out the fitting of the various types of model:

FIT	fits a linear model, a generalized linear model, a generalized additive model, or a generalized nonlinear model
FITCURVE	fits a standard nonlinear regression model
FITNONLINEAR	fits a user-defined nonlinear regression model or optimizes a scalar function

Further directives are provided to allow sequential modification of the set of explanatory variables:

ADD	adds extra terms to any type of regression model
DROP	drops terms from any type of regression model

SWITCH	adds terms to, or drops them from, any type of regression model
TRY	displays results of single-term changes to a linear or generalized linear model
STEP	selects terms to include in or exclude from a linear or generalized linear model

The results of fitting the models can be displayed or stored in data structures:

RDISPLAY	displays the fit of any type of regression model
RKEEP	stores the results from any type of regression model
RKESTIMATES	saves estimates and other information about individual terms in a regression analysis
PREDICT	forms predictions from a linear or generalized linear model
RFUNCTION	estimates functions of parameters of a regression model
RSPREADSHEET	puts results from a regression, generalized linear or nonlinear model into Genstat spreadsheets

Procedure in the Library relevant to regression analysis include:

RCHECK	checks the fit of a regression model
RGRAPH	draws a graph to display the fit of a regression model
RDESTIMATES	plots one- or two-way tables of regression estimates
RPERMTEST	does random permutation and exact tests for regression or generalized-linear-model analyses
RPOWER	calculates the power (probability of detection) for regression models
RCOMPARISONS	calculates comparison contrasts amongst the levels of a factor classifying a table of regression means
RCURVECOMMONNONLINEAR	refits a standard curve with common nonlinear parameters across groups to provide s.e.'s for linear parameters
RRETRIEVE	retrieves a regression save structure from an external file
RSTORE	stores a regression save structure in an external file
RTCOMPARISONS	calculates comparison contrasts within a multi-way table of means
RWALD	calculates Wald and F tests for dropping terms from a regression
RYPARALLEL	fits the same regression model to several response variates, and collates the output
SED2ESE	calculates effective standard errors that give good approximate sed's
SEDLSI	calculates least significant intervals
LSIPLOT	plots least significant intervals
MCOMPARISON	performs pairwise multiple comparison tests within a table of predictions
BRDISPLAY	displays a regression tree
BREGRESSION	constructs a regression tree
BRPREDICT	makes predictions using a regression tree
BRVALUES	forms values for nodes of a regression tree
DILUTION	calculates Most Probable Numbers from dilution series data
DSEPARATIONPLOT	creates a separation plot for visualising the fit of a model with a dichotomous (i.e. binary) or polytomous (i.e. multi-categorical) outcome
EXTRABINOMIAL	fits models to overdispersed proportions
FIELLER	calculates effective doses or relative potencies
FITINDIVIDUALLY	fits regression models one term at a time (useful for obtaining an accumulated analysis of deviance table containing the contributions of individual terms in a generalized linear model)

FITMULTINOMIAL	fits generalized linear models with multinomial distribution
GEE	fits models to longitudinal data by generalized estimating equations
GLM	analyses non-standard generalized linear models
GLMM	fits a generalized linear mixed model
GLDISPLAY	displays further output from a GLMM analysis
GLKEEP	saves results from a GLMM analysis
GLPERMTEST	does random permutation tests for generalized linear mixed models
GLPLOT	plots residuals from a GLMM analysis
GLPREDICT	forms predictions from a GLMM analysis
GLRTEST	calculates likelihood tests to assess random terms in a generalized linear mixed model
HGANALYSE	analyses data using a hierarchical generalized linear model (HGLM) or a double hierarchical generalized linear model (DHGLM)
HGDISPLAY	displays results from an HGLM or DHGLM
HGDRANDOMMODEL	adds random terms into the dispersion models of an HGLM, so that the whole model becomes a DHGLM
HGFIXEDMODEL	defines the fixed model for an HGLM or DHGLM
HGFTEST	calculates likelihood tests for fixed terms in a hierarchical generalized linear model
HGGRAPH	draws a graph to display the fit of an HGLM or DHGLM analysis
HGKEEP	saves information from an HGLM or DHGLM analysis
HGNONLINEAR	defines nonlinear parameters for the fixed model of an HGLM
HGPLOT	produces model-checking plots for an HGLM or DHGLM
HGPREDICT	forms predictions from an HGLM or DHGLM analysis
HGRANDOMMODEL	defines the random model for an HGLM
HGRTEST	calculates likelihood tests for random terms in a hierarchical generalized linear model
HGSTATUS	displays the current HGLM model definitions
HGWALD	prints or saves Wald tests for fixed terms in an HGLM
IFUNCTION	estimates implicit and/or explicit functions of parameters
MAREGRESSION	does regressions for single-channel microarray data
MINIMIZE	finds the minimum of a function calculated by a procedure
MIN1DIMENSION	finds the minimum of a function in one dimension
MICHAELISMENTEN	fits the Michaelis-Menten equation for substrate concentration versus time data
MMPREDICT	predicts the Michaelis-Menten curve for a particular set of parameter values
NLAR1	fits curves with an AR1 or a power-distance correlation model
PAIRTEST	performs t-tests for pairwise differences
PPAIR	displays results of t-tests for pairwise differences in compact diagrams
PROBITANALYSIS	fits probit models allowing for natural mortality and immunity
R0INFLATED	fits zero-inflated regression models to count data with excess zeros
R0KEEP	saves information from models fitted by R0INFLATED
RAR1	fits regressions with an AR1 or a power-distance correlation model
RBRADLEYTERRY	fits the Bradley-Terry model for paired-comparison preference tests
RCATENELSON	performs a Cate-Nelson graphical analysis of bivariate data
RCIRCULAR	does circular regression of mean direction for an angular response

RFINLAYWILKINSON	performs Finlay and Wilkinson's joint regression analysis of genotype-by-environment data
RIDGE	produces ridge regression and principal component regression analyses
LRIDGE	does logistic ridge regression
RLASSO	performs lasso using iteratively reweighted least-squares
RQLINEAR	fits and plots quantile regressions for linear models
RQNONLINEAR	fits and plots quantile regressions for nonlinear models
RQSMOOTH	fits and plots quantile regressions for loess or spline models
RLFUNCTIONAL	fits a linear functional relationship model
RMGLM	fits a model where different units follow different generalized linear models
RNEGBINOMIAL	fits a negative binomial generalized linear model estimating the aggregation parameter
RNONNEGATIVE	fits a generalized linear model with nonnegativity constraints (synonym FITNONNEGATIVE)
RPAIR	gives t-tests for all pairwise differences of means from linear or generalized linear models
RPARALLEL	carries out analysis of parallelism for nonlinear functions (synonym FITPARALLEL)
RQUADRATIC	fits a quadratic surface and estimates its stationary point
RSCHNUTE	fits a general four-parameter growth model to a non-decreasing response variate (synonym FITSCHNUTE)
RSCREEN	performs screening tests for generalized or multivariate linear models
RSEARCH	searches through models for a regression or generalized linear model (with methods including all-subsets, forward and backward stepwise regression)
R2LINES	fits two-straight-line (broken-stick) models to data
SIMPLEX	searches for the minimum of a function using the Nelder-Mead algorithm
SVGLM	fits generalized linear models to survey data
WADLEY	fits models for Wadley's problem, allowing alternative links and errors
XOCATEGORIES	performs analyses of categorical data from crossover trials
YTRANSFORM	estimates the parameter lambda of a single parameter transformation

3.3 Analysis of variance

Genstat has a comprehensive set of commands to do an analysis of variance. These directives define the models to be fitted:

BLOCKSTRUCTURE	defines the blocking structure of the design, and hence the strata and error terms
COVARIATE	specifies covariates for analysis of covariance
TREATMENTSTRUCTURE	defines the treatment (or systematic) terms

For unstructured designs with a single error term, BLOCKSTRUCTURE need not be specified, and COVARIATE is needed only for analysis of covariance. Balanced designs can be analysed using the ANOVA directive.

ANOVA	performs analysis of variance

Directives and procedures are available to produce plots, checks and further output from an ANOVA analysis, or to save information in Genstat data structures:

ADISPLAY	displays further output from analyses produced by ANOVA
AGRAPH	plots tables of means from ANOVA
APLOT	plots residuals from an ANOVA analysis
AFIELDRESIDUALS	display residuals in field layout
ABLUPS	calculates BLUPs for block terms in an ANOVA analysis
ACHECK	checks assumptions for an ANOVA analysis
AMCOMPARISON	performs pairwise multiple comparison tests for ANOVA means
AKEEP	copies information from an ANOVA analysis into Genstat data structures
ARESULTSUMMARY	provides a summary of results from an ANOVA analysis
ASPREADSHEET	saves results from an analysis of variance in a spreadsheet

Unbalanced designs with a single error term can be be analysed using the AUNBALANCED procedure. (Unbalanced designs with several error terms should be analysed using the commands for REML analysis of linear mixed models.)

AUNBALANCED	performs analysis of variance for unbalanced designs
AUDISPLAY	produces further output for an unbalanced design (after AUNBALANCED)
AUGRAPH	plots tables of means from AUNBALANCED
AUPREDICT	forms predictions from an unbalanced design (after AUNBALANCED)
AUSPREADSHEET	Saves results from an analysis of an unbalanced design (by AUNBALANCED) in a spreadsheet
AUMCOMPARISON	performs pairwise multiple comparison tests for means from an unbalanced analysis of variance, performed previously by AUNBALANCED
AUKEEP	saves output from analysis of an unbalanced design (by AUNBALANCED)

There are also specialized procedures for designs (balanced or unbalanced) with a single error term and one or two treatment factors.

A2WAY	performs analysis of variance of a balanced or unbalanced design with up to two treatment factors
A2DISPLAY	provides further output following an analysis of variance by A2WAY
A2KEEP	copies information from an A2WAY analysis into Genstat data structures
A2RESULTSUMMARY	provides a summary of results from an analysis by A2WAY

If you are unsure what method to use, you can use the AOVANYHOW procedure to see which method is most appropriate.

AOVANYHOW	performs analysis of variance using ANOVA, AUNBALANCED, A2WAY or REML as appropriate
AOVDISPLAY	provides further output from an analysis by AOVANYHOW

Other procedures relevant to analysis of variance include:

ABOXCOX	estimates the power λ in a Box-Cox transformation, that maximizes the partial log-likelihood in ANOVA
AFCOVARIATES	defines covariates from a model formula for ANOVA
AFMEANS	forms tables of means classified by ANOVA treatment factors

ASTATUS	provides information about the settings of ANOVA models and variates
APERMTEST	does random permutation tests for analysis-of-variance tables
ABIVARIATE	produces graphs and statistics for bivariate analysis of variance
ACONFIDENCE	calculates simultaneous confidence intervals
AMDUNNETT	forms Dunnett's simultaneous confidence interval around a control
AMTIER	analyses a multitiered design by analysis of variance specified by up to 3 model formulae
AMTDISPLAY	displays further output for multitiered designs analysed by AMTIER
AMTKEEP	saves information from the analysis of a multitiered design by AMTIER
ACANONICAL	determines the orthogonal decomposition of the sample space for a design, using an analysis of the canonical relationships between the projectors derived from two or more model formulae
ACDISPLAY	provides further output from an analysis by ACANONICAL
ACKEEP	saves information from an analysis by ACANONICAL
VSPECTRALCHECK	forms the spectral components from the canonical components of a multitiered design, and constrains any negative spectral components to zero
AN1ADVICE	aims to give useful advice if a design that is thought to be balanced fails to be analysed by ANOVA
APAPADAKIS	analysis of variance with an added Papadakis covariate, formed from neighbouring residuals
APOLYNOMIAL	forms the equation for a polynomial contrast fitted by ANOVA
ADPOLYNOMIAL	plots single-factor polynomial contrasts fitted by ANOVA
AREPMEASURES	produces an analysis of variance for repeated measurements
ARETRIEVE	retrieves an ANOVA save structure from an external file
ASTORE	stores an ANOVA save structure in an external file
ASCREEN	performs screening tests for designs with orthogonal block structure
AYPARALLEL	does the same analysis of variance for several y-variates, and collates the output
A2RDA	saves results from an analysis of variance in R data frames
AU2RDA	saves results from an unbalanced analysis of variance, by AUNBALANCED, in R data frames
FALIASTERMS	forms information about aliased model terms in analysis of variance
FWITHINTERMS	forms factors to define terms representing the effects of one factor within another factor
MAANOVA	does analysis of variance for a single-channel microarray design (parallel anova)
SED2ESE	calculates effective standard errors that give good approximate standard errors of differences
SEDLSI	calculates least significant intervals
LSIPLOT	plots least significant intervals, saved from SEDLSI
RTCOMPARISONS	calculates comparison contrasts within a multi-way table of means
A2PLOT	plots effects and robust s.e. estimates from designs with two-level factors
CENSOR	pre-processes censored data before analysis by ANOVA
CINTERACTION	clusters rows and columns of a two-way interaction table
DIALLEL	analyses full and half diallel tables with parents
AMMI	allows exploratory analysis of genotype × environment interactions
FMEGAENVIRONMENTS	forms mega-environments based on winning genotypes from an

	AMMI-2 model
FRIEDMAN	performs Friedman's nonparametric analysis of variance
NLCONTRASTS	fits non-linear contrasts to quantitative factors in ANOVA
TEQUIVALENCE	performs equivalence, non-inferiority and non-superiority tests
VHOMOGENEITY	tests homogeneity of variances
WSTATISTIC	calculates the Shapiro-Wilk test for Normality

3.4 Design of experiments

Genstat has a comprehensive set of facilities for design of experiments. Collectively, these are known as the *Genstat Design System*. Many different design types are covered, each with a procedure that allows you to view and choose from the available possibilities. Other procedure allow designs and data forms to be displayed. There is also a general procedure DESIGN that can be used interactively to provide a single point of access to all the design types. DESIGN and the AG... procedures that it calls provide the Select Design facilities in Genstat *for Windows*, while the alternative Standard Design menu uses AGHIERARCHICAL, AGLATIN and AGSQLATTICE to generate completely randomized designs, randomized blocks, Latin and Graeco-Latin squares, split-plots, strip-plots (or criss-cross designs) and lattices.

DESIGN	provides a menu-driven interface for selecting and generating experimental designs
AGALPHA	forms alpha designs for up to 100 treatments
AGBIB	generates balanced-incomplete-block designs
AGBOXBEHNKEN	generates Box-Behnken designs
AGCENTRALCOMPOSITE	generates central composite designs
AGCROSSOVERLATIN	generates Latin squares balanced for carry-over effects
AGCYCLIC	generates cyclic designs from standard generators
AGDESIGN	generates generally balanced designs – factorial designs with blocking, fractional factorial designs, Lattice squares etc.
AGFACTORIAL	generates minimum aberration complete and fractional factorial designs
AGFRACTION	generates fractional factorial designs
AGHIERARCHICAL	generates orthogonal hierarchical designs
AGINDUSTRIAL	provides a menu-driven interface for selecting and generating designs for industrial experiments
AGLATIN	generates mutually orthogonal Latin squares
AGLOOP	generates loop designs e.g. for time-course microarray experiments
AGMAINEFFECT	generates designs to estimate main effects of two-level factors
AGNEIGHBOUR	generates neighbour-balanced designs
AGNONORTHOGONALDESIGN	generates non-orthogonal multi-stratum designs
AGSPACEFILLINGDESIGN	generates space filling designs
AGQLATIN	generates complete and quasi-complete Latin squares
AGREFERENCE	generates reference-level designs e.g. for microarray experiments
AGSEMILATIN	generates semi-Latin squares
AGSQLATTICE	generates square lattice and lattice square designs
PDESIGN	prints treatment combinations tabulated by the block factors
DDESIGN	plots the plan of a design
ADSPREADSHEET	puts the data and plan of an experimental design into Genstat spreadsheets

There are also procedures that you can use to determine the sample size (i.e. replication) required for experiments that are to be analysed by analysis of variance, t-test or various non-parametric tests. You can also calculate the power (or probability of detection) for terms in analysis of variance or regression analyses.

APOWER	calculates the power (probability of detection) for terms in an analysis of variance
RPOWER	calculates the power (probability of detection) for regression models
VPOWER	uses a parametric bootstrap to estimate the power (probability of detection) for terms in a REML analysis
ASAMPLESIZE	finds the replication (sample size) to detect a treatment effect or contrast
VSAMPLESIZE	estimates the replication to detect a fixed term or contrast in a REML analysis, using parametric bootstrap
ADETECTION	calculates the minimum size of effect or contrast detectable in an analysis of variance
SBNTEST	calculates the sample size for binomial tests
SCORRELATION	calculates the sample size to detect specified correlations
SLCONCORDANCE	calculates the sample size for Lin's concordance coefficient
SMANNWHITNEY	calculates the sample size for the Mann-Whitney test
SMCNEMAR	calculates the sample size for McNemar's test
SPNTEST	calculates the sample size for a Poisson test
SPRECISION	calculates the sample size to obtain a specified precision
SSIGNTEST	calculates the sample size for a sign test
STTEST	calculates the sample size for t-tests, including equivalence tests and tests for non-inferiority
DSTTEST	plots power and significance for t-tests, including equivalence tests and tests for non-inferiority

The Design System is based on a range of standard generators. Some of these, such as the Galois fields used to generate Latin squares, can be formed when required – and so there is no limitation on the available designs. Repertoires of others, such as design keys, are stored in backing-store files which are scanned by the design generation procedures to form menus listing the available possibilities. Algorithms are available to form generators for new designs, and these can then be added to the design files to become an integral part of the system. Other design utilities include procedures for combining simple designs into more complicated arrangements, for forming augmented designs, and for determining how many replicates are needed. There are also directives for constructing response-surface designs and doubly resolvable row-column designs. The relevant commands include the directives

AFMINABERRATION	forms minimum aberration factorial or fractional-factorial designs
AFRESPONSESURFACE	uses the BLKL algorithm to construct designs for estimating response surfaces
AGRCRESOLVABLE	forms doubly resolvable row-column designs
GENERATE	generates values of factors in systematic order or as defined by a design key, or forms values of pseudo-factors
RANDOMIZE	puts units of vectors into random order, or randomizes units of an experimental design
FKEY	forms design keys for multi-stratum experimental designs, allowing for confounding and aliasing of treatments
FPSEUDOFACTORS	determines patterns of confounding and aliasing from design keys, and extends the treatment formula to incorporate the necessary pseudo-factors
SET2FORMULA	forms a model formula using structures supplied in a pointer

and the procedures

AEFFICIENCY	calculates efficiency factors for experimental designs
AFAUGMENTED	forms an augmented design
AFLABELS	forms a variate of unit labels for a design

AFRCRESOLVABLE	forms doubly resolvable row-column designs, with output
AFUNITS	forms a factor to index the units of the final stratum of a design
AKEY	generates values for treatment factors using the design key method
AMERGE	merges extra units into an experimental design
AFNONLINEAR	forms D-optimal designs to estimate the parameters of a nonlinear or generalized linear model
AFPREP	searches for an efficient partially-replicated design
APRODUCT	forms a new experimental design from the product of two designs
ARCSPLITPLOT	adds extra treatments onto the replicates of a resolvable row-column design, and generates factors giving the row and column locations of the plots within the design
ARANDOMIZE	randomizes and prints an experimental design
CDNAUGMENTEDDESIGN	constructs an augmented block design, using CycDesigN if the controls are in an incomplete-block design
CDNBLOCKDESIGN	constructs a block design using CycDesigN
CDNPREP	constructs a multi-location partially-replicated design using CycDesigN
CDNROWCOLUMNDESIGN	constructs a row-column design using CycDesigN
COVDESIGN	produces experimental designs efficient under analysis of covariance
FACCOMBINATIONS	forms a factor to indicate observations with identical combinations of values of a set of variates, texts or factors
FACDIVIDE	represents a factor by factorial combinations of a set of factors
FACPRODUCT	forms a factor with a level for every combination of other factors
FBASICCONTRASTS	forms the basic contrasts of a model term
FCOMPLEMENT	forms the complement of an incomplete block design
FDESIGNFILE	forms a backing-store file of information for AGDESIGN
FHADAMARDMATRIX	forms Hadamard matrices
FOCCURRENCES	forms a "concurrence" matrix recording how often each pair of treatments occurs in the same block of a design
FPLOTNUMBER	forms plot numbers for a row-by-column design
FPROJECTIONMATRIX	forms a projection matrix for a set of model terms
XOEFFICIENCY	calculates the efficiency for estimating effects in cross-over designs
XOPOWER	estimates the power of contrasts in cross-over designs

3.5 REML analysis of linear mixed models

The REML algorithm allows you to analyse linear mixed models i.e. linear models that can contain both fixed and random effects. In some applications these are known as "multi-level" models. It can thus be used to analyse unbalanced designs with several error terms (which cannot be analysed by ANOVA). It can also fit random correlation models to describe the covariances between random effects as can arise, for example, in the analysis of repeated measurements or spatial data.

REML	fits a variance-component model by residual (or restricted) maximum likelihood
VCOMPONENTS	defines the model for REML
VCYCLE	controls advanced aspects of the REML algorithm
VDISPLAY	displays further output from a REML analysis
VKEEP	copies information from a REML analysis into Genstat data structures
VSTRUCTURE	defines a variance structure for random effects in a REML model
VPEDIGREE	generates an inverse relationship matrix for use when fitting animal or plant breeding models by REML

VPREDICT	forms predictions from a REML model
VRESIDUAL	defines the residual term for a REML model
VSTATUS	prints the current model settings for REML

There are several procedures that may be useful, for example, to define the model, to produce additional output or for other REML-based analyses.

FCONTRASTS	modifies a model formula to contain contrasts of factors
FDIALLEL	forms the components of a diallel model for REML or regression
F2DRESIDUALVARIOGRAM	calculates and plots a 2-dimensional variogram from a 2-dimensional array of residuals
TOBIT	linear mixed model analysis of data with fixed-threshold censoring
VAIC	calculates the Akaike and Schwarz (Bayesian) information coefficients for REML
VALLSUBSETS	fits all subsets of the fixed terms in a REML analysis
VAYPARALLEL	does the same REML analysis for several y-variates, and collates the output
VBOOTSTRAP	performs a parametric bootstrap of the fixed effects in a REML analysis
VCRITICAL	uses a parametric bootstrap to estimate critical values for a fixed term in a REML analysis
VCHECK	checks standardized residuals from a REML analysis
VDEFFECTS	plots one- or two-way tables of effects estimated in a REML analysis
VDFIELDRESIDUALS	display residuals from a REML analysis in field layout
VFIXEDTESTS	saves fixed tests from a REML analysis
VFLC	performs an F-test of random effects in a linear mixed model based on linear combinations of the responses, i.e. an FLC test
VFPEDIGREE	checks and prepares pedigree information from several factors, for use by VPEDIGREE and REML
VFRESIDUALS	obtains residuals, fitted values and their standard errors from a REML analysis
VFUNCTION	calculates functions of variance components from a REML analysis
VGRAPH	plots tables of means from REML
VHERITABILITY	calculates generalized heritability for a random term in a REML analysis
VLSD	prints approximate least significant differences for REML means
VMCOMPARISON	performs pairwise comparisons between REML means
VMETA	performs a multi-treatment meta analysis using summary results from individual experiments
VPERMTEST	does random permutation tests for the fixed effects in a REML analysis
VPLOT	plots residuals from a REML analysis
VPOWER	uses a parametric bootstrap to estimate the power (probability of detection) for terms in a REML analysis
VRACCUMULATE	forms a summary accumulating the results of a sequence of REML random models
VRCHECK	checks effects of a random term in a REML analysis
VRMETAMODEL	forms the random model for a REML meta analysis
VRPERMTEST	performs permutation tests for random terms in REML analysis
VRFIT	fits terms from a REML fixed model in a Genstat regression
VRADD	adds terms from a REML fixed model into a Genstat regression
VRDISPLAY	displays output for a REML fixed model fitted in a Genstat regression

VRDROP	drops terms in a REML fixed model from a Genstat regression
VRKEEP	saves output for a REML fixed model fitted in a Genstat regression
VRSETUP	sets up Genstat regression to assess terms from a REML fixed model
VRSWITCH	adds or drops terms from a REML fixed model in a Genstat regression
VRTRY	tries the effect of adding and dropping individual terms from a REML fixed model in a Genstat regression
VSAMPLESIZE	estimates the replication to detect a fixed term or contrast in a REML analysis, using parametric bootstrap
VSCREEN	performs screening tests for fixed terms in a REML analysis
VSOM	analyses a simple REML variance components model for outliers using a variance shift outlier model
VSPREADSHEET	saves results from a REML analysis in a spreadsheet
VSURFACE	fits a 2-dimensional spline surface using REML, and estimates its extreme point
VTCOMPARISONS	calculates comparison contrasts within a multi-way table of predicted means from a REML analysis
VUVCOVARIANCE	forms the unit-by-unit variance-covariance matrix for specified variance components in a REML model

There is also a suite of procedures to provide automatic selection of REML random models for single trials, series of trials and meta analysis.

VABLOCKDESIGN	analyses an incomplete-block design by REML, allowing automatic selection of random and spatial covariance models
VAROWCOLUMNDESIGN	analyses a row-and-column design by REML, with automatic selection of the best random and spatial covariance model
VALINEBYTESTER	provides combinabilities and deviances for a line-by-tester trial analysed by VABLOCKDESIGN or VAROWCOLUMNDESIGN
VLINEBYTESTER	analyses a line-by-tester trial by REML
VASERIES	analyses a series of trials with incomplete-block or row-and-column designs by REML, automatically selecting the best random models
VASDISPLAY	displays further output from an analysis by VASERIES
VASKEEP	copies information from an analysis by VASERIES into Genstat data structures
VASMEANS	saves experiment × treatment means from analysis of a series of trials by VASERIES
VAMETA	performs a REML meta analysis of a series of trials
VFMODEL	forms a model-definition structure for a REML analysis
VFSTRUCTURE	adds a covariance-structure definition to a REML model-definition structure
VMODEL	specifies the model for a REML analysis using a model-definition structure defined by VFMODEL
VAOPTIONS	defines options for the fitting of models by VARANDOM and associated procedures
VARANDOM	finds the best REML random model from a set of models defined by VFMODEL
VARECOVER	recovers when REML, is unable to fit a model, by simplifying the random model

3.6 Multivariate and cluster analysis

Several standard multivariate methods are provided by Genstat directives. These include methods that analyse data in the form of units-by-variates, and methods that use a similarity or distance matrix.

The following directives carry out standard multivariate analyses:

CVA	canonical variates analysis
FCA	factor analysis
MDS	non-metric multidimensional scaling
PCP	principal components analysis
PCO	principal coordinates analysis
ROTATE	Procrustes rotation

Other directives and procedures are available to process results from multivariate analyses:

ADDPOINTS	adds points for new objects to a PCO
CVAPLOT	plots the mean and unit scores from a canonical variates analysis
CVASCORES	calculates scores for individual units in canonical variates analysis
DBIPLOT	plots a biplot from an analysis by PCP, CVA or PCO
DMST	gives a high resolution plot of an ordination with minimum spanning tree
FACROTATE	rotates factor loadings from a PCP, CVA or FCA
LRVSCREE	prints a scree diagram and/or a difference table of latent roots
PCORELATE	relates principal coordinates to original data variables

The following directives are used for hierarchical or non-hierarchical cluster analysis:

CLUSTER	non-hierarchical clustering from a data matrix
FSIMILARITY	forms a similarity matrix or a between-group similarity matrix from a units-by-variates data matrix
HREDUCE	forms a reduced similarity matrix (by groups)
HCLUSTER	hierarchical cluster analysis from a similarity matrix

Other directives and procedures that process the results from hierarchical cluster analyses are:

DDENDROGRAM	draws dendrograms with control over structure and style
DCLUSTERLABELS	labels clusters in a single-page dendrogram plotted by DDENDROGRAM
HBOOTSTRAP	performs bootstrap analyses to assess the reliability of clusters from hierarchical cluster analysis
HCOMPAREGROUPINGS	compares groupings generated, for example, from cluster analyses
HDISPLAY	displays results associated with hierarchical clustering
HFAMALGAMATIONS	forms an amalgamations matrix from a minimum spanning tree
HFCLUSTERS	forms a set of clusters from an amalgamations matrix
HLIST	lists a data matrix in abbreviated form
HPCLUSTERS	prints a set of clusters
HSUMMARIZE	summarizes data variates by clusters

Other multivariate techniques are provided by procedures in the Library:

AMMI	allows exploratory analysis of genotype × environment interactions
BCLASSIFICATION	constructs a classification tree
BCDISPLAY	displays a classification tree

BCIDENTIFY	identifies specimens using a classification tree
BCKEEP	saves information from a classification tree
BCVALUES	forms values for nodes of a classification tree
BCFOREST	constructs a random classification forest
BCFDISPLAY	displays information about a random classification forest
BCFIDENTIFY	identifies specimens using a random classification forest
BIPLOT	produces a biplot from a set of variates
BKEY	constructs an identification key
BKDISPLAY	displays an identification key
BKIDENTIFY	identifies specimens using a key
BKKEEP	saves information from an identification key
CANCORRELATION	does canonical correlation analysis
CCA	performs canonical correspondence analysis
CRBIPLOT	plots correlation or distance biplots after CCA or RDA
CRTRIPLOT	plots ordination biplots or triplots after CCA or RDA
CINTERACTION	clusters rows and columns of a two-way interaction table
CLASSIFY	obtains a starting classification for non-hierarchical clustering
CONVEXHULL	finds the points of a single or a full peel of convex-hulls
CORANALYSIS	does correspondence analysis, or reciprocal averaging
MCORANALYSIS	does multiple correspondence analysis
CABIPLOT	plots results from correspondence analysis or multiple correspondence analysis
DISCRIMINATE	performs discriminant analysis
SDISCRIMINATE	selects the best set of variates to discriminate between groups
QDISCRIMINATE	performs quadratic discrimination between groups i.e. allowing for different variance-covariance matrices
DPARALLEL	displays multivariate data using parallel coordinates
GESTABILITY	calculates stability coefficients for genotype-by-environment data
GGEBIPLOT	plots displays to assess genotype + genotype-by-environment variation
GENPROCRUSTES	performs a generalized Procrustes analysis
IDENTIFY	identifies an unknown specimen from a defined set of objects
KNEARESTNEIGHBOURS	classifies items or predicts their responses by examining their *k* nearest neighbours
MANOVA	performs multivariate analysis of variance and covariance
MANTEL	assesses the association between similarity matrices
MULTMISSING	estimates missing values for units in a multivariate data set
MVAOD	does an analysis of distance of multivariate data
NORMTEST	performs tests of univariate and/or multivariate normality
OPLS	performs orthogonal partial least squares regression
PCOPROCRUSTES	performs a multiple Procrustes analysis
PLS	fits a partial least squares regression model
RDA	performs redundancy analysis
RIDGE	produces ridge regression and principal component regression analyses
LRIDGE	does logistic ridge regression
RLFUNCTIONAL	fits a linear functional relationship model
RMULTIVARIATE	performs multivariate linear regression with accumulated testing of terms
ROBSSPM	forms robust estimates of sum-of-squares-and-products matrices
SAGRAPES	produces statistics and graphs for checking sensory panel performance
SKEWSYMMETRY	provides an analysis of skew-symmetry for an asymmetric matrix

3.7 Time series

Genstat provides several methods for examining and analysing time series. Sample correlation functions are produced by the directive CORRELATE:

CORRELATE	forms correlations between variates, autocorrelations of variates, and lagged cross-correlations between variates

The analysis of Box-Jenkins models is specified by several directives:

FTSM	forms preliminary estimates of parameters in time-series models
TRANSFERFUNCTION	specifies input series and transfer-function models for subsequent estimation of a model for an output series
TFIT	estimates parameters in Box-Jenkins models for time series (renamed version of ESTIMATE, which is retained as a synonym)

Information can be saved in Genstat data structures, or further output can be produced:

TDISPLAY	displays further output after an analysis by TFIT
TKEEP	saves results after an analysis by TFIT
TFORECAST	forecasts future values of a time series (renamed version of FORECAST, which is retained as a synonym)
TSUMMARIZE	displays characteristics of a time series model

It is also possible to filter a time series, or perform spectral analysis via the Fourier transform of a time series using the directives:

TFILTER	filters time series by time-series models (renamed version of FILTER, which is retained as a synonym)
FOURIER	calculates cosine or Fourier transforms of a real or complex series

Relevant procedures in the Library include:

BJESTIMATE	fits an ARIMA model, with forecasts and residual checks
BJFORECAST	plots forecasts of a time series using a previously fitted ARIMA
BJIDENTIFY	displays time series statistics useful for ARIMA model selection
DFOURIER	performs a harmonic analysis of a univariate time series
KALMAN	calculates estimates from the Kalman filter
DKALMAN	plots results from an analysis by KALMAN
MCROSSPECTRUM	performs a spectral analysis of a multiple time series
MC1PSTATIONARY	gives the stationary probabilities for a 1st-order Markov chain
MOVINGAVERAGE	calculates and plots the moving average of a time series
PERIODTEST	gives periodogram-based tests for white noise in time series
PREWHITEN	filters a time series before spectral analysis
REPPERIODOGRAM	gives periodogram-based analyses for replicated time series
SMOOTHSPECTRUM	forms smoothed spectrum estimates for univariate time series
TVARMA	fits a vector autoregressive moving average (VARMA) model
TVFORECAST	forecasts future values from a vector autoregressive moving average (VARMA) model
TVGRAPH	plots a vector autoregressive moving average (VARMA) model

3.8 Repeated measurements

A repeated-measurements study is one in which subjects (animals, people, plots, etc) are observed on several occasions. Each subject usually receives some randomly allocated treatment, either at the outset or

repeatedly through the investigation, and is then observed at successive occasions to see how the treatment effects develop. One way to analyse data sets like this is to use Genstat's REML facilities to model the correlation structure over time.

REML	fits a variance-component model by residual (or restricted) maximum likelihood
VCOMPONENTS	defines the model for REML
VSTRUCTURE	defines a variance structure for random effects in a REML model

Alternatively, Genstat has procedures for customized plotting of the observations (or profiles) against time, repeated measures analysis of variance, analyses based on ante-dependence structure or generalized estimating equations, and regression or nonlinear modelling of data where the residuals follow an AR1 or power-distance correlation model.

ANTORDER	assesses order of ante-dependence for repeated measures data
ANTTEST	calculates overall tests based on a specified order of ante-dependence
AREPMEASURES	produces an analysis of variance for repeated measurements
CUMDISTRIBUTION	fits frequency distributions to accumulated counts
DREPMEASURES	plots profiles and differences of profiles for repeated measurements
GEE	fits models to longitudinal data by generalized estimating equations
NLAR1	fits curves with an AR1 or a power-distance correlation model
RAR1	fits regressions with an AR1 or a power-distance correlation model
VORTHPOLYNOMIAL	calculates orthogonal polynomial time-contrasts for repeated measurements

3.9 Survival analysis

In survival data the response variate is the survival time of an individual like a medical patient or an industrial component. The responses are often *censored*, i.e. some individuals survive beyond the end of the study, and so their survival times are unknown. Genstat provides various ways of estimating the *survivor function* (i.e. the probability that an individual is still surviving at each time). You can do nonparametric tests to compare different survival distributions. Finally, you can model the survival times, by assuming that they follow exponential, Weibull or extremevalue distributions, or by fitting a proportional hazards model.

KAPLANMEIER	calculates the Kaplan-Meier estimate of the survivor function
RLIFETABLE	calculates the life-table estimate of the survivor function
RPHFIT	fits the proportional hazards model to survival data as a generalized linear model
RPHCHANGE	modifies a proportional hazards model fitted by RPHFIT
RPHDISPLAY	prints output for a proportional hazards model fitted by RPHFIT
RPHKEEP	saves information from a proportional hazards model fitted by RPHFIT
RPROPORTIONAL	fits a proportional hazards model by a direct maximization of the likelihood (this will be more efficient than RPHFIT for large data sets)
RSTEST	compares groups of right-censored survival data by nonparametric tests
RSURVIVAL	models survival times of exponential, Weibull or extreme-value distributions

3.10 Bayesian methods

Genstat provides convenient ways to define and run Bayesian analyses using WinBUGS or OpenBUGS. It also supports Bayesian computing using the Differential Evolution Markov Chain algorithm.

BGIMPORT	imports MCMC output in CODA format produced by WinBUGS or OpenBUGS.
BGPLOT	produces plots for output and diagnostics from MCMC simulations.
BGXGENSTAT	runs WinBUGS or OpenBUGS from Genstat in batch mode using scripts.
DEMC	performs Bayesian computing using the Differential Evolution Markov Chain algorithm

3.11 Spatial statistics

Commands are available form forming variograms and for producing kriged estimates.

FVARIOGRAM	forms experimental variograms
MVARIOGRAM	fits models to an experimental variogram
DVARIOGRAM	plots fitted models to an experimental variogram
KRIGE	calculates kriged estimates using a model fitted to a sample variogram
FCOVARIOGRAM	forms a covariogram structure containing auto-variograms of individual variates and cross-variograms for pairs from a list of variates
MCOVARIOGRAM	fits models to sets of variograms and cross-variograms
DCOVARIOGRAM	plots 2-dimensional auto- and cross-variograms
COKRIGE	calculates kriged estimates using a model fitted to the sample variograms and cross-variograms of a set of variates
KCROSSVALIDATION	computes cross validation statistics for punctual kriging
DHSCATTERGRAM	plots an h-scattergram

Relevant procedures in the Library include:

DKSTPLOT	produces diagnostic plots for space-time clustering
DPOLYGON	draws polygons using high-resolution graphics
DPTMAP	draws maps for spatial point patterns using high-resolution graphics
DPTREAD	adds points interactively to a spatial point pattern
DRPOLYGON	reads a polygon interactively from the current graphics device
DPSPECTRALPLOT	calculates an estimate of the spectrum of a spatial point pattern
FHAT	calculates an estimate of the F nearest-neighbour distribution function
FZERO	gives the F function expectation under complete spatial randomness
GHAT	calculates an estimate of the G nearest-neighbour distribution function
GRLABEL	randomly labels two or more spatial point patterns
GRTHIN	randomly thins a spatial point pattern
GRTORSHIFT	performs a random toroidal shift on a spatial point pattern
GRCSR	generates completely spatially random points in a polygon
KCSRENVELOPES	simulates K function bounds under complete spatial randomness
KHAT	calculates an estimate of the K function
KLABENVELOPES	gives bounds for K function differences under random labelling
KSED	calculates s.e. for K function differences under random labelling

KSTHAT	calculates an estimate of the K function in space, time and space-time
KSTMCTEST	performs a Monte-Carlo test for space-time interaction
KSTSE	calculates the standard error for the space-time K function
KTORENVELOPES	gives bounds for the bivariate K function under independence
K12HAT	calculates an estimate of the bivariate K function
MSEKERNEL2D	estimates the mean square error for a kernel smoothing
PTAREAPOLYGON	calculates the area of a polygon
PTBOX	generates a box bounding or surrounding a spatial point pattern
PTCLOSEPOLYGON	closes open polygons
PTDESCRIBE	gives summary and second order statistics for a point process
PTGRID	generates a grid of points in a polygon
PTINTENSITY	calculates the overall density for a spatial point pattern
PTKERNEL2D	performs kernel smoothing of a spatial point pattern
PTK3D	performs kernel smoothing of space-time data
PTREMOVE	removes points interactively from a spatial point pattern
PTROTATE	rotates a point pattern
PTSINPOLYGON	returns points inside or outside a polygon

3.12 Six sigma

Genstat has wide range of facilities to support the six-sigma approach to quality improvement. It can display many different types of control chart.

SPCCHART	plots c or u charts representing numbers of defective items
SPCUSUM	prints CUSUM tables for controlling a process mean
SPEWMA	plots exponentially weighted moving-average control charts
SPPCHART	plots p or np charts for binomial testing for defective items
SPSHEWHART	plots control charts for mean and standard deviation or range

It can test for Normality, display Pareto charts and calculate capability statistics.

NORMTEST	performs tests of univariate and/or multivariate normality
SPCAPABILITY	calculates capability statistics
TABSORT	sorts tables to put margins are in ascending or descending order for display as a Pareto chart

It also provides full statistical backup for wider-ranging investigations. The list below highlights some of the commands that may be useful.

AFRESPONSESURFACE	uses the BLKL algorithm to construct response-surface designs
AGBOXBEHNKEN	generates Box-Behnken designs
AGCENTRALCOMPOSITE	generates central composite designs
AGDESIGN	selects from a set of standard designs including factorials with interactions confounded with blocks
AGFRACTION	generates fractional factorial designs
AGMAINEFFECT	generates designs to estimate main effects of two-level factors (Plackett-Burman designs)
A2WAY	performs analysis of variance of a balanced or unbalanced design with up to two treatment factors
ANOVA	analyses y-variates by analysis of variance according to the model defined by earlier BLOCKSTRUCTURE, COVARIATE, and TREATMENTSTRUCTURE statements
AGRAPH	plots one- or two-way tables of means from ANOVA
APLOT	plots residuals from an ANOVA analysis

AMCOMPARISON	performs pairwise multiple comparison tests for ANOVA means
AUNBALANCED	performs analysis of variance for unbalanced designs
AUGRAPH	plots tables of means from AUNBALANCED
FIT	fits a linear, generalized linear, generalized additive, or generalized nonlinear model
FITCURVE	fits a standard nonlinear regression model
FITNONLINEAR	fits a nonlinear regression model or optimizes a function
FKEY	forms design keys for balanced designs with several error terms, allowing for confounded and aliased treatments
REML	fits an unbalanced linear mixed model and estimates variance components
RQUADRATIC	fits a quadratic surface and estimates its stationary point
YTRANSFORM	estimates the parameter lambda from various single-parameter transformations, includling power (Box-Cox), modulus, folded power, Guerrero-Johnson, Aranda-Ordaz and power logit

3.13 Survey analysis

Genstat has several commands for the analysis of simple of complex surveys, including facilities for modelling, imputation, calculations and manipulation. (For further details, see the *Guide to Survey Analysis in Genstat*.)

TABULATE	forms tables of summaries of the values of variates classified by one or more factors
MTABULATE	forms tables of summaries of variates classified by multiple-response factors
SVBOOT	bootstraps data from random surveys
SVCALIBRATE	performs generalized calibration of survey data
SVGLM	fits generalized linear models to survey data
SVHOTDECK	performs hot-deck and model-based imputation for survey data
SVMERGE	merges strata prior to survey analysis
SVREWEIGHT	modifies survey weights adjusting to ensure that their overall sum weights remains unchanged
SVSTRATIFIED	analyses stratified random surveys by expansion or ratio raising
SVTABULATE	tabulates data from random surveys, including multistage surveys and surveys with unequal probabilities of selection
SVWEIGHT	forms survey weights
COMBINE	combines or omits "slices" of tables, matrices or variates
CSPRO	reads a data set from a CSPro survey data file and dictionary, and loads it into Genstat or puts it into a spreadsheet file
DTABLE	plots tables
MARGIN	calculates or deletes margins of tables
PERCENT	expresses the body of a table as percentages of one of its margins
TABMODE	forms summary tables of modes of values
TABSORT	sorts tables so their margins are in ascending or descending order
T%CONTROL	expresses tables as percentages of control cells
VSUMMARY	Summarizes a variate, with classifying factors, into a data matrix of variates and factors

3.14 Data mining

Genstat has many conventional statistical techniques such as generalized linear models (e.g. log-linear models and logistic regression) and multivariate analysis (e.g. canonical variates analysis and cluster analysis) that are very useful for data mining. It also provides various more specialized techniques such as association rules, classification and regression trees, random forests, *k*-nearest-neighbours classification,

self-organizing maps, neural networks and radial basis functions.

ASRULES	derives association rules from transaction data
BCLASSIFICATION	constructs a classification tree
BCDISPLAY	displays a classification tree
BCIDENTIFY	identifies specimens using a classification tree
BCKEEP	saves information from a classification tree
BCVALUES	forms values for nodes of a classification tree
BCFOREST	constructs a random classification forest
BCFDISPLAY	displays information about a random classification forest
BCFIDENTIFY	identifies specimens using a random classification forest
BREGRESSION	constructs a regression tree
BRDISPLAY	displays a regression tree
BRKEEP	saves information from a regression tree
BRPREDICT	makes predictions using a regression tree
BRVALUES	forms values for nodes of a regression tree
BRFOREST	constructs a random regression forest
BRFDISPLAY	displays information about a random regression forest
BRFPREDICT	makes predictions using a random regression forest
KNEARESTNEIGHBOURS	classifies items or predicts their responses by examining their *k* nearest neighbours
NNFIT	fits a multi-layer perceptron neural network
NNDISPLAY	displays output from a multi-layer perceptron neural network fitted by NNFIT
NNPREDICT	forms predictions from a multi-layer perceptron neural network fitted by NNFIT
RBFIT	fits a radial basis function model
RBDISPLAY	displays output from a radial basis function model fitted by RBFIT
RBPREDICT	forms predictions from a radial basis function model fitted by RBFIT
SOM	declares a self-organizing map
SOMADJUST	performs adjustments to the weights of a self-organizing map
SOMDESCRIBE	summarizes values of variables at nodes of a self-organizing map
SOMESTIMATE	estimates the weights for self-organizing maps
SOMIDENTIFY	allocates samples to nodes of a self-organizing map
SOMPREDICT	makes predictions using a self-organizing map
SVMFIT	fits a support vector machine
SVMPREDICT	forms the predictions using a support vector machine

3.15 Statistical genetics and QTL estimation

Genstat has a suite of procedures for statistical genetics. Several of these make use of Genstat's REML facilities to estimate QTLs from single environment, multi-environment and multi-trait trials.

DQMAP	displays a genetic map
DQMKSCORES	plots a grid of marker scores for genotypes and indicates missing data
DQMQTLSCAN	plots the results of a genome-wide scan for QTL effects in multi-environment trials
DQRECOMBINATIONS	plots a matrix of recombination frequencies between markers
DQSQTLSCAN	plots the results of a genome-wide scan for QTL effects in single-environment trials
GPREDICTION	produces genomic predictions (breeding values) using phenotypic and molecular marker information
QCANDIDATES	selects QTLs on the basis of a test statistic profile along the

	genome
QDESCRIBE	prints summary statistics of genotypes
QEIGENANALYSIS	uses principal components analysis and the Tracy-Widom statistic to find the number of significant principal components to represent a set of variables
QEXPORT	exports genotypic data for QTL analysis
QFLAPJACK	creates a Flapjack project file from genotypic and phenotypic data
QGSELECT	obtains a representative selection of genotypes by means of genetic distance sampling or genetic distance optimization
QIBDPROBABILITIES	reads molecular marker data and calculates IBD probabilities
QIMPORT	imports genotypic and phenotypic data for QTL analysis
QKINSHIPMATRIX	forms a kinship matrix from molecular markers
QLDDECAY	estimates linkage disequilibrium (LD) decay along a chromosome
QLINKAGEGROUPS	forms linkage groups using marker data from experimental populations
QMAP	constructs genetic linkage maps using marker data from experimental populations
QMASSOCIATION	performs multi-environment marker-trait association analysis in a genetically diverse population using bi-allelic and multi-allelic markers
QMATCH	matches different data structures to be used in QTL estimation
QMBACKSELECT	performs a QTL backward selection for loci in multi-environment trials or multiple populations
QMESTIMATE	calculates QTL effects in multi-environment trials or multiple populations
QMKDIAGNOSTICS	generates descriptive statistics and diagnostic plots of molecular marker data
QMKRECODE	recodes marker scores into separate alleles
QMKSELECT	obtains a representative selection of markers by means of genetic distance sampling or genetic distance optimization
QMQTLSCAN	performs a genome-wide scan for QTL effects (Simple and Composite Mapping) in multi-environment trials or multiple populations
QMTBACKSELECT	performs a QTL backward selection for loci in multi-trait trials
QMTESTIMATE	calculates QTL effects in multi-trait trials
QMTQTLSCAN	performs a genome-wide scan for QTL effects (Simple and Composite Interval Mapping) in multi-trait trials
QMVAF	calculates percentage variance accounted for by QTL effects in a multi-environment analysis
QMVESTIMATE	replaces missing molecular marker scores using conditional genotypic probabilities
QMVREPLACE	replaces missing marker scores with the mode scores of the most similar genotypes
QRECOMBINATIONS	calculates the expected numbers of recombinations and the recombination frequencies between markers
QREPORT	creates an HTML report from QTL linkage or association analysis results
QSASSOCIATION	performs marker-trait association analysis in a genetically diverse population using bi-allelic and multi-allelic markers
QSBACKSELECT	performs a backward selection for loci in single-environment trials
QSESTIMATE	calculates QTL effects in single-environment trials
QSIMULATE	simulates marker data and QTL effects for single and multiple environment trials

QSQTLSCAN	performs a genome-wide scan for QTL effects (Simple and Composite Mapping) in single-environment trials
QTHRESHOLD	calculates a threshold to identify a significant QTL
VGESELECT	selects the best variance-covariance model for a set of environments

3.16 Microarray data

There is a suite of procedures for the design, analysis and visualization of two-colour and Affymetrix microarray data. These are used by the Microarray menus in Genstat *for Windows*.

AGBIB	generates balanced incomplete block designs
AGLOOP	generates loop designs e.g. for time-course microarray experiments
AGREFERENCE	generates reference-level designs e.g. for microarray experiments
BAFFYMETRIX	Estimates expression values from Affymetrix CED and CDF files
MADESIGN	assesses the efficiency of a two-colour microarray design
MACALCULATE	corrects and transforms two-colour microarray differential expressions
MNORMALIZE	normalizes two-colour microarray data
MAESTIMATE	estimates treatment effects from a two-colour microarray design
AFFYMETRIX	estimates expression values for Affymetrix slides.
MABGCORRECT	performs background correction of Affymetrix slides
MAROBUSTMEANS	does a robust means analysis for Affymetrix slides
MARMA	calculates Affymetrix expression values
MAVDIFFERENCE	applies the average difference algorithm to Affymetrix data
DMADENSITY	plots the empirical CDF or PDF (kernel smoothed) by groups
MAHISTOGRAM	plots histograms of microarray data
MAPLOT	produces two-dimensional plots of microarray data
MAANOVA	does analysis of variance of single-channel microarray data
MAREGRESSION	does regressions for single-channel microarray data
MASHADE	produces shade plots to display spatial variation of microarray data
MAVOLCANO	produces volcano plots of microarray data
MAPCLUSTER	clusters probes or genes with microarray data
MASCLUSTER	clusters microarray slides
MA2CLUSTER	performs a two-way clustering of microarray data by probes (or genes) and slides
FDRBONFERRONI	estimates false discovery rates by a Bonferroni-type procedure
FDRMIXTURE	estimates false discovery rates using mixture distributions
MAEBAYES	modifies t-values by an empirical Bayes method.
MPOLISH	performs a median polish of two-way data
QNORMALIZE	performs quantile normalization
THINPLATE	calculates the basis functions for thin-plate splines
TUKEYBIWEIGHT	estimates means using the Tukey biweight algorithm

3.17 Ecological data

The procedures listed below allow you to display, summarize and model ecological data.

ECABUNDANCEPLOT	produces rank/abundance, *ABC* and *k*-dominance plots
ECACCUMULATION	plots species accumulation curves for samples or individuals
ECANOSIM	performs an analysis of similarities (ANOSIM)
ECDIVERSITY	calculates measures of diversity with jackknife or bootstrap

	estimates
ECFIT	fits models to species abundance data
ECNICHE	generates relative abundance of species for niche-based models
ECNPESTIMATE	calculates nonparametric estimates of species richness
ECRAREFACTION	calculates individual or sample-based rarefaction
LORENZ	plots the Lorenz curve and calculates the Gini and asymmetry coefficients

4 Syntax summary

4.1 Commands

ABIVARIATE procedure

Produces graphs and statistics for bivariate analysis of variance (R.F.A. Poultney).

Options

PRINT = *string tokens*	Controls printing of statistics from the bivariate analysis (error, treatment); default erro, trea
APRINT = *string tokens*	Controls output from the (univariate) ANOVAs of Y1 and Y2 (usual ANOVA print options); default aovt
TREATMENTSTRUCTURE = *formula*	Treatment terms to be fitted in the analysis of variance; this option must be set
BLOCKSTRUCTURE = *formula*	Block model defining the error terms in the analysis of variance; if unset, the design is assumed to be unstratified (i.e. to have a single error term)
TERM = *formula*	Single model term identifying the treatment term whose means are to be plotted
STRATUM = *formula*	Stratum from which to extract treatment information; default is to take the bottom stratum
FACTORIAL = *scalar*	Limit on number of factors in a treatment term; default 3
PROBABILITY = *scalar*	Significance level to use in the calculation of the radius of the confidence region and the region of non-significance; default 0.95
GRAPHICS = *string token*	Type of graphical output (lineprinter, highresolution); default high
STYLE = *string token*	controls the style of axes in a high-resolution graph (xy, none); default xy
LABELS = *factor* or *text*	Plotting symbols for the means; default is to take the letters A to Z, then a to z

Parameters

Y1 = *variates*	First variate for the bivariate analysis
Y2 = *variates*	Second variate for the bivariate analysis
TITLE = *texts*	Title for the graph

ABLUPS procedure

Calculates BLUPs for block terms in an ANOVA analysis (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (blups); default blup
PTERMS = *formula*	Specifies the block terms whose BLUPs are to be printed; default is to print them all
PSE = *string tokens*	Types of standard errors to be printed with the BLUPs (differences, alldifferences, blups, allblups); default diff, blup
SAVE = *identifier*	Save structure for the ANOVA analysis; default is to take the most recent ANOVA analysis

Parameters

TERMS = *formula*	Block terms whose BLUPs etc are to be saved
BLUPS = *table* or *pointer* to *tables*	Saves the BLUPs
SEBLUPS = *table* or *pointer* to *tables*	Standard errors for the BLUPs of each term
SEDMEANS = *symmetric matrix* or *pointer* to *symmetric matrices*	Standard errors of differences between the BLUPs of each term

ABOXCOX procedure

Estimates the power λ in a Box-Cox transformation, that maximizes the partial log-likelihood in ANOVA (W. van den Berg).

Options

PRINT = *string tokens*	Controls printed output (aovtable, lambda, monitoring); default aovt, lamb
TREATMENTSTRUCTURE = *formula*	Defines the treatment model; if this is not set, the default is taken from any existing setting defined by the TREATMENTSTRUCTURE directive
BLOCKSTRUCTURE = *formula*	Defines any block model; if this is not set, the default is taken from any existing setting defined by the BLOCKSTRUCTURE directive
COVARIATE = *variates*	Specifies any covariates; if this is not set, the default is taken from any existing setting defined by the COVARIATE directive
FACTORIAL = *scalar*	Limit in the number of factors in the terms generated from the TREATMENTSTRUCTURE formula; default 3
CONTRASTS = *scalar*	Limit on the order of a contrast of a treatment term; default 4
DEVIATIONS = *scalar*	Limit on the number of factors in a treatment term for the deviations from its fitted contrasts to be retained in the model; default 9
PLOT = *string token*	Whether to plot the partial log-likelihood (partialloglikelihood); default part
CIPROBABILITY = *scalar*	Probability level for the confidence interval for lambda; default 0.95, i.e. a 95% confidence interval
TRIALVALUES = *variate*	Values of λ for which the partial log-likelihood is to be calculated; default !(-4, -3.75 ... 4)
TRANSFORM = *string token*	How to transform the y-variate (estimate, trialvalue); default tria
STEPLENGTH = *scalar*	Steplength for estimating λ; default 0.01
MAXCYCLE = *scalar*	Maximum number of iterations; default 100
TOLERANCE = *scalar*	Tolerance for convergence; default 0.00001
ASAVE = *identifier*	Saves the ANOVA save structure from the analysis of variance

Parameters

Y = *variates*	Response variate
NEWY = *variates*	Saves the transformed response variate
LAMBDA = *scalars*	Saves the estimated value of λ
LOWER = *scalar*s	Saves the lower confidence limit for λ
UPPER = *scalar*s	Saves the upper confidence limit for λ

ACANONICAL procedure

Determines the orthogonal decomposition of the sample space for a design, using an analysis of the canonical relationships between the projectors derived from two or more model formulae (C.J. Brien).

Options

PRINT = *string tokens*	What to print (decomposition, df, ecriteria, efficiencies); default deco
CRITERIA = *string tokens*	The efficiency criteria to be saved and/or printed (aefficiency, mefficiency, sefficiency, eefficiency, xefficiency, order, dforth); default aeff, eeff, orde
FACTORIAL = *scalar*	Limit on the number of factors and variates in each model term default * i.e. no limit
TOLERANCE = *variate*	Tolerances for zero in various contexts; default 10^{-8} for all of these

Parameters

FORMULAE = *pointers*	Each pointer contains two or more model formulae whose joint decomposition is required

ORTHOGONALMETHOD = *string tokens*	Specifies the method to use for each model formula when orthogonalizing a projection matrix to those for terms that occur earlier in the formula (differencing, eigenmethods, hybrid); default hybr
PROJECTIONSETS = *pointers*	Saves the projection pointers formed from the formulae
COMBINEDPROJECTIONSET = *pointers*	Saves the projector pointers that produce the orthogonal decomposition
EFFICIENCYFACTORS = *pointers*	Saves the canonical efficiency factors
ECRITERIA = *pointers*	Saves the unadjusted efficiency criteria
ADJECRITERIA = *pointers*	Saves the adjusted efficiency criteria
ADJDF = *pointers*	Saves the adjusted degrees of freedom
SAVE = *pointers*	Saves information about the analysis for use by ACDISPLAY and ACKEEP

ACDISPLAY procedure

Provides further output from an analysis by ACANONICAL (C.J. Brien).

Option

PRINT = *string tokens*	What to print (decomposition, df, ecriteria, efficiencies); default deco

Parameter

SAVE = *pointer*	Information saved from ACANONICAL; if this is not set, the information is saved from the most recent ACANONICAL analysis

ACHECK procedure

Checks assumptions for an ANOVA analysis (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (tests, confirmation); default conf
ASSUMPTION = *string tokens*	Which assumptions to test (homogeneity, normality, stability); default homo, norm, stab
PROBABILITY = *scalar*	Critical value for the test probabilities to decide whether to generate warning messages; default=0.025
SAVE = *ANOVA save structure*	Specifies the analysis to be checked; by default this will be the most recent ANOVA

No parameters

ACKEEP procedure

Saves information from an analysis by ACANONICAL (C.J. Brien).

Options

COMBINEDPROJECTIONSET = *pointer*	Saves the projector pointers that produce the orthogonal decomposition
EFFICIENCYFACTORS = *pointer*	Saves the canonical efficiency factors
ECRITERIA = *pointer*	Saves the unadjusted efficiency criteria
ADJECRITERIA = *pointer*	Saves the adjusted efficiency criteria
ADJDF = *pointer*	Saves the adjusted degrees of freedom
SAVE = *pointer*	Information saved from ACANONICAL; if this is not set, the information is saved from the most recent ACANONICAL analysis

No parameters

ACONFIDENCE procedure

Calculates simultaneous confidence intervals for ANOVA means (D.M. Smith).

Options

PRINT = *string token*	Controls printed output (intervals); default inte
METHOD = *string token*	Type of interval (individual, smm, product, Bonferroni, Scheffe); default smm
FACTORIAL = *scalar*	Limit on the number of factors in each term; default 3
PROBABILITY = *scalar*	The required significance level; default 0.05
SAVE = *ANOVA save structure*	Save structure to provide the tables of means and associated information; default uses the save structure from the most recent ANOVA

Parameters

TERMS = *formula*	Treatment terms whose means are to be required
MEANS = *pointer* or *table*	Saves the means
LOWER = *pointer* or *table*	Saves the lower limits
UPPER = *pointer* or *table*	Saves the upper limits

ADD directive

Adds extra terms to a linear, generalized linear, generalized additive or nonlinear model.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence); default mode, summ, esti
NONLINEAR = *string token*	How to treat nonlinear parameters between groups (common, separate, unchanged); default unch
CONSTANT = *string token*	How to treat the constant (estimate, omit, unchanged, ignore); default unch
FACTORIAL = *scalar*	Limit for expansion of model terms; default * i.e. that in previous TERMS statement
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
AOVDESCRIPTION = *text*	Description for line in accumulated analysis of variance (or deviance) table when POOL=yes

Parameter

formula	List of explanatory variates and factors, or model formula

ADDPOINTS directive

Adds points for new objects to a principal coordinates analysis.

Option

PRINT = *string tokens*	Printed output required (coordinates, residuals); default * i.e. no printing

Parameters

NEWDISTANCES = *matrices*	Squared distances of the new objects from the original points
LRV = *LRVs*	Latent roots and vectors from the PCO analysis
CENTROID = *diagonal matrices*	Centroid distances from the PCO analysis
COORDINATES = *matrices*	Saves the coordinates of the additional points in the space of the original points
RESIDUALS = *matrices* or *variates*	Saves the residuals of the new objects from that space

ADETECTION procedure

Calculates the minimum size of effect or contrast detectable in an analysis of variance (R.W. Payne).

Options

PRINT = *string token*	Prints the minimum size of response that can be detected (detected); default dete
TERM = *formula*	Treatment term to be assessed in the analysis
TREATMENTSTRUCTURE = *formula*	Treatment structure of the design; determined automatically from an ANOVA save structure if TREATMENTSTRUCTURE is unset or if SAVE is set
BLOCKSTRUCTURE = *formula*	Block structure of the design; determined automatically from an ANOVA save structure if BLOCKSTRUCTURE is unset or if SAVE is set
FACTORIAL = *scalar*	Limit on the number of factors in treatment terms; default 3
PROBABILITY = *scalar*	Significance level at which the response is required to be detected (assuming a one-sided test); default 0.05
TMETHOD = *string token*	Type of test to be made (onesided, twosided, equivalence, noninferiority); default ones
XCONTRASTS = *variate*	X-variate defining a contrast to be detected
CONTRASTTYPE = *string token*	Type of contrast (regression, comparison); default rege
TOLERANCE = *scalar*	Tolerance for the iterations to calculate the detectable response
SAVE = *ANOVA save structure*	Save structure to provide the information about the design

Parameters

POWER = *scalars* or *variates*	Specifies the power i.e. probability with which the response should be detected
RMS = *scalars*	Anticipated residual mean square corresponding to TERM; can be omitted if a SAVE structure is available
DETECTED = *scalars* or *variates*	Minimum size of difference or contrast between the effects of TERM that is to be detected

ADISPLAY directive

Displays further output from analyses produced by ANOVA.

Options

PRINT = *string tokens*	Output from the analyses of the y-variates, adjusted for any covariates (aovtable, information, covariates, effects, residuals, contrasts, means, cbeffects, cbmeans, stratumvariances, %cv, missingvalues); default * i.e. no printing
UPRINT = *string tokens*	Output from the unadjusted analyses of the y-variates (aovtable, information, effects, residuals, contrasts, means, cbeffects, cbmeans, stratumvariances, %cv, missingvalues); default * i.e. no printing
CPRINT = *string tokens*	Output from the analyses of the covariates, if any (aovtable,

	information, effects, residuals, contrasts, means, %cv, missingvalues); default * i.e. no printing
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
PFACTORIAL = *scalar*	Limit on number of factors in printed tables of means or effects; default 9
PCONTRASTS = *scalar*	Limit on order of printed contrasts; default 9
PDEVIATIONS = *scalar*	Limit on number of factors in a treatment term whose deviations from the fitted contrasts are to be printed; default 9
FPROBABILITY = *string token*	Printing of probabilities for variance ratios in the aov table (yes, no); default no
PSE = *string tokens*	Standard errors to be printed with tables of means, PSE=* requests s.e.'s to be omitted (differences, lsd, means); default diff
TWOLEVEL = *string token*	Representation of effects in 2^n experiments (responses, Yates, effects); default resp
NOMESSAGE = *string tokens*	Which warning messages to suppress (nonorthogonal, residual); default *
LSDLEVEL = *scalar*	Significance level (%) to use in the calculation of least significant differences; default 5

Parameter

identifiers	Save structure (from ANOVA) to provide details of each analysis from which information is to be displayed; if omitted, output is from the most recent ANOVA

ADPOLYNOMIAL procedure

Plots single-factor polynomial contrasts fitted by ANOVA (R.W. Payne).

Option

SAVE = *ANOVA save structure*	Save structure (from ANOVA) to provide details of the analysis from which the polynomials are to be plotted; default uses the save structure from the most recent ANOVA

Parameters

XFACTOR = *factors*	Factor over which the polynomial contrasts have been formed
GROUPS = *factors* or *pointers*	Factor(s) for which different polynomial coefficients should be plotted in the same graph
TRELLISGROUPS = *factors* or *pointers*	Factor or factors for which different polynomial coefficients should be plotted in a trellis plot
TITLE = *texts*	Title for the graph; default defines a title automatically
YTITLE = *texts*	Title for the y-axis; default ' '
XTITLE = *texts*	Title for the x-axis; default is to use the identifier of the XFACTOR
PENS = *variates*	Defines the pen to use to plot the points and/or line for each group defined by the GROUPS factors

ADSPREADSHEET procedure

Puts the data and plan of an experimental design into a spreadsheet (R.W. Payne).

Options

DATA = *factors* or *variates*	Data variables (e.g. design factors and covariates) to put into the data spreadsheet; default takes the factors defined by previous BLOCKSTRUCTURE and TREATMENTSTRUCTURE directives
NEWDATA = *variates*	New variates (e.g. measurements to be taken during the experiment) to create and put into the data spreadsheet; default * i.e. none
Y = *variate* or *factor*	Specifies the *y*-coordinates of the plots for the plan spreadsheet

X = *variate* or *factor*	Specifies the *x*-coordinates of the plots for the plan spreadsheet
CONSTANTFACTORS = *string tokens*	Whether to put factors whose levels are constant in the *y* or *x* direction in a separate row or column of the Plan spreadsheet (y, x); default * i.e. neither
SEPARATOR = *text*	Separator for factor values in the plan spreadsheet; default '; '
OMITGAPS = *string token*	Whether to omit gaps when the plots in the plan are equally spaced (yes, no); default no
FOREGROUND = *scalar*, *variate* or *text*	Foreground colours to use for the plots in the experiment; default 'Black'
BACKGROUND = *scalar*, *variate* or *text*	Background colours to use for the plots in the experiment; default 'BlanchedAlmond'
CFACTORS = *factors*	Factors to determine the colour to use for each plot; default uses the first block factor or no colouring otherwise
GAPFOREGROUND = *text* or *scalar*	Foreground colour for gaps and surrounding plots; default 'Black'
GAPBACKGROUND = *text* or *scalar*	Background colour for gaps and surrounding plots; default 'LightGreen'
YFOREGROUND = *text* or *scalar*	Foreground colour for factors constant in y-direction; default 'Black'
YBACKGROUND = *text* or *scalar*	Background colour for factors constant in y-direction; default 'PaleTurquoise'
XFOREGROUND = *text* or *scalar*	Foreground colour for factors constant in x-direction; default 'Black'
XBACKGROUND = *text* or *scalar*	Background colour for factors constant in x-direction; default 'LightCyan'
SPREADSHEET = *string tokens*	Which spreadsheets to form (data, plan); default data
OUTFILENAME = *texts*	Name of Genstat workbook file (.gwb) or Excel (.xls or .xlsx) file to create

Parameters

FACTOR = *factors*	Factors to include in the plan spreadsheet; if unset, includes the factors defined by a previous TREATMENTSTRUCTURE directive
LABELS = *texts*	Labels to be used for each factor if its own levels or labels are inappropriate

AEFFICIENCY procedure

Calculates efficiency factors for experimental designs (R.W. Payne).

Options

FACTORIAL = *scalar*	Limit on the number of factors in each treatment term generated from TERMS; default 3
METHOD = *string token*	Whether to eliminate or ignore earlier model terms from the TERMS formula (eliminate, ignore); default elim
FORCED = *formula*	Terms to be eliminated before fitting TERMS; default * i.e. none

Parameters

TERMS = *formula*	Model terms
DF = *pointer* or *scalar*	Saves the degrees of freedom of the terms
EFFICIENCY = *pointer* or *variate*	Saves the efficiency factors of the terms
DFALIASED = *pointer* or *scalar*	Saves the number of aliased degrees of freedom of the terms

AFALPHA procedure

Generates alpha designs (R.W. Payne).

Option

PRINT = *string token*	Whether to print the design (design); default * i.e. no printing

Parameters

GENERATOR = *matrices*	generating array (of size number-of-plots-per-block by number-of-reps)
LEVELS = *scalars* or *variates*	Defines the levels of each treatment factor; if this is omitted, the levels of the TREATMENT factor are used, if available, otherwise LEVELS is determined from the generating array on the assumption that the blocks are to be of equal size
SEED = *scalar*	Seed to be used to randomize the design, if required
TREATMENTS = *factors*	Specifies the treatment factor for each design
REPLICATES = *factors*	Specifies the replicate factor
BLOCKS = *factors*	Specifies the block factor
UNITS = *factors*	Specifies the factor to index the units within each block

AFAUGMENTED procedure

Forms an augmented design (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (design); default * i.e. none
TREATMENTSTRUCTURE = *formula*	Treatment terms, other than GENOTYPES, to be included in the analysis
BLOCKSTRUCTURE = *formula*	Defines the block structure of the basic design
COVARIATE = *variates*	Specifies any covariates to be included in the analysis
LEVTEST = *variate*	Levels to represent the test genotypes in the augmented GENOTYPES factor
LEVCONTROL = *scalar* or *variate*	Levels to represent the control genotype(s) if these are not already in the GENOTYPES factor
GENOTYPES = *factor*	Genotype factor
CONTROLS = *factor*	Factor identifying the controls
TESTVSCONTROL = *factor*	Factor representing the comparison between test and control genotypes
SUBPLOTS = *factor*	Factor to represent the subplots to be created for the test genotypes in the basic design
NSUBPLOTS = *scalar*	Number of subplots to create within each plot of the basic design
SUBCONTROLS = *scalar* or *variate*	Subplots to be used for control genotypes, if not already pre-allocated in the GENOTYPES and SUBPLOTS factors; default selects subplots for the controls at random within each whole-plot
NREPTEST = *scalar* or *variate*	Number of times to replicate the test genotypes; default 1
SEED = *scalar*	Seed for the random numbers used to randomize the allocation of the genotypes (a negative value implies no randomization); default 0

No parameters

AFCARRYOVER procedure

Forms factors to represent carry-over effects in cross-over trials (R.W. Payne).

Option

NONELEVEL = *scalar* or *text*	Level or label to use for the units with no carry-over

Parameters

TREATMENTS = *factors*	Factors identifying the (direct) effects of the treatments
SUBJECTS = *factors*	Factors identifying the subjects
PERIODS = *factors*	Factors identifying the periods
CARRYOVERFACTOR = *factors*	Factors to represent the carry-over effect of the treatments in the period immediately after the period in which they were applied
NOCARRYOVER = *factors*	Factors to represent the comparison between none and any carry-over effect of the treatments

AFCOVARIATES procedure

Defines covariates from a model formula for ANOVA (R.W. Payne).

Options

COVARIATES = *pointer*	Saves the covariates
COVGROUPS = *pointer*	Saves the pointers defined to contain the covariates formed for each term in TERMS
FACTORIAL = *scalar*	Limit on number of factors in the model terms formed from TERMS; default 3

Parameters

TERMS = *formula*	Model terms from which to define covariates

AFCYCLIC procedure

Generates block and treatment factors for cyclic designs (R.W. Payne).

Option

PRINT = *string token*	Whether to print the design (design); default * i.e. no printing

Parameters

INITIALBLOCKS = *variates* or *pointers*	Defines one (variate) or more (pointer to variates) initial blocks for a treatment factor
INCREMENT = *scalars* or *pointers*	Defines the size of the successive increment (scalar) or increments (pointer to scalars) for each initial block
LEVELS = *scalars* or *variates*	Defines the levels of each treatment factor; this need not be specified if the factor has already been declared
SEED = *scalar*	Seed to be used to randomize each design, if required
TREATMENTS = *factors*	Specifies treatment factors
BLOCKS = *factors*	Specifies block factors
UNITS = *factors*	Specifies factors to index the units within each block

AFDISCREPANCY procedure

Calculates the discrepancy of a design (B.M. Parker).

Options

PRINT = *string tokens*	Controls whether to print the discrepancy (results); default resu
METHOD = *string token*	Specifies the method to use to calculate the discrepancy (L2, maximin, entropy); default L2
SWAP = *variate*	A variate of length two indicating which design points have swapped when updating the discrepancy criterion for the maximin or entropy criteria; default none

Parameters

DESIGN = *matrices* or *pointers*	A matrix, or a pointer of variates, specifying the design points
DISCREPANCY = *scalars*	Saves the discrepancy
DISTANCES = *matrices*	Stores the distances, to allow fast updates with the maximin or entropy criteria

AFFYMETRIX procedure

Estimates expression values for Affymetrix slides (D.B. Baird).

Options

PRINT = *string tokens*	What to print (estimates, background, monitoring); default para
METHOD = *string token*	Method for calculating probe expression values (mas4, mas5, rma, rma2); default rma
BMETHOD = *string token*	Method to use for background values (mean, quantile, none); default mean for METHOD settings mas4 and mas5, but none for settings rma and rma2
BWEIGHTING = *string token*	Method for weighting background grids (affymetrix,

distance); default affy

TRANSFORMATION = *string token* — How to transform the data (log2, none); default log2

NMETHOD = *string token* — Method for normalization i.e. whether to use a mean, median or geometric mean for the averaged normalized distribution (means, medians, geometricmeans, none); default mean

REPLACEDATA = *string token* — Whether to replace the DATA variates with background corrected intensities (yes, no); default no

SPREADSHEET = *string token* — What to save in a spreadsheet (results); default * i.e. nothing

MAXCYCLE = *scalar* — Maximum number of iterations; default 50

TOLERANCE = *scalar* — Tolerance for convergence; default 0.0001

Parameters

DATA = *variates* — Intensities to be analysed

SLIDES = *factors* — Identify the slides (or chips)

PROBES = *factors* — Identify the probes (or genes) within each slide

ATOMS = *factors* — Identify the PM/MM pairs within each probe

PMMM = *factors* — Distinguish between PM and MM values

TYPEPROBES = *factors* — Defines the probe-type corresponding to each intensity

ROWS = *factors* — Identifies rows within each slide (required only if background corrections are to be made)

COLUMNS = *factors* — Identifies columns within each slide (required only if background corrections are to be made)

ESTIMATES = *variates* — Saves the estimated expression values for each slide and probe combination

SE = *variates* — Saves approximate standard errors for the estimates

IDSLIDES = *factors* — Saves factors to identify the slides in the ESTIMATES variates

IDPROBES = *factors* — Saves factors to identify the probes in the ESTIMATES variates

AFIELDRESIDUALS procedure

Display residuals in field layout (R.W. Payne & A.D.Todd).

Options

PRINT = *string tokens* — Controls output (contour, shade, table); default cont

GRAPHICS = *string token* — Type of graph (highresolution, lineprinter); default high

METHOD = *string token* — Type of residuals to take from the save structure when the RESIDUALS parameter is not specified (combined, finalstratum, standardizedfinal); default comb

MARGIN = *string token* — Whether to include margins in printed tables (yes, no); default no

YORIENTATION = *string token* — Y-axis orientation of the plot (reverse, normal); default norm

PENCONTOUR = *scalar* — Pen number to be used for the contours; default 1

PENFILL = *scalar* or *variate* — Pen number(s) defining how to fill the areas between contours; default 3

PENSHADE = *scalar* or *variate* — Pen(s) to use for the shade plot; default 3

Parameters

Y = *variates* or *factors* — Specifies the y-coordinates of the plots

X = *variates* or *factors* — Specifies the x-coordinates of the plots

RESIDUALS = *variates* — Residuals to be plotted; default is to take the residuals from the save structure specified by the SAVE option, or from the most recent ANOVA if that is unspecified

SAVE = *ANOVA, REML or regression save structures* — Save structure of the ANOVA, REML or regression analysis from which to take the residuals if the RESIDUALS parameter is not specified; default is to take the most recent ANOVA analysis

FIELDWIDTH = *scalars* — Field width for printing the residuals; default 12

DECIMALS = *scalars* — Number of decimal places to use when printing the residuals

TITLE = *texts* Titles for the plots

AFLABELS procedure

Forms a variate of unit labels for a design (R.W. Payne).

Options

UNITLABELS = *variate* Stores the labels
MAXDIGIT = *scalar* Number of available digits; default 8

Parameters

FACTOR = *factors* Factors indexing the units of the design; if this is unset, the factors from the most recent BLOCKSTRUCTURE command are used
NEWLEVELS = *variates* Allows new levels to be specified for each FACTOR; if this is unset, uses the levels already defined for the factor

AFMEANS procedure

Forms tables of means classified by ANOVA treatment factors (R.W. Payne).

Options

PRINT = *string tokens* What to print (means, sed, sedsummary, ese, lsd, lsdsummary); default mean, sed
MEANS = *table* Saves means; default *
SED = *symmetric matrix* Saves matrices of standard errors of differences between means; default *
ESE = *table* Saves effective standard errors; default *
LSD = *symmetric matrix* Saves least significant differences between means; default *
LSDLEVEL = *scalar* Significance level (%) for least significant differences; default 5
DFMEANS = *symmetric matrices* Saves degrees of freedom for comparisons between every pair of entries in the table of means
EQFACTORS = *factors* Factors whose levels are to be assumed to be equal within the comparisons between means, when calculating effective standard errors
SAVE = *ANOVA save structure* Save structure to provide the table of means; default uses the save structure from the most recent ANOVA

Parameter

CLASSIFY = *vectors* Factors to classify table of means (from those in the TREATMENTSTRUCTURE in the ANOVA analysis)

AFMINABERRATION directive

Forms minimum aberration designs using the algorithm of Laycock & Rowley (1995).

Options

PRINT = *string tokens* Controls printed output (summary, keyblocks, keydefining, monitoring); default *
NTIMES = *scalar* Number of designs to try in a random search; default 0 does the full search
SEED = *scalar* Seed for the random number generator used to search the designs randomly; default 0

Parameters

LEVELS = *scalars* Number of levels of the treatment factors, must be a power of a prime number
NTREATMENTFACTORS = *scalars* Number of treatment factors
NUNITS = *scalars* Number of units in each block of a block design or in the principal block of a fractional factorial
NSUBUNITS = *scalars* Number of units in each (sub-)block
KEYBLOCKS = *matrices* Design key for the blocks and sub-blocks
KEYDEFINING = *matrices* Design key specifying the defining contrasts
RESOLUTION = *scalars* Saves the resolution of the design

ABERRATION = *scalars*	Saves the aberration of the design
SUBRESOLUTION = *scalars*	Saves the resolution of the sub-design
SUBABERRATION = *scalars*	Saves the aberration of the sub-design
NDESIGN = *scalars*	Saves or defines the design number
NSUBDESIGN = *scalars*	Saves or defines the sub-design number

AFNONLINEAR procedure

Forms D-optimal designs to estimate the parameters of a nonlinear or generalized linear model (W. van den Berg).

Options

PRINT = *string token*	Controls printed output (results, monitoring); default resu, moni
PLOT = *string token*	Controls whether to plot the design (design); default desi
YARGUMENT = *identifier*	Data structure that stores the results of the function when it is calculated by expressions supplied by the FUNCTION option; must be set
XARGUMENT = *identifier*	Data structure representing the x-variate in the expressions supplied by the FUNCTION option; must be set
FUNCTION = *expression structures*	Specifies the function whose parameters are to be estimated; must be set
FNDERIVATIVES = *expression structures*	Specifies expressions to calculate derivative of the function with respect to each parameter; must be set
ITERATIVEWEIGHTS = *identifier*	Data structure that stores the iterative weights in the expressions supplied by the FNITERATIVEWEIGHTS option
FNITERATIVEWEIGHTS = *expression structures*	Specifies expressions to calculate the iterative weights when estimating the parameters of a generalized linear model
XSUPPORT = *variate*	Supplies the support points for the initial design, and saves those of the final design; if no initial values are supplied, an initial design is formed at random
XWEIGHTS = *variate*	Supplies the weights for the support points for the initial design, and saves those of the final design; if no initial values are supplied, equal weights are used initially
GRID = *variate*	Specifies the grid points where the design will be evaluated
A0 = *scalar*	Initial update weight; default 0.1
SEED = *scalar*	Seed for the random numbers used to select the initial design when not supplied by XSUPPORT and XWEIGHTS
NCYCLE = *scalar*	Number of iterations to make between at each value of A0, before halving it for the next batch of iterations; default 100
MAXCYCLE = *scalar*	Maximum number of iterations; default 2500
TOLERANCES = *variate*	Variate with two values specifying the convergence criterion and the tolerance for zero weights; default !(1.E-6, 1.E-5)

Parameters

PARAMETER = *scalars*	Parameters of the nonlinear or generalized linear model (with values giving an indication of their likely estimated values)
DERIVATIVE = *identifiers*	Data structures that store the results of the calculation of the derivative for each parameter, in the expressions specified by the FNDERIVATIVES option

AFORMS procedure

Prints data forms for an experimental design (R.W. Payne).

Options

BLOCKSTRUCTURE = *formula*	Defines the block factors to be used to label the units of the design; default takes those specified in an earlier BLOCKSTRUCTURE directive

TREATMENTSTRUCTURE = *formula*	Defines the treatment factors to be used, if any, to label the forms
NLINES = *scalar*	Number of lines to be allowed for each measurement; default 1

Parameters

LABEL = *texts*	Labels for the measurements to be recorded on the forms
FIELDWIDTH = *scalar*	Fieldwidth to be allowed for each label

AFPREP procedure

Searches for an efficient partially-replicated design (R.W. Payne).

Options

PRINT = *strings*	Controls printed output (design, efficiency, factors, monitoring); default * i.e. none
LEVELS = *scalar* or *variate*	Levels of the treatment factor; if unset, takes the levels declared for the factor specified by the TREATMENTS option
NREPEATS = *variate*	Number of times each treatment occurs in the design
NBLOCKS = *scalar*	Number of blocks
TREATMENTS = *factor*	Treatment factor
BLOCKS = *factor*	Block factor
UNITS = *factor*	Unit-within-block factor
EFFICIENCY = *variate*	Saves the efficiency factors of the treatment term within blocks
NSTARTS = *scalar*	Specifies the number of random starting configurations to take in the search for the best design
NTRIES = *scalar*	Number of designs to try from each starting configuration
SEED = *scalar*	Seed for the random numbers used to randomize the design; default 0
TRYSEED = *scalar*	Seed for the random numbers used to select the random starting configurations; default 0
SPREADSHEET = *string*	Whether to put the design factors into a spreadsheet (design); default *

No parameters

AFRCRESOLVABLE procedure

Forms doubly resolvable row-column designs, with output (D.B. Baird).

Options

PRINT = *string tokens*	Controls printed output (design, plotnumbers, factors, efficiency; default desi, effi
DESIGNPLOT = *string token*	What factors to display in the design plot (treatment, plotandtreatment); default * i.e. no plot
FIRSTPLOT = *string token*	Defines the starting location for allocating plots to the row-by-column grid (lowleft, lowright, upleft, upright); default uple
PLOTORDER = *string token*	Defines the order in which the blocks are filled (colserpentine, colbycol, rowserpentine, rowbyrow); default rowb
TIME = *scalar*	Time in seconds to spend searching for an optimal design; default 60
SEED = *scalar*	Seed for the randomization; default 0
MAXITERATIONS = *scalar*	The number of random designs to search for an optimal design; default 10000
SPREADSHEET = *string token*	What to save in a spreadsheet (data, plan); default *

Parameters

NROWS = *scalars*	Number of rows in the layout of each design
NCOLUMNS = *scalars*	Number of columns in the layout of each design
LEVELS = *scalar*, *variate* or *text*	Defines the number of levels or labels of the TREATMENT factor for each design
TREATMENTS = *factors*	Saves the treatment allocation in each design

ROWREPLICATES = *factors*	Saves the row replicates in each design
COLREPLICATES = *factors*	Saves the column replicates in each design
ROWS = *factors*	Saves the row locations of the plots in each design
COLUMNS = *factors*	Saves the column locations of the plots in each design
PLOTNUMBER = *factors*	Saves the plot numbers
TITLE = *texts*	The title for the design plot; default an automatic description of the design
OUTFILE = *texts*	Gives a file name (with extension .gsh, .gwb, or .xlsx) to save the factors in each design
EXIT = *scalars*	Saves the exit code from the design search program (0 for success, greater than 0 for failure)

AFRESPONSESURFACE directive

Uses the BLKL algorithm to construct designs for estimating response surfaces.

Options

PRINT = *string token*	Printed output required (monitoring); default * i.e. no printing
TERMS = *formula*	Model to be fitted when the design is used; no default i.e. this option must be specified
CONSTANT = *string token*	How to treat the constant in the model (estimate, omit); default esti
FACTORIAL = *scalar*	Limit for expansion of terms in the model; default 2
NUNITS = *scalar*	Number of units (or trials) in the design
NDELETION = *scalar*	Number of design points to consider for deletion; default takes NUNITS/4, or 4 is this is larger
NINCLUSION = *scalar*	Number of design points to consider for inclusion; default takes NUNITS/4, or 4 is this is larger
NRUNS = *scalar*	Number of times to run the algorithm; default 100
ADJUSTMENTSTEP = *scalar*	Maximum amount by which to perturb the design points in the adjustment algorithm; default * i.e. no adjustments are tried
NBLOCKS = *scalar*	Number of blocks; default 1 i.e. design not blocked
BLOCKFACTOR = *factor*	Saves the block factor (if any) for the design
BLOCKSIZE = *scalar* or *variate*	Number of units in each block of the design
PREVIOUSBLOCKS = *factor*	Supplies values of the blocking factor for any previous experiments that are to be included in the analysis of the results of the design
MIXTURE = *variates*	Lists any variates that are part of a mixture (their values must be greater than zero and sum to one)
SEED = *scalar*	Seed for random numbers used to construct the initial design; default 124195
DETERMINANT = *scalar*	Saves the determinant of the information matrix for the best design
MEANGRID = *scalar*	Saves the mean value of the standardized variance of predictions obtained from the design over a grid of x-values
MAXGRID = *scalar*	Saves the maximum value of the standardized variance of predictions obtained from the design over a grid of x-values
NGRIDPOINTS = *scalar*	Number of grid points in each x-direction to use for MEANGRID and MAXGRID; default 5

Parameters

X = *variates*	Lists the variates to be investigated in the design; these need not be supplied if none of the other parameters are required
X2 = *variates*	Lists identifiers to be used to represent squares of the x-variates in the model
X3 = *variates*	Lists identifiers to be used to represent squares of the x-variates in the model
SUPPORTPOINTS = *variates*	Support points for each x-variate in the design; if these are not

(all) specified, they are formed automatically

PREVIOUSVALUES = *variates* — Supplies values of the x-variates for any previous experiments that are to be included in the analysis of the results of the design

AFUNITS procedure

Forms a factor to index the units of the final stratum of a design (R.W. Payne & W. van den Berg).

Option

BLOCKSTRUCTURE = *formula* — Defines the block factors for the design; the default is to take those specified by the BLOCKSTRUCTURE directive

Parameter

UNITS = *factor* — Factor to be formed

AGALPHA procedure

Forms alpha designs by standard generators for up to 100 treatments (M.F. Franklin & R.W. Payne).

Option

PRINT = *string token* — Controls whether or not to print a plan or the generator of of the design (design, generator); if unset in an interactive run AGALPHA will ask whether the design and generator are to be printed, in a batch run the default is not to print anything

Parameters

LEVELS = *scalars* — Number of treatments
NREPLICATES = *scalars* — Number of replicates
NBLOCKS = *scalars* — Number of blocks per replicate
SEED = *scalars* — Seed for randomization; a negative value implies no randomization
TREATMENTS = *factors* — Identifier for the treatment factor
REPLICATES = *factors* — Identifier for the replicate factor
BLOCKS = *factors* — Identifier for the factor to index the blocks within replicates
UNITS = *factors* — Identifier for the factor to index the units (or plots) within each block
STATEMENT = *texts* — Saves a command to recreate each design (useful if the design information has been specified in response to questions from AGALPHA)

AGBIB procedure

Generates balanced incomplete block designs (R.W. Payne).

Options

PRINT = *string token* — Controls whether or not to print a plan of the design and whether to print a catalogue of the designs in the subfile (design, catalogue); if unset in an interactive run AGBIB will ask whether the design is to be printed, in a batch run the default is not to print anything

ANALYSE = *string token* — Controls whether or not to analyse the design, and produce a skeleton analysis-of-variance table using ANOVA (no, yes); default is to ask if this is unset in an interactive run, and not to analyse if it is unset in a batch run

Parameters

LEVELS = *scalars* — Number of treatments
NBLOCKS = *scalars* — Number of blocks
NUNITS = *scalars* — Number of units per block
SEED = *scalars* — Seed for randomization; a negative value implies no randomization
TREATMENTS = *factors* — Identifier for the treatment factor
BLOCKS = *factors* — Identifier for the factor to index the blocks
UNITS = *factors* — Identifier for the factor to index the units within each block

STATEMENT = *texts*	Saves a command to recreate each design (useful if the design information has been specified in response to questions from AGBIB)

AGBOXBEHNKEN procedure

Generates Box Behnken designs (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (design); if unset in an interactive run AGBOXBEHNKEN will ask whether the design is to be printed, in a batch run the default is not to print anything
NCENTRALPOINTS = *scalar*	Defines the number of central points to include; default 4
LEVELS = *variate*	Defines the outer levels to be used; default !(-1,1)
NCOMBINATIONS = *scalar*	Number of factors to vary in combination at once; default 2
SEED = *scalar*	Seed to be used to randomize each design; a negative value implies no randomization
STATEMENT = *text*	Saves a command to recreate the design (useful if the design information has been specified in response to questions from AGBOXBEHNKEN)

Parameter

TREATMENTFACTOR = *factors*	Treatment factors

AGCENTRALCOMPOSITE procedure

Generates central composite designs (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (design); if unset in an interactive run AGCENTRALCOMPOSITE will ask whether the design is to be printed, in a batch run the default is not to print anything
NCENTRALPOINTS = *scalar*	Defines the number of central points to include; default 4
NSTARPOINTS = *scalar*	Defines the number of star points to include; default 1
LFACTORIAL = *variate*	Defines the treatment levels in the factorial part of the design; default !(-1,1)
LSTAR = *variate*	Defines the treatment levels for the star points; default is to use the levels defined by LFACTORIAL
FRACTION = *scalar*	Denominator for fractional factorial; default 1 specifies a complete design
SEED = *scalar*	Seed to be used to randomize each design; a negative value implies no randomization
STATEMENT = *text*	Saves a command to recreate the design (useful if the design information has been specified in response to questions from AGCENTRALCOMPOSITE)

Parameter

TREATMENTFACTOR = *factors*	Treatment factors

AGCROSSOVERLATIN procedure

Generates Latin squares balanced for carry-over effects (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (design); if unset in an interactive run ACROSSOVERGLATIN will ask whether the design is to be printed, in a batch run the default is not to print anything
ANALYSE = *string token*	Controls whether or not to analyse the design, and produce a skeleton analysis-of-variance table using ANOVA (yes, no); default is to ask if this is unset in an interactive run, and not to analyse if it is unset in a batch run

Parameters

LEVELS = *scalars* or *variates*	Number of treatments (scalar) or levels for the treatments
SEED = *scalars*	Seed to be used to randomize the design; a negative value

	implies no randomization
TREATMENTS = *factors*	Identifier for a factor to represent the direct effects of the treatments
SUBJECTS = *factors*	Identifier for a factor to represent the subjects
PERIODS = *factors*	Identifier for a factor to represent the periods
CARRYOVERFACTOR = *factors*	Identifier for a factor to represent the carry-over (or "residual") effect of the treatments in the period immediately after the period in which they were applied
NOCARRYOVER = *factors*	Identifier for a factor to represent the comparison between none and any carry-over effect of the treatments
STATEMENT = *texts*	Saves a command to recreate each design (useful if the design information has been specified in response to questions from AGCROSSOVERLATIN)

AGCYCLIC procedure

Generates cyclic designs from standard generators (M.F. Franklin & R.W. Payne).

Options

PRINT = *string token*	Controls whether or not to print a plan of the design (design); if unset in an interactive run AGCYCLIC will ask whether the design is to be printed, in a batch run the default is not to print the design
METHOD = *string token*	Type of design – ordinary cyclic, cyclic change-over or cyclic superimposed (cyclic, changeover, superimposed); if unset in an interactive run AGCYCLIC will ask about the type of design, in a batch the default is assumed to be cyclic

Parameters

LEVELS = *scalars*	Number of treatments
NBLOCKS = *scalars*	Number of blocks
NUNITS = *scalars*	Number of units per block, or number of periods in a cyclic change-over design
SEED = *scalars*	Seed for randomization; a negative value implies no randomization
TREATMENTS = *factors*	Identifier for the treatment factor
SUPERIMPOSED = *factors*	Identifier for the second treatment factor in a cyclic superimposed design
BLOCKS = *factors*	Identifier for the factor to index the blocks
UNITS = *factors*	Identifier for the factor to index the units within each block, or the periods of a cyclic change-over design
INITIALBLOCKS = *variates* or *pointers*	To save one (variate) or more (pointer to variates) initial blocks
STATEMENT = *texts*	Saves a command to recreate the design (useful if the design information has been specified in response to questions from AGCYCLIC)

AGDESIGN procedure

Generates generally balanced designs (R.W. Payne).

Options

PRINT = *string token*	Controls whether or not to print a plan of the design and whether to print a catalogue of the designs in the subfile (design, catalogue); if unset in an interactive run AGDESIGN will ask whether the design is to be printed, in a batch run the default is not to print anything
ANALYSE = *string token*	Controls whether or not to analyse the design, and produce a skeleton analysis-of-variance table using ANOVA (no, yes); default is to ask if this is unset in an interactive run, and not to analyse if it is unset in a batch run

FILENAME = *text*	Name of the backing store file containing the design information; default uses the standard design file
SUBFILE = *identifier*	Subfile of the backing store file to be used

Parameters

DESIGN = *variates*	Contains codes to indicate the choice of design
TREATMENTFACTORS = *pointers*	Specifies identifiers for the treatment factors
BLOCKFACTORS = *pointers*	Specifies identifiers for the block factors
PSEUDOFACTORS = *pointers*	Specifies identifiers for any pseudo-factors
REPLICATEFACTOR = *factors*	Specifies the identifier of the factor to represent the replicates (if any) in each design
UNITLABELS = *variates*	Specifies the identifier of a variate to store a unique numerical label for each plot in the design
SEED = *scalars*	Seed to be used to randomize each design; a negative value implies no randomization
STATEMENT = *texts*	Saves a command to recreate each design (useful if the design information has been specified in response to questions from AGDESIGN)

AGFACTORIAL procedure

Generates minimum aberration block or fractional factorial designs (P.J. Laycock, P.J. Rowley & R.W. Payne).

Options

PRINT = *string token*	Controls whether or not to print a plan of the design (design); if unset in an interactive run AGFACTORIAL will ask whether the design is to be printed, in a batch run the default is not to print the design
ANALYSE = *string token*	Controls whether or not to analyse the design, and produce a skeleton analysis-of-variance table using ANOVA (yes, no); default is to ask if this is unset in an interactive run, and not to analyse if it is unset in a batch run
FACTORIAL = *scalar*	Limit on number of factors in treatments terms in the analysis of variance; default 3

Parameters

LEVELS = *scalars*	Number of levels of the treatment factors in each design
NTREATMENTFACTORS = *scalars*	Number of treatment factors
NUNITS = *scalars*	Number of units per block
NFRACTIONBLOCK = *scalars*	Defines the number of the block to use to define a fractional factorial, or can be set to zero to take a block at random; if unset in an interactive run AGFACTORIAL will ask whether to form a fractional factorial design, in a batch run the default is to form the full (block) design
NSUBUNITS = *scalars*	Number of units in each sub-block
SEED = *scalars*	Seed to be used to randomize each design; a negative value implies no randomization
TREATMENTFACTORS = *pointers*	Specifies identifiers for the treatment factors
BLOCKS = *factors*	Identifier for the block factor
SUBBLOCKS = *factors*	Identifier for the sub-block factor
PSEUDOFACTORS = *pointers*	Specifies identifiers for pseudo-factors
UNITLABELS = *variates*	Specifies the identifier of a variate to store a unique numerical label for each unit in the design
NDESIGN = *scalars*	Saves or defines the design number
NSUBDESIGN = *scalars*	Saves or defines the sub-design number
STATEMENT = *texts*	Saves a command to recreate each design (useful if the design information has been specified in response to questions from AGFACTORIAL)

AGFRACTION procedure

Generates fractional factorial designs (M.F. Franklin & R.W. Payne).

Options

PRINT = *string token*	Controls whether or not to print a plan of the design (design); if unset in an interactive run AGFRACTION will ask whether the design is to be printed, in a batch run the default is not to to print the design
ANALYSE = *string token*	Controls whether or not to analyse the design, and produce a skeleton analysis-of-variance table using ANOVA (no, yes); default is to ask if this is unset in an interactive run, and not to analyse if it is unset in a batch run
FACTORIAL = *scalar*	Limit on number of factors in treatments terms in the analysis of variance; default 2
FILENAME = *text*	Name of the backing store file containing the design information; default uses the standard fractional design file

Parameters

LEVELS = *scalars*	Number of levels of the treatment factors in each design
FRACTION = *scalars*	Denominator of required fraction
NTREATMENTFACTORS = *scalars*	Number of treatment factors
NUNITS = *scalars*	Number of units per block
SEED = *scalars*	Seed to be used to randomize each design; a negative value implies no randomization
TREATMENTFACTORS = *pointers*	Specifies identifiers for the treatment factors
BLOCKS = *factors*	Identifier for the block factor
UNITS = *factors*	Identifier for the factor to index the units (or plots) within each block
STATEMENT = *texts*	Saves a command to recreate each design (useful if the design information has been specified in response to questions from AGFRACTION)

AGHIERARCHICAL procedure

Generates orthogonal hierarchical designs (R.W. Payne).

Options

PRINT = *string token*	Controls whether or not to print a plan of the design (design); if unset in an interactive run AGHIERARCHICAL will ask whether the design is to be printed, in a batch run the default is not to print the design
ANALYSE = *string token*	Controls whether or not to analyse the design, and produce a skeleton analysis-of-variance table using ANOVA (no, yes); default is to ask if this is unset in an interactive run, and not to analyse if it is unset in a batch run
SEED = *scalar*	Seed to be used to randomize the design; a negative value implies no randomization
STATEMENT = *text*	Saves a command to recreate the design (useful if the design information has been specified in response to questions from AGHIERARCHICAL)
EXCLUDELEVELS = *scalars*	Levels of the first block factor to exclude during randomization

Parameters

BLOCKFACTORS = *factors*	Specifies the identifier for the block factor used to index the units of each stratum (or level of the hierarchy)
TREATMENTFACTORS = *factors* or *pointers*	Specifies the identifier of the treatment factor or factors applied to the units of each stratum
LEVELS = *scalars* or *pointers*	Number of levels for the treatment factors in each stratum; if required, a pointer can contain an extra scalar to specify

replication

AGINDUSTRIAL procedure

Helps to select and generate effective designs for use in industrial experiments (R.W. Payne).

Option

STATEMENT = *text*	Saves a command to recreate the design

No parameters

AGLATIN procedure

Generates mutually orthogonal Latin squares (I. Wakeling & R.W. Payne).

Options

PRINT = *string token*	Controls printed output (design, squares, list); if unset in an interactive run AGLATIN will ask whether the design is to be printed, in a batch run the default is not to print anything
ANALYSE = *string token*	Controls whether or not to analyse the design, and produce a skeleton analysis-of-variance table using ANOVA (no, yes); default is to ask if this is unset in an interactive run, and not to analyse if it is unset in a batch run

Parameters

NROWS = *scalars*	Specifies the number of rows (and columns) in each square
NSQUARES = *scalars*	Number of squares to form (i.e. number of treatment factors to generate)
SEED = *scalars*	Seed to be used to randomize each design; a negative value implies no randomization
TREATMENTFACTORS = *pointers*	Pointer to identifiers for the treatment factors
ROWS = *factors*	Identifier for the row factor
COLUMNS = *factors*	Identifier for the column factor
MAXNSQUARES = *scalars*	Returns the maximum number of squares available with the specified number of rows and columns
STATEMENT = *texts*	Saves a command to recreate each design (useful if the design information has been specified in response to questions from AGLATIN)

AGLOOP procedure

Generates loop designs e.g. for time-course microarray experiments (R.W. Payne).

Option

PRINT = *string token*	Controls whether or not to print a plan of the design (design); if unset in an interactive run AGLOOP will ask whether the design is to be printed, in a batch run the default is not to print the design

Parameters

LEVELS = *scalars*	Number of treatments
INCREMENTS = *scalars*, *variates* or *pointers*	Increment or increments to be used to form the loops
SEED = *scalars*	Seed for randomization; a negative value implies no randomization
TREATMENTS = *factors*	Identifier for the treatment factor
BLOCKS = *factors*	Identifier for the block (plate) factor
UNITS = *factors*	Identifier for the factor for the units within each block (or colours in a microarray experiment)
STATEMENT = *texts*	Saves a command to recreate the design (useful if the design information has been specified in response to questions from AGLOOP)

AGMAINEFFECT procedure

Generates designs to estimate main effects of two-level factors (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (design, catalogue); if unset in an interactive run AGMAINEFFECT will ask whether the design or catalogue are to be printed, in a batch run the default is not to print anything
ANALYSE = *string token*	Controls whether or not to analyse the design, and produce a skeleton analysis-of-variance table using ANOVA (no, yes); default is to ask if this is unset in an interactive run, and not to analyse if it is unset in a batch run
FOLDED = *string token*	Whether to include an extra "folded" replicate with the levels of each factor interchanged (no, yes); default no
SEED = *scalar*	Seed to be used to randomize each design; a negative value implies no randomization
STATEMENT = *texts*	Saves a command to recreate the design (useful if the design information has been specified in response to questions from AGMAINEFFECT)

Parameter

TREATMENTFACTOR = *factors*	Treatment factors

AGNATURALBLOCK procedure

Forms 1- and 2-dimensional designs with blocks of natural size (P.D. Johnstone & D.B. Baird).

Options

PRINT = *string token*	Controls printed output (design, search); default desi
DESIGNTYPE = *string token*	Type of design to create (block, rowcolumn); default rowc
NSIMULATIONS = *scalar*	Number of randomizations to search to find the best design; default 1000
SEED = *scalar*	Seed for the randomization; default 0
FIRSTPLOT = *string token*	Defines the starting location for allocating plots to the row-by-column grid (lowleft, lowright, upleft, upright); default uple
FILLMETHOD = *string token*	Defines the order in which the plots are filled (colserpentine, colbycol, rowserpentine, rowbyrow); default rows

Parameters

LEVELS = *scalars* or *variates*	Defines the levels of the treatment factor for each design
NROWS = *scalars*	Number of rows in the smallest rectangle containing the layout of each design; not required if the ROWS parameter is set to a factor with values
NCOLUMNS = *scalars*	Number of columns in the smallest rectangle containing the layout of each design; not required if the COLUMNS parameter is set to a factor with values
NUNITS = *scalar*	Number of plots that will be assigned a treatment in each design; not required if the either the ROWS or COLUMNS parameter is set to a factor with values
TREATMENTS = *factors*	Saves the treatment allocation for each design
ROWS = *factors*	Defines or saves the row locations of the plots to receive treatments in each design
COLUMNS = *factors*	Defines or saves the column locations of the plots to receive treatments in each design
BLOCKS = *factors*	Defines or saves the allocation of the plots to blocks
PLAN = *matrices*	Saves the treatment layout in each design

AGNEIGHBOUR procedure

Generates neighbour-balanced designs (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (catalogue, design); if unset in an interactive run AGNEIGHBOUR will ask whether the design is to be printed, in a batch run the default is not to print anything
METHOD = *string token*	Type of design, *n* − 1 blocks of *n* plots, or *n* blocks of *n* − 1 plots (N_1BLOCKS, NBLOCKS); if unset in an interactive run AGNEIGHBOUR will ask about the type of design, in a batch the default is assumed to be n blocks of n − 1 plots

Parameters

LEVELS = *scalars*	Number of treatments
SEED = *scalars*	Seed for randomization; in batch there is a default of 12345
TREATMENTS = *factors*	Identifier for the treatment factor
BLOCKS = *factors*	Identifier for the factor to index the blocks within replicates
UNITS = *factors*	Identifier for the factor to index the units within each block, or the periods of a cyclic change-over design
LEFTNEIGHBOUR = *factors*	To save the treatment on the left neighbouring unit
RIGHTNEIGHBOUR = *factors*	To save the treatment on the right neighbouring unit
STATEMENT = *texts*	Saves a command to recreate each design (useful if the design information has been specified in response to questions from AGNEIGHBOUR)

AGNONORTHOGONALDESIGN procedure

Generates non-orthogonal split-plot and other hierarchical designs (B. M. Parker).

Options

PRINT = *string token*	Controls printed output (design, debug); default * i.e. nothing
METHOD = *string token*	Specifies the algorithm to use (jonesgoos, trincagilmour); default trin
CRITERION = *string token*	Optimality criterion (a, d); default a
MODELMATRIX = *matrix*	Defines the model to be estimated
NSTARTS = *scalar*	Number of random starts for the jg algorithm; default 10
NTRIES = *scalar*	Number of exchanges to try from each start; default 10000
MINIMUM = *scalar*	Minimum value for levels; default −1
MAXIMUM = *scalar*	Maximum value for levels; default 1
SEED = *scalar*	Specifies the seed for the random numbers used by the algorithms; default 0

Parameters

BLOCKFACTORS = *factors*	Specifies the identifier for the block factor used to index the units of the whole-plots, the sub-plots and, if required, the sub-sub-plots
TREATMENTFACTORS = *factors* or *pointers*	Specifies the identifier of the treatment factor or factors applied to the whole, sub-plots and sub-sub-plots
BLEVELS = *scalars*	Numbers of levels for the block factors
LEVELS = *scalars* or *pointers*	Numbers of levels for the treatment factors
VARIANCES = *scalars*	Variances for the strata

AGQLATIN procedure

Generates complete and quasi-complete Latin squares (R.W. Payne).

Options

PRINT = *string token*	Controls printing of the design (design); if unset in an interactive run AGQLATIN will ask whether the design is to be printed, in a batch run the default is not to print anything
ANALYSE = *string token*	Controls whether or not to analyse the design, and produce a

skeleton analysis-of-variance table using ANOVA (no, yes); default is to ask if this is unset in an interactive run, and not to analyse if it is unset in a batch run

Parameters

NROWS = *scalars* — Specifies the number of rows (and columns) in the square
SEED = *scalars* — Seed to be used to randomize each design; a negative value implies no randomization
TREATMENTS = *factors* — Identifier for the treatment factor
ROWS = *factors* — Identifier for the row factor
COLUMNS = *factors* — Identifier for the column factor
STATEMENT = *texts* — Saves a command to recreate each design (useful if the design information has been specified in response to questions from AGQLATIN)

AGRAPH procedure

Plots tables of means from ANOVA (R.W. Payne).

Options

GRAPHICS = *string token* — Type of graph (highresolution, lineprinter); default high
METHOD = *string token* — What to plot (means, lines, data, barchart, splines); default mean
XFREPRESENTATION = *string token* — How to label the *x*-axis (levels, labels); default labels uses the XFACTOR labels, if available
PSE = *string token* — What to plot to represent variation (differences, lsd, means, allmeans); default diff
LSDLEVEL = *scalar* — Significance level (%) to use for least significant differences; default 5
DFSPLINE = *scalar* — Number of degrees of freedom to use when METHOD=splines
YTRANSFORM = *string tokens* — Transformed scale for additional axis marks and labels to be plotted on the right-hand side of the y-axis (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default iden i.e. none
PENYTRANSFORM = *scalar* — Pen to use to plot the transformed axis marks and labels; default * selects a pen, and defines its properties, automatically
†KEYMETHOD = *string token* — What to use for the key descriptions when GROUPS specifies more than one factor (labels, namesandlabels); default name
†PLOTTITLEMETHOD = *string token* — What to use for the titles of the plots when TRELLISGROUPS specifies more than one factor (labels, namesandlabels); default name
†PAGETITLEMETHOD = *string token* — What to use for the titles of the pages when PAGEGROUPS specifies more than one factor (labels, namesandlabels); default name
†USEAXES = *string token* — Which aspects of the current axis definitions of window 1 to use (none, limits, marks, mpositions, nsubticks,); default none
SAVE = *ANOVA* or *regression save structure* — Save structure to provide the table of means; default uses the save structure from the most recent ANOVA

Parameters

XFACTOR = *factors* — Factor providing the *x*-values for each plot
GROUPS = *factors* or *pointers* — Factor or factors identifying groups of points in each plot; by default chosen automatically
TRELLISGROUPS = *factors* or *pointers* — Factor or factors specifying the different plots of a trellis plot of a multi-way table
PAGEGROUPS = *factors* or *pointers* — Factor or factors specifying plots to be displayed on different

pages

NEWXLEVELS = *variates* — Values to be used for XFACTOR instead of its existing levels

TITLE = *texts* — Title for the graph; default defines a title automatically

YTITLE = *texts* — Title for the y-axis; default is to use the identifier of the y-variate, or to have no title if this is unnamed

XTITLE = *texts* — Title for the x-axis; default is to use the identifier of the XFACTOR

PENS = *variates* — Defines the pen to use to plot the points and/or line for each group defined by the GROUPS factors

AGRCRESOLVABLE directive

Forms doubly resolvable row-column designs.

Options

PLOTORDER = *string token* — Defines the order in which the pots are formed into replicates (colserpentine, colbycol, rowserpentine, rowbyrow); default rowb

TIME = *scalar* — Time in seconds to spend searching for an optimal design; default 60

SEED = *scalar* — Seed for the randomization; default 0

MAXITERATIONS = *scalar* — The number of random designs to search for an optimal design; default 10000

Parameters

NROWS = *scalars* — Number of rows in the design

NCOLUMNS = *scalars* — Number of columns in the design

LEVELS = *scalar*, *variate* or *text* — Defines the number of levels or labels of the TREATMENT factor for each design

TREATMENTS = *factors* — Saves the treatment allocation in each design

ROWREPLICATES = *factors* — Saves the row replicates in each design

COLREPLICATES = *factors* — Saves the column replicates in each design

ROWS = *factors* — Saves the row locations of the plots in each design

COLUMNS = *factors* — Saves the column locations of the plots in each design

EXIT = *scalars* — Saves the exit code from the design search program (0 for success, greater than 0 for failure)

AGREFERENCE procedure

Generates reference-level designs e.g. for microarray experiments (R.W. Payne).

Option

PRINT = *string token* — Controls whether or not to print a plan of the design (design); if unset in an interactive run AGREFERENCE will ask whether the design is to be printed, in a batch run the default is not to print the design

Parameters

LEVELS = *scalars* — Number of treatments

REFLEVEL = *scalars*, *variates* or *pointers* — Reference level(s); if this is unset in an interactive run you will be asked which reference level or levels you want, in a batch run the default is level 1

REFUNIT = *scalars*, *variates* or *pointers* — Unit(s) to which to allocate the reference level(s); if this is unset in an interactive run you will be asked which reference level or levels you want, in a batch run the default is to choose the unit at random within each block

SEED = *scalars* — Seed for randomization; a negative value implies no randomization

TREATMENTS = *factors* — Identifier for the treatment factor

BLOCKS = *factors* — Identifier for the block (plate) factor

UNITS = *factors*	Identifier for the factor for the units within each block (or colours in a microarray experiment)
STATEMENT = *texts*	Saves a command to recreate the design (useful if the design information has been specified in response to questions from AGREFERENCE)

AGSEMILATIN procedure

Generates semi-Latin squares (W. van den Berg).

Options

PRINT = *string token*	Controls whether or not to print a plan of the design (design); if unset in an interactive run AGSEMILATIN will ask whether the design is to be printed, in a batch run the default is not to print anything
METHOD = *string token*	Method to use to construct the semi-Latin square (Trojan, interleaving, inflated); if unset in an interactive run AGSEMILATIN will ask what type is required, in a batch run the default is Trojan
ANALYSE = *string token*	Controls whether or not to analyse the design, and produce a skeleton analysis-of-variance table using ANOVA (no, yes); default is to ask if this is unset in an interactive run, and not to analyse if it is unset in a batch run

Parameters

NROWS = *scalars*	Number of rows and columns of the semi-Latin square
NUNITS = *scalars*	Number of units (i.e. treatments) within each block
SEED = *scalars*	Seed for randomization; a negative value implies no randomization
TREATMENTS = *factors*	Identifier for the treatment factor
ROWS = *factors*	Identifier for the row factor
COLUMNS = *factors*	Identifier for the column factor
UNITS = *factors*	Identifier for the unit factor
PSEUDOFACTOR = *factors*	Identifier for the pseudo-factor
STATEMENT = *texts*	Saves a command to recreate the design (useful if the design information has been specified in response to questions from AGSEMILATIN)

AGSPACEFILLINGDESIGN procedure

Generates space filling designs (B.M. Parker).

Options

PRINT = *string tokens*	Controls whether to print the design and its properties (design, properties, monitor); default * i.e. none
METHOD = *string token*	Specifies the method to use (latinhypercube, random, quasirandom); default rand
AUGMENT = *string token*	Indicates whether to augment an existing design (yes, no); default no
CENTRED = *string token*	For the Latin hypercube method, determines whether the design should be centred (yes, no); default no
CRITERION = *string token*	For the Latin hypercube method, determines which criterion should be used to assess space filling; (none, L2, maximin, entropy); default none
QRSEQUENCE = *string token*	Specifies which sequence to use with the quasi-random method; (sobol, niederreiter, faure); default sobol
NUNITS = *scalars*	Specifies the number of design points
NDIMENSIONS = *scalars*	Specifies the number of dimensions of each of the design points
NTIMES = *scalars*	Specifies the number of times to run the ESE algorithm; default 10

DISCREPANCY = *scalars* Saves the discrepancy of the design
SEED = *scalars* Seed to be used to randomize each design; default 0

Parameter

X = *pointer to variates* A pointer to a set of variates, each variate representing a column of the design matrix

AGSQLATTICE procedure

Generates square lattice or lattice square designs (R.W. Payne).

Options

PRINT = *string token* Controls whether or not to print a plan of the design (design); if unset in an interactive run AGSQLATTICE will ask whether the design is to be printed, in a batch run the default is not to print the design

ANALYSE = *string token* Controls whether or not to analyse the design, and produce a skeleton analysis-of-variance table using ANOVA (no, yes); default is to ask if this is unset in an interactive run, and not to analyse if it is unset in a batch run

DESIGNTYPE = *string token* What type of design to form (squarelattice, latticesquare); default squa

Parameters

LEVELS = *scalars* Number of treatments in each design
NREPLICATES = *scalars* Number of replicates in each design, taken by default to be the maximum number available in a batch run
SEED = *scalars* Seed for randomization; a negative value implies no randomization
TREATMENTS = *factors* Identifier for the treatment factor for each design
PSEUDOFACTORS = *pointers* Identifier for the pseudofactors required if the design is not a balanced lattice
REPLICATES = *factors* Identifier for the replicate factor for each design
BLOCKS = *factors* Identifier for the factor to index the blocks within replicates of a square lattice
ROWS = *factors* Identifier for the factor to index the rows within replicates of a lattice square
COLUMNS = *factors* Identifier for the factor to index the columns within replicates of a lattice square
UNITS = *factors* Identifier for the factor to index the units (or plots) within the blocks of a square lattice
STATEMENT = *texts* Saves a command to recreate the design (useful if the design information has been specified in response to questions from AGSQLATTICE)
EXCLUDEREPLICATES = *scalars* or *variates* Replicates to exclude during randomization

AKAIKEHISTOGRAM procedure

Prints histograms with improved definition of groups (A. Keen).

Options

CHANNEL = *scalar* Channel number of output file; default is the current output file
TITLE = *text* General title; default 'Histogram of ...', where ... is the identifier of the structure specified by DATA
LOWER = *scalar* Lowest class limit
WIDTH = *scalar* Interval width
SCALE = *scalar* Number of units represented by each symbol; default 1 (or more if the page width is not sufficient)

Parameters

DATA = *identifiers* Data for the histograms (variate, table, factor or matrix)
NOBSERVATIONS = *tables* One-way table to save numbers in the groups

GROUPS = *factors*	Factor to save groups defined, with LEVELS the midpoints of the intervals and LABELS as LEVELS, but as text-vector
SYMBOLS = *texts*	Characters to be used to represent the bars of each histogram
DESCRIPTION = *texts*	Annotation for key

AKEEP directive

Copies information from an ANOVA analysis into Genstat data structures.

Options

FACTORIAL = *scalar*	Limit on number of factors in a model term; default 3
STRATUM = *formula*	Model term of the lowest stratum to be searched for effects; default * implies the lowest stratum
SUPPRESSHIGHER = *string token*	Whether to suppress the searching of higher strata if a term is not found in STRATUM (yes, no); default no
TWOLEVEL = *string token*	Representation of effects in 2^n experiments (responses, Yates, effects); default resp
RESIDUALS = *variate*	Saves residuals from the final stratum (as in the RESIDUALS parameter of ANOVA)
FITTEDVALUES = *variate*	Saves fitted values (data values or missing value estimates, minus the residuals from the final stratum – as in the FITTEDVALUES parameter of ANOVA)
CBRESIDUALS = *variate*	Saves the sum of the residuals from all the strata
CBCREGRESSION = *variate*	Saves the estimates of the covariate regression coefficients, combining information from all the strata
CBCVCOVARIANCE = *symmetric matrix*	Saves the variance-covariance matrix of the combined estimates of the covariate regression coefficients
TREATMENTSTRUCTURE = *formula structure*	Saves the treatment formula used for the analysis
BLOCKSTRUCTURE = *formula structure*	Saves the block formula used for the analysis
AFACTORIAL = *scalar*	Saves the setting of the FACTORIAL option used in the ANOVA command that performed the analysis
WEIGHTS = *variate*	Saves the weights used in the analysis
YVARIATE = *dummy*	Dummy to be set to the y-variate of the analysis
LSDLEVEL = *scalar*	Significance level (%) to use in the calculation of least significant differences; default 5
AOVTABLE = *pointer*	Saves the analysis-of-variance table as a pointer with a variate or text for each column (source, d.f., s.s., m.s. etc)
EQFACTORS = *factors*	Factors whose levels are to be assumed to be equal within the comparisons between means calculated for SEMEANS
RMETHOD = *string token*	Type of residuals to form if the RESIDUALS option or parameter is set (simple, standardized); default simp
EXIT = *scalar*	Saves an exit code indicating the properties of the design
SAVE = *identifier*	Defines the Save structure (from ANOVA) that provides details of the analysis; default * gives that from the most recent ANOVA

Parameters

TERMS = *formula*	Model terms for which information is required
MEANS = *tables*	Table to store means for each term (available for treatment terms only)
SEMEANS = *tables*	Table of effective standard errors for the means, usable for calculating standard errors for differences between means in the table, at equal levels of the factors specified by the EQFACTORS option
SEDMEANS = *symmetric matrices*	Standard errors for comparisons between every pair of entries in the table of means
VCMEANS = *symmetric matrices*	Variances and covariances of means
EFFECTS = *tables* or *scalars*	Table or scalar (for terms with 1 d.f. when

	TWOLEVEL=responses or Yates) to store effects (for treatment terms only)
PARTIALEFFECTS = *tables*	Table or scalar (for terms with 1 d.f. when TWOLEVEL=responses or Yates) to store partial effects (for treatment terms only)
REPLICATIONS = *tables* or *scalars*	Table to store replications or scalar if they are all equal
RESIDUALS = *tables*	Table to store residuals (for block terms only)
DF = *scalars*	Number of degrees of freedom for each term
LSDMEANS = *symmetric matrices*	Least significant differences of means
DFMEANS = *symmetric matrices*	Degrees of freedom for comparisons between every pair of entries in the table of means
SS = *scalars*	Sum of squares for each term
EFFICIENCY = *scalars*	Efficiency factor for each term
VARIANCE = *scalars*	Unit variance for the effects of each term
RTERM = *formula structures*	Residual terms: for a treatment term this saves the lowest stratum where the term is estimated (down to the stratum specified by the STRATUM option); for a block term it saves all the strata to which it would be appropriate to compare the term
CEFFICIENCY = *scalars*	Covariance efficiency factor for each term
CREGRESSION = *variates*	Estimated regression coefficients for the covariates in the specified stratum
CSSP = *symmetric matrices*	Covariate sums of squares and products in the specified stratum
CVCOVARIANCE = *symmetric matrix*	Variance-covariance matrix of the covariate regression coefficients in the specified stratum
CONTRASTS = *pointers*	Estimates for the fitted contrasts of each treatment term, stored in a pointer to scalars or tables; units of the pointer are labelled by the contrast name (as used in the analysis-of-variance table)
XCONTRASTS = *pointers*	X-variates used to fit contrasts, as orthogonalized by ANOVA, stored in a pointer to tables; units of the pointer are labelled as for CONTRASTS
SECONTRASTS = *pointers*	Standard errors for estimated contrasts, stored in a pointer to scalars or tables; units of the pointer are labelled as for CONTRASTS
DFCONTRASTS = *pointers*	Degrees of freedom for estimated contrasts, stored in a pointer to scalars; units of the pointer are labelled as for CONTRASTS
CBMEANS = *tables*	Table to store estimates of the means, combining information from all the strata (for treatment terms only)
SECBMEANS = *tables*	Table of standard errors for the combined means, usable for calculating standard errors for differences between means in the table, at equal levels of the factors specified by the EQFACTORS option
SEDCBMEANS = *symmetric matrices*	Standard errors for comparisons between every pair of entries in the table of combined means
VCCBMEANS = *symmetric matrices*	Variances and covariances of combined means
LSDCBMEANS = *symmetric matrices*	Least significant differences of combined means
DFCBMEANS = *symmetric matrices*	Effective degrees of freedom for comparisons between every pair of entries in the table of combined means
CBEFFECTS = *tables* or *scalars*	Table or scalar (for terms with 1 d.f. when TWOLEVEL=responses or Yates) to store estimates of the effects, combining information from all the strata (for treatment terms only)
CBVARIANCE = *scalars*	Unit variance for the combined estimates of the effects of each term
DFCEFFECTS = *scalars*	Effective degrees of freedom for the combined estimates of the effects of each term
CBCEFFICIENCY = *scalars*	Covariance efficiency factor for the combined estimates of

	each term
STRATUMVARIANCE = *scalars*	Estimates of the stratum variances (for block terms only)
COMPONENT = *scalars*	Stratum variance components (for block terms only)
STATUS = *scalars*	Status code describing how the term is estimated (together with its marginal terms, if the term is a treatment term)

AKEY procedure

Generates values for treatment factors using the design key method (R.W. Payne).

Options

PRINT = *string token*	Allows the generated TREATMENTFACTOR values to be printed, tabulated by the BLOCKFACTORS (design); default * i.e. no printing
BLOCKFACTORS = *factors*	Defines the block factors for the design; default is to take those in the formula already specified by the BLOCKSTRUCTURE directive, in the order in which they occur there
KEY = *matrix*	Matrix (number of treatment factors × number of block factors) key for the design
BASEVECTOR = *variate*	Base vector (length = number of treatment factors) for the design; default is a variate of zeros
ROWPRIMES = *variate*	Prime numbers for the rows of the KEY matrix
COLPRIMES = *variate*	Prime numbers for the columns of the KEY matrix
ROWMAPPINGS = *variate*	Mappings from the rows of the KEY to the TREATMENTFACTORS
COLMAPPINGS = *variate*	Mappings from the columns of the KEY to the BLOCKFACTORS

Parameter

TREATMENTFACTORS = *factors*	Defines the treatment factors for the design; default is to take those in the formula already specified by the TREATMENTSTRUCTURE directive, in the order in which they occur there

ALIAS procedure

Finds out information about aliased model terms in analysis of variance (R.W. Payne).

Options

TREATMENTSTRUCTURE = *formula*	Treatment model for the design
BLOCKSTRUCTURE = *formula*	Block model for the design
FACTORIAL = *scalar*	Value used in the FACTORIAL option of ANOVA if not the default
DESIGN = *pointer*	Design structure for the analysis

Parameter

TERM = *factors*	Factors defining the aliased model term

ALIGNCURVE procedure

Forms an optimal warping to align an observed series of observations with a standard series (D.B. Baird).

Options

PRINT = *string tokens*	What to print (criterion, ss, warps); default * i.e. nothing
PLOT = *string tokens*	What to plot (series, warping); default * i.e. no plots
WARPPENALTY = *scalar*	The relative penalty to add to the criterion when jumping a unit in one series but not the other; default 1
MAXSTEP = *scalar*	The largest jump that can be made between the two series at a single point; default 1
MAXDIFFERENCE = *scalar*	Sets a limit on size of difference between the series to be squared and added to the criterion (differences greater than this are truncated to MAXDIFFERENCE, thus allowing the effects of outliers to be down-weighted); default * i.e. no limit
USEMEANS = *string token*	Whether to use the means of points covered in one step, rather

than the final value, when calculating the sums of squares between the two series (yes, no); default no

FORCEENDALIGNMENT = *string token* Whether to force the ends of the two series to align, so that warping happens only in the middle of the series (yes, no); default no

WINDOW = *scalar* Window number for the plots; default 1

KEYWINDOW = *scalar* Window for the key (zero for no key); default 2

Parameters

Y = *variates* Series to be aligned with the standard series

STANDARD = *variates* Standard series for each Y

WEIGHTS = *variates* Weights for the contribution of each point to the criterion; default * no weighting

UWARP = *variates* The warped positions of the unit numbers, required to align Y with STANDARD

YWARP = *variates* The warped series for Y, i.e. the optimally aligned y-values

CRITERIONVALUE = *scalars* The criterion value (as optimized during the alignment)

TITLE = *text* Title for the plots

ALLDIFFERENCES procedure

Shows all pairwise differences of values in a variate or table (A.R.G. McLachlan).

Options

PRINT = *string token* What to print (differences); default diff

CLPRINT = *string token* How to print column labels (labels, integers); default labe

SORT = *string token* How to sort the DATA values (ascending, descending); default * i.e. not sorted

MVREMOVE = *string token* Whether to remove missing values (yes, no); default no

RCMETHOD = *string token* Which differences to calculate i.e. column−row, row−column, or absolute values (column, row, absolute); default colu

DIAGONAL = *string token* Whether to put the data values into the diagonal of the symmetric matrices of results (values); default * i.e. diagonal left as missing values

Parameters

DATA = *variates* or *tables* Data values whose pairwise differences are required

DIFFERENCES = *symmetric matrices* or *pointers* Saves the pairwise differences in a symmetric matrix if GROUPS is unset, otherwise in a pointer to several symmetric matrices

GROUPS = *factors* or *pointers* Defines groupings of the data values

LABELS = *texts* Labels for the rows (and columns) of the symmetric matrices of differences

NEWLABELS = *texts* or *pointers* Saves the row labels of the symmetrix matrices of differences in a text if GROUPS is unset, otherwise in a pointer to several texts

ALLPAIRWISE procedure

Performs a range of all pairwise multiple comparison tests (D.M. Smith).

Options

METHOD = *string token* Test to be performed (Tukey, SNK, REGWMR, Duncan, Scheffe, FPLSD, FULSD, Bonferroni, Sidak); default *

DIRECTION = *string token* How to sort means (ascending, descending); default asce

PROBABILITY = *scalar* The required significance level; default=0.05

ALSD = *string token* Whether to use the alternative LSD test where the Studentized Range statistic is used instead of Student's t (yes, no); default no

Parameters

MEANS = *variates* or *tables*	Mean values
REPLICATIONS = *scalars* or *tables* or *variates*	Number(s) of observations per mean
VARIANCE = *scalars*	Estimate of variance
DF = *scalars*	Degrees of freedom
LABELS = *texts*	Identifiers of mean values

AMCOMPARISON procedure

Performs pairwise multiple comparison tests for ANOVA means (D.M. Smith).

Options

PRINT = *string tokens*	Controls printed output (comparisons, critical, description, lines, letters, plot, mplot, pplot); default lett
METHOD = *string token*	Test to be performed (tukey, snk, regwmr, duncan, scheffe, fplsd, fulsd, bonferroni, sidak); default fplsd
FACTORIAL = *scalar*	Limit on the number of factors in each term; default 3
DIRECTION = *string token*	How to sort means (ascending, descending); default asce
PROBABILITY = *scalar*	The required significance level; default 0.05
STUDENTIZE = *string token*	Whether to use the alternative LSD test where the Studentized Range statistic is used instead of Student's t (yes, no); default no
SAVE = *ANOVA save structure*	Save structure to provide the tables of means and associated information; default uses the save structure from the most recent ANOVA

Parameters

TERMS = *formula*	Treatment terms whose means are to be compared
MEANS = *pointer* or *variate*	Saves the (sorted) means
LABELS = *pointer* or *text*	Saves labels for the (sorted) means
LETTERS = *pointer* or *text*	Saves letters indicating groups of means that do not differ significantly
SIGNIFICANCE = *pointer* or *symmetric matrix*	Indicators to show significant comparisons between (sorted) means

AMDUNNETT procedure

Forms Dunnett's simultaneous confidence interval around a control (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (interval); default inte
METHOD = *string token*	Form of the alternative hypothesis (twosided, greaterthan, lessthan); default twos
CIPROBABILITY = *scalar*	Probability level for the confidence interval; default 0.95, i.e. a 95% confidence interval
LOWER = *scalar*	Saves the lower confidence limit
UPPER = *scalar*	Saves the upper confidence limit
SAVE = *ANOVA save structure*	Save structure to provide the means; default uses the save structure from the most recent ANOVA

Parameters

FACTOR = *factors*	Define the model term whose means are to be compared
CONTROL = *scalars* or *texts*	Scalar or single-valued text for each factor to identify which of the means of the term is the control; default uses the reference level of the FACTOR

AMERGE procedure

Merges extra units into an experimental design (R.W. Payne).

Option

SORT = *string token*	Whether to sort the factors afterwards (no, yes); default no

Parameters

FACTOR = *factors*	Factors to which the new units are to be added
NEWUNITS = *factors*, *variates* or *scalars*	Extra units to be added to each factor

AMMI procedure

Allows exploratory analysis of genotype × environment interactions (M. Talbot, K. Brown & M.F. Smith).

Options

PRINT = *string tokens*	Results to be output (aovtable, genotype, environment, estimates, envtable, cluster, stability); default * i.e. none
NROOTS = *scalar*	Number of IPCA scores required; default is to take as many roots as possible up to a maximum of 9
DIMENSIONS = *scalars*	Two numbers specifying the dimensions to display in the biplots; default 1,2
PLOT = *string tokens*	Types of biplot to display (mean, ipca); default * i.e. none
SCALING = *string token*	Scaling to use for the ipca (AMMI2) biplot (genotype, environment, symmetric); default envi

Parameters

DATA = *variates* or *tables*	Provides the data to be analysed
GENOTYPES = *factors*	Specifies the genotypes
ENVIRONMENTS = *factors*	Specifies the environments
REPLICATES = *factors*	Replication factor; this should be omitted if the data comprises just the genotype by environment means
GSCORES = *pointers*	Pointer containing a set of variates (each of length equal to the number of genotypes) to save the genotype IPCA scores
ESCORES = *pointers*	Pointer to a set of variates to save the environment IPCA scores
RESIDUALS = *variates*	Saves the residuals from the AMMI model
FITTEDVALUES = *variates*	Saves the fitted values from the AMMI model
TITLEPREFIX = *texts*	Specifies a prefix to use for the titles of the plots
AOVTABLE = *pointers*	Saves the analysis-of-variance table
STABILITY = *variates*	Saves the AMMI stability values

AMTDISPLAY procedure

Displays further output for multitiered experiments analysed by AMTIER (C.J. Brien & R.W. Payne).

Option

PRINT = *string tokens*	Controls printed output from the analysis (aovtable, aovpseudotable, design, effects, fittedvalues); default * i.e. none

Parameter

SAVE = *pointers*	Save structure for each analysis (saved from AMTIER); if this is not set the output is from the most recent AMTIER analysis

AMTKEEP procedure

Saves information from the analysis of a multitiered design by AMTIER (C.J. Brien & R.W. Payne).

Options

RESIDUALS = *variate*	Saves the residuals
FITTEDVALUES = *variate*	Saves the fitted values
AOVTABLE = *pointer*	Saves the analysis-of-variance table

SKELETON = *string token*	Whether to save only the skeleton analysis-of-variance table (yes, no); default no
PSEUDOLINES = *string token*	Whether to include lines for pseudo-terms in the analysis-of-variance table (yes, no); default no
OMITMISSINGLINES = *string token*	Whether to omit lines of the analysis-of-variance table that contain only missing values (yes, no); default no
SAVE = *pointer*	Save structure for the analysis; if this is not set, information is saved from the most recent AMTIER analysis

No parameters

AMTIER procedure

Analyses a multitiered design by an analysis of variance specified by up to three model formulae (C.J. Brien & R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output from the analysis (aovtable, aovpseudotable, design, effects, fittedvalues); default aovt
F1 = *formula*	First model formula
F2 = *formula*	Second model formula
F3 = *formula*	Third model formula
FACTORIAL = *scalar*	Limit on the number of factors in a model term
F2BALANCETYPE = *string token*	Type of balance required for F2 (orthogonal, firstorder); default orth
F3BALANCETYPE = *string token*	Type of balance required for F3 (orthogonal, firstorder); default orth
PSEUDOTERMS = *formula structures*	Specifies pseudo-terms for terms in the F1, F2 or F3 formulae
DESIGN = *tree*	Saves or specifies details of the design and analysis
SEED = *scalar*	Seed for random numbers to generate dummy variate for determining the design; default 13579
TOLERANCE = *variate*	Tolerance for zero sweeps in dummy and y-variate analyses
DPRINT = *string tokens*	Controls debug output (setup, analysis, dummyanalysis); default * i.e. none

Parameters

Y = *variates*	Each of these contains the data values for an analysis
RESIDUALS = *variates*	Saves the residuals from each analysis
FITTEDVALUES = *variates*	Saves the fitted values from each analysis
SAVE = *pointers*	Save structure for each analysis (to use in AMTDISPLAY)

ANOVA directive

Analyses y-variates by analysis of variance according to the model defined by earlier BLOCKSTRUCTURE, COVARIATE, and TREATMENTSTRUCTURE statements.

Options

PRINT = *string tokens*	Output from the analyses of the y-variates, adjusted for any covariates (aovtable, information, covariates, effects, residuals, contrasts, means, cbeffects, cbmeans, stratumvariances, %cv, missingvalues); default aovt, info, cova, mean, miss
UPRINT = *string tokens*	Output from the unadjusted analyses of the y-variates (aovtable, information, effects, residuals, contrasts, means, cbeffects, cbmeans, stratumvariances, %cv, missingvalues); default * i.e. no printing
CPRINT = *string tokens*	Output from the analyses of the covariates, if any (aovtable, information, effects, residuals, contrasts, means, %cv, missingvalues); default * i.e. no printing
FACTORIAL = *scalar*	Limit on number of factors in a treatment term; default 3

CONTRASTS = *scalar*	Limit on the order of a contrast of a treatment term; default 4
DEVIATIONS = *scalar*	Limit on the number of factors in a treatment term for the deviations from its fitted contrasts to be retained in the model; default 9
PFACTORIAL = *scalar*	Limit on number of factors in printed tables of means or effects; default 9
PCONTRASTS = *scalar*	Limit on order of printed contrasts; default 9
PDEVIATIONS = *scalar*	Limit on number of factors in a treatment term whose deviations from the fitted contrasts are to be printed; default 9
FPROBABILITY = *string token*	Printing of probabilities for variance ratios (yes, no); default no
PSE = *string token*	Standard errors to be printed with tables of means, PSE=* requests s.e.'s to be omitted (differences, lsd, means); default diff
TWOLEVEL = *string token*	Representation of effects in 2^n experiments (responses, Yates, effects); default resp
DESIGN = *pointer*	Stores details of the design for use in subsequent analyses; default *
WEIGHTS = *variate*	Weights for each unit; default * i.e. all units with weight one
ORTHOGONAL = *string token*	Whether or not design to be assumed orthogonal notassumed, assumed, compulsory); default nota
SEED = *scalar*	Seed for random numbers to generate dummy variate for determining the design; default 12345
MAXCYCLE = *scalar*	Maximum number of iterations for estimating missing values; default 20
TOLERANCES = *variate*	Allows you to redefine the tolerances for zero used by various parts of the algorithm
NOMESSAGE = *string tokens*	Which warning messages to suppress (nonorthogonal, residual); default *
LSDLEVEL = *scalar*	Significance level (%) to use in the calculation of least significant differences; default 5
EXIT = *scalar*	Saves an exit code indicating the properties of the design

Parameters

Y = *variates*	Variates to be analysed
RESIDUALS = *variates*	Variate to save residuals for each y variate
FITTEDVALUES = *variates*	Variate to save fitted values
SAVE = *identifiers*	Save details of each analysis for use in subsequent ADISPLAY or AKEEP statements

ANTMVESTIMATE procedure

Estimates missing values in repeated measurements (M.G. Kenward & R.W. Payne).

Options

PRINT = *string tokens*	Controls output from the procedure (meanprofiles); default * i.e. none
GROUPS = *factor*	Factor indicating the plot on which each sequence of observations was made
ORDER = *scalar*	Order of ante-dependence structure (i.e. number of past times for which to adjust)

Parameters

DATA = *variates*	Observations at each time
NEWDATA = *variates*	Data variates with missing observations replaced by their estimates
MEANPROFILE = *tables*	Estimated mean profiles at each time

ANTORDER procedure

Assesses order of ante-dependence for repeated measures data (M.S. Ridout & R.W. Payne).

Options

TREATMENTSTRUCTURE = *formula*	Treatment formula for the model at each time; if this is not set, the default is taken from the setting (which must already have been defined) of the TREATMENTSTRUCTURE directive
BLOCKSTRUCTURE = *formula*	Block formula for the model at each time; if this is not set, the default is taken from any existing setting specified by the BLOCKSTRUCTURE directive and if neither has been set the design is assumed to be unstratified (i.e. to have a single error term)
MAXORDER = *scalar*	Maximum order against which to test; default is maximum possible order
FACTORIAL = *scalar*	Limit on the number of factors in a treatment term
TIME = *factor*	Indicates the time of each observation when there is a single DATA variate

Parameter

DATA = *variates*	Data observations either in a list of variates (one for each time), or a single variate (with TIME set to a factor indicating the time of each observation)

ANTTEST procedure

Calculates overall tests based on a specified order of ante-dependence (R.W. Payne & M.S. Ridout).

Options

TREATMENTSTRUCTURE = *formula*	Treatment formula for the model at each time; if this is not set, the default is taken from the setting (which must already have been defined) of the TREATMENTSTRUCTURE directive
BLOCKSTRUCTURE = *formula*	Block formula for the model at each time; if this is not set, the default is taken from any existing setting specified by the BLOCKSTRUCTURE directive and if neither has been set the design is assumed to be unstratified (i.e. to have a single error term)
ORDER = *scalar*	Number of past times for which to adjust; default is maximum possible order
FACTORIAL = *scalar*	Limit on the number of factors in a treatment term
TIME = *factor*	Indicates the time of each observation when there is a single DATA variate

Parameter

DATA = *variates*	Data observations either in a list of variates (one for each time), or a single variate (with TIME set to a factor indicating the time of each observation)

AN1ADVICE procedure

Aims to give useful advice if a design that is thought to be balanced fails to be analysed by ANOVA (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (advice, suspects); default advi
FACTORIAL = *scalar*	Limit on number of factors in a treatment term; default 3
METHOD = *string tokens*	Method to use to predict the correct pattern of replication (median, mode, proportional); default mode
WEIGHTS = *variate*	Weights for the analysis; default * i.e. all units have weight one
SUSPECTS = *variate*	Saves the numbers of the units whose factor values are suspected to be incorrect

Parameter

Y = *variates*	Data values to be analysed (this is needed only if the analysis

is to take place on a restricted set of units)

AONEWAY procedure

Performs one-way analysis of variance (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output from the analysis of variance (aovtable, information, covariates, effects, residuals, contrasts, means, cbeffects, cbmeans, stratumvariances, %cv, missingvalues, homogeneity, permutationtest); default aovt, mean, miss
GROUPS = *factor*	Defines the treatments for the analysis
COVARIATES = *variates*	Covariates (if any) for analysis of covariance
PLOT = *string tokens*	Which residual plots to provide (fittedvalues, normal, halfnormal, histogram, absresidual); default fitt, norm, half, hist
GRAPHICS = *string token*	Type of graphs (lineprinter, highresolution); default high
FPROBABILITY = *string token*	Probabilities for variance ratio (yes, no); default no
PSE = *string tokens*	Types of standard errors to be printed with the means (differences, lsd, means); default diff
LSDLEVEL = *scalar*	Significance level (%) for least significant differences; default 5
NTIMES = *scalar*	Number of random allocations to make when PRINT=perm; default 999
SEED = *scalar*	Seed for the random number generator used to make the allocations; default 0 continues from the previous generation or (if none) initializes the seed automatically

Parameters

Y = *variates*	Each of these contains the data values for an analysis
RESIDUALS = *variates*	Saves the residuals from each analysis
FITTEDVALUES = *variates*	Saves the fitted values from each analysis

AOVANYHOW procedure

Performs analysis of variance using ANOVA, regression or REML as appropriate (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output from the analysis (aovtable, information, means, residuals); default aovt, info, mean
METHOD = *string token*	Whether to complete the analysis or just form a recommendation (analyse, recommend); default anal
FACTORIAL = *scalar*	Limit on number of factors in a treatment term; default 3
FPROBABILITY = *string token*	Printing of probabilities for variance ratios in the analysis-of-variance table (yes, no); default no
PLOT = *string tokens*	Which residual plots to provide (fittedvalues, normal, halfnormal, histogram); default * i.e. none
COMBINATIONS = *string token*	Factor combinations for which to form predicted means (present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment to be made when predicting means (marginal, equal, observed); default marg
WEIGHTS = *variate*	Weights for each unit; default * i.e. all units with weight one
PSE = *string tokens*	Types of standard errors to be printed with the predicted means (differences, alldifferences, lsd, alllsd, means; default diff
LSDLEVEL = *scalar*	Significance level (%) for least significant differences; default 5

EFLOSS = *scalar* — Maximum loss of efficiency occurring on any treatment contrast if the analysis is done by regression

EFLIMIT = *scalar* — Limit on the loss of efficiency for the analysis to be done by regression; default 0.1

EXIT = *scalar* — Exit code indicating the recommended method of analysis

Parameters

Y = *variates* — Data values to be analysed

RESIDUALS = *variates* — Variate to save the residuals from each analysis

FITTEDVALUES = *variates* — Variate to save the fitted values from each analysis

SAVE = *identifiers* — To save details of each analysis to use subsequently with the AOVDISPLAY procedure

AOVDISPLAY procedure

Provides further output from an analysis by AOVANYHOW (R.W. Payne).

Options

PRINT = *string tokens* — Controls printed output from the analysis (aovtable, information, means, residuals); default aovt, info, mean

FPROBABILITY = *string token* — Printing of probabilities for variance ratios in the analysis-of-variance table (yes, no); default no

PLOT = *string tokens* — Which residual plots to provide (fittedvalues, normal, halfnormal, histogram); default * i.e. none

COMBINATIONS = *string token* — Factor combinations for which to form predicted means (present, estimable); default esti

ADJUSTMENT = *string token* — Type of adjustment to be made when predicting means (marginal, equal, observed); default marg

PSE = *string tokens* — Types of standard errors to be printed with the predicted means (differences, alldifferences, lsd, alllsd, means; default diff

LSDLEVEL = *scalar* — Significance level (%) for least significant differences; default 5

EFLOSS = *scalar* — Maximum loss of efficiency occurring on any treatment contrast if the analysis is done by regression

EXIT = *scalar* — Code indicating the method of analysis

Parameters

SAVE = *identifiers* — Save structure from AOVANYHOW; default uses the save structure from the most recent AOVANYHOW analysis

†APAPADAKIS directive

Analysis of variance with an added Papadakis covariate, formed from neighbouring residuals (D.B. Baird).

Options

PRINT = *string tokens* — Output from the analysis of the y-variates, adjusted for covariates (aovtable, information, covariates, effects, residuals, contrasts, means, cbeffects, cbmeans, stratumvariances, %cv, missingvalues); default aovt, info, cova, mean, miss

PLOT = *string token* — Whether to plot the residuals against the average of neighbouring residuals (residuals); default * i.e. no plot

NEIGHBOURS = *string token* — The neighbours whose residuals are averaged to form the residual covariate (adjacent, rows, columns, all); default adja

TREATMENTSTRUCTURE = *formula* — Defines the treatment structure of the model; default given by the most recent TREATMENTSTRUCTURE directive

BLOCKSTRUCTURE = *formula* — Defines the blockings structure of the model; default given by the most recent BLOCKSTRUCTURE directive

COVARIATE = *variates*	Specifies any covariates in addition to the residual (Papadakis) covariate; default given by the most recent COVARIATE directive
FACTORIAL = *scalar*	Limit on number of factors in a treatment term; default 3
CONTRASTS = *scalar*	Limit on the order of a contrast of a treatment term; default 4
DEVIATIONS = *scalar*	Limit on the number of factors in a treatment term for the deviations from its fitted contrasts to be retained in the model; default 9
PSE = *string token*	Standard errors to be printed with tables of means, PSE=* requests s.e.'s to be omitted (differences, lsd, means); default diff
LSDLEVEL = *scalar*	Significance level (%) to use in the calculation of least significant differences; default 5

Parameters

Y = *variates*	Variates to be analysed
ROWS = *factors* or *variates*	Factor giving the row location of each plot
COLUMNS = *factors* or *variates*	Factor giving the column location of each plot
UNITS = *factors* or *variates*	Factor giving the plot location of each unit
RCOVARIATE = *variates*	Saves the covariate formed from the mean of the neighbouring residuals
TITLE = *texts*	Title for the graph; default i.e. title created from the Y variate name and the neighbouring plots that are used
WINDOW = *scalars*	Window number for the graph; default 3
PEN = *scalars*, *variates* or *factors*	Pen number for the graph; default 1
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea

APERMTEST procedure

Does random permutation tests for analysis-of-variance tables (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (aovtable, critical); default aovt
PLOT = *string*	What to plot (histogram); default *
NTIMES = *scalar*	Number of permutations to make; default 999
EXCLUDE = *factors*	Factors in the block model of the design whose levels are not to be randomized
SEED = *scalar*	Seed for the random number generator used to make the permutations; default 0 continues from the previous generation or (if none) initializes the seed automatically
AOVTABLE = *pointer*	Saves the aov-table, with permutation probabilities
CRITICAL = *pointer*	Saves the aov-table, with critical values
SAVE = *ANOVA save structure*	Save structure from the analysis of variance; default uses the save structure from the most recent ANOVA

No parameters

APLOT procedure

Plots residuals from an ANOVA analysis (R.W. Payne & A.D. Todd).

Options

RMETHOD = *string token*	Type of residuals to plot (simple, standardized); default simp
INDEX = *variate* or *factor*	X-variable for an index plot; default !(1,2...)
STRATUM = *formula*	The stratum (or error term) whose residuals are to be plotted; the default is to plot the residuals from the final stratum
GRAPHICS = *string token*	What type of graphics to use (lineprinter, highresolution); default high
TITLE = *text*	Overall title for the plots; if unset, the identifier of the y-variate is used

SAVE = *ANOVA save structure*	Specifies the analysis from which the residuals and fitted values are to be taken; by default they are taken from the most recent ANOVA

Parameters

METHOD = *string tokens*	Type of residual plot (fittedvalues, normal, halfnormal, histogram, absresidual, index); default fitt, norm, half, hist
PEN = *scalars*, *variates* or *factors*	Pen(s) to use for each plot

APOLYNOMIAL procedure

Forms the equation for a polynomial contrast fitted by ANOVA (R.W. Payne).

Options

PRINT = *string token*	Whether to print the equation of the polynomial (equation); default equa
SAVE = *ANOVA save structure*	Save structure (from ANOVA) to provide details of the analysis from which the equations are to be formed; default uses the save structure from the most recent ANOVA

Parameters

TERMS = *formula*	Model terms whose polynomial equations are required
COEFFICIENTS = *pointers*	Saves the coefficients of each polynomial

APOWER procedure

Calculates the power (probability of detection) for terms in an analysis of variance (R.W. Payne).

Options

PRINT = *string token*	Prints the power (power); default powe
TERM = *formula*	Treatment term to be assessed in the analysis
TREATMENTSTRUCTURE = *formula*	Treatment structure of the design; determined automatically from an ANOVA save structure if TREATMENTSTRUCTURE is unset or if SAVE is set
BLOCKSTRUCTURE = *formula*	Block structure of the design; determined automatically from an ANOVA save structure if BLOCKSTRUCTURE is unset or if SAVE is set
FACTORIAL = *scalar*	Limit on the number of factors in treatment terms; default 3
PROBABILITY = *scalar*	Significance level at which the response is required to be detected (assuming a one-sided test); default 0.05
TMETHOD = *string token*	Type of test to be made (onesided, twosided, equivalence, noninferiority, fratio); default ones
XCONTRASTS = *variate*	X-variate defining a contrast to be detected
CONTRASTTYPE = *string token*	Type of contrast (regression, comparison) default rege
SAVE = *asave*	ANOVA save structure to provide the information about the design

Parameters

RESPONSE = *scalars*, *variates* or *tables*	Size of the difference or contrast between the effects of TERM that is to be detected, or (for TMETHOD=fratio) pattern of effects or means to be detected
RMS = *scalars*	Anticipated residual mean square corresponding to TERM; can be omitted if a SAVE structure is available
POWER = *scalars* or *variates*	Saves the power (i.e. probability of detection) for RESPONSE

APPEND procedure

Appends a list of vectors of compatible types (R.W. Payne).

Options

NEWVECTOR = *variate*, *factor* or *text*	Vector to store the appended values; by default uses the first vector of the OLDVECTOR list
FREPRESENTATION = *string token*	How to match the values of old factors (levels, labels, ordinals, renumbered); default leve

GROUPS = *factor* — Factor to represent the OLDVECTOR to which each unit originally belonged

Parameter

OLDVECTOR = *variates*, *factors*, *texts* or *scalars* — Values to be appended

APRODUCT procedure

Forms a new experimental design from the product of two designs (R.W. Payne).

Options

PRINT = *string token* — Controls printing of the design (design); default desi

ANALYSE = *string token* — Whether to analyse the design by ANOVA (yes, no); default no

METHOD = *string token* — How to combine the designs (cross, nest); default nest

BF1 = *formula* — Block formula for design 1

TF1 = *formula* — Treatment formula for design 1

BF2 = *formula* — Block formula for design 2

TF2 = *formula* — Treatment formula for design 2

No parameters

ARANDOMIZE procedure

Randomizes and prints an experimental design (R.W. Payne).

Options

PRINT = *string token* — Allows the (randomized) design to be printed; (design); default *

BLOCKSTRUCTURE = *formula* — Defines the block factors according to which the randomization is to be carried out; default takes the existing specification as defined by the BLOCKSTRUCTURE directive

EXCLUDE = *factors* — (Block) factors whose levels are not to be randomized

SEED = *scalar* — Seed to generate the random numbers used to define the randomization; default 0

LPERMUTE = *string token* — Whether to randomly permute treatment factor levels (no, yes); default no

Parameters

OLDVECTOR = *factors* or *variates* — Vectors whose values are to be randomized; default is to use the factors occurring in the formula (if any) specified by the most recent TREATMENTSTRUCTURE directive

NEWVECTOR = *factors* or *variates* — Vectors to store the randomized values; by default these overwrite the values in the original vectors

†ARCSPLITPLOT procedure

Adds extra treatments onto the replicates of a resolvable row-column design, and generates factors giving the row and column locations of the plots within the design (R.W. Payne).

Options

PRINT = *strings* — Controls printed output (design, factors, layout); default * i.e. none

LEVELS = *scalar* or *variate* — Numbers of levels of the extra treatment factors; if unset, takes the numbers of levels declared for the TREATMENTFACTORS

TREATMENTFACTORS = *factors* — Extra treatment factors to be imposed onto the replicates of the original row-column design

REPLICATES = *factor* — Replicates in the modified design (after adding the extra treatments)

WHOLEPLOTS = *factor* — Whole-plots in the modified design

ROWS = *factor* — Factor indexing the rows over the whole design

COLUMNS = *factor* — Factor indexing the columns over the whole design

RCREPLICATES = *factor* — Replicates in the row-column design

RCROWS = *factor* — Rows within replicates of the row-column design

RCCOLUMNS = *factor* — Columns within replicates of the row-column design

REPLOCATIONS = *variate* or *matrix*	Locations of the replicates of the row-column design
METHOD = *string*	How to form the replicates of the modified design (rowserpentine, columnserpentine, given); default rows
SEED = *scalar*	Seed for randomizing the allocation of the extra treatments; default 0
SPREADSHEET = *string*	Whether to put the design factors into a spreadsheet (design); default *

No parameters

AREPMEASURES procedure

Produces an analysis of variance for repeated measurements (R.W. Payne).

Options

PRINT = *string tokens*	Controls output about the covariance structure (vcovariance, correlation, epsilon, test); default epsi, test
APRINT = *string tokens*	Printed output from the analysis of variance (as for the ANOVA PRINT option); default *
TREATMENTSTRUCTURE = *formula*	Defines the treatments given to the subjects; if this is not set, the default is taken from any existing setting defined by the TREATMENTSTRUCTURE directive
BLOCKSTRUCTURE = *formula*	Defines any block structure over the subjects if this is not set, the default is taken from any existing setting defined by the BLOCKSTRUCTURE directive
COVARIATE = *variates*	Specifies any covariates on the subjects if this is not set, the default is taken from any existing setting defined by the COVARIATE directive
FACTORIAL = *scalar*	Limit in the number of factors in the terms generated from the TREATMENTSTRUCTURE formula
TIMEPOINTS = *variate*, *text* or *factor*	When the DATA parameter supplies a separate variate of observations for each time this can specify numbers or labels for the time points, when there is a single DATA variate this must supply a factor to indicate the time of each observation
CONTRASTS = *scalar*	Limit on the order of a contrast of a treatment term; default 4
DEVIATIONS = *scalar*	Limit on the number of factors in a treatment term for the deviations from its fitted contrasts to be retained in the model; default 9
FPROBABILITY = *string token*	Printing of probabilities for variance ratios in the aov table (no, yes); default no
PSE = *string tokens*	Standard errors to be printed with tables of means (differences, lsd, means); default diff
MAXCYCLE = *scalar*	Maximum number of iterations for estimating missing values; default 20
LSDLEVEL = *scalar*	Significance level (%) to use in the calculation of least significant differences; default 5
EPSILON = *scalar*	Saves the correction factor epsilon
SAVEFACTORS = *pointer*	Saves the factors used in the analysis of variance
ASAVE = *identifier*	Saves the ANOVA save structure from the analysis of variance

Parameter

DATA = *variates*	Data observations either in a list of variates (one for each time), or a single variate (with TIMEPOINTS set to a factor indicating the time of each observation)

ARESULTSUMMARY procedure

Provides a summary of results from an ANOVA analysis (R.W. Payne).

Options

PRINT = *string tokens*	What to print (description, means, significant); default desc, mean, sign
PSE = *string tokens*	Standard errors to be printed with the means (sed, sedsummary, lsd, lsdsummary, dfmeans); default sed, dfme
LSDLEVEL = *scalar*	Significance level (%) for least significant differences; default 5
SAVE = *ANOVA save structure*	Save structure for the analysis; default uses the save structure from the most recent ANOVA

No parameters

ARETRIEVE procedure

Retrieves an ANOVA save structure from an external file (R.W. Payne).

No options

Parameters

FILENAME = *texts*	Name of the file storing the save structure
EXIT = *scalars*	Scalar that contains the value one if the save structure was retrieved successfully, otherwise contains either zero or a missing value
SAVE = *asave* structures	Save structure that has been retrieved

ASAMPLESIZE procedure

Finds the replication to detect a treatment effect or contrast (R.W. Payne & P. Brain).

Options

PRINT = *string tokens*	Prints the replication or produces a printed summary of the power etc. for the various amounts of replication (power, replication); default powe, repl
TERM = *formula*	Treatment term to be assessed in the analysis
REPLICATES = *factor*	Factor identifying the replication in the design
MINREPLICATION = *scalar*	Minimum number of replicates to try; default 2
MAXREPLICATION = *scalar*	Maximum feasible number of replicates; default * i.e. no limit
TREATMENTSTRUCTURE = *formula*	Treatment structure of the design; determined automatically from an ANOVA save structure if TREATMENTSTRUCTURE is unset or if SAVE is set
BLOCKSTRUCTURE = *formula*	Block structure of the design; determined automatically from an ANOVA save structure if BLOCKSTRUCTURE is unset or if SAVE is set
COMPONENTS = *variate* or *scalar*	Variate of variance components of all the terms in the block structure or, if TERM is estimated in the final stratum of the design, scalar containing only the variance component of the final stratum itself; determined automatically (if possible) from an ANOVA save structure if unset
FACTORIAL = *scalar*	Limit on the number of factors in treatment terms; default 3
PROBABILITY = *scalar*	Significance level at which the response is required to be detected (assuming a one-sided test); default 0.05
POWER = *scalar*	The required power (i.e. probability of detection) of the test; default 0.9
TMETHOD = *string token*	Type of test to be made (onesided, twosided, equivalence, noninferiority, fratio); default ones
XCONTRASTS = *variate*	X-variate defining a a contrast to be detected
CONTRASTTYPE = *string token*	Type of contrast (regression, comparison) default rege
SAVE = *asave*	ANOVA save structure to provide the information about the design

Parameters

RESPONSE = *scalars*	Size of the difference or contrast between TERM effects that is to be detected
NREPLICATES = *scalars*	Number of replicates required to detect RESPONSE

ASCREEN procedure

Performs screening tests for designs with orthogonal block structure (R.W. Payne).

Options

PRINT = *string tokens*	Which tests to print (conditional, marginal, efficiency); default cond, marg
FACTORIAL = *scalar*	Limit on the number of factors in each treatment term; default 3
EXCLUDEHIGHER = *string token*	Whether to exclude higher-order interactions in the initial model for the conditional test of each term (yes, no); default no
FORCED = *formula*	Terms that must be included (together with any covariates) in the initial models for every term; default * i.e. none

Parameter

Y = *variates*	Variates to be analysed

ASPREADSHEET procedure

Saves results from an analysis of variance in a spreadsheet (R.W. Payne).

Options

MEANS = *pointer*	Pointer to tables to contain the treatment means; default means
SEMEANS = *pointer*	Pointer to tables to contain the effective standard errors of treatment means; default ese
SEDMEANS = *pointer*	Pointer to matrices to contain standard errors of differences of treatment means; default sed
EFFECTS = *pointer*	Pointer to tables to contain the treatment effects; default effects
REPLICATIONS = *pointer*	Pointer to tables of treatment replications; default replication
RESIDUALS = *variate*	Variate to save the residuals in the fittedvalues page; default residuals
FITTEDVALUES = *variate*	Variate to save the fitted values in the fittedvalues page; default fittedvalues
AOVTABLE = *pointer*	Pointer to a text and variates containing the information in the analysis-of-variance table; default aovtable
COVINFORMATION = *pointer*	Pointer to a text and variates containing the information about the estimated covariate regression coefficients; default cov
MVINFORMATION = *pointer*	Pointer to a text and variates containing the information the about estimated missing values; default missing
EQFACTORS = *factors*	Factors whose levels are to be assumed to be equal within the comparisons between means, when calculating effective standard errors
RMETHOD = *string token*	Type of residuals to form (simple, standardized); default simp
†LSDMEANS = *pointer*	Pointer to matrices to contain least significant differences for means
†LSDLEVEL = *scalar*	Significance level (as a percentage) for the least significant differences; default 5
SPREADSHEET = *string tokens*	What to include in the spreadsheet (aovtable, covariates, effects, means, semeans, sedmeans, lsdmeans, replications, fittedvalues, missingvalues); default aovt, cova, mean, sedm, repl, fitt, miss
OUTFILENAME = *texts*	Name of Genstat workbook file (.gwb) or Excel (.xls or .xlsx) file to create

SAVE = *ANOVA save structure*	Specifies which analysis to save; default * i.e. most recent regression

No parameters

ASRULES directive

Derives association rules from transaction data.

Options

PRINT = *string tokens*	Controls printed output (rules); default rule
METHOD = *string tokens*	What to use to calculate the support of a rule (allitems, antecedent); default ante
MINSUPPORT = *scalar*	Minimum amount of support for a rule to be included; default 0.1
MINCONFIDENCE = *scalar*	Minimum amount of confidence for a rule to be included; default 0.8
MAXITEMS = *scalar*	Maximum number of items that a rule may contain; default 10
MAXRULES = *scalar*	Maximum number of rules to generate; default 100

Parameters

ITEMS = *factors*	Items in the transactions
TRANSACTIONS = *factors*	Specifies the transaction to which each each item belongs
NRULES = *scalars*	Saves the number of rules that have been derived
RULES = *pointers*	Pointer to factors, each of which saves the antecedent items and then the consequent item in one of the rules
SUPPORT = *variates*	Saves the support values for the rules
CONFIDENCE = *variates*	Saves the confidence values for the rules

ASSIGN directive

Sets elements of pointers and dummies.

Options

NSUBSTITUTE = *scalar*	Number of times *n* to substitute a dummy in order to determine which structure to assign (if *n* is negative, the assigned structure is the −*n*th from the bottom of the chain of dummies, like the NTIMES option of EXIT); default 0 i.e. no substitution
METHOD = *string token*	Whether to replace or preserve the existing value in each dummy or pointer element (replace, preserve); default repl (note, pointer elements are never unset so METHOD=preserve with a pointer simply causes the assignment to be ignored)
RENAME = *string token*	Whether to reset the default name for the structure if it has only a suffixed identifier (yes, no); default no
SCOPE = *string token*	This allows dummies or pointer elements within a procedure to be set to point to structures in the program that called the procedure (SCOPE=external) or in the main program itself (SCOPE=global) rather than to structures within the procedure (local, external, global); default loca
NSTRUCTURESUBSTITUTE = *scalar*	Number of times *n* to substitute a dummy setting of the STRUCTURE parameter in order to determine which structure to assign to the setting of the POINTER parameter (if *n* is negative, the assigned structure is the −*n*th from the bottom of the chain of dummies, like the NTIMES option of EXIT); default 0 i.e. no substitution

Parameters

STRUCTURE = *identifiers*	Values for the dummies or pointer elements
POINTER = *dummies* or *pointers*	Structure that is to point to each of those in the STRUCTURE list
ELEMENT = *scalars* or *texts*	Unit or unit label indicating which pointer element is to be set; if omitted, the first element is assumed

ASTATUS procedure

Provides information about the settings of ANOVA models and variates (R.W. Payne).

Option

PRINT = *string tokens*	Controls printed output (y, model, weights); default mode

Parameters

Y = *pointers*	Pointer of length 1 to save the identifier of the y-variate of the most recent ANOVA or that used to form INSAVE
TREATMENTSTRUCTURE = *formula structures*	Saves the current setting of TREATMENTSTRUCTURE or the setting used to form INSAVE
BLOCKSTRUCTURE = *formula structures*	Saves the current setting of BLOCKSTRUCTURE or the setting used to form INSAVE
COVARIATE = *pointers*	Saves the current COVARIATE setting or the setting used to form INSAVE
DESIGN = *pointers*	Pointer of length 1 to save the design structure in the most recent ANOVA or the one used to form INSAVE
WEIGHTS = *pointers*	Pointer of length 1 to save the identifier of the variate of weights (if any) in the most recent ANOVA or that used to form INSAVE
SAVE = *asave* structures	Saves the save structure from the most recent ANOVA
INSAVE = *asave* structures	Provides a save structure from which to save Y, TREATMENTSTRUCTURE, BLOCKSTRUCTURE and COVARIATE; default * uses the current settings

ASTORE procedure

Stores an ANOVA save structure in an external file (R.W. Payne).

No options

Parameters

FILENAME = *texts*	Name of the file to store the save structure
EXIT = *scalars*	Scalar that contains the value one if the save structure was stored successfully, otherwise contains either zero or a missing value
SAVE = *asave* structures	Save structure to be stored; default stores the save structure from the most recent ANOVA

ASWEEP procedure

Performs sweeps for model terms in an analysis of variance (R.W. Payne).

Options

TERM = *formula*	Model term (or terms) involved in the sweep (this need not be specified if EMETHOD=calculated); default is to sweep for the grand mean
EFFICIENCY = *scalar*	Efficiency factor of the term(s)
EMETHOD = *string token*	Source of the effects (calculated, given); default calc
RMETHOD = *string token*	Method to be used to obtain the residual variate (subtract, replace); default subt

Parameters

Y = *variate*	Working variates to be swept
EFFECTS = *table*	Estimated effects
RESIDUALS = *variate*	New working variates, following the sweep
SS = *scalars*	Sum of squares due to the term(s)
RSS = *scalars*	Sum of squares of the working variate after the sweep

AUDISPLAY procedure

Produces further output for an unbalanced design after AUNBALANCED (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output from the analysis (aovtable, effects, means, residuals, %cv); default aovt, mean
PFACTORIAL = *scalar*	Limit on number of factors in printed tables of predicted means; default 3
FPROBABILITY = *string token*	Printing of probabilities for variance ratios in the analysis-of-variance table (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-tests of effects (yes, no); default no
PLOT = *string tokens*	Which residual plots to provide (fittedvalues, normal, halfnormal, histogram); default * i.e. none
COMBINATIONS = *string token*	Factor combinations for which to form predicted means (present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment to be made when predicting means (marginal, equal, observed); default marg
PSE = *string tokens*	Types of standard errors to be printed with the predicted means (differences, alldifferences, lsd, alllsd, means, ese); default diff
LSDLEVEL = *scalar*	Significance level (%) for least significant differences; default 5
RMETHOD = *string token*	Type of residuals to plot (simple, standardized); default simp
PMEANTERMS = *formula*	Treatment terms for which predicted means are to be printed; default * implies all the treatment terms

Parameter

SAVE = *identifiers*	Save structure (from AUNBALANCED) containing details of the analysis for which further output is required; if omitted, output is from the most recent use of AUNBALANCED

AUGRAPH procedure

Plots tables of means from AUNBALANCED (R.W. Payne).

Options

GRAPHICS = *string token*	Type of graph (highresolution, lineprinter); default high
METHOD = *string token*	What to plot (means, lines, data, barchart, splines); default mean
XFREPRESENTATION = *string token*	How to label the *x*-axis (levels, labels); default labels uses the XFACTOR labels, if available
PSE = *string token*	What to plot to represent variation (differences, lsd, means, allmeans); default diff
COMBINATIONS = *string token*	Factor combinations for which to form predicted means (present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment to be made when predicting means (marginal, equal, observed); default marg
LSDLEVEL = *scalar*	Significance level (%) to use for least significant differences; default 5
DFSPLINE = *scalar*	Number of degrees of freedom to use when METHOD=splines
YTRANSFORM = *string tokens*	Transformed scale for additional axis marks and labels to be plotted on the right-hand side of the y-axis (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default iden i.e. none
PENYTRANSFORM = *scalar*	Pen to use to plot the transformed axis marks and labels; default * selects a pen, and defines its properties, automatically

†KEYMETHOD = *string token* — What to use for the key descriptions when GROUPS specifies more than one factor (labels, namesandlabels); default name

†PLOTTITLEMETHOD = *string token* — What to use for the titles of the plots when TRELLISGROUPS specifies more than one factor (labels, namesandlabels); default name

†PAGETITLEMETHOD = *string token* — What to use for the titles of the pages when PAGEGROUPS specifies more than one factor (labels, namesandlabels); default name

†USEAXES = *string token* — Which aspects of the current axis definitions of window 1 to use (none, limits, marks, mpositions, nsubticks,); default none

SAVE = *regression save structure* — Save structure to provide the table of means; default uses the save structure from the most recent AUNBALANCED analysis (provided no other regression analysis has been done in the interim)

Parameters

XFACTOR = *factors* — Factor providing the *x*-values for each plot

GROUPS = *factors* or *pointers* — Factor or factors identifying groups of points in each plot; by default chosen automatically

TRELLISGROUPS = *factors* or *pointers* — Factor or factors specifying the different plots of a trellis plot of a multi-way table

PAGEGROUPS = *factors* or *pointers* — Factor or factors specifying plots to be displayed on different pages

NEWXLEVELS = *variates* — Values to be used for XFACTOR instead of its existing levels

TITLE = *texts* — Title for the graph; default defines a title automatically

YTITLE = *texts* — Title for the y-axis; default is to use the identifier of the y-variate, or to have no title if this is unnamed

XTITLE = *texts* — Title for the x-axis; default is to use the identifier of the XFACTOR

PENS = *variates* — Defines the pen to use to plot the points and/or line for each group defined by the GROUPS factors

AUKEEP procedure

Saves output from analysis of an unbalanced design (by AUNBALANCED) (R.W. Payne).

Options

FACTORIAL = *scalar* — Limit on number of factors in the model terms generated from the TERMS parameter; default 3

RESIDUALS = *variate* — To save residuals from the analysis

FITTEDVALUES = *variate* — To save fitted values

COMBINATIONS = *string token* — Factor combinations for which to form predicted means (present, estimable); default esti

ADJUSTMENT = *string token* — Type of adjustment to be made when predicting means (marginal, equal, observed); default marg

LSDLEVEL = *scalar* — Significance level (as a percentage) for the least significant differences

RMETHOD = *string token* — Type of residuals to form if the RESIDUALS option is set (simple, standardized); default simp

SAVE = *identifier* — Save structure (from AUNBALANCED) containing details of the analysis for which further output is required; if omitted, output is from the most recent use of AUNBALANCED

Parameters

TERMS = *formula* — Model terms for which information is required

MEANS = *table* or *pointer* to *tables* — Predicted means for each term

SEMEANS = *table* or *pointer* to *tables* — Standard errors of the means for each term

SEDMEANS = *symmetric matrix* or *pointer* to *symmetric matrices* — Standard errors of differences between means

ESEMEANS = *table* or *pointer* to *tables* Approximate effective standard errors of the means: these are formed by procedure SED2ESE with the aim of allowing good approximations to the standard errors for differences to be calculated by the usual formula $sed_{i,j} = \sqrt{(ese_i^2 + ese_j^2)}$

LSD = *symmetric matrix* or *pointer* to *symmetric matrices*
Least significant differences

AUMCOMPARISON procedure

Performs pairwise multiple comparison tests for means from an unbalanced analysis of variance, performed previously by AUNBALANCED (D.M. Smith).

Options

PRINT = *string tokens* Controls printed output (comparisons, critical, description, lines, letters, plot, mplot, pplot); default lett

METHOD = *string token* Test to be performed (flsd, bonferroni, sidak); default flsd

FACTORIAL = *scalar* Limit on the number of factors in each term; default 3

COMBINATIONS = *string token* Factor combinations for which to form predicted means (present, estimable); default esti

ADJUSTMENT = *string token* Type of adjustment to be made when predicting means (marginal, equal, observed); default marg

WEIGHTS = *table* Weights classified by some or all of the factors in the model

DIRECTION = *string token* How to sort means (ascending, descending); default asce

PROBABILITY = *scalar* The required significance level; default 0.05

STUDENTIZE = *string token* Whether to use the alternative LSD test where the Studentized Range statistic is used instead of Student's t (yes, no); default no

SAVE = *identifier* Save structure to provide the table of means; default uses the save structure from the most recent AUNBALANCED analysis

Parameters

TERMS = *formula* Treatment terms whose means are to be compared

MEANS = *pointer* or *variate* Saves the (sorted) means

LABELS = *pointer* or *text* Saves labels for the (sorted) means

LETTERS = *pointer* or *text* Saves letters indicating groups of means that do not differ significantly

SIGNIFICANCE = *pointer* or *symmetric matrix*
Indicators to show significant comparisons between (sorted) means

AUNBALANCED procedure

Performs analysis of variance for unbalanced designs (R.W. Payne).

Options

PRINT = *string tokens* Controls printed output from the analysis (aovtable, effects, means, residuals, screen, %cv); default aovt, mean

FACTORIAL = *scalar* Limit on number of factors in a treatment term; default 3

PFACTORIAL = *scalar* Limit on number of factors in printed tables of predicted means; default 3

NOMESSAGE = *string tokens* Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default * i.e. none

FPROBABILITY = *string token* Printing of probabilities for variance ratios in the analysis-of-variance table (yes, no); default no

TPROBABILITY = *string token* Printing of probabilities for t-tests of effects (yes, no); default no

PLOT = *string tokens* Which residual plots to provide (fittedvalues, normal, halfnormal, histogram); default * i.e. none

COMBINATIONS = *string token*	Factor combinations for which to form predicted means (present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment to be made when predicting means (marginal, equal, observed); default marg
PSE = *string tokens*	Types of standard errors to be printed with the predicted means (differences, alldifferences, lsd, alllsd, means, ese); default diff
LSDLEVEL = *scalar*	Significance level (%) for least significant differences; default 5
RMETHOD = *string token*	Type of residuals to plot (simple, standardized); default simp

Parameters

Y = *variates*	Data values to be analysed
RESIDUALS = *variates*	Variate to save the residuals from each analysis
FITTEDVALUES = *variates*	Variate to save the fitted values from each analysis
SAVE = *identifiers*	To save details of each analysis to use subsequently with the AUDISPLAY procedure

AUPREDICT procedure

Forms predictions from an unbalanced design (after AUNBALANCED) (R.W. Payne).

Options

PRINT = *string tokens*	What to print (description, predictions, se, sed, sedsummary, ese, lsd, lsdsummary, vcovariance); default pred, sed
MODEL = *formula*	Model to use to calculate the predictions; default * i.e. full model fitted by AUNBALANCED
FACTORIAL = *scalar*	Limit on number of factors or variates in each term specified by MODEL; default 3
COMBINATIONS = *string token*	Factor combinations for which to form predicted means (present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment to be made when predicting means (marginal, equal, observed); default marg
PREDICTIONS = *tables* or *scalars*	Saves predictions; default *
SE = *tables* or *scalars*	Saves standard errors of predictions; default *
SED = *symmetric matrices*	Saves matrices of standard errors of differences between predictions; default *
ESE = *table*	Saves effective standard errors
LSD = *symmetric matrix*	Saves least significant differences between predictions
LSDLEVEL = *scalar*	Significance level (%) for least significant differences; default 5
VCOVARIANCE = *symmetric matrices*	Saves variance-covariance matrices of predictions; default *
SAVE = *identifier*	Save structure (from AUNBALANCED) containing details of the analysis for which predictions are required; if omitted, output is from the most recent use of AUNBALANCED

Parameters

CLASSIFY = *vectors*	Variates and/or factors to classify table of predictions
LEVELS = *variates* or *scalars*	To specify values of variates, levels of factors

AUSPREADSHEET procedure

Saves results from an analysis of an unbalanced design (by AUNBALANCED) in a spreadsheet (R.W. Payne).

Options

MEANS = *pointer*	Pointer to tables to contain the treatment means; default means
SEMEANS = *pointer*	Pointer to tables to contain the standard errors of treatment means; default sem
SEDMEANS = *pointer*	Pointer to matrices to contain standard errors of differences of

treatment means; default sed

ESEMEANS = *pointer* — Pointer to matrices to contain effective standard errors of treatment means; default ese

EFFECTS = *pointer* — Pointer to contain the estimated effects, their standard errors, t-statistics and probabilities; default effects

REPLICATIONS = *pointer* — Pointer to tables of treatment replications; default replication

RESIDUALS = *variate* — Variate to save the residuals in the fittedvalues page; default residuals

FITTEDVALUES = *variate* — Variate to save the fitted values in the fittedvalues page; default fittedvalues

COMBINATIONS = *string token* — Factor combinations for which to form predicted means (present, estimable); default esti

ADJUSTMENT = *string token* — Type of adjustment to be made when predicting means (marginal, equal, observed); default marg

AOVTABLE = *pointer* — Pointer to a text and variates containing the information in the analysis-of-variance table; default aovtable

RMETHOD = *string token* — Type of residuals to form (simple, standardized); default simp

†LSDMEANS = *pointer* — Pointer to matrices to contain least significant differences for means

†LSDLEVEL = *scalar* — Significance level (as a percentage) for the least significant differences; default 5

SPREADSHEET = *string tokens* — What to include in the spreadsheet (aovtable, effects, means, semeans, sedmeans, esemeans, lsdmeans, replications, fittedvalues); default aovt, mean, sedm, repl, fitt

OUTFILENAME = *texts* — Name of Genstat workbook file (.gwb) or Excel (.xls or .xlsx) file to create

SAVE = *identifier* — Save structure (from AUNBALANCED) containing details of the analysis for which further output is required; if omitted, output is from the most recent use of AUNBALANCED

No parameters

AU2RDA procedure

Saves results from an unbalanced analysis of variance, by AUNBALANCED, in R data frames (R.W. Payne & Z. Zhang).

Options

TERM = *formula* — Treatment term whose means, effects etc. are to be saved; must be set if any of these are to be saved, unless there is only one treatment term

COMBINATIONS = *string token* — Factor combinations for which to form predicted means (present, estimable); default esti

ADJUSTMENT = *string token* — Type of adjustment to be made when predicting means (marginal, equal, observed); default marg

LSDLEVEL = *scalar* — Significance level (%) for least significant differences and multiple comparisons; default 5

RMETHOD = *string token* — Type of residuals to form (simple, standardized); default simp

MCOMPARISON = *string token* — Method to use to make multiple comparisons between the means (flsd, fstudentizedlsd, bonferroni, sidak); default * i.e. none

SAVE = *identifier* — Save structure (from AUNBALANCED) containing details of the analysis for which further output is required; if omitted, output is from the most recent use of AUNBALANCED

Parameters

INFORMATION = *string tokens* — What to save (aovtable, effects, means, semeans,

	esemeans, sedmeans, lsdmeans, replications, fittedvalues); must be set
OUTFILENAME = *texts*	Name of the R (.rda) file to create for each set of information; must be set
COLUMNNAMES = *texts*	Specifies names for the columns in the file; if this is not set, suitable names are chosen automatically
EXIT = *scalars*	Records the exit status, 0 if the information was saved successfully, 1 otherwise

AXES directive

Defines the axes in each window for high-resolution graphics.

Options

EQUAL = *string tokens*	Whether/how to make axes equal (no, scale, lower, upper); default no
RESET = *string token*	Whether to reset the axes definitions to the default values (no, yes); default no

Parameters

WINDOW = *scalars*	Numbers of the windows
YTITLE = *texts*	Title for the y-axis in each window
XTITLE = *texts*	Title for the x-axis in each window
YLOWER = *scalars*	Lower bound for y-axis
YUPPER = *scalars*	Upper bound for y-axis
XLOWER = *scalars*	Lower bound for x-axis
XUPPER = *scalars*	Upper bound for x-axis
YMARKS = *scalars* or *variates*	Distance between each tick mark on y-axis (scalar) or positions of the marks (variate)
XMARKS = *scalars* or *variates*	Distance between each tick mark on x-axis (scalar) or positions of the marks (variate)
YMPOSITION = *string tokens*	Position of the tick marks across the y-axis (left, right, centre)
XMPOSITION = *string tokens*	Position of the tick marks across the x-axis (above, below, centre)
YLABELS = *texts*	Labels at each mark on y-axis
XLABELS = *texts*	Labels at each mark on x-axis
YLPOSITION = *string tokens*	Position of the labels for the y-axis (left, right)
XLPOSITION = *string tokens*	Position of the labels for the x-axis (above, below)
YORIGIN = *scalars*	Position on y-axis at which x-axis is drawn
XORIGIN = *scalars*	Position on x-axis at which y-axis is drawn
STYLE = *string tokens*	Style of axes (none, x, y, xy, box, grid)
PENTITLE = *scalar*	Pen to use for the title
PENAXES = *scalar*	Pen to use for the axes and their labelling
PENGRID = *scalar*	Pen to use for the grid
SAVE = *pointers*	Saves details of the current settings for the axes concerned

AXIS directive

Defines an oblique axis for high-resolution graphics.

Option

RESET = *string token*	Whether to reset the axis definition to the default values (yes, no); default no

Parameters

IDENTIFIER = *identifiers*	Name to be used inside Genstat to identify each axis
TITLE = *texts*	Title for each axis
TPOSITION = *string tokens*	Position of title (middle, end)
TDIRECTION = *string tokens*	Direction of title (parallel, perpendicular)
LOWER = *scalars*	Lower bound for each axis
UPPER = *scalars*	Upper bound for each axis

MARKS = *scalars* or *variates*	Distance between each tick mark (scalar) or positions of the marks along each axis (variate)
MPOSITION = *string tokens*	Positioning of the tick marks on each axis (inside, outside, across)
LABELS = *texts* or *variates*	Labels at each major tick mark
LPOSITION = *string tokens*	Position of the axis labels (inside, outside)
LDIRECTION = *string tokens*	Direction of the axis labels (parallel, perpendicular)
LROTATION = *scalars* or *variates*	Rotation of the axis labels
NSUBTICKS = *scalars*	Number of subticks per interval (ignored if MARKS is a variate)
XZERO = *scalars*	Position of the axis origin in the x-dimension
YZERO = *scalars*	Position of the axis origin in the y-dimension
ZZERO = *scalars*	Position of the axis origin in the z-dimension
XSTEP = *scalars*	Step in the x-direction corresponding to a step of length one along the axis
YSTEP = *scalars*	Step in the y-direction corresponding to a step of length one along the axis
ZSTEP = *scalars*	Step in the z-direction corresponding to a step of length one along the axis
PENTITLE = *scalars*	Pen to use to write the axis title
PENAXIS = *scalars*	Pen to use to draw the axis
PENLABELS = *scalar*	Pen to use to write the axis labels
ARROWHEAD = *string tokens*	Whether the axis should have an arrowhead (include, omit)
ACTION = *string tokens*	Whether to display or hide the axis (display, hide)
TRANSFORM = *string tokens*	Transformed scale for the axis marks and labels (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default iden
DECIMALS = *scalars* or *variates*	Number of decimal places to use for numbers printed at the marks
DREPRESENTATION = *scalars* or *variates*	Format to use for dates and times printed at the marks
VREPRESENTATION = *string tokens*	Format to use for numbers printed at the marks (decimal, engineering, scientific); default deci
ZEROOFFSET = *scalars*	Point on the axis corresponding to XZERO, YXERO and ZZERO
SAVE = *pointers*	Saves details of the current settings for the axis concerned

AYPARALLEL procedure

Does the same analysis of variance for several y-variates, and collates the output (R.W. Payne & D.B. Baird).

Options

PRINT = *string tokens*	Controls printed output (summary, monitoring); default * i.e. none
TREATMENTSTRUCTURE = *formula*	Treatment formula for the analysis; if this is not set, the default is taken from the setting (which must already have been defined) of the TREATMENTSTRUCTURE directive
BLOCKSTRUCTURE = *formula*	Block formula for the analysis; if this is not set, the default is taken from any existing setting specified by the BLOCKSTRUCTURE directive and if neither has been set the design is assumed to be unstratified (i.e. to have a single error term)
COVARIATE = *variates*	Defines any covariates
FACTORIAL = *scalar*	Limit on the number of factors in a treatment term
SAVETERMS = *formula*	Treatment terms for which to save information; if this is not set, information is saved for all the treatment terms
REPLICATION = *pointer*	Pointer to tables saving the replication of the SAVETERMS
SPREADSHEET = *string tokens*	What results to save in spreadsheets (aov, means, vcmeans, effects, vareffects, seeffects, contrasts,

	secontrasts, tcontrasts, prcontrasts); default * i.e. none
CONTRASTSLIMIT = *scalar*	Limit on the order of a contrast of a treatment term; default 4
DEVIATIONSLIMIT = *scalar*	Limit on the number of factors in a treatment term for the deviations from its fitted contrasts to be retained in the model; default 9

Parameters

Y = *variates* or *pointers*	Y-variates for each analysis
VFACTOR = *factors*	Identifies the individual y-variates when they are supplied in a single Y variate
RESIDUALS = *variates* or *matrices*	Saves the residuals
FITTEDVALUES = *variates* or *matrices*	Saves the fitted values
MEANS = *pointers*	Pointer to a matrix for each of the SAVETERMS, saving the means from each analysis
VCMEANS = *pointers*	Pointer to matrices saving variances and covariances for the means
EFFECTS = *pointers*	Pointer to matrices saving effects
VAREFFECTS = *pointers*	Pointer to variates saving unit variances for effects
SEEFFECTS = *pointers*	Pointer to variates saving standard errors of effects
DF = *pointers*	Pointer to variates saving degrees of freedom
SS = *pointers*	Pointer to variates saving sums of squares
MS = *pointers*	Pointer to variates saving mean squares
RDF = *pointers*	Pointer to variates saving degrees of freedom for the residual corresponding to each of the SAVETERMS
RSS = *pointers*	Pointer to variates saving residual sums of squares
RMS = *pointers*	Pointer to variates saving residual mean squares
VR = *pointers*	Pointer to variates saving variance ratios
PRVR = *pointers*	Pointer to variates saving probabilities for the variance ratios
CONTRASTS = *pointers*	Pointer to matrices saving estimates of contrasts
SECONTRASTS = *pointers*	Pointer to matrices saving standard errors of contrasts
TCONTRASTS = *pointers*	Pointer to matrices saving t-statistics for contrasts
PRCONTRASTS = *pointers*	Pointer to matrices saving probabilities for t-statistics of contrasts
OUTFILENAME = *texts*	Name of Genstat workbook file (.gwb) or Excel (.xls or .xlsx) file to create

A2DISPLAY procedure

Provides further output following an analysis of variance by A2WAY (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output from the analysis (aovtable, information, covariates, effects, residuals, means, %cv, missingvalues); default *
FPROBABILITY = *string token*	Probabilities for variance ratio (yes, no); default no
PLOT = *string tokens*	Which residual plots to provide (fittedvalues, normal, halfnormal, histogram, absresidual); default *
GRAPHICS = *string token*	Type of graphs (lineprinter, highresolution); default high
COMBINATIONS = *string token*	Factor combinations for which to form predicted means (present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment to be made when predicting means (marginal, equal, observed); default marg
PSE = *string tokens*	Types of standard errors to be printed with the means (differences, lsd, means); default diff
LSDLEVEL = *scalar*	Significance level (%) for least significant differences; default 5
RMETHOD = *string token*	Type of residuals to display (simple, standardized);

default simp

Parameter

SAVE = *pointers*	Save structure (from A2WAY) for the analysis; if omitted, output is from the most recent A2WAY analysis

A2KEEP procedure

Copies information from an A2WAY analysis into Genstat data structures (R.W. Payne).

Options

FACTORIAL = *scalar*	Sets a limit on the number of factors in the terms formed from the TERMS formula; default 2
RESIDUALS = *variate*	Saves the residuals
FITTEDVALUES = *variate*	Saves the fitted values
COMBINATIONS = *string token*	Factor combinations for which to form predicted means (present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment to be made when predicting means (marginal, equal, observed); default marg
LSDLEVEL = *scalar*	Significance level (%) for least significant differences; default 5
AOVTABLE = *pointer*	To save the analysis-of-variance table as a pointer with a variate or text for each column (source, d.f., s.s., m.s. etc)
RMETHOD = *string token*	Type of residuals to form if the RESIDUALS option is set (simple, standardized); default simp
EXIT = *scalar*	Saves an exit code indicating the properties of the design
SAVE = *pointer*	Save structure (from A2WAY) for the analysis; if omitted, output is from the most recent A2WAY analysis

Parameters

TERMS = *formula*	Specifies the treatment terms whose means &c are to be saved
MEANS = *table* or *pointer* to *tables*	Saves tables of means for the terms or pointer to tables
SEMEANS = *table* or *pointer* to *tables*	Saves approximate effective standard errors of means
SEDMEANS = *table* or *pointer* to *tables*	Saves standard errors of differences between means
LSD = *table* or *pointer* to *tables*	Saves least significant differences

A2PLOT procedure

Plots effects from two-level designs with robust s.e. estimates (Eric D. Schoen & Enrico A.A. Kaul).

Options

PRINT = *string tokens*	Which ANOVA output to print, as in ADISPLAY; default aovt, effe
CHANNEL = *scalar*	What channel to use for anova and line-printer output; default * i.e. the current output channel
FACTORIAL = *scalar*	Limit for factorial expansion of TREATMENT formula; default 3
STRATUM = *formula*	Error strata from which Yates effects are to be plotted; if unset, plots are made for all the strata
GRAPHICS = *string token*	What type of graphics (highresolution, lineprinter); default high
TITLE = *string tokens*	Separate titles for each of the plots
METHOD = *string token*	Whether to make half-Normal or Normal plots (halfnormal, normal); default half
ROBUSTNESS = *string token*	Robustness of scale estimators against contamination with active effects (low, medium, high); default medi
ALPHALEVEL = *scalar*	Type I error (0.20, 0.15, 0.10, 0.05, 0.01); default 0.05
EXCLUDE = *scalars*	How many of the largest effects to withhold from each of the half-Normal plots; default 0

Parameters

Y = *variates*	Data to be analysed
EFFECTS = *pointers*	To save a variate for each error stratum containing the (sorted) Yates effects estimated there

SE = *pointers*	To save a scalar with the standard error of the Yates effects for each error stratum
SIGNIFICANT = *pointers*	To save formulae containing the significant Yates effects in each stratum

A2RDA procedure

Saves results from an analysis of variance in R data frames (R.W. Payne & Z.Zhang).

Options

TERM = *formula*	Treatment term whose means, effects etc. are to be saved; must be set if any of these are to be saved, unless there is only one treatment term
STRATUM = *formula*	Model term of the lowest stratum to be searched for effects and contrasts; default * implies the lowest stratum
SUPPRESSHIGHER = *string token*	Whether to suppress the searching of higher strata if a term is not found in STRATUM (yes, no); default no
LSDLEVEL = *scalar*	Significance level (%) for least significant differences and multiple comparisons; default 5
EQFACTORS = *factors*	Factors whose levels are to be assumed to be equal within the comparisons between means calculated for effective standard errors of treatment means
RMETHOD = *string token*	Type of residuals to form (simple, standardized, combined); default simp
MCOMPARISON = *string token*	Method to use to make multiple comparisons between the means (tukey, regwmr, duncan, scheffe, fplsd, fulsd, fpstudentizedlsd, fustudentizedlsd, bonferroni, sidak); default * i.e. none
SAVE = *ANOVA save structure*	Specifies the analysis from which to save the results; default * i.e. most recent one

Parameters

INFORMATION = *string tokens*	What to save (aovtable, covariates, effects, cbeffects, partialeffects, contrasts, means, semeans, sedmeans, lsdmeans, dfmeans, cbmeans, secbmeans, sedcbmeans, replications, fittedvalues, missingvalues, stratumvariances, %cv, fixedcoefficients, randomcoefficients); must be set
OUTFILENAME = *texts*	Name of the R (.rda) file to create for each set of information; must be set
COLUMNNAMES = *texts*	Specifies names for the columns in the file; if this is not set, suitable names are chosen automatically
EXIT = *scalars*	Records the exit status, 0 if the information was saved successfully, 1 otherwise

A2RESULTSUMMARY procedure

Provides a summary of results from an analysis by A2WAY (R.W. Payne).

Options

PRINT = *string tokens*	What to print (description, means, significant); default desc, mean, sign
PSE = *string tokens*	Standard errors to be printed with the means (sed, sedsummary, lsd, lsdsummary, dfmeans); default sed, dfme
LSDLEVEL = *scalar*	Significance level (%) for least significant differences; default 5
SAVE = *pointer*	Save structure from A2WAY; default uses the save structure from the most recent A2WAY analysis

No parameters

A2WAY procedure

Performs analysis of variance of a balanced or unbalanced design with up to two treatment factors (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output from the analysis (aovtable, information, covariates, effects, residuals, means, %cv, missingvalues); default aovt, mean
TREATMENTS = *factors*	Defines either one or two treatment factors
BLOCKS = factor	Can specify a blocking factor e.g. for a randomized block design
COVARIATES = *variates*	Specifies any covariates
FACTORIAL = *scalar*	Can be set to 1 to fit only the main effects of the treatments factors; default 2 also fits their interaction
FPROBABILITY = *string token*	Probabilities for variance ratio (yes, no); default no
PLOT = *string tokens*	Which residual plots to provide (fittedvalues, normal, halfnormal, histogram, absresidual); default fitt, norm, half, hist
GRAPHICS = *string token*	Type of graphs (lineprinter, highresolution); default high
COMBINATIONS = *string token*	Factor combinations for which to form predicted means (present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment to be made when predicting means (marginal, equal, observed); default marg
PSE = *string tokens*	Types of standard errors to be printed with the means (differences, lsd, means); default diff
LSDLEVEL = *scalar*	Significance level (%) for least significant differences; default 5
RMETHOD = *string token*	Type of residuals to save or display (simple, standardized); default simp
MVINCLUDE = *string token*	Whether to include units with missing y-values when using ANOVA (yvariate); default * i.e. not included
EXIT = *scalar*	Saves an exit code indicating the properties of the design

Parameters

Y = *variates*	Each of these contains the data values for an analysis
RESIDUALS = *variates*	Saves the residuals from each analysis
FITTEDVALUES = *variates*	Saves the fitted values from each analysis
SAVE = *pointers*	Save structure for each analysis (to use in A2DISPLAY or A2KEEP)

BACKTRANSFORM procedure

Calculates back-transformed means with approximate standard errors and confidence intervals (V.M. Cave).

Options

PRINT = *string tokens*	Controls printed output (description, means, backmeans); default desc, back
PLOT = *string tokens*	The confidence intervals of the back-transformed means to plot (backtransformed, approximate, both); default * i.e. none
TRANSFORMATION = *string tokens*	Transformation (identity, logarithm, log10, logit, squareroot, reciprocal, power, probit, complementaryloglog,logratio, angular, arcsinesquareoot, calculated); default iden (i.e. no transformation)
CLOG = *scalar*	Constant *c* for the logarithm and log10 transformations, in form log(mean+*c*); default 0
EXPONENT = *scalar*	Exponent for power transformation; default -2

KLOGRATIO = *scalar* — Parameter *k* for logratio transformation, in form log(mean/(mean+*k*)); default 1

BACKTRANSFORMATION = *expression* — Expression, formed using argument Y, that defines the inverse of the transformation; must be specified when TRANSFORMATION = calculated

DERIVATIVE = *expression* — Expression, formed using argument Y, that defines the first derivative of the transformation; must be specified when TRANSFORMATION = calculated

CIPROBABILITY = *scalar* — Probability for the confidence intervals; default 0.95

DIRECTION = *string tokens* — Order in which the back-transformed means are plotted (ordinal, ascending, descending); default ordi

USEPENS = *string tokens* — Whether to use the current pen definitions for plotting; (yes, no); default no

WINDOW = *scalar* — Window to use for plot; default 3

Parameters

MEANS = *tables*, *variates* or *scalars* — Supplies the transformed mean(s)

SEMEANS = *tables*, *variates* or *scalars* — Supplies the standard error(s) of the transformed mean(s)

DF = *scalars* — Degrees of freedom to construct the confidence intervals; default *

DECIMALS = *scalars* — Number of decimal places for printing; default *

BACKTRANSFORMEDMEANS = *tables*, *variates* or *scalars* — Saves the back-transformed means

SEBACKTRANSFORMEDMEANS = *tables*, *variates* or *scalars* — Saves the approximate standard errors for the back-transformed means

CIAPPROXIMATE = *pointers* — Saves the approximate confidence intervals for the back-transformed means

CIBACKTRANSFORMED = *pointers* — Saves the back-transformed confidence intervals for the back-transformed means

TITLE = *texts* — Title for plot; default * i.e. none

YTITLE = *texts* — Title for y-axis; default * i.e. none

XTITLE = *texts* — Title for x-axis; default * i.e. formed automatically

BAFFYMETRIX procedure

Estimates expression values from an Affymetrix CED and CDF file (D.B. Baird).

Options

METHOD = *string token* — Method for calculating probe expression values (mas4, mas5, rma, rma2); default rma

TRANSFORMATION = *string* — How to transform the data (log2, none); default none when METHOD=mas4, otherwise log2

Parameters

CELFILES = *texts* — Affymetrix CEL files

CDFFILE = *texts* — Associated CDF file

GSHFILE = *texts* — Genstat spreadsheet file containing the estimated expression values, together with the associated slide and probe information

BANK procedure

Calculates the optimum aspect ratio for a graph (J. Ollerton & S.A. Harding).

Option

WINDOW = *scalar* — Window number; default 1

Parameters

Y = *variates* — Vertical coordinates

X = *variates* — Horizontal coordinates

ASPECTRATIO = *scalars* — Store the calculated aspect ratios

BARCHART directive

Plots bar charts in high-resolution graphics.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the bar charts; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
BARWIDTH = *scalar*, *variate* or *table*	Width(s) of the bars; default * sets equal widths to fill the x-axis
BARCOVERING = *scalar*	What proportion of the space allocated along the x-axis each bar should occupy; default * gives proportion 1 for a DATA variate, and 0.8 for a factor or table (thus giving a gap between each bar)
LABELS = *text*	Labels for the bars or groups of bars; default *
APPEND = *string token*	Whether or not the bars of the bar charts are appended together (yes, no); default no
YSCALING = *string token*	What scale to use to label the y-axis (absolute, proportion, percentage); default abso
ORIENTATION = *string token*	Direction of the plot (horizontal, vertical); default vert
OUTLINE = *string token*	Where to draw outlines (bars, perimeter); default bars
PENOUTLINE = *scalar*	Pen to use for the outlines; default -9
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key; default *
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

DATA = *tables* or *variates*	Heights of the bars in each bar chart
ERRORBARS = *scalars*, *tables* or *variates*	Heights of error bars plotted above the bars of each bar chart; default 0 i.e. none
LOWERERRORBARS = *scalars*, *tables* or *variates*	Heights of error bars plotted below the bars of each bar chart; if any of these is omitted, the corresponding setting of ERRORBARS is used as the default so that the error bars will have equal heights above and below the bars of the bar chart
GROUPS = *factors*	Which factor of a 2-way table to use as the groups factor; default uses the second classifying factor
PEN = *scalars*, *tables* or *variates*	Pen number(s) for each bar chart; default * uses pens 2, 3, and so on for the successive structures specified by DATA
PENERRORBARS = *scalars*, *tables* or *variates*	Pen number(s) for the error bars; default -11
DESCRIPTION = *texts*	Annotation for key

BASELINE procedure

Estimates a baseline for a series of numbers whose minimum value is drifting. (D.B. Baird).

Options

PLOT = *string token*	Whether to plot the series and the fitted baseline (baseline); default * i.e. no plot
BANDWIDTH = *scalar*	Bandwidth for the moving minimum; default 50
WINDOW = *scalar*	Window number for the plot; default 1
KEYWINDOW = *scalar*	Window for the key (zero for no key); default 2

Parameters

Y = *variates*	Series whose baseline is to be estimated
NEWY = *variates*	Saves the y-values corrected to a zero baseline
BASELINE = *variates*	Saves the estimated baseline

TITLE = *text*	Title for the plot

BASSESS directive

Assesses potential splits for regression and classification trees.

Options

Y = *variate* or *factor*	Response variate for a regression tree, or factor specifying the groupings for a classification tree
SELECTED = *dummy*	Returns the identifier of X variate or factor used in the best split
TESTSPLIT = *expression structure*	Logical expression representing the best split
MAXSPLITPOINT = *scalar* or *variate*	When SELECTED is a variate or a factor with ordered levels this returns a scalar containing the boundary between the two splits, when the SELECTED is a factor with unordered levels it returns a variate containing the levels allocated to the first split
MAXCRITERION = *scalar*	Maximum value obtained for the selection criterion
NOSELECTION = *scalar*	Returns the value 1 if no split has been selected, otherwise 0
FMETHOD = *string token*	Selection method to use when Y is a factor (Gini, MPI); default Gini
ANTIENDCUTFACTOR = *string token*	Anti-end-cut factor to use when Y is a factor (classnumber, reciprocalentropy); default * i.e. none
WEIGHTS = *variate*	Weights; default * i.e. all weights 1
TOLERANCE = *scalar*	Tolerance multiplier used e.g. to check for equality of x-values; default * i.e. set automatically for the implementation concerned

Parameters

X = *variates* or *factors*	Variables available to make the split
ORDERED = *string tokens*	Whether factor levels are ordered (yes, no); default no
SPLITPOINT = *scalars* or *variates*	Saves details of the best split found for each X variable; when X is a variate or a factor with ordered levels this returns a scalar containing the boundary between the two splits, when the X is a factor with unordered levels it returns a variate containing the levels allocated to the first split
CRITERIONVALUE = *scalars*	Saves the value of the selection criterion for the best split found for each X variable

BBINOMIAL procedure

Estimates the parameters of the beta binomial distribution (D.M. Smith).

Options

PRINT = *string tokens*	Controls printed output (estimates, loglikelihood); default esti
MAXCYCLE = *scalar*	Maximum number of iterations; default 50
TOLERANCE = *scalar*	Convergence criterion; default 10^{-5}

Parameters

RBINOMIAL = *variates*	Numerator of binomial data
NBINOMIAL = *variates*	Denominator of binomial data or scalars
MU = *scalars*	Mean, expectation of underlying beta distribution
THETA = *scalars*	Shape-determining parameter of underlying beta distribution
SEMU = *scalars*	Standard error of mu
SETHETA = *scalars*	Standard error of theta
LOGLIKELIHOOD = *scalars*	Log likelihood
NCYCLES = *scalars*	Number of iterations
EXIT = *scalars*	Indicator of faults

BCDISPLAY procedure

Displays a classification tree (R.W. Payne).

Option

PRINT = *string tokens*	Controls printed output (summary, details, indented, bracketed, labelleddiagram, numbereddiagram, graph); default * i.e. none

Parameter

TREE = *tree*	Tree to be displayed

BCFDISPLAY procedure

Displays information about a random classification forest (R.W. Payne).

Option

PRINT = *string tokens*	Controls printed output (outofbagerror, confusion, importance, orderedimportance); default * i.e. none

Parameter

SAVE = *pointers*	Save structure from BCFOREST providing information about the random forest

BCFIDENTIFY procedure

Identifies specimens using a random classification forest (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (identification); default * i.e. none
IDENTIFICATION = *scalar* or *variate*	Saves the identification of each specimen
VOTES = *matrix*	Saves numbers of the terminal nodes reached by the specimens
SAVE = *pointers*	Save structure from BCFOREST providing information about the random forest

Parameters

X = *variates* or *factors*	Explanatory variables
VALUES = *scalars*, *variates* or *texts*	Values to use for the explanatory variables; if these are unset for any variable, its existing values are used

BCFOREST procedure

Constructs a random classification forest (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (outofbagerror, confusion, importance, orderedimportance, monitoring); default outo, conf, impo
NTREES = *scalar*	Number of trees in the forest; no default – must be specified
NXTRY = *scalar*	Number of X variables to select at random at each node from which to choose the X variable to use there; default is the square root of number of X variables
NUNITSTRY = *scalar*	Number of units of the X variables to select at random to use in the construction of each tree; default is two thirds of the number of units
METHOD = *string token*	Selection criterion to use when constructing the trees (Gini, MPI); default Gini
GROUPS = *factor*	Groupings of the individuals to identify in the trees
NSTOP = *scalar*	Number of individuals in a group at which to stop selecting tests; default 5
ANTIENDCUTFACTOR = *string token*	Adaptive anti-end-cut factor to use (classnumber, reciprocalentropy); default * i.e. none
SEED = *scalar*	Seed for random numbers to select the NXTRY X-variables and NUMITSTRY units; default 0
OWNBSELECT = *string token*	Indicates whether or not your own version of the BSELECT

	procedure is to be used, as explained in the Method section (`yes`, `no`); default `no`
`OUTOFBAGERROR` = *scalar*	Saves the "out-of-bag" error rate
`CONFUSION` = *matrix*	Saves the confusion matrix
`SAVE` = *pointer*	Saves details of the forest that has been constructed

Parameters

`X` = *factors* or *variates*	X-variables available for constructing the tree
`ORDERED` = *string tokens*	Whether factor levels are ordered (`yes`, `no`); default `no`
`IMPORTANCE` = *scalars*	Saves the importance of each x-variable

`BCIDENTIFY` procedure

Identifies specimens using a classification tree (R.W. Payne).

Options

`PRINT` = *string tokens*	Controls printed output (`identification`, `transcript`); if `PRINT` is unset in an interactive run `BCIDENTIFY` will ask what you want to print, in a batch run the default is `iden`
`TREE` = *tree*	Specifies the tree
`IDENTIFICATION` = *text*	Saves the identification of each specimen
`TERMINALNODES` = *pointer*	Saves the numbers of the terminal nodes reached by each specimen
`PROBABILITIES` = *matrix*	Specimen × group matrix giving the probability that the specimens belong to each group
`MVINCLUDE` = *string token*	Whether to provide identifications for specimens with missing or unavailable values of the x-variables (`explanatory`); default `expl`

Parameters

`X` = *variates* or *factors*	Explanatory variables
`VALUES` = *scalars*, *variates* or *texts*	Values to use for the explanatory variables; if these are unset for any variable, its existing values are used

`BCKEEP` procedure

Saves information from a classification tree (R.W. Payne).

No options

Parameters

`TREE` = *trees*	Tree from which the information is to be saved
`SUMMARY` = *variates*	Saves summary information about each tree
`XVARIABLES` = *pointers*	Saves the identifiers of the x-variables in each tree

`BCLASSIFICATION` procedure

Constructs a classification tree (R.W. Payne).

Options

`PRINT` = *string tokens*	Controls printed output (`summary`, `details`, `indented`, `bracketed`, `labelleddiagram`, `numbereddiagram`, `graph`, `monitoring`); default `*` i.e. none
`METHOD` = *string token*	Selection criterion to use when constructing the tree (`Gini`, `MPI`); default `Gini`
`GROUPS` = *factor*	Groupings of the individuals in the tree
`TREE` = *tree*	Saves the tree that has been constructed
`NSTOP` = *scalar*	Number of individuals in a group at which to stop selecting tests; default 5
`ANTIENDCUTFACTOR` = *string token*	Adaptive anti-end-cut factor to use (`classnumber`, `reciprocalentropy`); default `*` i.e. none
`OWNBSELECT` = *string token*	Indicates whether or not your own version of the `BSELECT` procedure is to be used, as explained in the Method section (`yes`, `no`); default `no`

Parameter

X = *factors* or *variates*	X-variables available for constructing the tree
ORDERED = *string tokens*	Whether factor levels are ordered (yes, no); default no

BCONSTRUCT procedure

Constructs a tree (R.W. Payne).

Option

PRINT = *string token*	Whether to print monitoring information (monitoring); default * i.e. none

Parameters

TREE = *trees*	Saves the trees that have been constructed
DATA = *identifiers*	Data available for constructing the trees

BCUT directive

Cuts a tree at a defined node, discarding the nodes and information below it.

Option

RENUMBER = *string token*	Whether or not to renumber the nodes of the tree (yes, no); default no

Parameters

TREE = *trees*	Trees to be cut
NODE = *scalars*	Node at which to cut each tree
NEWTREE = *trees*	New trees with the information cut; if unspecified, the new tree replaces the original tree
CUTTREE = *trees*	Tree formed from the branches cut from the original tree
OLDNODES = *variates*	Mapping from old nodes to node numbers in a renumbered new tree (as positive numbers) or to nodes in the CUTTREE (as negative numbers)
NEWNODES = *variates*	Mapping from new node numbers in a renumbered tree to the original nodes
CUTNODES = *variates*	Mapping from node numbers in the CUTTREE tree to the original nodes

BCVALUES procedure

Forms values for nodes of a classification tree (R.W. Payne).

Options

GROUPS = *factor*	Groupings of the observations in the data set
TREE = *tree*	Tree for which predictions and accuracy values are to be formed
REPLACE = *string token*	Whether to replace the values stored in the tree (yes, no); default no
PREDICTION = *pointer*	New predictions for the nodes of the tree
ACCURACY = *pointer*	New accuracy values for the nodes of the tree
REPLICATION = *pointer*	New replication tables for the nodes of the tree

Parameter

X = *factors* or *variates*	Values of the factors or variates used in the tree for the new data set

BGIMPORT procedure

Imports MCMC output in CODA format produced by WinBUGS or OpenBUGS (D.A. Murray).

Options

INDEXFILE = *text*	Name of file containing index for output files
OUTPREFIX = *text*	Prefix name for the output files
WORKDIRECTORY = *text*	Working directory to use; default current Genstat working directory
PNAMES = *text*	Saves the names of the simulated nodes

NOUTFILES = *scalar*	Number of output files or chains to read; default 1

Parameter

SIMULATIONS = *pointers*	Saves the simulations in a list of pointers, one for each Markov chain

BGPLOT procedure

Produces plots for output and diagnostics from MCMC simulations (D.A. Murray).

Options

PRINT = *string tokens*	Controls printed output (summary); default *
PLOT = *string tokens*	Controls the type of plot (trace, density, autocorrelation, gelmanrubin); default trac
ARRANGEMENT = *string tokens*	Specifies whether to draw the plots individually or 4 to a page (single, multiple); default sing
START = *scalar*	Start iteration number for plots
END = *scalar*	End iteration number for plots
MAXLAG = *scalar*	Maximum lag for autocorrelation plots; default 50
BANDWIDTH = *scalar*	The bandwidth value to be used for the density plots.
GRMETHOD = *scalar*	Controls the method of the Gelman-Rubin diagnostic plot (gr, bgr); default bgr
BINWIDTH = *scalar*	Number of values in each bin in the Gelman-Rubin plot; default 50
USEALLSAMPLES = *text*	Whether to use all the samples for Gelman-Rubin plot, or to discard the first half of the observations (yes, no); default no

Parameter

SIMULATIONS = *pointers*	List of pointers containing simulations, one for each Markov chain

BGRAPH procedure

Plots a tree (R.W. Payne).

Option

SIZE = *scalar*	Provides a multiplier by which to scale the node labels

Parameters

TREE = *trees*	Trees to be plotted
XTERMINAL = *scalars* or *variates*	X-spacing (scalar) or x-values (variate) for the terminal nodes of each tree; default 2

BGROW directive

Adds new branches to a node of a tree.

No options

Parameters

TREE = *trees*	Trees to be extended
NODE = *scalars*	Node at which to extend each tree
NBRANCHES = *scalars*	Number of branches to add to each node; default 2
POSITION = *scalars*	Position at which to add the branches to each node; default * i.e. after all the current braches from the node
NEWNODES = *variates*	Returns the number(s) allocated to the new nodes

BGXGENSTAT procedure

Runs WinBUGS or OpenBUGS from Genstat in batch mode using scripts (D.A. Murray).

Options

PRINT = *string tokens*	Controls printed output (bugslog, nodestatistics, dic); default node
WPATH = *text*	Path specifying the location of the WinBUGS executable
WEXE = *text*	Name of the WinBUGS or OpenBUGS executable to run; default 'WinBUGS14.exe'

MODELFILE = *text*	Name of a file containing the model in WinBUGS code; the file should have an extension of .txt
DATA = *pointer*	A pointer to the data used by the WinBUGS model
IDATANAMES = *text*	A text containing the names for the data
MONITOR = *text*	The names of the variables that are to monitored
NCHAINS = *scalar*	Number of Markov chains; default 3
NBURNIN = *scalar*	Length of burn-in per chain; default 1000
NSAMPLES = *scalar*	Number of samples to run after burn-in; default 5000
THIN = *scalar*	Thinning rate where the samples from every kth iteration are stored; default 1
INAMES = *text*	The names for the initial parameters
DIC = *string token*	Whether to calculate the deviance information criterion (yes, no); default no
SEED = *scalar*	Specifies a seed to use for the random number generator in BUGS; default uses a pseudo-random number generated from the uniform distribution
WORKDIRECTORY = *texts*	Working directory to use; default current Genstat working directory
BUGS = *string token*	Whether to use WinBUGS or OpenBUGS (winbugs, openbugs); default winb
VIEWBUGS = *string token*	Whether to leave WinBUGS open after the run (yes, no); default no
CONTINUE = *string token*	Whether to continue Genstat server without waiting for WinBUGS to complete; (yes, no); default no
CODA = *string token*	Whether to save CODA files (yes, no); default no
WLOG = *text*	Name of file to save log from WinBUGS or OpenBUGS
Parameters	
INITIAL = *pointers*	List of pointers, one for each set of initial values for each Markov chain
SIMULATIONS = *pointers*	List of pointers to save simulations, one for each Markov chain

BIDENTIFY directive

Identifies specimens using a tree (R.W. Payne).

Options

TREE = *tree*	Specifies the tree
TESTELEMENT = *scalar*	Specifies which element of the pointer of information stored at each node of the tree contains the test to be done there to determine which subsequent branch to take
TERMINALNODES = *scalar*, *variate* or *pointer*	Scalar or variate saving the number or numbers of the terminal nodes reached by a single specimen, or pointer of scalars or variates saving the numbers of the terminal nodes reached by several specimens
Parameters	
X = *factors* or *variates*	Variables involved in the tests performed in the tree
VALUES = *scalars*, *variates* or *texts*	Values of the variables for the specimens to be identified

BIPLOT procedure

Produces a biplot from a set of variates (S.A. Harding).

Options

PRINT = *string tokens*	Printed output from the analysis (singular, scores); default * i.e. no output
GRAPHICS = *string token*	What sort of graphics to use (lineprinter, highresolution); default high
WINDOW = *scalar*	Window number for the graph; default 3
SCREEN = *string token*	Whether to clear the screen before plotting or to continue

plotting on the old screen (clear, keep); default clea

METHOD = *string token* — Type of analysis required (principalcomponent, variate, diagnostic); default prin

STANDARDIZE = *string tokens* — Whether to centre the configurations (at the origin), and/or to normalize them (to unit sum of squares) prior to analysis (centre, normalize); default cent, norm

LABELS = *factor* or *text* — Labels to identify the points for the individuals

VLABELS = *factor* or *text* — Labels to identify the points for the variates

NDIMENSIONS = *scalar* — Number of dimensions to save with COORDINATES and VCOORDINATES; default 2

Parameters

DATA = *pointers* — Each pointer contains a set of variates to be analysed

COORDINATES = *matrices* — Used to store the scores for the individuals

VCOORDINATES = *matrices* — Used to store the scores for the variates

BJESTIMATE procedure

Fits an ARIMA model, with forecast and residual checks (G. Tunnicliffe Wilson & S.J. Welham).

Options

PRINT = *string tokens* — Controls printed output (description, monitoring, model); default desc, moni, mode

GRAPHICS = *string token* — What type of graphics to use (lineprinter, highresolution); default high

WINDOWS = *scalar* or *variate* — Windows to be used for residual plots: a scalar N indicates that plots are to be produced on separate pages in window N (as currently defined), whereas a variate specifies four separate windows to be redefined (within the procedure) for plotting four graphs on one page; default 1

PENS = *variate* — The three pens to be used (after being defined appropriately) for drawing the plots; default !(1,2,3)

Parameters

SERIES = *variates* — Holds the time series to which the model is to be fitted

LENGTH = *scalars* or *variates* — Specifies the units to be used from each series: a scalar N indicates that the first N units of the series are to be used, a variate of length 2 gives the index of the first and last units of the subseries to be used; by default the whole series is used

ORDERS = *variates* — Variate holding the orders for the ARIMA model to be fitted to each series

PARAMETERS = *variates* — Variate specifying the initial values for the parameters (to be used by the TFIT directive)

TSM = *TSMs* — TSM to store each fitted model, also to supply values for orders and parameters if ORDERS and PARAMETERS are unset

RESIDUALS = *variates* — Variate to save the residuals from fitting the model to each series

BJFORECAST procedure

Plots forecasts of a time series using a previously fitted ARIMA (G. Tunnicliffe Wilson & S.J. Welham).

Options

PROBABILITY = *scalar* — Probability value used for forecast limits; default 0.9

GRAPHICS = *string token* — What type of graphics to use (lineprinter, highresolution); default high

WINDOW = *scalar* — Window to be used for plotting; default 1

PENS = *variate* — The three pens to be used (after being defined appropriately) for drawing the plots; default !(1,2,3)

Parameters

SERIES = *variates* — Variates holding the time series to be used for producing forecasts

LENGTH = *scalars* or *variates*	Specifies the units to be used from each series: a scalar N specifies that the first N units of the series are to be used, a variate of length 2 gives the time index of the first and last units of the subseries to be used; by default the whole series is used
TSM = *TSMs*	ARIMA model to be used for forecasting
TIMERANGE = *variates*	The first and second elements of each variate specify respectively the first and last time index, relative to the whole series, of the range to be forecast
ORIGIN = *scalars*	The time of the latest observation to be used to construct forecasts with increasing leadtimes for each series; if ORIGIN is unset, the default is to take the latest time point in the series prior to the range given by TIMERANGE, unless parameter LEADTIME is set, in which case fixed leadtime forecasts are constructed
LEADTIME = *scalars*	The fixed leadtime to be used to construct forecasts if ORIGIN is unset
FORECAST = *variates*	Save the values of the constructed forecasts
LOWER = *variates*	Save the lower limits of the forecasts
UPPER = *variates*	Save the upper limits of the forecasts
SFE = *variates*	Save the standardized forecast errors, available only for LEADTIME=1

BJIDENTIFY procedure

Displays time series statistics useful for ARIMA model selection (G. Tunnicliffe Wilson & S.J. Welham).

Options

PRINT = *string token*	Controls printed output (description); default desc
GRAPHICS = *string token*	What type of graphics to use (lineprinter, highresolution); default high
WINDOWS = *scalar* or *variate*	Windows to be used for the plots: a scalar N indicates that plots are to be produced on separate pages in window N (as currently defined), whereas a variate specifies four separate windows to be redefined (within the procedure) for plotting four graphs on one page; default 1
PENS = *variate*	The three pens to be used (after being defined appropriately) for drawing the plots; default !(1,2,3)

Parameters

SERIES = *variates*	Variates holding the time series for which the statistics are to be produced
LENGTH = *scalars* or *variates*	Specifies the units to be used from each series: a scalar N indicates that the first N units of the series are to be used, a variate of length 2 gives the index of the first and last units of the subseries to be used; by default the whole series is used

BJOIN directive

Extends a tree by joining another tree to a terminal node.

No options

Parameters

TREE = *trees*	Trees to be extended
NODE = *scalars*	Node at which to join the tree
JOINTREE = *trees*	Trees to be joined onto the tree
NEWNODES = *variates*	New node numbers allocated to each node in JOINTREE in the new tree

BKDISPLAY procedure

Displays an identification key (R.W. Payne).

Option

PRINT = *string tokens*	Controls printed output (indented, bracketed, diagram, graph); default * i.e. none

Parameter

KEY = *tree*	Key to be displayed

BKEY procedure

Constructs an identification key (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (indented, bracketed, diagram, graph); default * i.e. none
TAXONNAMES = *text*	Names of the taxa in the key; default * uses textual versions of the numbers 1, 2 onwards
GROUPS = *factor*	Groupings of the taxa, if the key is to identify the group of a specimen rather than its taxon
CRITERION = *string token*	Criterion to use to select the character to use at each node of the key (CME, CMV, GME); default GME when GROUPS is set, otherwise CME
PARTIAL = *string token*	Controls whether or not to use partial separation; (yes, no) default no
KEY = *tree*	Saves the key

Parameters

CHARACTER = *factors*	Characters available to construct the key
COST = *scalars*	Cost of each character; default 1

BKIDENTIFY procedure

Identifies specimens using a key (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (identification, transcript); if PRINT is unset in an interactive BKIDENTIFY will ask what you want to print, in a batch run the default is iden
KEY = *tree*	Specifies the key
IDENTIFICATION = *variate*	Saves the identification of each specimen
TERMINALNODE = *variate*	Saves numbers of the terminal nodes reached by the specimens

Parameter

CHARACTER = *factors*	Character values of the specimens

BKKEEP procedure

Saves information from an identification key (R.W. Payne).

No options

Parameters

KEY = *trees*	Identification key from which the information is to be saved
SUMMARY = *variates*	Saves summary information about each key
CHARACTERS = *pointers*	Saves the identifiers of the characters in each key

BLANDALTMAN procedure

Produces Bland-Altman plots to assess the agreement between two variates (A.R.G. McLachlan).

Options

PRINT = *string tokens*	Controls printed output (summary, estimates); default * i.e. none
PLOT = *string tokens*	What to plot (blandaltman, normal); default blan
DMETHOD = *string token*	Method for calculating differences (differences, ratios, %differences, percentages); default diff

LMETHOD = *string token* — Method for calculating limits of agreement when regression is not used (normaldistribution, percentile); default norm

REGMETHOD = *string tokens* — Whether to use regression to calculate bias (i.e. mean) or limits (bias, mean, limits, auto); default * i.e. none

CIPROBABILITY = *scalar* — Probability level for limits of agreement, confidence intervals and percentiles; default 0.95

LOWERLIMIT = *scalar* — Lower limit of agreement to use instead of a calculated limit

UPPERLIMIT = *scalar* — Upper limit of agreement to use instead of a calculated limit

ALPHALEVEL = *scalar* — Critical probability level used for regression when REGMETHOD=auto; default 0.05

XBLANDALTMAN = *string token* — X-values to use for the Bland-Altman plot (mean, Y1, Y2); default mean

REFERENCELINECHOICE = *string tokens* — Reference lines to plot on a Bland-Altman plot (bias, mean, limits, zero); default bias

GRAPHICS = *string token* — Type of graph (highresolution, lineprinter); default high

WINDOW = *scalar* — Window for the plot; default 3

SCREEN = *string token* — Whether to clear or keep the screen before displaying the plot (keep, clear); default clea

PENZEROLINE = *scalar* — Pen to use for the zero reference line

PENMEANLINE = *scalar* — Pen to use for the mean reference line

PENLIMITLINES = *scalar* — Pen to use for the reference lines showing limits of agreement

Parameters

Y1 = *variates* — First variate

Y2 = *variates* — Second variate

LABELS = *texts* — Labels for individual points on the Bland-Altman plot

MEANS = *variates* — Saves the means

DIFFERENCES = *variates* — Saves the differences, ratios or % differences (according to the DMETHOD option)

TITLE = *texts* — Title for the Bland-Altman plot

YTITLE = *texts* — Title for y-axis of the Bland-Altman plot

XTITLE = *texts* — Title for x-axis of the Bland-Altman plot

PEN = *scalars*, *variates* or *factors* — Pen for plotting points on the Bland-Altman plot; default 1

BLOCKSTRUCTURE directive

Defines the blocking structure of the design and hence the strata and the error terms.

No options

Parameter

formula — Block model (defines the strata or error terms for subsequent ANOVA statements)

BNTEST procedure

Calculates one- and two-sample binomial tests (D.A. Murray).

Options

PRINT = *string tokens* — Controls printed output (test, summary, confidence); default test, summ, conf

METHOD = *string token* — Type of test required (twosided, greaterthan, lessthan); default twos

TEST = *string token* — Form of the test for one-sample test (exact, normalapproximation) or for two-sample (normalapproximation, oddsratio); default norm

CIPROBABILITY = *scalar* — The probability level for the confidence interval; default 0.95

NULL = *scalar* — The value of the probability of success under the null hypothesis for the one-sample test; default 0.5

Parameters

R1 = *scalars*	Number of successes in the first sample
N1 = *scalars*	Sample size of the first sample
R2 = *scalars*	Number of successes in the second sample
N2 = *scalars*	Sample size of the second sample
STATISTIC = *scalars*	Saves the Normal approximation from the one-sample or two-sample tests, or the odds ratio
PROBABILITY = *scalars*	Saves the probability value from the one-sample or two-sample tests
LOWER = *scalars*	Saves the lower limit of the confidence interval
UPPER = *scalars*	Saves the upper limit of the confidence interval

BOOTSTRAP procedure

Produces bootstrapped estimates, standard errors and distributions (P.W. Lane).

Options

PRINT = *string token*	Controls printed output (estimates, graphs, vcovariance); default esti
DATA = *variates*, *factors* or *texts*	Data vectors from which the statistics are to be calculated; no default
AUXILIARY = *pointers*	Further sets of data vectors, each set to be resampled independently
ANCILLARY = *any type*	Other relevant information needed to calculate the statistics
NTIMES = *scalar*	Number of times to resample; default 100
SEED = *scalar*	Seed for random number generator; default continue from previous generation or use system clock
GRAPHICS = *string token*	Type of graphics (lineprinter, highresolution); default high
PROBABILITY = *scalar*	Probability level for confidence interval; default 0.95
METHOD = *string token*	What type of bootstrapping to use (random, balance, permute); default rand
BLOCKSTRUCTURE = *formula*	Block structure to use for random permutations
CIMETHOD = *string token*	What type of confidence intervals to provide (bca, percentile); default perc
VCOVARIANCE = *symmetric matrix*	Saves the variance-covariance matrix of the statistics

Parameters

LABEL = *texts*	Texts, each containing a single line, to label the statistics; default 'Statistic'
ESTIMATE = *scalars*	Saves the bootstrap mean for each statistic
SE = *scalars*	Saves the bootstrap standard error for each statistic
LOWER = *scalars*	Saves the bootstrap lower confidence limit for each statistic
UPPER = *scalars*	Saves the bootstrap upper confidence limit for each statistic
STATISTIC = *variates*	Saves the series of bootstrap estimates of each statistic
WINDOW = *scalars*	Graphical window to use for displaying bootstrap distribution for each statistic; default 4
SCREEN = *string tokens*	Whether to clear graphical frame or draw on top (clear, keep); default clea

BOXPLOT procedure

Draws box-and-whisker diagrams or schematic plots (P.W. Lane & S.D. Langton).

Options

GRAPHICS = *string token*	What type of graphics to use (highresolution, lineprinter); default high
TITLE = *text*	Title for diagram; default *
AXISTITLE = *text*	Title for axis representing data values; default *
WINDOW = *scalar*	Window in which to draw a high-resolution plot; default 4
ORIENTATION = *string token*	Orientation of plots (horizontal, vertical, across,

down); default vert

YORIENTATION = *string token* — Direction of the y-axis for horizontal plots (reverse, normal); default reve

SCREEN = *string token* — Whether to clear screen before a high-resolution plot (clear, keep); default clea

METHOD = *string token* — Type of representation of data in a high-resolution plot (boxandwhisker, schematic); default boxa

BOXTITLE = *text* — Title for axis representing different variates or groups; default *

BOXWIDTH = *string token* — Whether to relate box width to size of sample in high-resolution plot (fixed, variable); default fixe

WHISKER = *number* — Linestyle for whiskers (0...10); default 1

BAR% = *scalar* — Size of bar at the end of the whiskers, as a percentage of the box-width; default 0 (i.e. no bar)

WIDTH% = *scalar* — Width of the boxes, expressed as a percentage of the default width; default 100

Parameters

DATA = *variates* — Data to be summarized; no default

GROUPS = *factor* — Factor to divide values of a single variate into groups; default *

BOXLABELS = *texts* — Labels for individual boxes; default *, i.e. identifiers of variates or labels or levels of factor

UNITLABELS = *texts* — Labels for extreme points in schematic plot; default is to use unit labels

BOXPOSITIONS = *variates* — Positions of the boxes on the appropriate axis; default defines positions in an equal spacing

†BPCONVERT procedure

Converts bit patterns between integers, pointers of set bits and textual descriptions (R.W. Payne).

Options

PRINT = *string token* — Controls printed output (description); default desc

BITS = *text*, *variate* or *pointer* — Labels for the individual bits; default !(1...31)

SEPARATOR = *string* — Separator between the bit labels in the description; default '.'

Parameters

DATA = *scalars*, *texts* or *pointers* — Bit patterns to convert

BP = *scalars* — Bit patterns as integers

CONTENTS = *pointers* — Bits that are set in each bit pattern

DESCRIPTION = *texts* — Textual description of each bit pattern

BPRINT procedure

Displays a tree (R.W. Payne).

Option

PRINT = *string tokens* — Controls printed output (indented, bracketed, labelleddiagram, numbereddiagram); default inde

Parameter

TREE = *trees* — Trees to be displayed

BPRUNE procedure

Prunes a tree using minimal cost complexity (R.W. Payne).

Option

PRINT = *string tokens* — Controls printed output (graph, table, monitoring); default tabl

Parameters

TREE = *trees* — Trees to be pruned

ACCURACY = *pointers* — Accuracy values for the nodes of each tree; default is to use those stored with the tree

NEWTREES = *pointers*	Saves the trees generated during the pruning of each tree
RTPRUNED = *variates*	Accuracy of the pruned trees of each tree
NTERMINAL = *variates*	Number of terminal nodes in the pruned trees of each tree

BRDISPLAY procedure

Displays a regression tree (R.W. Payne).

Option

PRINT = *string tokens*	Controls printed output (summary, details, indented, bracketed, labelleddiagram, numbereddiagram, graph); default * i.e. none

Parameter

TREE = *tree*	Tree to be displayed

BREAK directive

Suspends execution of the statements in the current channel or control structure and takes subsequent statements from the channel specified.

Option

CHANNEL = *scalar*	Channel number; default 1

Parameter

expression	Logical expression controlling whether or not the break takes place

BREGRESSION procedure

Constructs a regression tree (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (summary, details, indented, bracketed, labelleddiagram, numbereddiagram, graph, monitoring); default * i.e. none
Y = *variate*	Response variate for the regression
TREE = *tree*	Saves the tree that has been constructed
MSLIMIT = *scalar*	Limit on the mean square of the observations at a node at which to stop making splits; default 0
NSTOP = *scalar*	Specifies the number of observations at a node at which to stop making splits; default 1
OWNBSELECT = *string token*	Indicates whether or not your own version of the BSELECT procedure is to be used, as explained in the Method section (yes, no); default no

Parameter

X = *variates* or *factors*	Independent variables available for constructing the tree
ORDERED = *string tokens*	Whether factor levels are ordered (yes, no); default no

BRFDISPLAY procedure

Displays information about a random regression forest (R.W. Payne).

Option

PRINT = *string tokens*	Controls printed output (outofbagerror, youtofbagestimates, importance orderedimportance); default * i.e. none

Parameter

SAVE = *pointers*	Save structure from BRFOREST providing information about the random forest

BRFOREST procedure

Constructs a random regression forest (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (outofbagerror,

	youtofbagestimates, importance, orderedimportance, monitoring); default outo, impo
Y = *variate*	Response variate for the regression
NTREES = *scalar*	Number of trees in the forest; no default – must be specified
NXTRY = *scalar*	Number of X variables to select at random at each node from which to choose the X variable to use there; default is the square root of number of X variables
NUNITSTRY = *scalar*	Number of units of the X variables to select at random to use in the construction of each tree; default is two thirds of the number of units
MSLIMIT = *scalar*	Limit on the mean square of the observations at a node at which to stop making splits; default 0
NSTOP = *scalar*	Specifies the number of observations at a node at which to stop making splits; default 1
SEED = *scalar*	Seed for random numbers to select the NXTRY X-variables and NUMITSTRY units; default 0
OWNBSELECT = *string token*	Indicates whether or not your own version of the BSELECT procedure is to be used, as explained in the Method section (yes, no); default no
OUTOFBAGERROR = *string token*	Saves the "out-of-bag" error rate
YOUTOFBAGESTIMATES = *variate*	Saves the "out-of-bag" estimates of Y
SAVE = *pointer*	Saves details of the forest that has been constructed

Parameters

X = *factors* or *variates*	X-variables available for constructing the tree
ORDERED = *string tokens*	Whether factor levels are ordered (yes, no); default no
IMPORTANCE = *scalars*	Saves the importance of each x-variable

BRFPREDICT procedure

Makes predictions using a random regression forest (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (prediction); default pred
PREDICTION = *variate*	Saves the prediction for the observations
SAVE = *pointer*	Save structure from BRFOREST providing information about the random forest

Parameters

X = *variates* or *factors*	Explanatory variables
VALUES = *scalars*, *variates* or *texts*	Values to use for the explanatory variables; if these are unset for any variable, its existing values are used

BRKEEP procedure

Saves information from a regression tree (R.W. Payne).

No options

Parameters

TREE = *trees*	Tree from which the information is to be saved
SUMMARY = *variates*	Saves summary information about each tree
XVARIABLES = *pointers*	Saves the identifiers of the x-variables in each tree

BRPREDICT procedure

Makes predictions using a regression tree (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (prediction, transcript); if PRINT is unset in an interactive run BRPREDICT will ask what you want to print, in a batch run the default is pred
TREE = *tree*	Specifies the tree
PREDICTIONS = *variate*	Saves the prediction for the observations
TERMINALNODES = *pointer*	Saves the numbers of the terminal nodes from which each

	prediction was obtained
MVINCLUDE = *string token*	Whether to provide predictions for units with missing or unavailable values of the x-variables (explanatory); default expl

Parameters

X = *variates* or *factors*	Explanatory variables
VALUES = *scalars*, *variates* or *texts*	Values to use for the explanatory variables; if these are unset for any variable, its existing values are used

BRVALUES procedure

Forms values for nodes of a regression tree (R.W. Payne).

Options

Y = *variate*	Values of the response variate for the new data set
TREE = *tree*	Tree for which predictions and accuracy values are to be formed
REPLACE = *string token*	Whether to replace the values stored in the tree (yes, no); default no
PREDICTION = *pointer*	New predictions for the nodes of the tree
ACCURACY = *pointer*	New accuracy values for the nodes of the tree
NOBSERVATIONS = *pointer*	New numbers of observarions for the nodes of the tree

Parameter

X = *variates*	Values of the x-variates for the new data set

CABIPLOT procedure

Plots results from correspondence analysis or multiple correspondence analysis (A.I. Glaser).

Options

DIMENSIONS = *scalars*	Two numbers specifying which axes of the ordinations to plot; default 1,2
PLOT = *string tokens*	Which scores to plot (rowscores, rowactive, rowpassive, colscores, colactive, colpassive); default rows, cols for correspondence analysis and cols for multiple correspondence analysis
ROWSCALING = *string token*	Scaling to use for row coordinates (principal, standard, mass, sqrtmass); default prin
COLSCALING = *string token*	Scaling to use for column coordinates (principal, standard, mass, sqrtmass); default prin
COLOURMETHOD = *string tokens*	Whether colour of symbol should show level of inertia of rows or columns (rowinertia, colinertia); default *
SIZEMETHOD = *string tokens*	Whether size of symbol should show row or column masses (rowmass, colmass); default *
FACCOLOURS = *text*, *variate* or *scalar*	Specifies a colour or colours for the factors in a multiple correspondence analysis; if this is unset, a different colour is selected automatically for every factor
WINDOW = *scalar*	Which graphical window to use; default 1
KEYWINDOW = *scalar*	Graphical window for the key
SAVE = *pointer*	Supplies results from a analysis by CORANALYSIS or MCORANALYSIS; default uses the most recent analysis

Parameters

TITLE = *texts*	Titles for the plot
LMROWVARIABLES = *string tokens*	How to label the row scores (identifiers, labels, none, numbers); default labe if LROWVARIABLES is set, otherwise iden
LMCOLVARIABLES = *string tokens*	How to label the column scores (identifiers, labels, none, numbers); default labe if LCOLVARIABLES is set, otherwise iden
LROWVARIABLES = *texts*	Labels for row variables

LCOLVARIABLES = *texts*	Labels for column variables
LSPECIES = *texts*	Labels for species scores
LSITES = *texts*	Labels for site scores

CALCULATE directive

Calculates numerical values for data structures.

Options

PRINT = *string token*	Printed output required (summary); default * i.e. no printing
ZDZ = *string token*	Value to be given to zero divided by zero (missing, zero); default miss
TOLERANCE = *scalar*	If the scalar is non missing, this defines the smallest non-zero number; otherwise it accesses the default value, which is defined automatically for the computer concerned
SEED = *scalar*	Seed to use for any random number generation during the calculation; default 0
INDEX = *scalar*	If the calculation has a list of structures before the assignment operator (=), the scalar indicates the position within the list of the structure currently being evaluated
RESTRICTEDUNITS = *variate*	Defines a "restriction" on the vectors in the expression; if this is set the calculations on those vectors will take place only on the units listed in the variate (and any restrictions of their own will be ignored)

Parameter

expression	Expression defining the calculations to be performed

CALLS directive

Lists library procedures called by a procedure.

No options

Parameter

identifiers	Names of the called procedures

CANCORRELATION procedure

Does canonical correlation analysis (P.G.N. Digby).

Option

PRINT = *string tokens*	Printed output from the analysis (correlations, pcoeff, qcoeff, pscores, qscores); default * i.e. no output

Parameters

PVARIATES = *pointers*	Pointer to P-set of variates to be analysed
QVARIATES = *pointers*	Pointer to Q-set of variates to be analysed
CORRELATIONS = *diagonal matrices*	Stores the canonical correlations from each analysis
PCOEFF = *matrices*	Stores the coefficients for the P-set of variates
QCOEFF = *matrices*	Stores the coefficients for the Q-set of variates
PSCORES = *matrices*	Stores the unit scores from the P-set of variates
QSCORES = *matrices*	Stores the unit scores from the Q-set of variates

CAPTION directive

Prints captions in standardized formats.

Option

PFIRST = *string tokens*	What to print first (dots, page, outprint); default * i.e. none

Parameters

TEXT = *texts*	Contents of the captions
STYLE = *string tokens*	Style for each caption (plaintext, stress, minor, major, meta, note, status); default plai

CASE directive

Introduces a "multiple-selection" control structure.

No options

Parameter

expression	Expression which is evaluated to an integer, indicating which set of statements to execute

CASSOCIATION procedure

Calculates measures of association for circular data (S.J. Clark).

Options

PRINT = *string token*	What to print (tests); default test
NRANDOMIZATIONS = *scalar*	Number of randomizations to use in the randomization tests; default 999
ASCALE = *string token*	Units of the circular variables (degrees, radians); default degr

Parameters

Y = *variates*	Response variable
X = *variates*	Circular explanatory variable
YTYPE = *string tokens*	Type of response variable (circular, linear); default circ
SEED = *variates*	Vraiete of length two, firstly to supply a seed for the randomization tests and secondly to supply a seed to use for randomly-selecting sets of data points; default !(0,0)
STATISTICS = *variates*	Saves the test statistics

CATALOGUE directive

Displays the contents of a backing-store file.

Options

PRINT = *string tokens*	What to print (subfiles, structures); default subf, stru
CHANNEL = *scalar*	Channel number of the backing-store file; default 0, i.e. the workfile
LIST = *string token*	How to interpret the list of subfiles (inclusive, exclusive, all); default incl
SAVESUBFILE = *text*	To save the subfile identifiers; default *
UNNAMED = *string token*	Whether to list unnamed structures (yes, no); default no

Parameters

SUBFILE = *identifiers*	Identifiers of subfiles in the file to be catalogued
SAVESTRUCTURE = *texts*	To save the identifiers of the structures in each subfile

CATRENDTEST procedure

Calculates the Cochran-Armitage chi-square test for trend (A.I. Glaser).

Option

PRINT = *string token*	Output required (test); default test

Parameters

DATA = *tables*	Table containing observed data
TREND = *factors*	Dimension of the table representing the trend; can default if only one dimension of size greater than 2
CHISQUARE = *scalars*	Saves the chi-square for trend
PROBABILITY = *scalars*	Saves the probability value for trend
DEVCHISQUARE = *scalars*	Saves the chi-square for deviations from a linear trend
DEVDF = *scalars*	Saves the degrees of freedom for the chi-square for deviations
DEVPROBABILITY = *scalars*	Saves the probability value for the chi-square for deviations

CCA procedure

Performs canonical correspondence analysis (A.I. Glaser).

Options

PRINT = *string tokens*	Controls printed output (variance, loadings, roots, evalues, evectors, speciesscores, sitescores, fitsitescores, correlations, fitcorrelations); default vari, root
NROOTS = *scalar*	Number of eigenvalues and eigenvectors to include in output; default * takes all the non-zero eigenvalues
NORMALIZE = *string tokens*	Whether to normalize the Y, X and/or Z variates to have unit sums-of-squares before the analysis (x, y, z); default x, z
SCALING = *string tokens*	Whether to scale for species or site score (species, site); default spec
TOLERANCE = *scalar*	Tolerance for detecting non-zero eigenvalues; default 10^{-5}

Parameters

Y = *pointers*	Each pointer defines a set of response variates to be modelled
X = *pointers*	Explanatory variates or factors to use for for each pointer of y-variates
Z = *pointers*	Conditioning variates to remove ("partial out") before the analysis
LRV = *LRVs*	LRV structure from each analysis, storing the eigenvectors, eigenvalues and total variance
SPECIESSCORES = *matrices*	Save the "species scores" from each analysis
SITESCORES = *matrices*	Save the "site scores" from each analysis
FITSITESCORES = *matrices*	Save the fitted "site scores" from each analysis
CORRELATIONS = *matrices*	Saves the correlations between the site scores and the x-variates
FITCORRELATIONS = *matrices*	Saves the correlations between the fitted site scores and the x-variates
SAVE = *pointers*	Save structure which provides information for use in CRBIPLOT and CRTRIPLOT

CCOMPARE procedure

Tests whether samples from circular distributions have a common mean direction or have identical distributions (S.J. Clark).

Options

PRINT = *string token*	What to print (tests); default test
TEST = *string token*	Which tests to perform (compare, identical); default comp, iden
ASCALE = *string token*	Units of the circular variables (degrees, radians); default degr
STATISTICS = *variate*	Saves the test statistics
COMMON = *scalar*	Saves the common mean direction
LOWER = *scalar*	Saves the lower 95% confidence limit for common mean
UPPER = *scalar*	Saves the upper 95% confidence limit for common mean

Parameter

DATA = *variates*	Circular response variables to be compared

CDESCRIBE procedure

Calculates summary statistics and tests of circular data (P.W. Goedhart & R.W. Payne).

Options

PRINT = *string tokens*	What to print (summary, fittedvalues); default summ
SEGMENT = *scalar*	Width of sectors (in degrees) into which to group an ANGLES variate for calculation of the test of randomness and the chi-square goodness of fit statistic for the von Mises distribution; default 20

MSEGMENT = *scalar*	Defines the centre (in degrees) of the sectors; default 0
DIRECTION = *scalar*	Direction (in degrees) of the unimodal alternative distribution for the Rayleigh test; default * i.e. not known

Parameters

ANGLES = *factors* or *variates*	Directional observations (in degrees)
RESULTS = *variates*	Saves the summary statistics
VONMISESCOUNTS = *pointers*	Saves structures relevant for calculation of the chi-square goodness of fit statistic for the von Mises distribution

CDNAUGMENTEDDESIGN procedure

Constructs an augmented block design, using CycDesigN if the controls are in an incomplete-block design (R.W. Payne).

Options

PRINT = *strings*	Controls printed output (design, controldesign, factors, monitor); default * i.e. none
LEVELS = *scalar* or *variate*	Levels for the unreplicated treatments
LEVCONTROLS = *scalar* or *variate*	Levels for the control treatments
NROWS = *scalar*	Number of rows
NCOLUMNS = *scalar*	Number of columns
NRBLOCKS = *scalar*	Number of rows in each block
NCBLOCKS = *scalar*	Number of columns in each block
NCONTROLSPERBLOCK = *scalar*	Number of control treatments in each block
TREATMENTS = *factor*	Treatment factor
ROWS = *factor*	Row factor
COLUMNS = *factor*	Column factor
BLOCKS = *factor*	Block factor
ROWBLOCKS = *factor*	Row block factor
COLBLOCKS = *factor*	Column block factor
NTIMES = *scalar*	Number of times to try allocations of controls within blocks
SEED = *scalar* or *variate*	Scalar or variate with three values specifying seeds for the random numbers used by CycDesigN to search for the control design, for the allocation of controls within blocks, and for the allocation of the unreplicated treatments – if a scalar is specified the same seed is used for all purposes; default 0 i.e. set automatically
SPREADSHEET = *string*	Whether to put the design factors into a spreadsheet (design); default *
TIMELIMIT = *scalar*	Time in minutes for CycDesigN to search; default 1

No parameters

CDNBLOCKDESIGN procedure

Constructs a block design using CycDesigN (R.W. Payne).

Options

PRINT = *strings*	Controls printed output (design, report, factors); default * i.e. none
LEVELS = *scalar* or *variate*	Numbers of levels of the treatment factors; if unset, takes the numbers of levels declared for the factors in the TREATMENTSTRUCTURE model
NREPLICATES = *scalar*	Number of replicates
NBLOCKS = *scalar*	Number of blocks
NUNITS = *scalar*	Number of units per block
NGROUPS = *variate*	Group sizes for a two-factor nested treatment structure
TREATMENTFACTORS = factors	Up to four factors to use in the treatment model: one factor for a one-way treatment model, two factors for a nested structure when NGROUPS is set, or two to four factors for a factorial treatment structure when NGROUPS is not set

REPLICATES = *factor*	Replicate factor
BLOCKS = *factor*	Block factor
UNITS = *factor*	Unit-within-block factor
RESOLVABLE = *string*	Whether the design is resolvable (yes, no); default no
ALPHADESIGN = *string*	Whether an alpha design is constructed for a resolvable design (yes, no); default no
CYCLIC = *string*	Whether a cyclic design is constructed for a non-resolvable design (yes, no); default no
NBLATIN = *scalar*	Number of contiguous blocks to latinize; default 0 i.e. not latinized
REPLATINGROUPS = *variate*	Sizes of groups defining the positions of the replicates when constructing latinized designs; default * i.e. no groupings
SPATIALMODEL = *string*	Spatial model to use with a single-treatment-factor resolvable design (integer, linearvariance, seconddifference, ev); default * i.e. none
EVDECAY = *scalar*	Decay parameter to use when SPATIALMODEL=ev; default 0.5
WEIGHTS = *variate*	Variate with two values specifying weightings for the main effects and for the interactions in factorial treatment structures; default !(1, 0.25)
SEED = *scalar* or *variate*	Scalar or variate with two values specifying seeds for the random numbers used by CycDesigN to search for the best design and to randomize it – if a scalar is specified the same seed is used for both purposes; default 0 i.e. set automatically
SPREADSHEET = *string*	Whether to put the design factors into a spreadsheet (design); default *
TIMELIMITS = *scalar* or *variate*	A scalar or a variate containing up to three numbers defining the time in minutes to spend on the first phase, the second phase and the spatial phase of the search (if the 2nd or 3rd numbers are omitted they default to the maximum of those specified); default 1
NRANDOMIZATIONS = *scalar*	Number of randomizations to generate from the best design; default 1
TRIALS = *factor*	Trials factor

No parameters

CDNPREP procedure

Constructs a multi-location partially-replicated design using CycDesigN (R.W. Payne).

Options

PRINT = *strings*	Controls printed output (design, report, factors, blocknumbers); default * i.e. none
LEVELS = *scalar*	Numbers of levels of the treatment factor; if unset, takes the numbers of levels declared for the factor specified by the TREATMENTS option
NLOCATIONS = *scalar*	Number of locations
NBLOCKS = *scalar*	Number of blocks at each location
NUNITSPERLOCATION = *scalar*	Number of units at each location
NREPLICATEDPERBLOCK = *scalar*	Number of treatments in each block that are replicated at the location containing the block
TREATMENTS = *factor*	Treatment factor
LOCATIONS = *factor*	Locations factor
BLOCKS = *factor*	Block factor
UNITS = *factor*	Unit-within-block factor
SEED = *scalar* or *variate*	Scalar or variate with two values specifying seeds for the random numbers used by CycDesigN to search for the best design and to randomize it – if a scalar is specified the same seed is used for both purposes; default 0 i.e. set automatically

SPREADSHEET = *string*	Whether to put the design factors into a spreadsheet (design); default *
TIMELIMIT = *scalar*	Time in minutes to search; default 1

No parameters

CDNROWCOLUMNDESIGN procedure

Constructs a row-column design using CycDesigN (R.W. Payne).

Options

PRINT = *strings*	Controls printed output (design, report, factors); default * i.e. none
LEVELS = *scalar* or *variate*	Numbers of levels of the treatment factors; if unset, takes the numbers of levels declared for the factors in the TREATMENTSTRUCTURE model
NREPLICATES = *scalar*	Number of replicates
NROWS = *scalar*	Number of rows
NCOLUMNS = *scalar*	Number of columns
NGROUPS = *variate*	Group sizes for a two-factor nested treatment structure
TREATMENTFACTORS = factors	Up to four factors to use in the treatment model: one factor for a one-way treatment model, two factors for a nested structure when NGROUPS is set, or two to four factors for a factorial treatment structure when NGROUPS is not set
REPLICATES = *factor*	Replicate factor
ROWS = *factor*	Row factor
COLUMNS = *factor*	Column factor
RESOLVABLE = *string*	Whether the design is resolvable (yes, no); default no
METHOD = *string*	How to construct the design (onestage, twostage, unrestrictedtwostage); default ones
NRLATIN = *scalar*	Number of contiguous rows to latinize; default 0 i.e. not latinized
NCLATIN = *scalar*	Number of contiguous columns to latinize; default 0 i.e. not latinized
REPLATINGROUPS = *variate*	Specifies the number of replicates in each column when constructing latinized designs; default * i.e. all in one column
SPATIALMODEL = *string*	Spatial model to use with a single-treatment-factor resolvable design (integer, linearvariance, seconddifference, ev); default * i.e. none
EVDECAY = *scalar*	Decay parameter to use when SPATIALMODEL=ev; default 0.5
WEIGHTS = *variate*	Variate with two values specifying weightings for the main effects and for the interactions in factorial treatment structures; default !(1, 0.25)
RCWEIGHTS = *variate*	Variate with three values specifying weightings for the within-row-and-column, between-row and between-column information; default has weight one for the within-row-and-column information, and the reciprocal of their numbers of levels for the rows and columns
SEED = *scalar* or *variate*	Scalar or variate with two values specifying seeds for the random numbers used by CycDesigN to search for the best design and to randomize it – if a scalar is specified the same seed is used for both purposes; default 0 i.e. set automatically
SPREADSHEET = *string*	Whether to put the design factors into a spreadsheet (design); default *
TIMELIMITS = *scalar* or *variate*	A scalar or a variate containing up to three numbers defining the time in minutes to spend on the first phase, the second phase and the spatial phase of the search (if the 2nd or 3rd numbers are omitted they default to the maximum of those specified); default 1

NRANDOMIZATIONS = *scalar* — Number of randomizations to generate from the best design; default 1

TRIALS = *factor* — Trials factor

No parameters

CENSOR procedure

Pre-processes censored data before analysis by ANOVA (P.W. Lane).

Options

PRINT = *string token* — Whether to monitor convergence (monitor); default * implies no monitoring

TERM = *formula* — Formula for lowest stratum residual term; no default – this option must be set

DESIGN = *pointer* — Identifier specifying design information for ANOVA, or to save design information; default *

MAXCYCLE = *scalar* — Maximum number of iterations; default 20

Parameters

Y = *variates* — Observed variate with censored values represented by values greater than or equal to the bound; no default – this parameter must be set

BOUND = *scalars* or *variates* — Upper bound for censoring for each unit; no default – this parameter must be set

DF = *scalars* — Estimated residual d.f. for lowest stratum, adjusting for censoring; default *

NEWY = *variates* — Saves a variate with the censored values replaced by their estimates; if unset, the censored values are replaced in the original Y variate

SAVE = *identifiers* — Save details of each analysis for use in subsequent ADISPLAY or AKEEP statements

CHECKARGUMENT procedure

Checks the arguments of a procedure (R.W. Payne).

Option

ERROR = *scalar* — This scalar is given the value 1 if any errors are detected; it should have the value 0 on entry

Parameters

STRUCTURE = *identifiers* — Lists the structures (arguments) to be checked

VALUES = *variates* or *texts* — Defines the allowed values for a structure of type variate or text

DEFAULT = *identifiers* — Default to be used if STRUCTURE is set to an unset dummy

SET = *texts* — Indicates whether or not each structure must be set (no, yes); default no

DECLARED = *texts* — Indicates whether or not each structure must have been declared (no, yes); default no

TYPE = *texts* — Text for each structure whose values indicate the types allowed (scalar, factor, text, variate, matrix, diagonalmatrix, symmetricmatrix, table, expression, formula, dummy, pointer, LRV, SSPM, TSM, tree, asave, rsave, tsave, vsave); default *

PRESENT = *texts* — Indicates whether or not each structure must have values (no, yes); default no

CHIPERMTEST procedure

Performs a random permutation test for a two-dimensional contingency table (L.H. Schmitt, M.C. Hannah & S.J. Welham).

Options

PRINT = *string tokens* — Output required (summary, observed, expected); default

summ

PLOT = *string token* — What to plot (histogram); default hist
METHOD = *string token* — Method for calculating chi-square (pearson, maximumlikelihood); default pear
NTIMES = *scalar* — Number of permutations to make; default 999
SEED = *scalar* — Seed for the random number generator used to make the permutations; default 0 continues from the previous generation or (if none) initializes the seed automatically

Parameters

DATA = *tables* — Table containing observed data
CHISQUARE = *scalars* — Saves the observed chi-square value
CHIPERMUTED = *variates* — Saves the chi-square values from the permuted data sets
PROBABILITY = *scalars* — Saves the probability value from the test

CHISQUARE procedure

Calculates chi-square statistics for one- and two-way tables (A.D. Todd & P.K. Leech).

Options

PRINT = *string tokens* — Output required (test, probability, fittedvalues, tchisquare); default test, prob
METHOD = *string token* — Method for calculating chi-square (pearson, maximumlikelihood); default pear
GOODNESSOFFIT = *string token* — Whether to carry out a goodness-of-fit test for the DATA values against a supplied set of FITTEDVALUES (yes, no); default no

Parameters

DATA = *tables* — Table containing observed data
CHISQUARE = *scalars* — Scalar to save the chi-square value
DF = *scalars* — Scalar to supply or save the degrees of freedom
PROBABILITY = *scalars* — Scalar to save the probability value
FITTEDVALUES = *tables* — Table of expected values
RESIDUALS = *tables* — Table of standardized residuals
TCHISQUARE = *tables* — Table whose cells show the individual contributions to the chi-square value

CINTERACTION procedure

Clusters rows and columns of a two-way interaction table (J.T.N.M. Thissen & J. de Bree).

Options

PRINT = *string tokens* — What information to print (sortedtable, aovtable, summary, monitoring, variance, amalgamations, dendrogram); default sort, aov, summ, moni, vari, amal, dend
PRMONITOR = *scalar* — If option VARIANCE is set this provides a P-value to indicate when to start monitoring, if VARIANCE is unset PRMONITOR is ignored; default 0.95
VARIANCE = *scalar* — Variance of a mean in TABLE; default *
DF = *scalar* — Degrees of freedom of VARIANCE; default *
SSTHRESHOLD = *scalar* — Specifies a value of cumSS at which to partition the dendrograms and to define factors ROWGROUPS and COLGROUPS; default 0 i.e. no partitioning
TITLE = *text* — General title for the high-resolution graph; default *
PENSIZE = *scalar* — Pen size for y-labels of dendrograms; default 1

Parameters

TABLE = *tables* — Two-way table whose interaction structure is to be clarified
ROWAMALGAMATIONS = *matrices* — To either save or specify amalgamations for rows
COLAMALGAMATIONS = *matrices* — To either save or specify amalgamations for columns
ROWPERMUTATIONS = *variates* — To specify order of labels in the row dendrogram
COLPERMUTATIONS = *variates* — To specify order of labels in the column dendrogram

ROWGROUPS = *factors*	To save the grouping of the rows specified by the SSTHRESHOLD option
COLGROUPS = *factors*	To save the grouping of the columns specified by the SSTHRESHOLD option
SORTEDTABLE = *tables*	To save the sorted TABLE with increasing row and column means

CLASSIFY procedure

Obtains a starting classification for non-hierarchical clustering (S.A. Harding).

No options

Parameters

DATA = *pointers*	Each pointer contains a set of variates giving the properties of the units to be grouped
NGROUPS = *scalars*	Indicates the number of groups required
GROUPS = *factors*	Stores the classifications formed

CLOSE directive

Closes files.

No options

Parameters

CHANNEL = *scalars* or *texts*	Numbers of the channels to which the files are attached, or identifiers of texts used for input (which, after "closing", can then be re-read)
FILETYPE = *string tokens*	Type of each file (input, output, unformatted, backingstore, procedurelibrary, graphics); default inpu
DELETE = *string tokens*	Whether to delete the file on closure (yes, no); default no

CLUSTER directive

Forms a non-hierarchical classification.

Options

PRINT = *string tokens*	Printed output required (criterion, optimum, units, typical, initial, random); default * i.e. no printing
DATA = *matrix* or *pointer*	Data from which the classification is formed, supplied as a units-by-variates matrix or as a pointer containing the variates of the data matrix
CRITERION = *string token*	Criterion for clustering (sums, predictive, within, Mahalanobis); default sums
INTERCHANGE = *string token*	Permitted moves between groups (transfer, swop); default tran (implies swop also)
START = *factor*	Initial classification; default * i.e. splits the units, in order, into NGROUPS classes of nearly equal size
NSTARTS = *scalar*	Number of starting configurations to be used; default 0
SEED = *scalar*	Seed for the random numbers used to form random starting configurations; default 0

Parameters

NGROUPS = *scalars*	Numbers of classes into which the units are to be classified: note, the values of the scalars must be in descending order
GROUPS = *factors*	Saves the classification formed for each number of classes
CRITERIONVALUE = *scalars*	Saves the criterion values (representing within-class homogeneity)
BCRITERIONVALUE = *scalars*	Saves the subsidiary criterion values (representing between-class heterogeneity for maximal predictive classification)
MEANS = *matrices*	Saves the variate means for the groups of each classification
PREDICTORS = *matrices*	Saves the group predictors from maximal predictive classification

CMHTEST procedure

Performs the Cochran-Mantel-Haenszel test (D.A. Murray).

Options

PRINT = *string token*	Controls printed output (test); default test
CLASSIFICATION = *factors*	Classifying factors for a DATA variate or classifying factors for the *R*×*C* tables in a DATA table
CONTINUITY = *string token*	Continuity correction for 2×2×*K* Mantel-Haenszel test (correct, none); default corr
CIPROBABILITY = *scalar*	Size of confidence interval for common odds ratio in 2×2×*K* tables; default 0.95

Parameters

DATA = *tables* or *variates*	Data values
STATISTIC = *scalars*	Save the test statistic
PROBABILITY = *scalars*	Save the probability for the test
ODDSRATIO = *scalars*	Save the common odds ratio for the 2×2×*K* table case
LOWER = *scalars*	Save lower limit of the confidence interval of odds ratio
UPPER = *scalars*	Save upper limit of the confidence interval of odds ratio

COKRIGE directive

Calculates kriged estimates using a model fitted to the sample variograms and cross-variograms of a set of variates.

Options

PRINT = *string token*	Controls printed output (description, search, weights, conditionalprobabilities, quantiles, crossvalidations); default desc
Y = *variate*	Variate to predict in the coKriging
METHOD = *string token*	Type of kriging (Normal, LogNormal); default Norm
X1OUTER = *variate*	Variate containing 2 values to define the bounds of the region to be examined in the first direction; by default the whole region is used
X2OUTER = *variate*	Variate containing 2 values to define the bounds of the region to be examined in the second direction; by default the whole region is used
X3OUTER = *variate*	Variate containing 2 values to define the bounds of the region to be examined in the third direction; by default the whole region is used
X1INNER = *variate*	Variate containing 2 values to define the bounds of the interpolated region in the first direction; no default
X2INNER = *variate*	Variate containing 2 values to define the bounds of the interpolated region in the second direction; no default
X3INNER = *variate*	Variate containing 2 values to define the bounds of the interpolated region in the third direction; no default
X1INTERVAL = *scalar*	Distance between successive interpolations in the first direction; default 1.0
X2INTERVAL = *scalar*	Distance between successive interpolations in the second direction; default 1.0
X3INTERVAL = *scalar*	Distance between successive interpolations in the third direction; default 1.0
POINTS = *matrix*	Allows the point where predictions are required to be specified explicitly if the X1-3INNER and X1-3INTERVAL options are unset, otherwise if these are set, saves the locations of the prediction points
BLOCKDIMENSIONS = *variate* or *matrix*	Dimensions of the block(s) in the 3 directions, a variate defines identical blocks for each prediction point, a matrix can be used to define different block sizes for each point when the

points are defined by the POINTS option; default `!(0,0,0)` i.e. punctual kriging at every point

`POOLRADIUS` = *scalar* Specifies the minimum distance for which points are pooled; default `*` i.e. no pooling

`SEARCHNEIGHBOURHOOD` = *string token* Search neighbourhood to be used (`global`, `local`); default `glob`

`MINPOINTS` = *scalars* Minimum number of data points from which to compute elements

`MAXPOINTS` = *scalars* Maximum number of data points in each direction from which to compute elements

`RADII` = *scalars* or *variates* Scalar defining the maximum distance between target point in block and usable data for each variable in 1 dimension, or radii of the ellipse or ellipsoid enclosing the usable points in 2 or 3 dimensions

`ELLIPSEAXIS` = *scalar* or *variate* Angle or angles defining the direction of the axis of the ellipse or ellipsoid, scalar for 2 dimensions and variate containing 3 values for 3 dimensions

`DRIFT` = *string token* Mean function for universal cokriging (`constant`, `linear`, `quadratic`, `polygon`); default `cons`

`X1EXV` = *variate* Variate containing locations of the explanatory model in the first dimension

`X2EXV` = *variate* Variate containing locations of the explanatory model in the second dimension (if recorded in 2 or 3 dimensions)

`X3EXV` = *variate* Variate containing locations of the explanatory model in the third dimension (if recorded in 3 dimensions)

`TERMS` = *variates* List of variates for explanatory model; default `*` i.e. none

`POLYGONCOORDINATES` = *pointer* Pointer containing the coordinates of polygons in 2 variates and the map unit numbers within a factor

`COORDSYSTEM` = *string token* Coordinate system used for the geometry for discretizing the lag (`mathematical`, `geographical`); default `math`

`CPTHRESHOLD` = *scalar or variate* Threshold(s) for calculating the conditional probabilities

`PERCENTQUANTILES` = *scalar or variate* Percentage points for which quantiles are required; default 5 and 95

`LOGBASE` = *string token* Base of antilog transformation to be applied to the predictions and variances for lognormal (co)kriging (`ten`, `e`); default `*` i.e. none

Parameters

`DATA` = *variates* Measurements as one or more variates

`X1` = *variates* Locations of the measurements in the first dimension

`X2` = *variates* Locations of the measurements in the second dimension (if recorded in 2 or 3 dimensions)

`X3` = variates Locations of the measurements in the third dimension (if recorded in 3 dimensions)

`PREDICTIONS` = *variate* Kriged estimates

`VARIANCES` = *variate* Estimation variances

`MEASUREMENTERROR` = *scalars* Variance of measurement error for punctual (co)kriging

`ESTIMATES` = *pointers* Estimates for the model structure

`CONDITIONALPROBABILITIES` = *pointers* Structure to save conditional probabilities

`QUANTILES` = *pointers* Structure to save estimated quantiles

`SAMPLESUPPORT` = *scalars* Sampling size (length, area or volume according to the dimensionality of the data) of the data points

COLOUR directive

Defines the red, green and blue intensities to be used for the Genstat colours for certain graphics devices.

Option

RESET = *string token*	Whether to reset values to their defaults (no, yes); default no

Parameters

NUMBER = *scalars*	Numbers of the colours to be set
RED = *scalars*	Red intensity of each colour (between 0 and 255)
GREEN = *scalars*	Green intensity of each colour (between 0 and 255)
BLUE = *scalars*	Blue intensity of each colour (between 0 and 255)
MATCH = *scalars*	Number of a Genstat colour to define any unset values of RED, GREEN or BLUE; default is to restore the original values of the colour
SAVE = *pointers*	Pointers each containing three scalars to save the red, green and blue intensities of the colours

COMBINE directive

Combines or omits "slices" of a multi-way data structure (table, matrix or variate).

Options

OLDSTRUCTURE = *identifier*	Structure whose values are to be combined; no default i.e. this option must be set
NEWSTRUCTURE = *identifier*	Structure to contain the combined values; no default i.e. this option must be set

Parameters

OLDDIMENSION = *factors* or *scalars*	Dimension number or factor indicating a dimension of the OLDSTRUCTURE
NEWDIMENSION = *factors* or *scalars*	Dimension number or factor indicating the corresponding dimension of the NEWSTRUCTURE; this can be omitted if the dimensions are in numerical order, while zero settings (each in conjunction with a single OLDPOSITION) allows a slice of an old table to be mapped into a new table with fewer dimensions
OLDPOSITIONS = *pointers*, *texts*, *variates* or *scalars*	These define positions in each OLDDIMENSION: pointers are appropriate for matrices whose rows or columns are indexed by a pointer; texts are for matrices indexed by a text, variates with a textual labels vector, or tables whose OLDDIMENSION factor has labels; and variates either refer to levels of table factors or numerical labels of matrices or variates, if these are present, otherwise they give the (ordinal) number of the position. If omitted, the positions are assumed to be in (ordinal) numerical order. Margins of tables are indicated by missing values
NEWPOSITIONS = *pointers*, *texts*, *variates* or *scalars*	These define positions in each NEWDIMENSION, specified similarly to OLDPOSITIONS; these indicate where the values from the corresponding OLDDIMENSION positions are to be entered (or added to any already entered there)
WEIGHTS = *variates*	Define weights by which the values from each OLDDIMENSION coordinate are to be multiplied

COMMANDINFORMATION directive

Provides information about whether (and how) a command has been implemented.

No options

Parameters

NAME = *texts*	Single-line texts supplying the names of the commands
IMPLEMENTATION = *texts*	Single-line texts set to 'directive', 'procedure' or a null string ('') according to the type of command

CHANNEL = *scalars*	Saves the channel for a procedure from a procedure library
PRESENTNOW = *scalars*	Logical set to one if the command is now present, or zero otherwise

CONCATENATE directive

Concatenates and truncates lines (units) of text structures; allows the case of letters to be changed.

Options

NEWTEXT = *text*	Text to hold the concatenated/truncated lines; default is the first OLDTEXT vector
CASE = *string token*	Case to use for letters (given, lower, upper, changed); default give leaves the case of each letter as given in the original string

Parameters

OLDTEXT = *texts*	Texts to be concatenated
WIDTH = *scalars* or *variates*	Number of characters to take from the lines of each text, a negative value takes all the (unskipped) characters other than trailing spaces; if * or omitted, all the (unskipped) characters are taken
SKIP = *scalars* or *variates*	Number of characters to skip at the left-hand side of the lines of each text, a negative value skips all initial spaces; if * or omitted, no characters are skipped

CONFIDENCE procedure

Calculates simultaneous confidence intervals (D.M. Smith).

Options

PRINT = *string token*	Controls printed output (intervals); default inte
METHOD = *string token*	Type of interval (individual, smm, product, Bonferroni, Scheffe); default smm
MU = *scalar*	Value for population mean checked as to whether in the confidence interval; default * i.e. no checking
PROBABILITY = *scalar*	The required significance level; default 0.05

Parameters

MEANS = *tables* or *variates*	Mean values
REPLICATIONS = *scalars* or *tables* or *variates*	Number(s) of observations per mean
VARIANCE = *scalars*	Estimate of variance
DF = *scalars*	Degrees of freedom
XCONTRASTS = *matrices*	Matrix of coefficients of orthogonal contrasts
LABELS = *texts*	Identifiers of mean values
LOWER = *tables* or *variates*	Lower values of confidence intervals
UPPER = *tables* or *variates*	Upper values of confidence intervals

CONTOUR directive

Produces contour maps of two-way arrays of numbers (on the terminal/printer).
This directive was replaced in Release 10 by the directive LPCONTOUR (with exactly the same options and parameters). It is currently retained as a synonym of LPCONTOUR, but may be removed in a future release.

CONVEXHULL procedure

Finds the points of a single or a full peel of convex hulls (P.G.N. Digby).

Options

PEELING = *string token*	Specifies whether the procedure is to form the full set of peels, or just the convex hull (no, yes); default no
SCALE = *scalar*	Scaling factor for hulls; default 1.0

Parameters

Y = *variate*	Y-coordinates of the points

X = *variate*	X-coordinates of the points
YHULL = *variate* or *pointer*	Variate storing the y-coordinates of the points defining the convex hull (for PEELING=no) or pointer to a set of variates storing the y-coordinates of the convex hulls forming the complete set of peels
XHULL = *variate* or *pointer*	Variate storing the x-coordinates of the points defining the convex hull (for PEELING=no) or pointer to a set of variates storing the x-coordinates of the convex hulls forming the complete set of peels
PEEL = *variate*	Stores the number of the peel to which each point belongs

COPY directive

Forms a transcript of a job.

Option

PRINT = *string tokens*	What to transcribe (statements, output); default stat

Parameter

scalar	Channel number of output file

CORANALYSIS procedure

Does correspondence analysis, or reciprocal averaging; synonym CORRESP (P.G.N. Digby).

Options

PRINT = *string tokens*	Printed output from the analysis (roots, rowscores, rowinertias, rowchisquare, rowmass, rowquality, colscores, colinertias, colchisquare, colmass, colquality); default * i.e. no output
METHOD = *string token*	Type of analysis required (correspondence, digbycorrespondence, biplot, reciprocal); default corr
NROOTS = *scalar*	Number of latent roots for printed output; default * requests them all to be printed
%METHOD = *string token*	How to represent proportions or %s in quality statistics (permills, percentages, proportions); default prop
NDIMENSIONS = *scalar*	Number of dimensions for which quality statistics are required; default 2
ROWSUBSET = *scalars*	Indexes of subset rows
COLSUBSET = *scalars*	Indexes of subset columns
ROWPASSIVE = *scalars*	Indexes of passive rows
COLPASSIVE = *scalars*	Indexes of passive columns

Parameters

DATA = *matrices* or *data matrices*	Data to be analysed
ROOTS = *diagonal matrices*	Saves the squared singular values from each analysis
ROWSCORES = *matrices*	Saves the scores for the rows of the data matrix
COLSCORES = *matrices*	Saves the scores for the columns of the data matrix
ROWINERTIAS = *matrices*	Saves the inertias for the rows of the data matrix
COLINERTIAS = *matrices*	Saves the inertias for the columns of the data matrix
ROWQUALITY = *matrices*	Saves the quality statistics for rows of the data
COLQUALITY = *matrices*	Saves the quality statistics for columns of the data
SAVE = *pointers*	Saves details of the analysis for use by CAPLOT

CORRELATE directive

Forms correlations between variates, autocorrelations of variates, and lagged cross-correlations between variates.

Options

PRINT = *string tokens*	What to print (correlations, autocorrelations, partialcorrelations, crosscorrelations); default *
GRAPH = *string tokens*	What to display with graphs (autocorrelations,

	partialcorrelations, crosscorrelations); default *
MAXLAG = *scalar*	Maximum lag for results; default * i.e. value inferred from variates to save results
CORRELATIONS = *symmetric matrix*	Stores the correlations between the variates specified by the SERIES parameter

Parameters

SERIES = *variates*	Variates from which to form correlations
LAGGEDSERIES = *variates*	Series to be lagged to form crosscorrelations with first series
AUTOCORRELATIONS = *variates*	To save autocorrelations, or to provide them to form partial autocorrelations if SERIES=*
PARTIALCORRELATIONS = *variates*	To save partial autocorrelations
CROSSCORRELATIONS = *variates*	To save crosscorrelations
TESTSTATISTIC = *scalars*	To save test statistics
VARIANCES = *variates*	To save prediction error variances
COEFFICIENTS = *variates* or *matrices*	To save prediction coefficients: in a variate to keep only those for the maximum lag, or in a matrix to keep the coefficients for all lags up to the maximum

COUNTER directive

Increments a multi-digit counter using non base-10 arithmetic.

Options

NREQUIRED = *scalar*	Specifies the number of values required for the counter; default 2
NFOUND = *scalar*	Saves the number of counter values that could be formed
DIRECTION = *string token*	Specifies the direction of the sequence of increments to the counter (ascending, descending); default asce

Parameters

START = *scalars*	Provides the starting values for the digits in the counter
END = *scalars*	Can provide values to define the end of the sequence of counter values
STEP = *scalars*	Specifies the amount by which to increment each digit of the counter
BASE = *scalars*	Specifies the base of the numbers used for each digit
DIGITSEQUENCE = *variates*	Saves the sequence of values generated for each digit

COVARIATE directive

Specifies covariates for use in subsequent ANOVA statements.

No options

Parameter

variates or *pointers*	Covariates

COVDESIGN procedure

Produces experimental designs efficient under analysis of covariance (D.B. Baird).

Options

PRINT = *string tokens*	Controls printed output (design, cefficiency, means, histogram, cutoff); default desi, ceff, cuto
TREATMENTSTRUCTURE = *formula*	Treatment terms to be fitted
BLOCKSTRUCTURE = *formula*	Block model for the design
COVARIATES = *variates*	Covariates for the design
FACTORIAL = *scalar*	Limit on number of factors in a treatment term; default 3
GRBLOCKSTRUCTURE = *formula*	Formula use for randomization; default uses BLOCKSTRUCTURE
EXCLUDE = *factors*	(Block) factors whose levels are not to be randomized
UNITS = *text*, *variate* or *factor*	Labels for the units of the design

Parameters

PROPORTION = *scalars*	Upper proportion of the combined cov. ef. distribution from

	which the design is to be chosen (or zero to take the best design found); default 0.5
NSIMULATIONS = *scalars*	Number of designs to simulate for the empirical distribution of combined cov. ef.'s; default 100
WEIGHTS = *variates*	Weighting for the treatment terms to use when calculating the combined cov. ef.; default 1 (i.e. all equal)
CEFLIMIT = *scalars*	Minimum value of the cov. ef. for each or variates treatment term for a design to be included in the set of acceptable designs; default 0 (i.e. all designs acceptable).
ORDER = *scalars*	Order of polynomial to fit for each covariate; or variates default 1 (i.e. only linear covariates)
SEED = *scalars*	Seed for random number generator for randomizing the simulated designs; default 0
SAVE = *pointers*	Saves the treatment factor allocations for the selected design; if unset, these overwrite the values of the treatment factors themselves
CUTOFF = *scalars*	Critical value of the combined cov. ef. from the simulated distribution
CEFFICIENCY = *variates*	Covariate efficiencies for the treatment terms from the selected design
SIMULATIONS = *variates*	Simulated combined cov. ef.'s

CRBIPLOT procedure

Plots correlation or distance biplots after RDA, or ranking biplots after CCA (A.I. Glaser).

Options

DIMENSIONS = *scalars*	Two numbers specifying which axes of the ordinations to plot; default 1,2
PLOT = *string token*	Whether to plot site or species scores (sitescores, speciesscores); default spec
WINDOW = *scalar*	Which graphical window to use; default 1
KEYWINDOW = *scalar*	Which graphical window to use for the key (zero for none); default 2
SAVE = *pointer*	Supplies results from an ordination analysis by CCA or RDA; default uses the most recent analysis

Parameters

X1 = *scalars*, *variates* or *texts*	First explanatory variable to plot; default 1
X2 = *scalars*, *variates* or *texts*	Second explanatory variable to plot; default * i.e. none
LMXVARIABLES = *string tokens*	How to label the x-variables (identifiers, labels, none, numbers); default labe if LXVARIABLES is set, otherwise iden
LMSPECIES = *string tokens*	How to label the species scores (identifiers, labels, none, numbers); default labe if LSPECIES is set, otherwise numb
LMSITES = *string tokens*	How to label the site scores (labels, none, numbers); default labe if LSITES is set, otherwise numb
LXVARIABLES = *texts*	Labels for variables
LSPECIES = *texts*	Labels for species scores
LSITES = *texts*	Labels for site scores

CRTRIPLOT procedure

Plots ordination biplots or triplots after CCA or RDA (A.I. Glaser).

Options

DIMENSIONS = *scalars*	Which dimensions of the ordinations to display; default 1,2
PLOT = *string token*	What to plot (sitescores, speciesscores, xvariables); default spec, site, xvar
DGROUPS = *string token*	Features to plot for the XGROUPS variate (ellipse, hull,

	lines, spider); default * i.e. none
DBINARY = *string token*	What to plot for binary variables (biplot, centroid); default bipl
MULTIPLIER = *scalar*	Value to multiply species and environmental variables scores by when plotting RDA; default *, i.e. none chosen
WINDOW = *scalar*	Which graphical window to use; default 1
KEYWINDOW = *scalar*	Which graphical window to use for the key (zero for none); default 2
SAVE = *pointer*	Supplies results from an ordination analysis by CCA or RDA; default uses the most recent analysis

Parameters

LMXVARIABLES = *string tokens*	How to label the x-variables (identifiers, labels, none, numbers); default labe if LXVARIABLES is set, otherwise iden
LMSPECIES = *string tokens*	How to label the species scores (identifiers, labels, none, numbers); default labe if LSPECIES is set, otherwise numb
LMSITES = *string tokens*	How to label the site scores (labels, none, numbers); default labe if LSITES is set, otherwise numb
LXVARIABLES = *texts*	Labels for variables
LSPECIES = *texts*	Labels for species scores
LSITES = *texts*	Labels for site scores
XGROUPS = *variates*, *factors* or *scalars*	X-variate to generate grouping information to appear on the plot (see the DGROUPS option)

CSPRO procedure

Reads a data set from a CSPro survey data file and dictionary, and loads it into Genstat or puts it into a spreadsheet file (D.B. Baird).

Options

PRINT = *string token*	What to print (catalogue); default cata
FACMETHOD = *string token*	Which factors to create (convertall, keepandconvertall, none, noranges); default keep
MISSINGCODES = *string tokens*	Which special values to convert to Genstat missing values (missing, na); default miss
FVALUESETS = *string token*	Whether form to a set of columns containing all the valueset information (yes, no); default no
SUBITEMS = *string token*	Whether to create a set of columns for the sub-items (yes, no); default no
MERGE = *string token*	Whether to merge the records into a single set of columns all of the same length (yes, no); default no
FUNKNOWNGROUP = *string token*	Whether to create a specific level for values not in the value set, rather than setting them to missing values (yes, no); default no
INCLUDEEXTRA = *string token*	Whether to include a row of column descriptions in the Excel output file after the column heading row (yes, no); default no
WARNONEMPTYGROUPS = *string token*	Whether to warn that groups in a factor are empty and offer to remove them when loading the data from a saved GWB file (yes, no); default no
DUPLICATELABELS = *string token*	What to do with factor groups that have identical labels (combine, ignore, rename); default comb
SCOPE = *string token*	Whether to read the data into global data structures or into data structures local to a procedure calling CSPRO (local, global); default loca
INOPTIONS = *text*	Optional extra input options to be passed to the Dataload.dll
OUTOPTIONS = *text*	Optional extra output options to be passed to the Dataload.dll

Parameters

FILENAME = *text*	Survey data file to be read
DICTIONARY = *text*	Survey dictionary for interpreting the data file
OUTFILENAME = *text*	Name of the output file to be created, if required
SURVEYLEVEL = *scalar*	Level of the survey (1, 2 or 3) to read; default 1
RECORDS = *scalar* or *variate*	Defines the records to be read within the SURVEYLEVEL; by default they are all read
ITEMS = *text*	Names of the survey items to be read
ISAVE = *text* or *pointer*	Saves the identifiers of the columns that are created

CUMDISTRIBUTION procedure

Fits frequency distributions to accumulated counts (R.C. Butler, M.E. O'Neill, P. Brain & H. Turner).

Options

PRINT = *string tokens*	Controls printed output (model, summary, estimates, correlations, fittedvalues, monitoring); default mode, summ, esti
DISTRIBUTION = *string token*	Which distribution to use (normal, logistic, complementaryloglog, acomplementaryloglog, inversenormal, weibull, exponential); default norm
TRANSFORMATION = *string token*	Whether to use log(TIME) if DISTRIBUTION = normal, logistic, complementarylog or acomplementarylog (log, none); default * uses log except when DISTRIBUTION = inversenormal, weibull or exponential
LAG = *string token*	Type of lag to add to TIME (none, positive, unconstrained); default none
ALLRESPOND = *string token*	If TOTUNITS is set, whether all units are constrained to respond (yes, no); default no
FORM = *string token*	Whether DATA are cumulated or differences (cumulated, differences); default cumu
LOSTUNITS = *string token*	Whether data are left-censored (yes, no); default no
SEPARATE = *string token*	Which parameters to estimate separately for each group (lag, b, m, propn, gamma); default *
POPSEPARATE = *string token*	Which parameters to estimate separately for populations in each group (b, m, lag); default *
PLOT = *string token*	Which graphs to draw (cumulative, density, trcumulative, trdensity); default cumu
MAXCYCLE = *scalar*	Number of iterations for fitting, as in RCYCLE; default 30

Parameters

DATA = *variates* or *pointers*	Specifies the accumulated counts
TIME = *variates* or *pointers*	Defines the time at which each count was recorded
GROUPS = *factors*	Factor indicating groups
INITIAL = *variates*	Initial values for all parameters
IB = *scalars* or *variates*	Initial values for *b*
IM = *scalars* or *variates*	Initial values for *m*
ILAG = *scalars* or *variates*	Initial values for *lag*
IGAMMA = *scalars* or *variates*	Initial values for *gamma*
IPROPN = *scalars* or *variates*	Initial values for proportions
STEPLENGTHS = *variates*	Steplengths for all parameters
SB = *scalars* or *variates*	Steplengths for *b*
SM = *scalars* or *variates*	Steplengths for *m*
SLAG = *scalars* or *variates*	Steplengths for *lag*
SGAMMA = *scalars* or *variates*	Steplengths for *gamma*
SPROPN = *scalars* or *variates*	Steplengths for proportions
TOTUNITS = *scalars* or *variates*	Total number
NPOPULATION = *scalars*	Number of populations (1, 2 or 3); default 1
SAVE = *pointers*	Saves the results

CVA directive

Performs canonical variates analysis.

Options

PRINT = *string tokens*	Printed output required (roots, loadings, means, residuals, distances, tests); default * i.e. no printing
NROOTS = *scalar*	Number of latent roots for printed output; default * requests them all to be printed
SMALLEST = *string token*	Whether to print the smallest roots instead of the largest (yes, no); default no

Parameters

WSSPM = *SSPMs*	Within-group sums of squares and products, means etc (input for the analyses)
LRV = *LRVs*	Saves loadings, roots, and trace from each analysis
SCORES = *matrices*	Saves canonical variate means
RESIDUALS = *matrices*	Saves distances of the means from the dimensions fitted in each analysis
DISTANCES = *symmetric matrices*	Saves inter-group-mean Mahalanobis distances
ADJUSTMENTS = *matrices*	Saves the adjustment terms
SAVE = *pointers*	Saves details of the analysis; if unset, an unnamed save structure is saved automatically (and this can be accessed using the GET directive)

CVAPLOT procedure

Plots the mean and unit scores from a canonical variates analysis (D.A. Murray).

Options

PLOT = *string tokens*	Type of plot to be drawn (meanscores, unitscores, confidenceregion); default mean, conf
GROUPS = *factor*	Group allocations in the CVA
MSCORES = *matrix*	Mean scores from the CVA; if unset these are calculated using the CVA directive
USCORES = *matrix*	Unit scores from the CVA; if unset these are calculated using the CVASCORES procedure
WSSPM = *SSPM*	Within-group sums of squares and products, means etc. for the CVA; must be supplied if the scores and groupings are not provided
CREGION = *string tokens*	Type of confidence region to be drawn (mean, population); default mean
CIPROBABILITY = *scalar*	The probability level for the confidence region; default 0.95
TAREA = *scalar*	Defines the transparency to use to shade the confidence regions; default 255 i.e. no shading

Parameters

YDIMENSION = *scalars*	Dimensions to be plotted in the y direction of each graph
XDIMENSION = *scalars*	Dimension to be plotted in the x direction
TITLE = *texts*	Title for each plot
WINDOW = *scalars*	Window for each graph; default 1
SCREEN = *string tokens*	Whether to clear the screen before plotting (clear, keep); default clea

CVASCORES procedure

Calculates scores for individual units in canonical variates analysis (S.A. Harding).

Option

PRINT = *string tokens*	What output to print (scores, adjustments); default scor

Parameters

WSSPM = *SSPMs*	Within-group sums of squares and products structure
LRV = *LRVs*	Loadings, roots and trace saved from CVA of the WSSPM

SCORES = *matrices* Unit scores
ADJUSTMENTS = *matrices* Mean Adjustments

DARROW procedure

Adds arrows to an existing plot (D. B. Baird).

Options

WINDOW = *scalar* Window number for the graphs; default 3
COORDINATETYPE = *string token* Type of coordinate to use for the locations of the arrows (frame, graph); default grap
YUPPER = *scalar* Maximum vertical coordinate in the frame; default 1
XUPPER = *scalar* Maximum horizontal coordinate in the frame; default 1
ISTYLE = *string token* The type of symbol at the start of the arrow (none, open, closed, circle); default none
ESTYLE = *string token* The type of symbol at the end of the arrow (none, open, closed, circle); default open
ISIZE = *scalar* The size of the symbol at the start of the arrow; default 1
ESIZE = *scalar* The size of the symbol at the end of the arrow; default 1
IANGLE = *scalar* The angle in degrees of the starting arrowhead when ISTYLE is open or closed; default 45
EANGLE = *scalar* The angle in degrees of the ending arrowhead when ESTYLE is open or closed; default 45
LAYER = *scalar* The plot layer for the arrows; default is a new layer above the previous plot items

Parameters

IY = *variates, scalars* or *factors* The starting y-positions of the arrows
IX = *variates, scalars* or *factors* The starting x-positions of the arrows
EY = *variates, scalars* or *factors* The ending y-position of the arrows
EX = *variates, scalars* or *factors* The ending x-position of the arrows
COLOUR = *variates, scalars*, *texts* or *factors* Colour of the arrows; default 'black'
LINESTYLE = *variates, scalars* or *factors* Linestyle of the line in the arrows; default 1
THICKNESS = *variates, scalars* or *factors* Thickness of the line in the arrows; default 1
TRANSPARENCY = *variates, scalars* or *factors* Transparency of the arrows; default 0

DAYLENGTH procedure

Calculates daylengths at a given period of the year (R.J. Reader & K. Phelps).

Option

LATITUDE = *scalar* Latitude at which the daylength is to be calculated, positive for northern hemisphere and negative for southern hemisphere; default 52.205 N (Wellesbourne)

Parameters

DAYNUMBER = *variate* Days of year for which daylengths are required
DAYLENGTH = *variate* Calculated daylengths in hours

DBARCHART procedure

Produces bar charts for one or two-way tables (A.R.G. McLachlan & R.C. Butler).

Options

TITLE = *text* Title for the chart; no default
WINDOW = *scalar* Window for the chart; default 1
KEYWINDOW = *scalar* Window for the key, no key is produced for one-way tables; default 2
LABELS = *text* Labels for clusters of bars; by default the labels or levels of the first classifying factor of TABLE are used

APPEND = *string token* Whether to append bars (no, yes); default no
SCREEN = *string token* Whether to clear screen before displaying chart (keep, clear); default clea
KEYDESCRIPTION = *text* Title for key; default is the name of the second factor of TABLE
YSCALE = *expression structure* Defines a transformation of the data, the expression must be a function of either Y or X, for example !e(log(X)), and should be valid for the range of the data in TABLE; default no transformation
BELOWORIGIN = *string token* Whether to include or values in TABLE less than ORIGIN (omit, include); default omit
ORIENTATION = *string token* Direction of the plot (horizontal, vertical); default vert
BARCOVERING = *scalar* What proportion of the space allocated along the x-axis each bar should occupy; default * gives proportion 0.8 (thus giving a gap between each bar or each group of bars)
XPOSITION = *string token* Position of the x-axis on the y-axis (lower, origin); default lowe
OMITEMPTYLEVELS = *string token* Whether to omit levels where there are only missing values (yes, no); default no

Parameters

TABLE = *tables* One or two-way table of data
ORIGIN = *scalars* Origin for y-axis; default 0
PEN = *variates* or *scalars* Pen (or pens) to use; default is !(1 ... nlevel(last_classifying_factor))
DESCRIPTION = *texts* Annotation for Key for two-way tables; default uses the labels or levels of the factor that is not being used as the XFACTOR
YMARKS = *variates* Position of the tick-marks on the y-axis
XFACTOR = *factors* X-axis factor for a 2-way TABLE; default first factor of TABLE
LOWERERRORBARS = *tables*, *variates* or *scalars* Lower bounds of the error bars on the y-axis
UPPERERRORBARS = *tables*, *variates* or *scalars* Upper bounds of the error bars on the y-axis
YERRORBARS = *tables*, *variates* or *scalars* Y-axis position of any error bar symbols; by default no symbols are plotted
XERRORBARS = *tables*, *variates* or *scalars* X-axis position of the error bars; default midpoints of bar-chart bars
PENERRORBARS = *tables*, *variates* or *scalars* Pen (or pens) to use for plotting error bars; default 1

DBCOMMAND procedure

Runs an SQL command on an ODBC database, PC Windows only (D.B. Baird).

Options

WARNINGDIALOGS = *string token* Whether dialogs giving ODBC error and warning messages are presented (display, omit); default disp
DRIVER = *scalar* Driver version (either 32 or 64) to use with the 64-bit version of Genstat; default 64

Parameters

COMMAND = *texts* Specifies SQL commands to run on the database
DB = *texts* Database connection string for each command
GDBFILE = *texts* Name of GDB file to be used in specifying the database for each command
EXIT = *scalars* The exit code (0=success, 1=failure) from each command

DBEXPORT procedure

Update data in an ODBC database table using Genstat data, PC Windows only (D.B. Baird).

Options

METHOD = *string token*	Type of update on table (create, insert, merge); default crea
ROWMERGEMETHOD = *string token*	For METHOD=merge, what action to take when rows do not match any in the existing table (none, matched, all); default all
COLMERGEMETHOD = *string token*	What to do with unmatched columns (add, omit); default add
OMIT = *string token*	Which rows to omit from the data for METHOD settings other than merge (none, restricted); default rest
ERRORACTION = *string token*	What to do when any non-fatal errors occur, (continue, stop); default stop
WARNINGDIALOGS = *string token*	If any errors occur, pop up warning dialogs (display, omit); default disp
GLKFILE = *text*	Name of existing Genstat ODBC Update link file (*.GLK) to use
DRIVER = *scalar*	Driver version (either 32 or 64) to use for the 64-bit version of Genstat; default 64
ODBCPATH = *text*	Path for the folder containing the executable program (Odbcload.exe) used by the 64-bit version of Genstat to export the data when DRIVER=32; default is the folder containing the Genstat executable program

Parameters

DATA = *pointer* or *text*	Pointer to a compatible set of data structures to add to the table or text with a name of an existing Genstat spreadsheet file containing data to be added
DB = *text*	Database connection string specifying the ODBC database to connect to
TABLENAME = *text*	Name of the table in the ODBC database (if METHOD is set to insert or merge, then this must already exist in the database)
COLUMNNAMES = *text*	Names of the columns in the table to be updated; if this is not provided, it will be assumed that the columns in the table have the same names as the Genstat data structures
SUBSET = *variate* or *text*	Column numbers or names of the subset of data columns (only if a pointer is used for the DATA parameter) to be added to the table; if SUBSET is not set, all columns are added to the table
MATCH = *variate*	Numbers of the columns in the table to be matched with the column in the table (the names are provided by WITH)
WITH = *text*	Names of the columns in the table to be matched with the Column; if this not provided, it is assumed that these columns have the same names as those of the Genstat data structures

DBIMPORT procedure

Loads data from an ODBC database, PC Windows only (D.B. Baird).

Options

PRINT = *string token*	What information to print (catalogue); default cata
OUTTYPE = *string token*	Whether to form a Genstat command file or spreadsheet file as output (GEN, GSH, GWB); default GWB
METHOD = *string token*	Whether to load data into the Genstat server after creating the file, or merely to create the file, or to run a command with no output (create, load, command); default load
IMETHOD = *string token*	Whether to read the column names from the first row of data, or to use default column names (read, supply, none, default); default read
ENDSTATEMENT = *string token*	Ending statement to use in a GEN output file (RETURN,

ENDBREAK); default RETURN

WARNINGDIALOGS = *string token* — Whether dialogs giving ODBC error and warning messages are presented (display, omit); default disp

DRIVER = *scalar* — Driver version (either 32 or 64) to use for the 64-bit version of Genstat; default 64

ODBCPATH = *text* — Path for the folder containing the executable program (Odbcload.exe) used by the 64-bit version of Genstat to load the data when DRIVER=32; default is the folder containing the Genstat executable program

NROWSFETCH = *scalar* — Number of rows to fetch per driver transaction; default 40

Parameters

DB = *text* — Database connection string

SQL = *text* — SQL Query string to run against the ODBC database

GDBFILE = *text* — Name of GDB file to be used in reading from ODBC database

OUTFILE = *text* — Output file to be created; if this is not provided a temporary file will be created, and then deleted if the data is loaded

COLUMNS = *text* — Names and/or type codes for the columns read (the type of column can be forced by ending the column name, if supplied, with the code ! for a factor, # for a variate, and $ for a text)

ISAVE = *pointer* — Name of a pointer to save the column identifiers

NROWSALLOCATE = *scalars* — Specifies how many rows to allow space for, in the initial allocation of memory, before the data are read; default 1000

DBINFORMATION procedure

Loads information on the tables and columns in an ODBC database, PC Windows only (D.B. Baird).

Options

PRINT = *string token* — What to print (information); default info

INFORMATION = *string token* — What information to read from the database (tables, columns); default tabl

DRIVER = *scalar* — Driver version (either 32 or 64) to use with the 64-bit version of Genstat; default 64

Parameters

DB = *texts* — Database connection string

GDBFILE = *texts* — GDB file specifying an ODBC query

ISAVE = *pointers* — Specifies pointers to save the information

DBIPLOT procedure

Plots a biplot from an analysis by PCP, CVA or PCO (A.I. Glaser).

Options

PLOT = *string tokens* — Additional features for the plot (convexhull, means); default * i.e. none

METHOD = *string token* — Type of axes to plot (predictive, interpolative); default pred

HORIZONTAL = *identifer* — Which axis to make horizontal; default * i.e. none

PREDICTIONS = *matrix* — Saves predicted values

GROUPS = *factor* — Factor defining groupings of individuals for a PCP biplot; default * i.e. none

LMINDIVIDUALS = *string tokens* — How to label the individuals (labels, none, numbers, unitlabels); default labe if LINDIVIDUALS is set, otherwise unit

LMVARIABLES = *string tokens* — How to label the variables (identifiers, labels, none, numbers); default labe if LVARIABLES is set, otherwise iden

LINDIVIDUALS = *texts* — Labels for individuals (i.e. scores)

LVARIABLES = *texts* — Labels for variables (i.e. biplot axes)

MULTIPLIER = *scalar* — Value to multiply vector loadings; default * i.e. determined

automatically

WINDOW = *scalar* — Which graphical window to use; default 1 when there are groups, otherwise 3

KEYWINDOW = *scalar* — Which graphical window to use for the key when there are groupings of individuals (0 for none); default 2

SCREEN = *string token* — Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea

SIZEMULTIPLIER = *scalar* — Multiplier used in the calculation of the size in which to draw symbols and labels; default 1

SAVE = *pointer* — Supplies results from an ordination analysis by PCP, CVA or PCO; default uses the most recent analysis

Parameters

VARIABLE = *identifiers* — Axis variables

DISPLAY = *string tokens* — Whether to show, hide or omit each axis (show, hide, omit); default disp

COLOUR = *texts* or *scalars* — Colour to use to plot each axis

DBITMAP directive

Plots a bit map of RGB colours.

Options

TITLE = *text* — General title; default *

WINDOW = *scalar* — Window number for the graph; default 1

YORIENTATION = *string token* — Y-axis orientation of the plot (reverse, normal); default reve

GRIDMETHOD = *string token* — How to draw a grid around the elements of the matrix (present, complete); default * i.e. none

PENGRID = scalar — Pen to use for the grid; default −7

SCREEN = *string token* — Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea

ENDACTION = *string token* — Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

BITMAP = *symmetric matrix*, *matrix*, *table*, *pointer to variates* or *variate* — Data to be plotted

ROWS = *variate* — Row indexes for a BITMAP variate

COLUMNS = *variate* — Column indexes for a BITMAP variate

DCLEAR directive

Clears a graphics screen.

Options

DEVICE = *scalar* — Device whose screen is to be cleared; default is to clear the screen of the current graphics device

ENDACTION = *string token* — Action to be taken after clearing the screen (continue, pause); default * uses the setting from the last DEVICE statement

No parameters

DCLUSTERLABELS procedure

Labels clusters in a single-page dendrogram plotted by DDENDROGRAM (R.W. Payne).

Options

WINDOW = *scalar* — Window containing the dendrogram; default 1

UNITS = *variate* or *text* — Names used for the units in the clusters supplied by CLUSTER

PEN = *scalar* — Pen to use to plot the labels; default 1

Parameters

CLUSTER = *variates* or *texts* — Specifies clusters to be labelled

LABEL = *texts*	Specifies the label to be plotted where each cluster is formed
YSAVE = *scalars*	Saves the y-coordinate where each label is plotted
XSAVE = *scalars*	Saves the x-coordinate where each label is plotted

DCOLOURS procedure

Forms a band of graduated colours for graphics (P.W. Goedhart).

Options

METHOD = *string token*	Type of colour band required (spectral, blackbody, linear); default line
PLOT = *string token*	What to plot (testgraph); default *

Parameters

START = *scalar* or *text*	Start value for the colour band; default * gives an appropriate default for the METHOD concerned
END = *scalar*, *text* or *variate*	End value(s) for the colour band; default * gives an appropriate default for the METHOD concerned
GAMMA = *scalar* or *variate*	The gamma-correction exponent(s) for the colour band; default 1
NCOLOURS = *scalar* or *variate*	Number(s) of colours in the colour band; default 20
RGB = *variates*	Saves the RGB colour values of each colour band
RED = *variates*	Saves the red component of the RGB colour values
GREEN = *variates*	Saves the green component of the RGB colour values
BLUE = *variates*	Saves the blue component of the RGB colour values
TITLE = *text*	General title for each test graph; default forms an informative title automatically
WINDOW = *scalar*	Window number for each test graph; default 0 does not display a test graph
SCREEN = *string token*	Whether to clear the screen before plotting each test graph or to continue plotting on the old screen (clear, keep); default clea

DCOMPOSITIONAL procedure

Plots 3-part compositional data within a barycentric triangle (S.J. Clark).

Options

PRINT = *text*	What to print (proportions); default *
VERTEXLABELS = *text*	Labels for the vertices of the triangle; default * uses the names of the corresponding variates given in the DATA pointer
TITLE = *text*	Title for the barycentric triangle; default * (i.e. no title)
PERPENDICULARS = *text*	Whether to draw perpendiculars from each vertex to its opposite side (yes, no); default no
WINDOW = *number*	Which high-resolution graphics window to use; default 3
SCREEN = *string token*	Whether to clear the graphics screen before plotting (clear, keep); default clea

Parameters

DATA = *pointers*	Contains variates which form the three-part compositions
SCALE = *scalars*	Scale factor for adjusting size of triangle to represent a fourth category; default 1
SAVECOORDINATES = *pointers*	Saves the two-dimensional x- and y-coordinates into the first and second elements of the pointer, respectively
PEN = *scalars* or *variates* or *factors*	Pen number to draw points within the barycentric triangle; default 1

DCONTOUR directive

Draws contour plots on a plotter or graphics monitor.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the plots; default 1

KEYWINDOW = *scalar* — Window number for the key (zero for no key); default 2
YORIENTATION = *string token* — Y-axis orientation of the plot (`reverse`, `normal`); default `reve`
ANNOTATION = *string token* — How to annotate the contours (`levels`, `ordinals`); default `ordi` if there is a key, and `leve` if there is no key
SCREEN = *string token* — Whether to clear the screen before plotting or to continue plotting on the old screen (`clear`, `keep`); default `clea`
KEYDESCRIPTION = *text* — Overall description for the key
ENDACTION = *string token* — Action to be taken after completing the plot (`continue`, `pause`); default * uses the setting from the last `DEVICE` statement

Parameters

GRID = *identifier* — Pointer (of variates representing the columns of a data matrix), matrix or two-way table specifying values on a regular grid
PENCONTOUR = *scalar* — Pen number to be used for the contours; default 1
PENFILL = *scalar* or *variate* — Pen number(s) defining how to fill the areas between contours, or 0 to leave the areas in the background colour; default 3
PENHIGHLIGHT = *scalar* — Pen number to use for highlighted contours; default 0 i.e. no highlighting
HIGHLIGHTFREQUENCY = *scalar* — Frequency at which contours are to be highlighted; default 10
NCONTOURS = *scalar* — Number of contours; default 10
CONTOURS = *variate* — Positions of contours
INTERVAL = *scalar* — Interval between contours
DESCRIPTION = *text* — Annotation for key

DCORRELATION procedure

Plots a correlation matrix (A.I. Glaser).

Options

PLOT = *string tokens* — Type of plot (`together`, `separate`); default `sepa`
SHOW = *string tokens* — What features to include on the plots (`axes`, `diagonal`); default `axes`
NCOLOURS = *scalar* — Number of distinct colour to use from 0 to −1 or 1; default 20
COLOURS = *text* or *variate* — Text or variate with three values, defining the colours to use for correlations of −1, 0 and 1; default * chooses the colours automatically
WEIGHTS = *variate* — Provides weights for the units of the variates; default * assumes that they all have weight one

Parameters

PVARIATES = *pointers* or *symmetric matrices* — Pointer to either the first (P-) set or the only set of variates to be correlated, or symmetric matrix containing the correlations themselves
QVARIATES = *pointers* — Pointer to the second (Q-) set of variates to be correlated
PROWS = *scalars* — Specifies the number of rows corresponding the first (P-) set of variates in a correlation matrix supplied by `PVARIATES`, when this contains two sets
TITLE = *text* — Title for the plot

DCOVARIOGRAM procedure

Plots 2-dimensional auto- and cross-variograms (D.A. Murray).

Options

PLOT = *string token* — Controls how to display the plotted variograms (`separate`, `scattermatrix`); default `scat`
ESTIMATES = *pointer* — Pointer containing model estimates saved from `MCOVARIOGRAM`

Parameter

COVARIOGRAM = *pointer*	Pointer to supply the semi-variances, distances and associated information as saved from FCOVARIOGRAM

DDENDROGRAM procedure

Draws dendrograms with control over structure and style (P.G.N. Digby).

Options

STYLE = *string token*	Style to use for the links of the dendrogram (average, centroid, lower, full); default aver
ORDERING = *string tokens*	How to define the order of the units for the dendrogram (given, ziggurat, size, first); default zigg, size, firs
REVERSE = *string token*	Whether to reverse the order of the units in the dendrogram (no, yes); default no
ORIENTATION = *string token*	Specifies the orientation of a dendrogram produced by high-resolution graphics (north, south, east, west); default west
METHOD = *string token*	Method used to represent the scale on which the amalgamations have been made: settings other than the default are relevant only for data not generated by HCLUSTER or HDISPLAY (similarities, percentages, distances); default simi
SCREEN = *string token*	Setting to use for the SCREEN option of DGRAPH (clear, keep); default clea
CHANGE = *string token*	If a dendrogram-save structure from a previous DDENDROGRAM is used as the DATA parameter then this option specifies the area of the process where the first changes occur: see the description of the SAVE parameter (order, dendrogram, display); default orde
GRAPHICS = *string token*	Form of graphics to be used (lineprinter, highresolution); default high
DSIMILARITY = *string token*	Whether to display an axis for the similarities in high-resolution graphics (no, yes); default no
LOWSIMILARITY = *scalar*	Lower value to be used for the axis showing the similarities; default * i.e. determined from the data
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the current setting

Parameters

DATA = *matrices* or *pointers*	Data defining each dendrogram in the form of either a matrix saved using the AMALGAMATIONS parameter of HCLUSTER (methods other than single linkage), or a matrix from the TREE parameter of HDISPLAY, or a SAVE structure from a previous use of DDENDROGRAM
PERMUTATION = *variates*	Specify or save permutations of the units for drawing each dendrogram, according to ORDERING option
LABELS = *variates* or *texts*	Supply labels to use for the units of each dendrogram; these should be in the natural order of the units, not in a permuted order
TITLE = *texts*	Titles for the dendrograms
WINDOW = *scalars*	Window to use for each dendrogram (window 1 if unset); if this is set to zero the dendrogram is not drawn, but results can still be saved using the PERMUTATION, ZIGGURAT and SAVE parameters
PENS = *scalars*, *variates*, *strings* or *texts*	Scalar or string specifying the graphics pen or symbol in which to draw each (high-resolution or line-printer) dendrogram; alternatively use of a variate or text allows the structure of

	each dendrogram to be highlighted by drawing different links with different graphics pens or symbols
ZIGGURAT = *variates*	Save the "ziggurat-degree" of the links in each dendrogram
SAVE = *pointers*	Save the information required to plot a dendrogram, for use as input for the DATA parameter in a subsequent call to DDENDROGRAM

DDESIGN procedure

Plots the plan of an experimental design (K.E. Bicknell & R.W. Payne).

Options

Y = *variate*	Specifies the *y* position of the plots in standard coordinates 1 ... number of rows of plots in the experiment (taking 1 as the top row of the window)
X = *variate*	Specifies the *x*-coordinate of the plots in standard coordinates 1 ... number of columns of experimental plots
TITLE = *text*	Title for the plan
WINDOW = *scalar*	Window number for the plan; default 3
KEYWINDOW = *scalar*	Window number for the key; default 0
SCREEN = *string token*	Whether to clear the screen before plotting (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key; default *
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement
CHARACTERS = *scalar*	Sets a limit on the length of each factor label; default * i.e. none
SIZE = *scalar*	Provides a multiplier by which to scale the sizes of the factor labels on the plan

Parameters

FACTOR = *factors*	Factors to be listed on the plan and to define the layout (the procedure determines the style of line to divide each pair of plots in the design from the grid pen of the first factor in the list with which they have different levels); default * forms the list from first the factors specified by a preceding BLOCKSTRUCTURE statement, and then those specified by a preceding TREATMENTSTRUCTURE statement
PEN = *scalars*	Pen to be used to write the levels of each factor on the plan (if PEN=0 the levels of that factor are not included); default 1 if the FACTOR parameter is specified, otherwise 0 for block factors and 1 for treatment factors
PENGRID = *scalars*	Pens to be used to draw the boundaries between the plots in the design (according to the first FACTOR with which they have different levels but ignoring factors with PENGRID=0); default 1,2...
LABELS = *texts*	Labels to be used for each factor if its own levels or labels are inappropriate

DDISPLAY directive

Redraws the current graphical display.

Options

DEVICE = *scalar*	Device on which to redraw the display (on some systems it may only be possible to redisplay the picture on an interactive graphics device); default uses the current graphics device
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

No parameters

DEBUG directive

Puts an implicit BREAK statement after the current statement and after every NSTATEMENTS subsequent statements, until an ENDDEBUG is reached.

Options

CHANNEL = *scalar*	Channel number; default 1
NSTATEMENTS = *scalar*	Number of statements between breaks; default 1
FAULT = *string token*	Whether to invoke DEBUG only at the next fault (yes, no); default no

No parameters

DECIMALS procedure

Sets the number of decimals for a structure, using its round-off (A. Keen).

Options

SETATTRIBUTE = *string token*	Attributes to be redefined for STRUCTURE (decimals); default deci
SIGNIFICANTFIGURES = *scalar*	Required number of significant figures; default takes the system default, which can be modified by SET

Parameters

STRUCTURE = *identifiers*	Numerical structure for which the number of decimals is to be set
DECIMALS = *scalars*	To save the number of decimals
ROUND = *scalars*	To save the round-off
VDECIMALS = *structures*	To save numbers of decimals for every value of each structure
VROUND = *structures*	To save the round-off for every value of each structure

DECLARE directive

declares one or more customized data structures.

Options

TYPE = *text*	Single-valued text defining the type of structure to declare
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no

Parameters

IDENTIFIER = *identifiers*	Identifiers of the structures
VALUES = *pointers*	Values for each structure
EXTRA = *text*	Extra text associated with each identifier

DELETE directive

Deletes the attributes and values of structures.

Options

REDEFINE = *string token*	Whether or not to delete the attributes of the structures so that the type etc can be redefined (yes, no); default no
LIST = *string token*	How to interpret the list of structures (inclusive, exclusive, all); default incl
PROCEDURE = *string token*	Whether the list of identifiers is of procedures instead of data structures (yes, no); default no
NSUBSTITUTE = *scalar*	Number of times *n* to substitute a dummy in order to determine which structure to delete; default * i.e. full substitution
REMOVE = *string token*	Whether or not to remove the structures from Genstat completely i.e. to delete their identifiers as well as their attributes and values (yes, no); default no

Parameter

identifiers	Structures whose values (and attributes, if requested) are to be deleted

†DELLIPSE procedure

Draws a 2-dimensional scatter plot with confidence, prediction and/or equal-frequency ellipses superimposed (V.M. Cave).

Options

PLOT = *string tokens*	What type of ellipse to plot (confidence, prediction, equalfrequency); default conf
PROBABILITY = *scalar* or *variate*	Probability level(s) for the ellipse(s); default 0.95
NPOINTS = *scalar*	Number of points used to draw the ellipses; default 1000
DISPLAY = *string token*	Whether to include the data points on the graph (show, hide); default show
PAXES = *string token*	Whether to plot the principal axes on the graph (no, yes); default no
TFILL = *scalar*	Transparency used to fill the area inside the ellipses, on a scale of 0 (opaque) to 255 (completely transparent); default 255
USEPENS = *string token*	Whether to use the current pen definitions for drawing the ellipses, drawing the principal axes and plotting the data (no, yes); default no
CMATCH = *string token*	When USEPENS=yes and groups are to be plotted, indicates whether the colours for the ellipses and principal axes are matched to the corresponding group, or to the colours defined by the pens for the different ellipse types and principal axes (group, pen); default group
WINDOW = *scalar*	Window to use for the graph(s); default 1
KEYWINDOW = *scalar*	Window to use for the key; by default the key is drawn on the right, in window 255
KEYDESCRIPTION = *text*	Overall title for the key; default * i.e. none
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clear

Parameters

Y = *variates* or *pointers*	Vertical coordinates (i.e. variable to plot on the y-axis)
X = *variates* or *pointers*	Horizontal coordinates (i.e. variable to plot on the x-axis)
GROUPS = *factors*	Defines groupings of the data points
DESCRIPTION = *texts*	Labels for the groups; default generates the labels automatically
TITLE = *text*	Title for the plot; default * i.e. none
YTITLE = *text*	Title for the y-axis; by default a title is generated automatically
XTITLE = text	Title for the x-axis; by default a title is generated automatically

DEMC procedure

Performs Bayesian computing using the Differential Evolution Markov Chain algorithm (W. van den Berg & R.W. Payne).

Options

PRINT = *string token*	What to print (results, monitoring, scatterplot, histogram); default resu, moni, scat, hist
CALCULATION = *expression structures*	Calculation(s) of logposterior, involving explanatory or pointer variate; if unset, this is calculated by the procedure specified by the PROCEDURE option
LOGPOSTERIOR = *scalar*	Identifier of scalar holding log-posterior within CALCULATION (must be set if CALCULATION is set)
MULTIPLE = *scalar*	Number of populations is number of parameters times MULTIPLE; default 3
UNIFORMLIMIT = *scalar*	Uniform random numbers are drawn from (-UNIFORMLIMIT, UNIFORMLIMIT) and added to candidate parameter sets; default 0.00001
DATA = *identifiers*	Data structures used in CALCULATION or by PROCEDURE

NGENERATIONS = *scalar*	Maximum number of iterations; default 1000
STEP1 = *scalar* or *variate*	Generations for which gamma is set to 1; default 0
FRACTIONBURNIN = *scalar*	Fraction of iterations used for burn-in; default 0.5
GRVARIANCE = *scalar* or *variate*	Variance to generate populations from initial values of the parameters; default 0.1
PERCENTAGES = *variate*	Percentages for which quantiles has to be calculated; default !(2.5, 25, 50, 75, 97.5)
PROCEDURE = *identifier*	Identifier of procedure to calculate LOGPOSTERIOR if CALCULATION is unset; default _DEMCLOGPOSTERIOR
SEED = *scalar*	Seed for the random numbers; default 0
NWINDOWS = *scalar*	Number of histograms and scatterplots per screen when plotting estimates and logposterior from all iterations
SDLOGPOSTERIOR = *scalar*	Saves the s.d. for LOGPOSTERIOR
QUANTILESLOGPOSTERIOR = *variate*	Saves quantiles for LOGPOSTERIOR
RHATLOGPOSTERIOR = *scalar*	Saves the convergence criterion for LOGPOSTERIOR
ALLLOGPOSTERIOR = *variate*	Saves the parameter estimates for LOGPOSTERIOR from all the iterations
IPOPULATIONS = *pointers*	Pointer to supply initial populations of the parameters and the corresponding log-posteriors
FPOPULATIONS = *pointers*	Pointer to save final populations of the parameters and the corresponding log-posteriors

Parameters

PARAMETER = *scalars*	Parameters to estimate
INITIAL = *scalars*	Initial values of the parameters; must be set unless IPOPULATIONS is set
SD = *scalars*	Standard errors of the estimates
QUANTILES = *variates*	Saves the quantiles for each parameter
RHAT = *scalars*	Convergence criteria
ALLESTIMATES = *variates*	Saves the parameter estimates from all the iterations

DERRORBAR procedure

Adds error bars to a graph (R.W. Payne).

Options

ORIENTATION = *string token*	Direction of the line (horizontal, vertical); default vert
BARCAPWIDTH = *scalars*	Width of the cap drawn at the ends of the error bar; default 1
WINDOW = *scalar*	Window in which to draw the bar; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2

Parameters

BARLENGTH = *scalars*	Lengths of the bars
Y = *identifiers*	Vertical coordinates for the midpoints of the bars
X = *identifiers*	Horizontal coordinates for the midpoints of the bars
PEN = *scalars*	Pen to use for each bar
LABEL = *texts*	Text to plot alongside each bar
YLPOSITION = *string tokens*	Position of each label in the y-direction (above, below, centre, center); default belo
XLPOSITION = *string tokens*	Position of each label in the x-direction (centre, center, left, right); default righ
PENLABEL = *scalars*	Pen to use for each label
DESCRIPTION = *texts*	Annotation for the key

DESCRIBE procedure

Saves and/or prints summary statistics for variates (R.C. Butler & D.A. Murray).

Options

PRINT = *string token*	Controls whether or not the summaries are printed (summaries); default summ
SELECTION = *string tokens*	Selects the statistics to be produced (nval, nobs, nmv, mean,

	median, min, max, range, q1, q3, sd, sem, var, sevar, %cv, sum, ss, uss, skew, seskew, kurtosis, sekurtosis, all); default mean, min, max, nobs, nmv, medi, q1, q3
GROUPS = *factor*	Allows groups to be defined, so that summaries are produced for each group in turn

Parameters

DATA = *variates*	Data to summarize
SUMMARIES = *variates* or *pointers*	To save summaries for each DATA variate, in a variate if GROUPS is unset, or in a pointer to a set of variates (one for each group) if groups have been specified; will be redefined if necessary

DESIGN procedure

Helps to select and generate effective experimental designs (R.W. Payne, M.F. Franklin & A.E. Ainsley).

Option

STATEMENT = *text*	Saves a command to recreate the design

No parameters

DEVICE directive

Switches between (high-resolution) graphics devices.

No options

Parameters

NUMBER = *scalar*	Device number
ENDACTION = *string token*	Action to be taken after completing each plot (continue, pause)
ORIENTATION = *string token*	Orientation of the pictures, if relevant (landscape, portrait); default * retains the current setting for this device
PALETTE = *string token*	How to represent colour (monotone, greyscale, grayscale, colour); default * retains the current setting for this device
RESOLUTION = *scalar*	Specifies the height of the image for hard-copy output, in pixels
ACTION = *string token*	How to create graphs for file types such as .emf, .jpg, .tif or .png (asynchronous, synchronous); default asyn

DFINISH directive

Ends a sequence of related high-resolution plots.

No options or parameters

DFONT directive

Defines the default font for high-resolution graphics.

No options

Parameter

text	specifies or saves the default graphics font

DFOURIER procedure

Performs a harmonic analysis of a univariate time series (G. Tunnicliffe Wilson & R.P. Littlejohn).

Options

PRINT = *string tokens*	Controls printed output (accumulated, means, tsm); default *
PLOT = *string tokens*	What to plot (periodogram, harmonics, means, residuals, cumulative, range); default peri, harm, mean, resid
MODELTYPE = *string token*	What harmonic regression model to fit (none, best, full); default none

GROUPS = *factor*	Groups for plot of means
ORDER = *variate*	Order for time series model; default !(1,0,0)
COLOURS = *text* or *variate*	Colour for each level of GROUPS
FACSHORTCYCLE = *factor*	Factor giving levels of the short cycle
NCOMPONENTS = *scalar*	Number of nested cycles, must be 0, 1, or 2; default 0
SHORTCYCLE = *scalar*	Length of the short cycle; default 24
LONGCYCLE = *scalar*	Length of the long cycle; default 365.225
LABSHORTCYCLE = *text*	Label for the short cycle; default 'daily'
LABLONGCYCLE = *text*	Label for the long cycle; default 'annual'
NHSHORTCYCLE = *scalar*	Number of harmonics for the short cycle; default 5
NHLONGCYCLE = *scalar*	Number of harmonics for the long cycle; default 3
RANGE = *variate*	Variate with two values, defining the frequency range within [0,0.5] to draw a portion of the periodogram

Parameters

DATA = *variates*	Time series
PERIODOGRAM = *variates*	Saves the periodogram of DATA
FREQUENCY = *variates*	Saves the frequencies at which the periodogram is calculated
MEANS = *tables*	Saves the table of means of the fitted model for each value of FACSHORTCYCLE by each level of GROUPS
RESIDUALS = *variates*	Saves the residuals from the fitted model
FITTEDVALUES = *variates*	Saves the fitted values from the model

DFRTEXT procedure

Adds text to a graphics frame (W. van den Berg).

No options

Parameters

Y = *variates* or *scalars*	Vertical coordinates in the frame
X = *variates* or *scalars*	Horizontal coordinates in the frame
TEXT = *texts*	Text to plot
PEN = *scalars*, *variates* or *factor*s	Pens to use; default 1
YUPPER = *scalars*	Maximum vertical coordinate in the frame; default 1
XUPPER = *scalars*	Maximum horizontal coordinate in the frame; default 1

DFUNCTION procedure

Plots a function (R.W. Payne).

Options

FUNCTION = *expression*	Function to plot
TITLE = *text*	Title for the plot; default shows the function
COLOUR = *text* or *scalar*	Colour of the function curve; default 'green'
WINDOW = *scalar*	Which graphics window to use; default 3
ELEVATION = *scalar*	Elevation of the viewpoint for the surface that is plotted when there are two arguments; default 25 (degrees)
AZIMUTH = *scalar*	Rotation about the horizontal plane for the viewpoint of a surface plot; default 225 (degrees)
DISTANCE = *scalar*	Distance of the viewpoint of a surface plot from the centre of the grid on the base plane; default * gives a distance of 100 times the maximum of the x-range and the y-range
ZSCALE = *scalar*	defines the scaling of the z-axis relative to the horizontal (x-y) axes in a surface plot; default 1
SCREEN = *string token*	Whether to clear the screen before plotting (clear, keep, resize); default clea

Parameters

ARGUMENT = *scalars*	Arguments of the function
LOWER = *scalars*	Lower values of the arguments for the plot
UPPER = *scalars*	Upper values of the arguments for the plot
STEP = *scalars*	Steps at which to evaluate the function

DGRAPH directive

Draws graphs on a plotter or graphics monitor.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the graphs; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep, resize); default clea
KEYDESCRIPTION = *text*	Overall description for the key; default *
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement
HOTMENU = *matrices*	Defines sets of "hot" components for the user to select as shown or hidden by a menu in the Graphics Viewer
HOTCHOICE = *string token*	Whether one or several "hot" components can be displayed at a time (one, several); default seve

Parameters

Y = *identifiers*	Vertical coordinates
X = *identifiers*	Horizontal coordinates
PEN = *scalars*, *variates* or *factors*	Pen number for each graph (use of a variate or factor allows different pens to be defined for different sets of units); default * uses pens 1, 2, and so on for the successive graphs
DESCRIPTION = *texts*	Annotation for key
YLOWER = *identifiers*	Lower values for vertical bars
YUPPER = *identifiers*	Upper values for vertical bars
XLOWER = *identifiers*	Lower values for horizontal bars
XUPPER = *identifiers*	Upper values for horizontal bars
YBARPEN = *scalars*, *variates* or *factors*	Pens to use to draw the vertical bars; default −11
XBARPEN = *scalars*, *variates* or *factors*	Pens to use to draw the horizontal bars; default −11
LAYER = *scalars*	"Layer" of the plot
UNITNUMBERS = *identifiers*	Specifies unit numbers to be used when points are selected in the graphics viewer; default * uses the actual unit numbers of the values in the X and Y structures
DISPLAY = *string tokens*	Whether to display each component initially in the graph (show, hide); default show
HOTCOMPONENT = *scalars*	Allows components of the graph (specified by pairs of Y and X settings) to be defined as "hot" components that can be shown or hidden through their association with "hot" points or using a menu in the Graphics Viewer
HOTDEFINITION = *matrices*	Define how to use points defined by the Y and X parameters as "hot" points in the Graphics Viewer to allow the user to decide whether other components of the graph are shown or hidden

DHELP procedure

Provides information about Genstat graphics (S.A. Harding).

No options

Parameter

TOPIC = *string tokens*	Lists the required graphics topics (current, possible); default poss

DHISTOGRAM directive

Draws histograms on a plotter or graphics monitor.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the histograms; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
LIMITS = *variate*	Variate of group limits for classifying DATA variates into groups; default *
LOWER = *scalar*	For a DATA variate, this specifies the lower limit of the first bar; default * takes the minimum value of the variates
UPPER = *scalar*	For a DATA variate, this specifies the upper limit of the last bar; default * takes the maximum value of the variates
NGROUPS = *scalar*	When LIMITS and BINWIDTH are not specified, this defines the number of groups into which a DATA variate is to be classified; default is then 10 or the integer value nearest to the square root of the number of values in the variate if that is smaller
BINWIDTH = *scalar*	When LIMITS is unset the range of a DATA variate is split into equal intervals known as "bins" to form the groups, this option can set the bin widths (alternative is to set the number of groups using NGROUPS)
FIXEDBARWIDTH = *string token*	Whether to plot the histogram with bars of equal width (no, yes); default no
BARCOVERING = *scalar*	What proportion of the space allocated along the x-axis each bar should occupy; default * gives proportion 1 for a DATA variate, and 0.8 for a factor or table (thus giving a gap between each bar)
LABELS = *text*	Group labels; default *
APPEND = *string token*	Whether or not the bars of the histograms are appended together (yes, no); default no
ORIENTATION = *string token*	Direction of the plot (horizontal, vertical); default vert
OUTLINE = *string token*	Where to draw outlines (bars, perimeter); default bars
PENOUTLINE = *scalar*	Pen to use for the outlines; default -8
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key; default *
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

DATA = *identifiers*	Data for the histograms; these can be either a factor indicating the group to which each unit belongs, a variate whose values are to be grouped, or a one-way table giving the height of each bar
NOBSERVATIONS = *tables*	One-way table to save numbers in the groups
GROUPS = *factors*	Factor to save groups defined from a variate
PEN = *scalars* or *variates*	Pen number(s) for each histogram; default * uses pens 2, 3, and so on for the successive structures specified by DATA
DESCRIPTION = *texts*	Annotation for key

DHSCATTERGRAM procedure

Plots an h-scattergram (D.A. Murray).

Options

LAGCLASS = *scalar* or *variate*	The lag classes to be displayed in the plots; default all lag classes
ARRANGEMENT = *text*	Specifies whether to display the plots individually or with

multiple plots on the same page (`single`, `multiple`); default `mult`

Parameters

`DATA` = *variates*	Observations as a variate
`LAGPOINTS` = *pointers*	Lag classes, indexes to observations and directions for plotting

`DIAGONALMATRIX` directive

Declares one or more diagonal matrix data structures.

Options

`ROWS` = *scalar*, *vector*, *pointer* or *text*	Number of rows, or labels for rows (and columns); default `*`
`VALUES` = *numbers*	Values for all the diagonal matrices; default `*`
`MODIFY` = *string token*	Whether to modify (instead of redefining) existing structures (`yes`, `no`); default `no`
`IPRINT` = *string tokens*	Information to be used by default to identify the diagonal matrices in output (`identifier`, `extra`); if this is not set, they will be identified in the standard way for each type of output

Parameters

`IDENTIFIER` = *identifiers*	Identifiers of the diagonal matrices
`VALUES` = *identifiers*	Values for each diagonal matrix
`DECIMALS` = *scalars*	Number of decimal places for printing
`EXTRA` = *texts*	Extra text associated with each identifier
`MINIMUM` = *scalars*	Minimum value for the contents of each structure
`MAXIMUM` = *scalars*	Maximum value for the contents of each structure
`DREPRESENTATION` = *scalars* or *texts*	Default format to use when the contents represent dates and times

`DIALLEL` procedure

Analyses full and half diallel tables with parents (J.F. Potter).

Options

`PRINT` = *string tokens*	Controls printed output (`data`, `vrwr`, `regression`, `aov`, `means`, `griffingaov`); default `data`, `vrwr`, `regr`, `aov`, `mean`
`LABELS` = *text*	Labels for rowcols, one text value for each, column *j* has the same label as row *j*, so each value of `LABELS` is the label for a pair of parents, applying to a rowcol; default 1...*N*, where *N* is the dimension of each diallel table
`METHOD` = *string token*	Whether to perform full or half diallel analysis (`half`, `full`); default `full`

Parameter

`DATA` = *matrices*	Each matrix contains the data for one block in the analysis, half diallel tables are presented as square matrices with the upper triangles and leading diagonals containing the values of interest, the matrices must be of the same size

`DILUTION` procedure

Calculates Most Probable Numbers from dilution series data (M.S. Ridout & S.J. Welham).

Options

`PRINT` = *string tokens*	Output required (`estimates`, `fitted`); default `esti`, `fitt`
`%LIMITS` = *scalar*	Percentage points for confidence limits; default 95
`RMETHOD` = *string token*	Which type of residuals to form (deviance, Pearson); default `devi`
`MAXCYCLE` = *scalar*	Maximum number of iterations allowed for the Newton-Raphson algorithm to converge; default 10
`TOLERANCE` = *scalar*	Defines the convergence criterion; default 0.0005

Parameters

POSITIVE = *variates*	Number of positive subsamples at each dilution
NSAMPLE = *variates*	Total number of subsamples tested at each dilution
VOLUME = *variates*	Volume of original sample present in each dilution
FITTED = *variates*	To store the fitted values
RESIDUAL = *variates*	To store the residuals, as specified by option RMETHOD
MPN = *scalars*	To store the maximum likelihood estimate of Most Probable Number
UPPER = *scalars*	To store the upper confidence limit for MPN
LOWER = *scalars*	To store the lower confidence limit for MPN
DEVIANCE = *scalars*	To store the residual deviance
PEARSONCHISQUARE = *scalars*	To store Pearson's chi-square statistic
DF = *scalars*	To store the degrees of freedom for goodness-of-fit tests (zero if no test is available)

DIRECTORY procedure

Prints or saves a list of files and/or subdirectories with names matching a specified mask (D.B. Baird).

Options

PRINT = *string tokens*	What to print (filenames, subdirectories); default file
SAVEPATH = *string token*	Whether to include the path in FILENAMES (yes, no); default no
MASKTYPE = *string token*	The type of mask specified by MASK (file, directory); default file

Parameters

MASK = *texts*	Mask identifying the files that are to be included in the each listing, if no directory path is included, the current working directory is searched; default '*.*'
FILENAMES = *texts*	Saves the list of files that match each mask
SUBDIRECTORIES = *texts*	Saves the list of subdirectories that match each mask

DISCRIMINATE procedure

Performs discriminant analysis (L.H. Schmitt & P.G.N. Digby).

Options

PRINT = *string tokens*	Printed output from the analysis (counts, lrv, tests, ccorrelations, icorrelations, correlations, adjustments, means, gdistances, scores, distances, newgroups, table, validation); default coun
NROOTS = *scalar*	The number of dimensions to be used for printed and saved output, and used in calculating the distances and the allocation of units; default is to use the full dimensionality
REALLOCATE = *string token*	Whether units from the training set are to be reallocated to groups (no, yes); default no
PLOT = *string tokens*	Features for the plots (means, mlabels, scores, polygons, confidencecircle); default mean, scor, poly (Note: * suppresses plotting)
VALIDATIONMETHOD = *string token*	Validation method to use to calculate error rates (bootstrap, crossvalidation, jackknife); default cros
NSIMULATIONS = *variate*	Number of bootstraps or cross-validation sets to use for selection and for validation; default !(10,50)
NCROSSVALIDATIONGROUPS = *scalar*	Number of groups for cross-validation, default 10
SEED = *scalar*	Seed for random number generation; default 0
YROOT = *scalars*	Specifies roots for plotting on y-axes
XROOT = *scalars*	Specifies roots for plotting on x-axes
TITLE = *strings*	Titles for plots
WINDOW = *scalars*	Windows for plots
SCREEN = *string tokens*	Action before each plot (keep, clear); default clea

Parameters

DATA = *pointers*	Each pointer contains a set of variates to be analysed
GROUPS = *factors*	Define groupings for the units in each training set, or missing values for the units to be allocated
NEWGROUPS = *factors*	Saves allocations (and reallocations)
ALLOCATION = *factors*	Saves allocations to groups including those not present in the training set
MEANS = *matrices* or *pointers*	Saves scores for group means
SCORES = *matrices* or *pointers*	Saves scores for units
DISTANCES = *matrices*	Saves unit to group-mean squared distances
LRV = *LRVs*	Saves the LRVs from the canonical variates analyses
ADJUSTMENTS = *matrices*	Saves adjustments to the canonical variates analyses
GDISTANCES = *symmetric matrices*	Saves the distances between groups
CCORRELATIONS = *matrices*	Saves canonical correlation coefficients
ICORRELATIONS = *symmetric matrices*	Saves within-group correlation matrices of the input variates
CORRELATIONS = matrices	Saves within-group correlations between the input and canonical variates

DISPLAY directive

Prints, or reprints, diagnostic messages.

Options

PRINT = *string token*	What information to print (diagnostic); default diag
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
FAULT = *text*	Specifies the fault message to print (for example, FAULT='VA 4' prints the message "Values not set"); default is to print the last diagnostic message

No parameters

DISTRIBUTION directive

Estimates the parameters of continuous and discrete distributions.

Options

PRINT = *string tokens*	Printed output required from each individual fit (parameters, samplestatistics, fittedvalues, proportions, monitoring); default para, samp, fitt
CBPRINT = *string tokens*	Printed output required from a fit combining all the input data (parameters, samplestatistics, fittedvalues, proportions, monitoring); default *
DISTRIBUTION = *string token*	Distribution to be fitted (Poisson, geometric, logseries, negativebinomial, NeymanA, PolyaAeppli, PlogNormal, PPascal, Normal, dNvequal, dNvunequal, logNormal, exponential, gamma, Weibull, b1, b2, Pareto); default * i.e. fit nothing
CONSTANT = *string token*	Whether to estimate a location parameter for the gamma, logNormal, Pareto or Weibull distributions (estimate, omit); default omit
LIMITS = *variate*	Variate to specify or save upper limits for classifying the data into groups; default *
NGROUPS = *scalar*	When LIMITS is not specified, this defines the number of groups (of approximately equal size) into which the data are to be classified; default is the integer value nearest to the square root of the number of data values
XDEVIATES = *variate*	Variate to specify points up to which the CUMPROPORTIONS are to be estimated
JOINT = *string token*	Requests joint estimates from the combined fit to be used for a re-fit to the separate data sets (dispersion,

	variancemeanratio, Poissonindex); default *
PARAMETERS = *variate*	Estimated parameters from the combined fit
SE = *variate*	Standard errors for the estimated parameters of the combined fit
VCOVARIANCE = *symmetric matrix*	Variance-covariance matrix for the estimated parameters of the combined fit
CUMPROPORTIONS = *variate*	Estimated cumulative proportions of the combined distribution up to the values specified by the XDEVIATES option
MAXCYCLE = *scalar*	Maximum number of iterations; default 30
TOLERANCE = *scalar*	Convergence criterion; default 0.0001

Parameters

DATA = *variates* or *tables*	Data values either classified (table) or unclassified (variate)
NOBSERVATIONS = *tables*	One-way table to save the data classified into groups
RESIDUALS = *tables*	Residuals from each (individual) fit
FITTEDVALUES = *tables*	Fitted values from each fit
PARAMETERS = *variates*	Estimated parameters from each fit
SE = *variates*	Standard errors of the estimates
VCOVARIANCE = *symmetric matrices*	Variance-covariance matrix for each set of estimated parameters
CUMPROPORTIONS = *variates*	Estimated cumulative proportions of each distribution up to the values specified by the XDEVIATES option
CBRESIDUALS = *tables*	Residuals from the combined fit
CBFITTEDVALUES = *tables*	Fitted values from the combined fit
STEPLENGTH = *variates*	Initial step lengths for each fit
INITIAL = *variates*	Initial values for each set fit

DKALMAN procedure

Plots vector time series (A.I. Glaser).

Options

TIMEPOINTS = *variate*	X-coordinates for the graphs; default uses the integers 1, 2...
TITLE = *texts*	Overall title for the graphs
YTITLE = *texts*	Titles for the y-axes; default * forms titles automatically from the identifiers or labels of the y-variables
XTITLE = *texts*	Title for the x-axis in each set of graphs; default * uses the identifier of TIMEPOINTS (if set)
NROWS = *scalar*	Specifies the number of rows of graphs to appear on the graphics screen; default * takes the number of y-variables
NCOLUMNS = *scalar*	Specifies the number of columns of graphs to appear on the graphics screen; default 1

Parameter

SAVE = *pointers*	Save structure from KALMAN with information about the analysis; default plots information from the most recent KALMAN analysis

DKEEP directive

Saves information from the last plot on a particular device.

No options

Parameters

DEVICE = *scalars*	The devices for which information is required, if the scalar is undefined or contains a missing value, this returns the current device number
WINDOW = *scalars*	Window about which the information is required; default * gives information about the last window
XLOWER = *scalars*	Lower bound for the x-axis in last graph in the specified device and window
XUPPER = *scalars*	Upper bound for the x-axis in last graph in the specified device

and window

YLOWER = *scalars* Lower bound for the y-axis in last graph in the specified device and window

YUPPER = *scalars* Upper bound for the y-axis in last graph in the specified device and window

ZLOWER = *scalars* Lower bound for the z-axis in last graph in the specified device and window

ZUPPER = *scalars* Upper bound for the z-axis in last graph in the specified device and window

FILE = *scalars* Returns the value 1 or 0 to indicate whether a file is required for this device

DESCRIPTION = *texts* Description of the device

DREAD = *scalars* Returns the value 1 or 0 to indicate whether graphical input is possible from this device

ENDACTION = *texts* Returns the current ENDACTION setting ('continue' or 'pause')

DKEY procedure

Adds a key to a graph (D.B. Baird & V.M. Cave).

Options

WINDOW = *scalar* Window in which to draw the key; default 2

NCOLUMNS = *scalar* Number of columns forming the grid in which the key is displayed; default * (i.e. set automatically)

NROWS = *scalar* Number of rows forming the grid in which the key is displayed; default * (i.e. set automatically)

TITLE = *text* Title for the key

PENTITLE = *scalar* Pen used to write the title of the key; default is that set for the window in which the key is plotted

PENLABELS = *variate* Pens to use to plot the labels; default is to plot the labels using the settings of LFONT, LSIZE and LCOLOUR

TPOSITION = *string* Position of the title (inside, outside, left, centre, center, right); default cent, outs

ORDER = *string* Order in which to fill the key's row by column grid (rows, columns); default rows

LSIZE = *scalar* Relative size of the labels; default 1

LFONT = *scalar* or *text* Font to use for the labels; default 1

LCOLOUR = *scalar* or *text* Colour used to write the labels; default 'black'

XLOFFSET = *scalar* or *variate* Offset in the x-direction between the items (i.e. symbols/lines) and labels in the key; default 0

COLSPACING = *string* Column spacing (equal, unequal); default equa

ROWGAP = *scalar* Multiplier for gaps between rows; default 1

COLGAP = *scalar* Multiplier for gaps between columns; default 1

BORDER = *string* Border around the key (fit, given, none); default fit

CBORDER = *string* Colour for the border around the key; default 'black'

Parameters

DESCRIPTIONS = *texts* Labels for the key

PEN = *variates* Pens to use for the items in the key; default uses the integers 1, 2 ...

METHOD = *texts* Method for plotting the items in the key (fill, point, line, both, none); default is to use the method defined for the corresponding PEN

SYMBOL = *variates*, *scalars*, *factors* or *texts* Symbols to be drawn in the key; default is to use those specified by PEN

COLOUR = *variate*s, *scalars*, *factors* or *texts* Colours of lines, or of filled areas when METHOD='fill';

default is to use those specified by PEN

CSYMBOL = *variates*, *scalars*, *factors* or *texts*
Colours of symbols; default is to use those specified by PEN

CFILL = *variates*, *scalars*, *factors* or *texts*
Colours used to fill hollow symbols; default is to use those specified by PEN

SIZEMULTIPLIER = *variates*, *scalars* or *factors*
Relative sizes of symbols and filled area; default is to use those specified by PEN

LINESTYLE = *variates*, *scalars*, *factors* or *texts*
Numbers or names of the linestyles to use; default is to use those specified by PEN

THICKNESS = *variates*, *scalars* or *factors*
Thicknesses of the lines; default is to use those specified by PEN

TRANSPARENCY = *variates*, *scalars* or *factors*
Transparencies of the filled areas when METHOD='fill'; default is to use those specified by PEN

DKSTPLOT procedure

Produces diagnostic plots for space-time clustering (D.A. Murray).

Options

PLOT = *string token* — Whether to produce plots separately or in composite (separate, combined); default comb

DZERO = *string token* — Whether to produce a DZERO plot (yes, no); default no

Parameters

Y = *variates* — Vertical coordinates of the spatial point patterns

X = *variates* — Horizontal coordinates of the spatial point patterns

KS = *variates* — Estimates of spatial K function

KT = *variates* — Estimates of temporal K function

KST = *matrices* — Estimates of space-time K function

KSE = *matrices* — Estimates of standard errors of space-time K function

DLOAD directive

Loads the graphics environment settings from an external file.

No options

Parameter

text — File from which to lead the environment settings

DMADENSITY procedure

Plots the empirical CDF or PDF (kernel smoothed) by groups (D.B. Baird).

Options

PLOT = *string tokens* — What to plot (cdf, pdf, histogram); default cdf, pdf

TRANSFORMATION = *string token* — Whether to transform the data to log base 2 (log2, none); default none

BANDWIDTH = *scalar* — Bandwidth to use in kernel density estimates for PDF

ARRANGEMENT = *string token* — Whether to use trellis or single plots (single, trellis); default trel

WINDOW = *scalar* — Window number for the graphs; default 3

KEYWINDOW = *scalar* — Window number for the key; default 0 i.e. none

DEVICE = *scalar* — Device number on which to plot the graphs

GRAPHICSFILE = *text* — What graphics filename template to use to save the graphs; default *

Parameters

DATA = *variates* or *pointers* — Data coordinates

GROUPS = *factors* or *texts* — Groups

DMASS procedure

Plots discrete data like mass spectra, discrete probability functions (J.W. McNicol).

Options

X = *variate*	Positions on the x-axis at which to plot the lines; default uses 1, 2 ...
TITLE = *text*	Title for the graph; default * i.e. none
WINDOW = *scalar*	Window for the graph; default 3
YTITLE = *texts*	Title for the y-axis
XTITLE = *texts*	Title for the x-axis
YMARKS = *scalars* or *variates*	Distance between each tick mark on y-axis (scalar) or positions of the marks (variate)
XMARKS = *scalars* or *variates*	Distance between each tick mark on x-axis (scalar) or positions of the marks (variate)
YMPOSITION = *string tokens*	Position of the tick marks across the y-axis (left, right, centre); default left
XMPOSITION = *string tokens*	Position of the tick marks across the x-axis (above, below, centre); default * i.e. none
YLABELS = *texts*	Labels at each mark on y-axis
XLABELS = *texts*	Labels at each mark on x-axis
PENAXES = *scalar*	Pen to be used for axes and their titles; default 1
PENTITLE = *scalar*	Pen to use for the title; default 1
LINETHICKNESS = *scalar*	Thickness for the vertical lines representing the mass heights; default 1
SCREEN = *string token*	Whether to clear screen before displaying the graph (keep, clear); default clea

Parameters

Y = *variates*	Heights for the masses
LINECOLOUR = *texts* or *scalars*	Colours for the vertical lines representing mass heights; default * sets suitable colours automatically

DMSCATTER procedure

Produces a scatter-plot matrix for one or two sets of variables (J. Ollerton & R.W. Payne).

Options

PLOT = *string tokens*	Additional information to include in the scatter plots (correlation, histograms, boxplots, densities, dothistograms); default *
SCALING = *string token*	How to scale the x- and y-axes (common, equal, none); default none
PEN = *scalar* or *variate* or *factor*	Pens to plot the scatter plots; default 1
PENHISTOGRAM = *scalar*	Pens to plot the histograms; if PEN is a factor the default plots the histogram for each group separately using the pen used for that group in the scatter plots, otherwise the default is to use pen 2
PENCORRELATION = *scalar*	Pen to use to write the correlations; default 1
PENTITLE = *scalar*	Pen to use to write the axis titles; default uses the pens currently defined for the axes in the windows that are used for the plots
PENAXIS = *scalar*	Pen to use to draw the axes; default uses the currently defined pens
PENLABELS = *scalar*	Pen to use to write the axis labels; default uses the currently defined pens
NROWS = *scalar*	Number of rows of graphs to put in a single frame (i.e. page); default puts them all in one frame
NCOLUMNS = *scalar*	Number of columns of graphs to put in a single frame; default uses the same value as NROWS

ASPECTRATIO = *scalar* — Ratio of the length of the y-axis to the length of the x-axis in each graph

FRAMESHAPE = *string token* — Shape of the plotting frame (`landscape`, `portrait`, `square`); default `squa`

MARGINSIZE = *scalar* — Specifies the size of the margins at the bottom and left-hand edge of the frame

Parameters

Y = *pointers* — Each pointer contains a set of variates and/or factors to be plotted

YTITLES = *texts* — Labels for the axes for the `Y` variates and factors, to use instead of their identifiers

YMARKS = *variates*, *scalars* or *pointers* — Marks to use on the axes for the `Y` variates and factors, if any of these contains missing values, the marks and their labels are suppressed for that variate or factor

X = *pointers* — Each pointer contains a set of variates and/or factors to be plotted as the x-variables in a rectangular scatter-plot matrix; if unset `Y` specifies both the x-variables and y-variables for a symmetric scatter-plot matrix

XTITLES = *texts* — Labels for the axes for the `X` variates and factors, to use instead of their identifiers

YMARKS = *variates*, *scalars* or *pointers* — Marks to use on the axes for the `Y` variates and factors, if any of these contains missing values, the marks and their labels are suppressed for that variate or factor

XMARKS = *variates*, *scalars* or *pointers* — Marks to use on the axes for the `X` variates and factors, if any of these contains missing values, the marks and their labels are suppressed for that variate or factor

DMST procedure

Gives a high resolution plot of an ordination with minimum spanning tree (A.W.A. Murray).

Options

DIMENSIONS = *scalars* — Two numbers specifying the dimensions to display on the y- and x-axes; default 2,1

TITLE = *text* — Title for the graph

WINDOW = *scalar* — Window for the graph; default 1

KEYWINDOW = *scalar* — Window for the key; default 2

SCREEN = *string token* — Controls screen (`clear`, `keep`); default `clea`

Parameters

COORDINATES = *matrices* or *datamatrices* — Coordinates from ordination

TREE = *matrices* — Minimum spanning tree

SIMILARITY = *symmetric matrices* — Association matrix used to derive ordination

SYMBOLS = *factors* or *texts* — Symbols to label the coordinates

PENCOORDINATES = *scalars* — Pen to use for the coordinates

PENTREE = *scalars* — Pen to use for the minimum spanning tree

DOTHISTOGRAM procedure

Plots dot histograms (L.S. Schmitt).

Options

TITLE = *text* — Title for the plot; default `*` i.e. none

AXISTITLE = *text* — Title for the axis representing the data values; default `*` uses the name of the `DATA` variate if there is only one, otherwise no title

WINDOW = *scalars* — Window for the plot; default `*` uses window 1 when `PEN` is set,

and window 3 when PEN is unset

ORIENTATION = *string token* — Direction of the plot (horizontal, vertical); default vert

YORIENTATION = *string token* — Direction of the y-axis for horizontal plots (reverse, normal); default reve

SCREEN = *string token* — Whether to clear screen before displaying chart (keep, clear); default clea

JUSTIFICATION = *string token* — How to position the dots; (right, left, centre, center, bottom, top, backtoback); default cent

CREATEMISSINGLEVEL = *text* — Whether to create a level for missing GROUPS data (yes, no); default no

OMITEMPTYLEVELS = *text* — Whether to omit levels of GROUPS for which there are no DATA values to plot (yes, no); default no

SIZE = *scalar* — Size of the pen used to plot the dots; default 1

KEYWINDOW = *scalar* — Window to use for a key when PEN is set; default 2

KEYDESCRIPTION = *text* — Overall title for a key when PEN is set; default * uses name of PEN data structure

SELECTION = *string tokens* — Selects the statistics to be plotted (mean, median, interquartilerange); default * i.e. none

BARWIDTH = *scalar* — Width of the bars for the selected statistics; default * sets an appropriate width automatically

BARTHICKNESS = *scalar* — Thickness of the bars for the selected statistics; default 2

CMEAN = *scalar*, *variate* or *text* — Colour of the bars for the means

CMEDIAN = *scalar*, *variate* or *text* — Colour of the bars for the medians

CINTERQUARTILE = *scalar*, *variate* or *text* — Colour of the bars for the inter-quartile ranges

Parameters

DATA = *variates* or *pointers* — Data to be plotted

GROUPS = *factors* — Factor to divide values of a DATA variate into groups

COLOURS = *scalars*, *variates*, *texts* or *factors* — Colours for the histograms in each plot, a scalar to use the same colour for all the histograms, or a variate or factor to plot each histogram in a different colour; default 'black'

NOBSERVATIONS = *tables* — Save tables of count

PEN = *variates*, *factors* or *pointers* — Pens to define colours for the individual dots; default uses those defined by the COLOURS parameter

SYMBOLS = *scalars*, *variates*, *texts* or *factors* — Symbols for the points

DESCRIPTION = *texts* — Annotation for key when PEN is set; default uses unique values of PEN

DOTPLOT procedure

Produces a dot-plot using line-printer or high-resolution graphics (J. Ollerton & S.A. Harding).

Options

GRAPHICS = *string token* — Whether to use high-resolution graphics or line-printer graphics (lineprinter, highresolution); default high

TITLE = *text* — Title for the Dot Plot; default *

WINDOW = *scalar* — Window number for the graph; default 1

SCREEN = *string token* — Whether to clear the screen before plotting or to or continue plotting on the old screen (clear, keep); default clea

ENDACTION = *string token* — Action to be taken after completing the plot (continue, pause); default * uses the current setting

DIRECTION = *string token* — Order in which to sort the data before plotting, DIRECTION=* implies plot unsorted data (ascending, descending); default asce

LINES = *string token* — How to draw guide lines on the plot, LINES=* omits the guide lines (todot, full); default todot draws lines from the *x*-

origin to the dots

Parameters

YLABELS = *texts*	Text specifying Y labels for each dotplot
X = *variates*	Data to be plotted
PENDOTS = *scalars*	Pen to draw the dots; default 1
PENLINES = *scalars*	Pen to draw the lines; default 2

DPARALLEL procedure

Displays multivariate data using parallel coordinates (Z. Karaman).

Options

TITLE = *text*	Title for the plot
GROUPS = *factor*	Defines grouping of the units (if any); by default, different pens are used for the observations in different groups
PERMUTATIONSALL = *string token*	Whether to display all necessary permutations so that any two variates will be adjacent in at least one plot, or just display once in the order given by the DATA pointer (yes, no); default no
SCALING = *string token*	Whether to do scaling overall (scale all variates on the same scale), or to scale each variate separately (overall, separate); default sepa
PEN = *variate*	Pens to be used for different groups (if any); default * uses pens from 1 up to the number of groups (number of levels of the GROUPS factor)

Parameter

DATA = *variates*	Data variables to be plotted

DPIE directive

Draws a pie chart on a plotter or graphics monitor.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the pie chart; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
ANNOTATION = *string token*	Whether to annotate the slices by their percentages (percentages); default perc
OUTLINE = *string token*	Where to draw outlines (slices, perimeter); default slices
PENOUTLINE = *scalar*	Pen to use for the outlines; default −10
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

SLICE = *scalars*	Amounts in each of the slices (or categories)
PEN = *scalars*	Pen number for each slice; default * uses pens 1, 2, and so on for the successive slices
DESCRIPTION = *texts*	Description of each slice

DPOLYGON procedure

Draws polygons using high-resolution graphics (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Options

TITLE = *text*	Main title for the plot; default *
WINDOW = *scalar*	Which graphics window to use for the plot; default 1
KEYWINDOW = *scalar*	Which graphics window to use for the key; default 2

YTITLE = *text* Title for the vertical axis; default *
XTITLE = *text* Title for the horizontal axis; default *
YLOWER = *scalar* Lower limit for the vertical axis
YUPPER = *scalar* Upper limit for the vertical axis
XLOWER = *scalar* Lower limit for the horizontal axis
XUPPER = *scalar* Upper limit for the horizontal axis
SCREEN = *string token* Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text* Overall description for the key; default *
ENDACTION = *string token* Action to be taken after completing the plot (continue, pause); default paus

Parameters

YPOLYGON = *variates* Vertical coordinates of one or more polygons; no default – this parameter must be set
XPOLYGON = *variates* Horizontal coordinates of one or more polygons; no default – this parameter must be set
PEN = *scalars* or *variates* or *factors* Pen number for each graph
DESCRIPTION = *texts* Annotation for the key

DPROBABILITY procedure

Creates a probability distribution plot of the values in a variate (D.B. Baird).

Options

PRINT = *string tokens* Controls whether to print estimated parameters of the distribution or test statistics (parameters, tests); default para
DISTRIBUTION = *string token* Distribution for expected values against which to plot values (normal, stdnormal, lognormal, exponential, gamma, weibull, beta, b2, pareto, chisquare, cauchy, logistic, ev1, ev2, ev3, gev, invnormal, t, f, uniform, stduniform, laplace, gpareto, ubetamix, ugammamix, loggamma, loglogistic, paralogistic, igamma, iweibull, burr, iburr); default norm
METHOD = *string token* Method used for the plot axes (quantile, probability, stabilizedprobability); default quan
GRAPHICS = *string token* Type of graphics (highresolution, lineprinter); default high
PLOT = *string tokens* Whether to plot differences from expectations or the 1-1 reference line (differences, reference); default refe
CONSTANT = *string token* Whether to estimate the constant for the distribution (estimate, omit) default omit
BANDS = *string token* What type of confidence bands to plot, if any (simultaneous, pointwise); default simu
NSIMULATIONS = *scalar* Number of simulations for pointwise bands; default 100
ALPHA = *scalar* Acceptance limits for confidence bands; default 0.95
DF = *scalar* Number of degrees of freedom of chi-square or t distribution; default 1
DFNUMERATOR = *scalar* Numerator degrees of freedom of F distribution; default 1
DFDENOMINATOR = *scalar* Denominator degrees of freedom of F distribution; default 1
WINDOW = *scalar* Window to use for the plot; default 3
XMETHOD = *string token* Scaling of X / Expected Plot axes (quantile, probability, stabilizedprobability); if unset, takes the same setting as METHOD
QMETHOD = *string token* Whether to standardize plotted score in expected quantiles (standardized, unstandardized); default stan
TMETHOD = *string tokens* Specifies the method used to perform the goodness-of-fit tests (likelihoodratio, traditional); default like

NTIMES = *scalar* — Number of Monte-Carlo simulations to perform for likelihood-ratio tests; default 999

SEED = *scalar* — Seed for random number generation for the likelihood-ratio tests; default 0 continues an existing sequence or, if none, selects a seed automatically

Parameters

DATA = *variates* — Values to plot

TITLE = *text* — Title for the graph; default * generates an appropriate title automatically

ESTIMATES = *variates* — Saves the estimated parameters for the distribution

SE = *variates* — Saves standard errors for the estimated parameters

LOWERTRUNCATION = *scalars* — Lower truncation points for Loss distributions

UPPERTRUNCATION = *scalars* — Upper truncation points for Loss distributions

DEVIANCE = *scalars* — Saves the deviance for the fitted distribution

PROBABILITIES = *variates* — Saves the probabilities from the goodness-of-fit tests

DPSPECTRALPLOT procedure

Calculates an estimate of the spectrum of a spatial point pattern (C.J. Alexander & D.A.Murray).

Options

PLOT = *string tokens* — Which graphs to plot (periodogram, rspectrum, thetaspectrum, weights); default peri, rspe, thet, weig

NROWS = *scalar* — Number of rows for periodogram; default 17

NCOLUMNS = *scalar* — Number of columns for periodogram; default 32

SCALING = *string token* — Whether to normalize the coordinates of the points within the study region to a unit square (normalize, none); default norm

Parameters

Y = *variates* — Vertical coordinates of each spatial point pattern

X = *variates* — Horizontal coordinates of each spatial point pattern

YPOLGON = *variates* — Y-coordinates for the rectangular study region

XPOLYGON = *variates* — X-coordinates for the rectangular study region

YHOLEPOLGON = *variates* — Y-coordinates for the missing region polygons

XHOLEPOLYGON = *variates* — X-coordinates for the missing region polygons

HOLEGROUPS = *variates* — Grouping factor where each level represents a different polygon for the missing regions.

PERIODOGRAM = *matrices* — Saves the periodogram

WEIGHTS = *variates* — Saves the weights used for the inter-event calculation

YINTEREVENT = *variates* — Saves the y-coordinates for the inter-event calculation

XINTEREVENT = *variates* — Saves the x-coordinates for the inter-event calculation

DPTMAP procedure

Draws maps for spatial point patterns using high-resolution graphics (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Options

TITLE = *text* — Main title for the plot; default *

WINDOW = *scalar* — Which graphics window to use for the plot; default 1

KEYWINDOW = *scalar* — Which graphics window to use for the key; default 2

YTITLE = *text* — Title for the vertical axis; default *

XTITLE = *text* — Title for the horizontal axis; default *

YLOWER = *scalar* — Lower limit for the vertical axis

YUPPER = *scalar* — Upper limit for the vertical axis

XLOWER = *scalar* — Lower limit for the horizontal axis

XUPPER = *scalar* — Upper limit for the horizontal axis

SCREEN = *string token* — Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea

KEYDESCRIPTION = *text* — Overall description for the key; default *
ENDACTION = *string token* — Action to be taken after completing the plot (continue, pause); default paus

Parameters

Y = *variates* — Vertical coordinates of one or more spatial point patterns; no default – this parameter must be set
X = *variates* — Horizontal coordinates of one or more spatial point patterns; no default – this parameter must be set
PEN = *scalars* or *variates* or *factors* — Pen number for each graph
DESCRIPTION = *texts* — Annotation for the key

DPTREAD procedure

Adds points interactively to a spatial point pattern (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Options

PRINT = *string token* — What to print (summary, monitoring); default summ, moni
WINDOW = *scalar* — Which graphics window to use for the plot; default 1

Parameters

OLDY = *variates* — Vertical coordinates of each spatial point pattern; no default – this parameter must be set
OLDY = *variates* — Horizontal coordinates of each spatial point pattern; no default – this parameter must be set
NEWY = *variates* — Variates to receive the vertical coordinates of the original points and added points
NEWX = *variates* — Variates to receive the horizontal coordinates of the original points and added points

DQMAP procedure

Displays a genetic map (D.A. Murray).

Options

ORIENTATION = *string token* — Orientation of map (vertical, horizontal); default vert
DCHROMOSOMES = *variate*, *text* or *scalar* — To specify a subset of the linkage groups to be displayed
TITLE = *text* — General title; default *

Parameters

CHROMOSOMES = *factors* — Factor defining the linkage groups
POSITIONS = *variates* or *pointers* — Positions of markers within the linkage groups
MKNAMES = *texts* — Names of the markers
QCHROMOSOMES = *factors* — Factor defining the linkage groups of the QTLs
QPOSITIONS = *variates* — Positions of QTLs within the linkage groups
QNAMES = *texts* — Names of the QTLs
QINTERACTIONS = *variates* — Logical variate indicating whether the QTL has significant (1) or non-significant (0) QTL-by-environment interaction

DQMKSCORES procedure

Plots a grid of marker scores for genotypes and indicates missing data (D.A. Murray).

Options

PLOT = *string token* — Type of plot (missing, all); default miss
LOWERGENOTYPE = *scalar* — Lower genotype for the display
UPPERGENOTYPE = *scalar* — Upper genotype for the display
DCHROMOSOMES = *variate*, *text* or *scalar* — Specify a subset of the linkage groups to be displayed
POPULATIONTYPE = *string token* — Type of population (BC1, DH1, F2, RIL, BCxSy, AMP); must be set
COLOURS = *text* or *variate* — Colours to use for the different marker scores
TITLE = *text* — Title for the graph

Parameters

MKSCORES= *pointers*	Marker score code for each marker
CHROMOSOMES = *factors*	Linkage group for each marker
PARENTS = *pointers*	Parent information
IDPARENTS = *texts*	Labels to identify the parents

DQMQTLSCAN procedure

Plots the results of a genome-wide scan for QTL effects in multi-environment trials (M.P. Boer & J.T.N.M. Thissen).

Options

POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set
METHOD = *string token*	Method to be used for plotting (line, spikes); default line
THRESHOLD = *scalar*	Threshold value for test statistic; default 0
DCHROMOSOMES = *scalar*, *text* or *variate*	Allows a subset chromosomes to be specified to display; default * i.e. all the chromosomes
SUPPRESSLINES = *string token*	Whether to suppress the vertical lines between the chromosomes (yes, no); default no
SYMBOL = *scalar*	Defines the plotting symbol for each point, as in the SYMBOL option of PEN, when METHOD=manhattan; default 2 i.e. circle
SIZEMULTIPLIER = *scalar*	Multiplier used in the calculation of sizes of symbols when METHOD=manhattan; default 1
BLACKOUTLINE = *string token*	Whether to draw the outer line the SYMBOL in black when METHOD=manhattan (yes, no); default no
COLOURS = *scalar*, *variate* or *text*	Colours to use for the chromosomes; default * uses the default colours of pens 1, 2 up to the number of chromosomes
TITLE = *text*	General title
YLOWERTITLE = *text*	Title for the y-axis of the lower graph; default 'Environments'
YUPPERTITLE = *text*	Title for the y-axis of the upper graph; default uses the identifier of the STATISTICS variate or pointer
XTITLE = *text*	Title for the x-axis; default 'Chromosomes'
YAXUPPER = *scalar*	Upper bound for y-axis of the upper graph
ANNOTATION = *string token*	Whether to include annotation of the effects in the plot (include, omit); default incl

Parameters

STATISTICS = *variates* or *pointers*	Test statistics to be plotted; must be set
CHROMOSOMES = *factors*	Chromosome for each locus; must be set
POSITIONS = *variates*	Positions on the chromosome of each locus; must be set
QEFFECTS = *pointers*	QTL effects in the different environments; must be set
QSE = *pointers*	Standard errors of the QTL effects in the different environments; must be set
ENVNAMES = *texts*	Labels for the different environments; must be set
IDEFFECTS = *texts*	Labels to use to identify the effects
IDPARENTS = *texts*	Labels to use to identify the parents
DFILENAME = *texts*	Name of the graphics file for the plots

DQRECOMBINATIONS procedure

Plots a matrix of recombination frequencies between markers (S.J. Welham & D.A. Murray).

Options

DCHROMOSOMES = *scalar*, *variate* or *text*	Specifies a subset of the linkage groups to be displayed
TITLE = *text*	General title for the plot
WINDOW = *scalar*	Window number for the graph; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2

PALETTE = *string token* Colour scheme for plot (colour, color, greyscale, grayscale); default colo

Parameters

RECFREQUENCIES = *symmetric matrices* Recombination frequencies to plot

CHROMOSOMES = *factors* Linkage group for each marker

DQSQTLSCAN procedure

Plots the results of a genome-wide scan for QTL effects in single-environment trials (M.P. Boer & J.T.N.M. Thissen).

Options

POPULATIONTYPE = *string token* Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set when QEFFECTS are supplied

METHOD = *string token* Method to be used for plotting (line, manhattan, spikes); default line

THRESHOLD = *scalar* Threshold value for test statistic; default 0

DCHROMOSOMES = *scalar*, *text* or *variate* Allows a subset chromosomes to be specified to display; default * i.e. all the chromosomes

SUPPRESSLINES = *string token* Whether to suppress the vertical lines between the chromosomes (yes, no); default no

SYMBOL = *scalar* Defines the plotting symbol for each point, as in the SYMBOL option of PEN, when METHOD=manhattan; default 2 i.e. circle

SIZEMULTIPLIER = *scalar* Multiplier used in the calculation of sizes of symbols when METHOD=manhattan; default 1

BLACKOUTLINE = *string token* Whether to draw the outer line the SYMBOL in black when METHOD=manhattan (yes, no); default no

COLOURS = *scalar*, *variate* or *text* Colours to use for the chromosomes; default * uses the default colours of pens 1, 2 up to the number of chromosomes

TITLE = *text* General title

YTITLE = *text* Title for the y-axis; default uses the identifier of the STATISTICS variate or pointer

XTITLE = *text* Title for the x-axis; default 'Chromosomes'

YUPPER = *scalar* Upper bound for y-axis

WINDOW = *scalar* Window number for the graphs; default 1

KEYWINDOW = *scalar* Window number for key (zero for none); default 2

SCREEN = *string token* Whether to clear the screen before displaying the graph (clear, keep); default clea

Parameters

STATISTICS = *variates* or *pointers* Test statistic(s) to be plotted; must be set

CHROMOSOMES = *factors* Chromosome for each locus; must be set

POSITIONS = *variates* Position on the chromosome for each locus; must be set

QEFFECTS = *variates* or *pointers* QTL effects along the genome,

QSE = *variates* or *pointers* Standard errors of the QTL effects

IDEFFECTS = *texts* Labels along the x-axis to identify the effects when QEFFECTS are supplied

IDPARENTS = *texts* Labels to use to identify the parents

DFILENAME = *texts* Name of the graphics file for the plots

DREAD directive

Reads the locations of points from an interactive graphical device.

Options

PRINT = *string tokens* What to print (data, summary); default summ

CHANNEL = *scalar* Number of the graphics device from which to read; default * takes the current graphics device

WINDOW = *scalar* Window from which to read; default 1

CURSORTYPE = *scalar* — Type of cursor; default 1
SETNVALUES = *string token* — Whether to set number of values of structures from the number of values read (yes, no); default no causes the number of values to be set only for structures whose lengths are not defined already
ENDACTION = *string token* — Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

Y = *variates* — Variate to receive the y-values that have been read
X = *variates* — Variate to receive the x-values that have been read
YGIVEN = *variates* — Y-coordinates of points that may be located on the graph
XGIVEN = *variates* — X-coordinates of points that may be located
SAVESET = *variates* — Unit numbers of the located points
PEN = *scalars* — Pen number to use to echo points; default 0
YSAVE = *variates* — Variate to receive the y-coordinates of the located points
XSAVE = *variates* — Variate to receive the x-coordinates of the located points

DREFERENCELINE procedure

Adds reference lines to a graph (R.W. Payne).

Options

ORIENTATION = *string token* — Direction of the line (horizontal, vertical); default hori
WINDOW = *scalar* — Window in which to draw the line; default 1

Parameters

POSITION = *scalars* — Positions of the lines
PEN = *scalars* — Pen to use for each line
LABEL = *texts* — Text to plot alongside each line
YLPOSITION = *string tokens* — Position of the label in the y-direction (above, below, centre, center); default belo
XLPOSITION = *string tokens* — Position of the label in the x-direction (centre, center, left, right); default left
PENLABEL = *scalars* — Pen to use for each label

DREPMEASURES procedure

Plots profiles and differences of profiles for repeated measures data (J.T.N.M. Thissen).

Options

TITLE = *text* — Title for the plots; default *
GROUPS = *factors* — List of one or two factors; one factor gives one plot while a list with two factors gives as many plots as the number of levels of the first factor in the list; must be set
TIMEPOINTS = *variate* or *factor* — When the DATA parameter is set to a pointer containing a separate variate of observations for each time this can specify the actual time points (otherwise the suffixes of the DATA pointer are used), when there is a single DATA variate this must supply a factor to indicate the time of each observation
DIFFERENCES = *string token* — Can suppress plotting of the differences (no, yes); default no

Parameters

DATA = *pointers* or *variates* — Data observations either in a pointer to a list of variates (one for each time), or a single variate (with TIMEPOINTS set to a factor indicating the time of each observation)
GROUPMEANS = *tables* — To save the calculated treatment means at each timepoint

DRESIDUALS procedure

Plots residuals (R.W. Payne).

Options

RESIDUALS = *variate* — Residuals to plot

FITTEDVALUES = *variate* — Fitted values against which to plot the residuals
INDEX = *variate* or *factor* — X-variable for an index plot; default !(1,2...)
GRAPHICS = *string token* — What type of graphics to use (lineprinter, highresolution); default high
TITLE = *text* — Overall title for the plots; default * i.e. none

Parameters

METHOD = *string tokens* — Type of residual plot (fittedvalues, normal, halfnormal, histogram, absresidual, index); default fitt, norm, half, hist
PEN = *scalars*, *variates* or *factors* — Pen(s) to use for each plot

DROP directive

Drops terms from a linear, generalized linear, generalized additive or nonlinear model.

Options

PRINT = *string tokens* — What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence); default mode, summ, esti
NONLINEAR = *string token* — How to treat nonlinear parameters between groups (common, separate, unchanged); default unch
CONSTANT = *string token* — How to treat the constant (estimate, omit, unchanged, ignore); default unch
FACTORIAL = *scalar* — Limit for expansion of model terms; default * i.e. that in previous TERMS statement
POOL = *string token* — Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token* — Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens* — Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, df, inflation); default *
FPROBABILITY = *string token* — Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token* — Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens* — Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
PROBABILITY = *scalar* — Probability level for confidence intervals for parameter estimates; default 0.95
AOVDESCRIPTION = *text* — Description for line in accumulated analysis of variance (or deviance) table when POOL=yes

Parameter

formula — List of explanatory variates and factors, or model formula

DRPOLYGON procedure

Reads a polygon interactively from the current graphics device (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Options

PRINT = *string token* — What to print (summary); default summ
WINDOW = *scalar* — Window from which to read default 1

Parameters

YPOLYGON = *variates*	Variates to receive the vertical coordinates of the polygons that are read
XPOLYGON = *variates*	Variates to receive the horizontal coordinates of the polygons that are read
PEN = *scalars*	Pen numbers to use to echo points

DSAVE directive

Saves the current graphics environment settings to an external file.

No options

Parameters

FILENAME = *text*	File in which to save the environment settings
DESCRIPTION = *text*	Description for these settings

DSCATTER procedure

Produces a scatter-plot matrix using high-resolution graphics (J. Ollerton).

Options

PEN = *scalar* or *variate* or *factor*	Pen number for the graph; default 1
EQUALSCALING = *string token*	Whether to have equal scaling of x- and y-axes in each plot (yes, no); default no
XDATA = *variates* or *factors*	Variables to be plotted as x-coordinates (DATA then specifies the y-coordinates); if unset DATA specifies both x-coordinates and y-coordinates
ASPECTRATIO = *scalar*	Ratio of the length of the y-axis to the length of the x-axis in each plot

Parameter

DATA = *variates* or *factors*	A list of variables to be plotted

DSEPARATIONPLOT procedure

Creates a separation plot for visualising the fit of a model with a dichotomous (i.e. binary) or polytomous (i.e. multi-categorical) outcome (V.M. Cave).

Options

METHOD = *string token*	Method used to plot the predicted probabilities (rectangles, lines, rbands, lbands); default rect
PLOT = *string tokens*	Information to be plotted on the graph (key, traceline, expectednumber); default key, trac, expe when METHOD=rectangles or lines, and key when METHOD=rbands or lbands
SUCCESSLEVEL = *string token*	Specifies which level corresponds to success when GROUPS supplies a factor with 2 levels (first, second); default seco
LINEORDER = *string token*	If METHOD=lines, whether the failures or successes are plotted first (failurefirst, successfirst); default fail
NGROUPS = *scalar*	Number of discrete bands used to group the predicted probabilities when METHOD=rbands or lbands; default 10
TIES = *string token*	How tied data values in PROBABILITIES are handled when METHOD=rectangles or lines (permute, same); default perm
SEED = *scalar*	Seed for random number generator used to permute the tied data; default 0
COLOURS = *variate or text*	The two colours used to plot the predicted probabilities
THICKNESS = *scalar*	Thickness of the line for plotting the predicted probabilities when METHOD=lines or lbands; default 1
BACKGROUND = *scalar or text*	Colour of the background when METHOD=lines or lbands; default ligh
BORDER = *string token*	Whether to draw borders around the rectangles when METHOD=rectangles or rbands (yes, no); default no

USEPENS = *string token*	Whether to use the current pen definitions of pens 2 and 3 for plotting the traceline and expectednumber. respectively (yes, no); default no
SAVE = *rsave* or *pointer*	Regression or HGLM save structure to provide the data if PROBABILITIES, GROUPS, NSUCCESSES and NBINOMIAL are not specified

Parameters

PROBABILITIES = *variate or matrix*	Variate containing probabilities of success for a binary outcome (i.e. for binary or binomial data), or matrix containing probabilities of membership in each group for a polytomous outcome
GROUPS = *variate or factor*	Actual outcome, when NSUCCESSES and NBINOMIAL are not supplied
NSUCCESSES = *variate*	Number of successes when PROBABILITIES supplies predicted probabilities from binomial data
†NBINOMIAL = *variate* or *scalar*	Number of trials when PROBABILITIES supplies predicted probabilities from binomial data
TITLE = *text*	Title for the plot; default generates the title automatically
XTITLE = *text*	Title for the x-axis; default * i.e. none

DSHADE directive

Plots a shade diagram of 3-dimensional data.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the graph; default 1
KEYWINDOW = *scalar*	Window number for the key (0 for no key); default 2
YORIENTATION = *string token*	Y-axis orientation of the plot (reverse, normal); default reve
GRIDMETHOD = *string token*	How to draw a grid around the elements of the matrix (present, complete); default pres
PENGRID = scalar	Pen to use for the grid; default -7
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

GRID = *symmetric matrix*, *matrix* or *pointer to variates*	Data to be plotted
PEN = *scalar* or *variate*	How to draw each shade
LIMITS = *variate*	Boundary values for changes in shade
NGROUPS = *scalar*	Number of groups to form from the data values (i.e. number of different shades)
INTERVAL = *scalar*	Interval between changes in shade
PERMUTATION = *variate*	Can define permutations to be done to the units of symmetric matrices prior to plotting
DESCRIPTION = *text*	Annotation for key

DSPIDERWEB procedure

Displays spider-web and star plots (W. van den Berg).

Options

METHOD = *string token*	Type of plot (spiderweb, star); default spid
MARKS = *scalar* or *variate*	Distances between the strands of the web or marks on the axes of the star (scalar), or positions of those strands or marks (variate); default 0.25

ANGLE = *scalar*	Angle to rotate the plot, in degrees; default 0
SIZEMULTIPLIER = *scalar*	Controls the size of the labels identifying the categories; default selects a size appropriate to the number of plots in the frame
FRAMESHAPE = *string token*	Shape of the plotting frame (landscape, portrait, square); default squa

Parameters

DATA = *tables*	Values to plot in each frame
CATEGORIES = *factors*	Factor specifying the categories that define the axes in the plots
GROUPS = *factors* or *pointers*	Factor specifying the groups to appear in each plot
TRELLISGROUPS = *factors* or *pointers*	Factor or factors specifying the different plots of a trellis plot of a multi-way table
PAGEGROUPS = *factors* or *pointers*	Factor or factors specifying plots to be displayed on different pages
TITLE = *texts*	Title for the graph; default none
COLOURS = *texts* or *variates*	Colours to be used for the groups

DSTART directive

Starts a sequence of related high-resolution plots.

Options

TITLE = *text*	Overall title for the plots
PEN = *scalar*	Pen to use for the title; if this is not set, pen −12 is used

DSTTEST procedure

Plots power and significance for t-tests, including equivalence tests (R.W. Payne).

Options

NSAMPLES = *scalar*	Number of samples for the t-test (1 or 2); default 2
PROBABILITY = *scalar*	Significance level at which the response is to be tested; default 0.05
TMETHOD = *string token*	Type of test to be done (onesided, twosided, equivalence, noninferiority); default ones
RATIOREPLICATION = *scalar*	Ratio of replication sample2:sample1 (i.e. the size of sample 2 should be RATIOREPLICATION times the size of sample 1); default 1

Parameters

RESPONSE = *scalars*	Response to be detected
VAR1 = *scalars*	Anticipated variance of sample 1
VAR2 = *scalars*	Anticipated variance of sample 2; default * assumes the same variance as sample 1
NREPLICATES = *scalars*	Number of replicates
RDF = *scalars*	Number of residual degrees of freedom; default * calculates these automatically, assuming a standard t-test

DSURFACE directive

Produces perspective views of two-way arrays of numbers.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the plots; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
ELEVATION = *scalar*	The elevation of the viewpoint relative to the surface; default 25 (degrees)
AZIMUTH = *scalar*	Rotation about the horizontal plane; the default of 225 degrees ensures that, with a square matrix M, the element M$[1;1] is nearest to the viewpoint
DISTANCE = *scalar*	Distance of the viewpoint from the centre of the grid on the

base plane; default * gives a distance of 100 times the maximum of the x-range and the y-range

ZSCALE = *scalar* — defines the scaling of the z-axis relative to the horizontal (x-y) axes; default 1

SCREEN = *string token* — Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea

KEYDESCRIPTION = *text* — Overall description for the key; default *

ENDACTION = *string token* — Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

GRID = *identifier* — Pointer (of variates representing the columns of a data matrix), matrix or two-way table specifying values on a rectangular grid

PEN = *scalar* — Pen number to be used for the plot; default 1

PENFILL = *scalar* or *variate* — Pen number(s) defining how to fill the areas between contours (0 or * leaves the areas in the background colour); default 3

PENMESH = *scalar* — Pen number to use to draw the mesh (omitted if set to 0 or *); default 1

PENSIDE = *scalar* — Pen number to use to shade the sides of the surface (omitted if set to 0 or *); default *

NCONTOURS = *scalar* — Number of contours; default 10

CONTOURS = *variate* — Positions of contours

INTERVAL = *scalar* — Interval between contours

DESCRIPTION = *text* — Annotation for key

DTABLE procedure

Plots tables (R.W. Payne).

Options

GRAPHICS = *string token* — Type of graph (highresolution, lineprinter); default high

METHOD = *string token* — What to plot (points, linesandpoints, onlylines, data, barchart, splines); default poin

XFREPRESENTATION = *string token* — How to label the *x*-axis (levels, labels); default labels uses the XFACTOR labels, if available

DFSPLINE = *scalar* — Number of degrees of freedom to use when METHOD=splines

YTRANSFORM = *string tokens* — Transformed scale for additional axis marks and labels to be plotted on the right-hand side of the y-axis (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default iden i.e. none

PENYTRANSFORM = *scalar* — Pen to use to plot the transformed axis marks and labels; default * selects a pen, and defines its properties, automatically

†KEYMETHOD = *string token* — What to use for the key descriptions when GROUPS specifies more than one factor (labels, namesandlabels); default name

†PLOTTITLEMETHOD = *string token* — What to use for the titles of the plots when TRELLISGROUPS specifies more than one factor (labels, namesandlabels); default name

†PAGETITLEMETHOD = *string token* — What to use for the titles of the pages when PAGEGROUPS specifies more than one factor (labels, namesandlabels); default name

†USEAXES = *string token* — Which aspects of the current axis definitions of window 1 to use (none, limits, marks, mpositions, nsubticks,); default none

Parameters

TABLE = *tables* — Tables to plot

DATA = *variates*	Data values to plot with each table when METHOD=data
XFACTOR = *factors*	Factor providing the *x*-values for the plot of each table
GROUPS = *factors* or *pointers*	Factor or factors identifying the different lines from a multi-way table
TRELLISGROUPS = *factors* or *pointers*	Factor or factors specifying the different plots of a trellis plot of a multi-way table
PAGEGROUPS = *factors* or *pointers*	Factor or factors specifying plots to be displayed on different pages
BAR = *scalars*, *tables* or *pointers*	Scalar defining the length of error bar to be plotted to indicate the overall (or average) variability of the values in each table, or table defining the variability of each individual table value, or pointer containing either two scalars or two tables defining the upper and lower positions of the error bar(s)
NEWXLEVELS = *variates*	Values to be used for XFACTOR instead of its existing levels
TITLE = *texts*	Title for the graph; default uses the identifier of the TABLE
YTITLE = *texts*	Title for the y-axis; default ' '
XTITLE = *texts*	Title for the x-axis; default is to use the identifier of the XFACTOR
BARDESCRIPTION = *texts*	Descriptions for the bars
PENS = *variates*	Defines the pen to use to plot the points and/or line for each group defined by the GROUPS factors

DTEXT procedure

Adds text to a graph (S.A. Harding).

Option

WINDOW = *scalar*	Window number of the graph; default 1

Parameters

Y = *variates* or *scalars*	Vertical coordinates
X = *variates* or *scalars*	Horizontal coordinates
TEXT = *texts*	Text to plot
PEN = *scalars*, *variates* or *factors*	Pens to use; default 1

DTIMEPLOT procedure

Produces horizontal bars displaying a continuous time record (S.J. Clark).

Options

TITLE = *text*	Title for the plot; default * i.e. none
WINDOW = *numbers*	Which high-resolution graphics windows to use; default 3 for single plots and 5...8 for the composite plot
SCREEN = *string token*	Whether to clear the graphics screen before plotting (clear, keep); default clea

Parameters

DATA = *variates*	Bout lengths
GROUPS = *factors*	Factor defining act performed during each bout
LABELS = *texts*	Labels for each act
METHOD = *texts*	Type of plot to produce for each DATA variate (barplot, cumulative, log, survivor, composite); default comp

DUMMY directive

Declares one or more dummy data structures.

Options

VALUE = *identifier*	Value for all the dummies; default *
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used by default to identify the dummies in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the dummies
VALUE = *identifiers*	Value for each dummy
EXTRA = *texts*	Extra text associated with each identifier

DUMP directive

Prints information about data structures, and internal system information.

Options

PRINT = *string tokens*	What information to print about structures (attributes, values, identifiers, space); default attr
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
INFORMATION = *string tokens*	What information to print for each structure (brief, full, extended); default brie
TYPE = *string tokens*	Which types of structure to include in addition to those in the parameter list (all, diagonalmatrix, dummy, expression, factor, formula, LRV, matrix, pointer, scalar, SSPM, symmetricmatrix, table, text, TSM, variate); default * i.e. none
SYSTEM = *string token*	Whether to display Genstat system structures (yes, no); default no
UNNAMED = *string token*	Whether to display unnamed structures (yes, no); default no

Parameter

identifiers or *numbers*	Identifier or reference number of a structure whose information is to be printed

DUPLICATE directive

Forms new data structures with attributes taken from an existing structure.

Option

ATTRIBUTES = *string tokens*	Which attributes to duplicate (all, nvalues, values, nlevels, levels, labels (of factors or pointers), extra, decimals, characters, rows, columns, classification, margins, suffixes, minimum, maximum, restriction, referencelevel); default all
REDEFINE = *string token*	Whether or not to delete the attributes of the new structures beforehand so that their types can be redefined (yes, no); default no

Parameters

OLDSTRUCTURE = *identifiers*	Data structures to provide attributes for the new structures
NEWSTRUCTURE = *identifiers*	Identifiers of the new structures
VALUES = *identifiers*	Values for each new structure
DECIMALS = *scalars*	Number of decimals for printing numerical structures
CHARACTERS = *scalars*	Number of characters for printing texts or labels of a factor
EXTRA = *texts*	Extra text associated with each identifier
MINIMUM = *scalars*	Minimum value for numerical structures
MAXIMUM = *scalars*	Maximum value for numerical structures

DVARIOGRAM procedure

Plots fitted models to an experimental variogram (S.A. Harding, D.A. Murray & R. Webster).

Options

MODELTYPE = *string token*	Defines which model to plot (power, boundedlinear, circular, spherical, doublespherical, pentaspherical, exponential, besselk1, gaussian, affinepower, linear, cubic, stable, cardinalsine, matern); default powe
ISOTROPY = *string token*	Defines whether this is an isotropic or geometrical anisotropic

	model (`isotropic`, `geometrical`); default `isot`
`WINDOW` = *scalar*	Window in which to plot a graph; default 1
`TITLE` = *text*	Title for the graph

Parameters

`VARIOGRAM` = *variates*	Experimental variogram to which the model or matrices has been fitted, as a variate if in only one direction or as a matrix if there are several
`DISTANCE` = *variates*	Mean lag distances for the points in each or matrices variogram
`DIRECTION` = *variates*	Directions in which each variogram was computed
`ESTIMATES` = *variates*	Estimated parameter values
`XUPPER` = *scalar*	Upper limit for the x-axis in the graph
`PENDATA` = *scalar*	Pen to be used to plot the data; default 1
`PENMODEL` = *scalar*	Pen to be used to plot the model; default 2

`DXDENSITY` procedure

Produces one-dimensional density (or violin) plots (D. B. Baird).

Options

`BANDWIDTH` = *scalar*	Bandwidth for kernel smoothing (0-1); default density is chosen according to the number of observations
`GAP` = *scalar*	The size of the gap (0-1) between envelopes when there several densities are to be plotted; default 0.1
`TRANSFORM` = *string token*	Transformed scale for the data (`identity`, `log`, `log10`, `logit`, `probit`, `cloglog`, `square`, `exp`, `exp10`, `ilogit`, `iprobit`, `icloglog`, `root`); default is to use the transform defined for `XAXIS`
`AXISTITLE` = *text*	The title for the data axis; default is the name of the `DATA` variate
`GROUPSTITLE` = *text*	The title for the groups or variates axis; default is to use the name of the `GROUPS` factor
`WINDOW` = *scalar*	Window number for the graphs; default 3
`ORIENTATION` = *string token*	Orientation of plots (`horizontal`, `vertical`); default `vert`
`METHOD` = *string token*	Method for plotting the density envelope (`fill`, `line`); default `fill`
`SCREEN` = *string token*	Whether to clear screen before the plot (`clear`, `keep`, `resize`); default `clea`

Parameters

`DATA` = *variates* or *pointer*s	The data whose density is to be plotted
`GROUPS` = *factor*s	Factor to divide values of a single variate into groups; default `*` i.e. none
`TITLE` = *texts*	Title for graph; default uses the names of the data variates and type of plot

`DXYDENSITY` procedure

Produces density plots for large data sets (D. B. Baird).

Options

`PLOT` = *string tokens*	How to plot the density (`pointplot`, `shadeplot`, `contourplot`, `histogram`, `surface`); default `poin`
`NGROUPS` = *scalar*	Number of sections into which to divide each axis (4-400); default 50
`METHOD` = *string token*	Method to use to smooth the density (`thinplate`, `radialspline`, `tensorspline`, `kernel`); default `*` i.e. none
`DF` = *scalar*	Degrees of freedom for smoothing methods (2-50); default 12
`BANDWIDTH` = *scalar*	Bandwidth for kernel smoothing (0-1); default 0.2
`MEANFIT` = *string tokens*	What smooth regression fits to the means to plot (`yx`, `xy`);

	default * i.e. none
NCONTOURS = *scalar*	Number of contours in the contour plot; default 9
SYMBOL = *string token*	Symbol to use in a point plot (circle, square); default circ
COLOURS = *text*, *variate* or *scalar*	Colour to use to draw the symbols, shades, contours or surface; default !t(red, blue, black)
XTRANSFORM = *string token*	Transformed scale for the x-axis (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default iden
YTRANSFORM = *string token*	Transformed scale for the y-axis (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default iden
ZTRANSFORM = *string token*	Transformed scale for the z-axis (identity, percentile, root); default iden
WINDOW = *scalar*	Window number for the graphs; default 3
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep, resize); default clea

Parameters

Y = *variate* or *factor*	Y-coordinates of the data
X = *variate* or *factor*	X-coordinates of the data
TITLE = *text*	Title for graph; default uses the names of the data and type of plot

DXYGRAPH procedure

Draws two-dimensional graphs with marginal distribution plots alongside the y- and x-axes (D.A. Murray).

Options

YMETHOD = *string token*	Distribution plot to display in the margin of the y-axis (histogram, rugplot, boxplot); default hist
XMETHOD = *string token*	Distribution plot to display in the margin of the x-axis (histogram, rugplot, boxplot); default hist
YNGROUPS = *scalar*	Defines the number of groups in a margin plot of a histogram of the Y variate; default is then 10, or the integer value nearest the square root of the number of values in the Y variate if that is smaller
XNGROUPS = *scalar*	Defines the number of groups in a margin plot of a histogram of the X variate; default is then 10, or the integer value nearest the square root of the number of values in the X variate if that is smaller
YCOLOUR = *scalar* or *text*	Colour to use for the Y margin plot
XCOLOUR = *scalar* or *text*	Colour to use for the X margin plot

Parameters

Y = *variates* or *factors*	Vertical coordinates
X = *variates* or *factors*	Horizontal coordinates
TITLE = *texts*	General title for the plot; default *
WINDOW = *scalars*	Window number for the graphs; default 1
KEYWINDOW = *scalars*	Window number for the key (zero for no key); default 2
PEN = *scalars*, *variates* or *factors*	Pen number for each graph; default * uses pens 1, 2, and so on for the successive graphs
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea

DYPOLAR procedure

Produces polar plots (D. B. Baird).

Options

MODULUS = *scalar*	Number of units required to give a complete revolution in X; default 360
TOPANGLE = *scalar*	Angle at the top of the plot; default is a quarter of the MODULUS
COLOUR = *scalar* or *text*	Colour for the lines marking rings and sectors; default 'black'
LINESTYLE = *scalar*	Linestyle for the lines marking rings and sectors; default 1
YORIGIN = *scalar*	Origin for the y-values; default 0 or the minimum of Y if this is less than 0
YMARKS = *variate*	Y-values for the rings, plotted in the background of the plot
XMARKS = *variate*	X-values for the lines marking the sectors, plotted in the background of the plot
YLABELS = *text*	Labels for the rings
XLABELS = *text*	Labels for the sectors
YTRANSFORM = *string token*	Transformed scale for the y-values (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default is to use the transform defined for YAXIS
NRINGS = *scalar*	Number of rings to be plotted, if YMARKS is not set; default 9
NSECTORS = *scalar*	Number of sectors to be plotted, if XMARKS is not set; default 12
WINDOW = *scalar*	Window number for the graph; default 1
KEYWINDOW = *scalar*	Window number for the graph key; default 2
SCREEN = *string token*	Whether to clear the screen before the plot (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key; default *

Parameters

Y = *variates, factors* or *pointer*s	Y-values specifying the amplitudes of the points
X = *variates, factors* or *pointer*s	X-values specifying the angles of the points
GROUPS = *factor*s	Factor to divide the points into groups; default * i.e. none
TITLE = *texts*	Title for the graph; default forms a title automatically with the names of the Y and X structures
PEN = *scalar* or *variates*	Pens used to plot the data; default 1
DESCRIPTION = *texts*	Annotation for key; default uses the names of the Y and X structures, or the labels of GROUPS if set

D3GRAPH directive

Plots a 3-dimensional graph.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the plots; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
ELEVATION = *scalar*	The elevation of the viewpoint relative to the surface; default 25 (degrees)
AZIMUTH = *scalar*	Rotation about the horizontal plane; the default of 225 degrees ensures that a point at the minimum x- and y-value is nearest to the viewpoint
DISTANCE = *scalar*	Distance of the viewpoint from the centre of the grid on the base plane; default * ensures that the data points fill the viewing area
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep, resize); default clea

KEYDESCRIPTION = *text* — Overall description for the key; default *
ENDACTION = *string token* — Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

X = *identifiers* — X-coordinates
Y = *identifiers* — Y-coordinates
Z = *identifiers* — Z-coordinates
PEN = *scalars*, *variates* or *factors* — Pen number for each graph (use of a variate or factor allows different pens to be defined for different sets of units); default * uses pens 1, 2, and so on for the successive graphs
DESCRIPTION = *texts* — Annotation for key
UNITNUMBERS = *identifiers* — Specifies unit numbers to be used when points are selected in the graphics viewer; default * uses the actual unit numbers of the values in the X and Y structures

D3HISTOGRAM directive

Plots three-dimensional histograms.

Options

TITLE = *text* — General title; default *
WINDOW = *scalar* — Window number for the plots; default 1
KEYWINDOW = *scalar* — Window number for the key (zero for no key); default 2
ELEVATION = *scalar* — The elevation of the viewpoint relative to the surface; default 25 (degrees)
AZIMUTH = *scalar* — Rotation about the horizontal plane; the default of 225 degrees ensures that, with a square matrix M, the element M$[1;1] is nearest to the viewpoint
DISTANCE = *scalar* — Distance of the viewpoint from the centre of the grid on the base plane; default * gives a distance of 25 times the number of y points in the grid
SCREEN = *string token* — Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text* — Overall description for the key; default *
ENDACTION = *string token* — Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

GRID = *identifier* — Pointer (of variates representing the columns of a data matrix), matrix or two-way table specifying values on a regular grid
PEN = *scalar* — Pen number to be used for the plot; default 3
DESCRIPTION = *texts* — Annotation for key

ECABUNDANCEPLOT procedure

Produces rank/abundance, *ABC* and *k*-dominance plots (D.A. Murray).

Options

PRINT = *string token* — Controls printed output (summary); default summ
PLOT = *string token* — Controls the type of plot (rankabundance, kdominance, abc); default rank, kdom
GROUPS = *factor* — Defines the groups if there is more than one sample

Parameters

INDIVIDUALS = *variates* — Number of individuals per species
SPECIES = *variates* — Number of species
BIOMASS = *variates* — Biomass data for each species for an ABC plot

ECACCUMULATION procedure

Plots species accumulation curves for samples or individuals (D.A. Murray).

Options

PRINT = *string token*	Controls printed output (summary); default summ
CURVE = *string token*	Controls the type of species accumulation curve (collector, random, coleman); default coll
PLOT = *string token*	Controls plot type (sac); default sac
METHOD = *string token*	Controls collector curve when data supplied in variate or factor with groups (individual, sample); default samp
GROUPS = *factor*	Grouping factor for samples when data are supplied in variate of factor of individuals
NPERMUTATIONS = *scalar*	A scalar defining the number of permutations to be performed for the random method; default 100
SEED = *scalar*	Seed for random number generator; default 0
SCREEN = *string token*	Whether to clear screen before displaying the graph (keep, clear); defaul clea
WINDOW = *scalar*	Window for the graph; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
PEN = *scalar*	Pen number to draw the curve; default 1

Parameters

DATA = *variates*, *factors*, *matrices* or *pointers*	For individual-based collector curves, a variate or factor containing the individuals in the order they were collected; for sample-based species accumulation curves, a pointer or matrix specifying the number of individuals for each species for different sites/samples
RICHNESS = *variates*	Saves the observed number of species for the collector method and the average or expected number of species at each sample size for the Coleman and random methods
VARIANCE = *variates*	Saves the variance for the richness (Coleman and random methods only)

ECANOSIM procedure

Performs an analysis of similarities i.e. *ANOSIM* (D.A. Murray).

Options

PRINT = *string token*	Controls printed output (test); default test
PLOT = *string token*	Type of plot (boxplot, histogram); default hist
NTIMES = *scalar*	Number of permutations to make; default 999
BLOCKS = *factor*	Factor specifying groups for a stratified test; default * i.e. none
SEED = *scalar*	Seed for the random number generator used to make the permutations; default 0 continues from the previous generation or (if none) initializes the seed automatically

Parameters

DATA = *symmetric matrices*	Similarity matrix
GROUPS = *factors*	Specify the different groups for each matrix
STATISTIC = *scalars*	Save the *R* statistics
PROBABILITY = *scalars*	Save the probabilities

ECDIVERSITY procedure

Calculates measures of diversity with jackknife or bootstrap estimates (D.A. Murray).

Options

PRINT = *string tokens*	Controls printed output (index, estimate); default inde
INDEX = *string token*	Controls the type of measurement to be calculated (hshannon, qstatistic, simpsonyule, bergerparker, ibrillouin, ebrillouin, dmcintosh, emcintosh, evar, logseriesalpha, lognormallambda, jshannon,

	`margalef`, `isimpson`, `richness`); default `hsha`
`GROUPS` = *factor*	Defines the groups if there is more than one sample
`BMETHOD` = *string token*	Controls whether to use the bootstrap or jackknife method (`jackknife`, `bootstrap`); default `jack` for multiple samples and `boot` for individual samples
`NBOOT` = *scalar*	Number of times to resample in bootstrap; default 100
`SEED` = *scalar*	Seed for random number generator for bootstrap; default 0
`CIPROBABILITY` = *scalar*	Probability for the confidence interval produced by either jackknife or bootstrap method; default 0.95

Parameters

`INDIVIDUALS` = *variates*	Number of individuals per species
`SPECIES` = *variates*	Number of species
`INDICES` = *variate* or *pointer*	Saved the diversity indices

`ECFIT` procedure

Fits models to species abundance data (D.A. Murray).

Options

`PRINT` = *string tokens*	Controls printed output (`summary`, `estimates`, `fittedvalues`); default `summ`, `esti`
`MODELTYPE` = *string token*	The model or distribution fitted to the data (`logseries`, `plognormal`, `negativebinomial`, `geometric`, `zipf`, `mandelbrotzipf`); default `logs`
`GROUPS` = *factor*	Defines the groups if there is more than one sample
`LOGBASE` = *string token*	Log base to use to form the octaves for the logseries, Poisson log-Normal and negative binomial distributions (`two`, `ten`); default `two`
`PLOT` = *string token*	Plots the fitted values (`fittedabundance`, `rankabundance`); default `fitt`

Parameters

`INDIVIDUALS` = *variates*	Number of individuals per species
`SPECIES` = *variates*	Number of species
`ESTIMATES` = *variates*	Saves the model estimates
`EGROUPS` = *factors*	Saves the grouping of the estimates

`ECNICHE` procedure

Generates relative abundance of species for niche-based models (D.A. Murray).

`PRINT` = *string token*	Controls printed output (`model`, `expected`, `replications`); default `mode`, `expe`
`MODELTYPE` = *string token*	The niche model (`powerfraction`, `fixedratio`, `preemption`, `randomfraction`, `macarthurfraction`); default `powe`
`METHOD` = *string token*	Whether to use the Fortran DLL to calculate the relative abundance (`dll`, `commands`); default `*` uses the DLL in Windows implementations, and commands for other platforms
`POWER` = *scalar*	Power for the Power fraction model, must be in the range 0 to 1
`URATIO` = *scalar*	Ratio for the fixed ratio model
`SEED` = *scalar*	Seed for random number generator for the random division of the niche space; default 0
`PLOT` = *string token*	Plots the average relative abundance (`relativeabundance`); default rela

Parameters

`NREPLICATES` = *scalars*	Number of replications
`NSPECIES` = *scalars*	Number of species
`EXPECTED` = *variates*	Saves the expected average relative abundance
`SDEXPECTED` = *variates*	Saves the standard deviation for the expected mean relative

abundance

ECNPESTIMATE procedure

Calculates nonparametric estimates of species richness (D.A. Murray).

Options

PRINT = *string token*	Controls printed output (summary, estimates); default summ, esti
GROUPS = *factor*	Grouping factor for different samples
NBOOT = *scalar*	A scalar defining the number of bootstrap samples to be performed; default 100
SEED = *scalar*	Seed for random number generator; default 0

Parameters

DATA = *variates*, *matrices* or *pointers*	A variate containing abundances of species or a pointer or matrix specifying the individuals for each species for different sites/samples
ESTIMATE = *variates* or *pointer*	Saves the estimated species richness in a variate, or in a pointer if GROUPS are specified
SE = *variates* or *pointers*	Saves the analytic standard errors in a variate, or in a pointer if groups are specified
BSE = *variates* or *pointers*	Saves the bootstrap standard errors in a variate, or in a pointer if groups are specified

ECRAREFACTION procedure

Calculates individual or sample-based rarefaction (D.A. Murray).

Options

PRINT = *string token*	Controls printed output (summary); default summ
METHOD = *string token*	Controls the type of rarefaction (individual, sample); default indi
PLOT = *string token*	Controls plot type (expected); default expe
SAMPLESIZES = *scalar* or *variate*	A scalar defining a step between sample sizes or number of samples to estimate the number of species; alternatively, a variate specifing the actual sample size values or number of samples
CIPROBABILITY = *scalar*	Probability for the confidence interval; default 0.95

Parameters

DATA = *variates*, *matrices* or *pointers*	For individual-based rarefaction, a variate containing the number of individuals for each species; for sample-based rarefaction, a pointer or matrix specifying the number of individuals for each species for different sites/samples
EXPECTED = *variates*	Saves the expected number of species at each sample size
VARIANCE = *variates*	Saves the variance for the expected number of species
LOWER = *variates*	Saves the lower confidence limit at each sample size
UPPER = *variates*	Saves the upper confidence limit at each sample size

EDDUNNETT procedure

Calculates equivalent deviates for Dunnett's simultaneous confidence interval around a control (R.W. Payne).

Options

METHOD = *string token*	Form of the alternative hypothesis (twosided, greaterthan, lessthan); default twos
NTREATMENTS = *scalar*	Number of treatments being compared
DF = *scalar*	Number of residual degrees of freedom
REPTREATMENTS = *scalar* or *variate*	Specifies the replication of the treatments
REPCONTROL = *scalar*	Specifies the replication of the control
TOLERANCE = *scalar*	Tolerance for the difference between the probability for the calculated equivalent deviate and that requested by

CIPROBABILITY; default 0.0001

Parameters

CIPROBABILITY = *scalars*	Specifies the probability for the confidence interval
ED = *scalars*	Saves the equivalent deviate

EDFTEST procedure

Performs empirical-distribution-function goodness-of-fit tests (V. M. Cave).

Options

PRINT = *string tokens*	Controls printed output (summary, tests); default summ, test
PLOT = *string tokens*	What graphs to plot (kerneldensity, histogram); default *
TEST = *string tokens*	Specifies the type of goodness-of-fit test to perform (andersondarling, cramervonmises, kolmogorovsmirnov); default ande, cram, kolm
DISTRIBUTION = *string tokens*	Continuous distribution that is hypothesized to have generated the DATA; (beta, b2, burr, cauchy, chisquare, ev1 (or gumbel), ev2 (or frechet), ev3, exponential, fdistribution, gamma, gev, gpareto, iburr, igamma, invnormal, iweibull, laplace, loggamma, logistic, loglogistic, lognormal, normal, paralogistic, pareto, stdnormal, stduniform, tdistribution, ubetamix, ugammamix, uniform, weibull, calculated); default norm
CONSTANT = *string tokens*	Whether to estimate a constant for the distribution, when the parameter values are estimated from the DATA (estimate, omit); default omit
TMETHOD = *string tokens*	Specifies the method used to perform the goodness-of-fit tests (likelihoodratio, traditional); default like
PARAMETERS = *scalar* or *variate*	Parameter values for the hypothesized distribution; if this is not set, parameter values are estimated from the DATA
NAMES = *text*	Names to identify the parameters in PARAMETERS; if this is not set, the default parameter ordering is assumed
CDFCALCULATION = *expression*	Expression, formed using argument X, that defines the cumulative distribution function of the hypothesized distribution; must be specified when DISTRIBUTION = calculated
MCPARAMETERS = *string tokens*	Whether the parameters are re-estimated or fixed during the Monte-Carlo simulations, when the parameter values are estimated from the DATA (fix, estimate); default esti
NTIMES = *scalar*	Number of Monte-Carlo simulations to perform; default 999
SEED = *scalar*	Seed for random number generation; default 0 continues an existing sequence or, if none, selects a seed automatically
TITLE = *text*	Title for the graphs; default generates the title automatically
YTITLE = *text*	Y-axis title for the graphs; default generates the title automatically
XTITLE = *text*	X-axis title for the graphs; default generates the title automatically
WINDOW = *scalar*	Window to use for the graphs; default 3
SCREEN = *string tokens*	Whether to clear the screen before plotting the graph or to continue plotting on the old screen, when a single graph is requested (clear, keep); default clear

Parameters

DATA = *variate*	Identifier of the variate holding the data
STATISTIC = *pointer*	Pointer to scalar(s) to save the test statistic(s)
MCSTATISTICS = *pointer*	Pointer to variates(s) to save the Monte-Carlo simulated test statistic(s)

PROBABILITY = *pointer*	Pointer to scalar(s) to save the probability value(s) of the test statistic(s)

EDIT directive

Edits text vectors.

Options

CHANNEL = *scalar* or *text*	Text structure containing editor commands or a scalar giving the number of a channel from which they are to be read; default is the current input channel
END = *text*	Character(s) to indicate the end of the commands read from an input channel; default is the character colon (:)
WIDTH = *scalar*	Limit on the line width of the text; default *
SAVE = *text*	Text to save the editor commands for future use; default *

Parameters

OLDTEXT = *texts*	Texts to be edited
NEWTEXT = *texts*	Text to store each edited text; if any of these is omitted, the corresponding OLDTEXT is used

ELSE directive

Introduces the default set of statements in block-if or in multiple-selection control structures.

No options or parameters

ELSIF directive

Introduces a set of alternative statements in a block-if control structure.

No options

Parameter

expression	Logical expression to indicate whether or not the set of statements is to be executed.

ENDBREAK directive

Returns to the original channel or control structure and continues execution.

No options or parameters

ENDCASE directive

Indicates the end of a "multiple-selection" control structure.

No options or parameters

ENDDEBUG directive

Cancels a DEBUG statement.

No options or parameters

ENDFOR directive

Indicates the end of the contents of a loop.

No options or parameters

ENDIF directive

Indicates the end of a block-if control structure.

No options or parameters

ENDJOB directive

Ends a Genstat job.

No options or parameters

ENDPROCEDURE directive

Indicates the end of the contents of a Genstat procedure.

No options or parameters

ENQUIRE directive

Provides details about files opened by Genstat.

No options

Parameters

CHANNEL = *scalars*	Channel numbers to enquire about; for FILETYPE=input or output, a scalar containing a missing value will be set to the number of the current channel of that type and a negative value can be used to check the existence of a file that is not yet connected to a channel
FILETYPE = *string tokens*	Type of each file (input, output, unformatted, backingstore, procedurelibrary, graphics); default inpu
OPEN = *scalars*	To indicate whether or not the corresponding channels are currently open (0=closed, 1=open)
NAME = *texts*	External name of the file, if channel is open
EXIST = *scalars*	To indicate whether files on corresponding channels currently exist (0=not yet created, 1=exist)
WIDTH = *scalars*	Maximum width of records in each file (only relevant for input and output files, set to * for other types)
PAGE = *scalars*	Number of lines per page (relevant only for output files)
ACCESS = *texts*	Allowed type of access: set to 'readonly', 'writeonly' or 'both'
LINE = *scalars*	Number of the current line (input files only)
STYLE = *texts*	Underlying style of an output channel: set to 'plaintext', 'html', 'rtf' or 'latex' (see OPEN)
OUTSTYLE = *texts*	Current style of an output channel: set to 'plaintext' or 'formatted' (see OUTPUT)

EQUATE directive

Transfers data between structures of different sizes or types (but the same modes i.e. numerical or text) or where transfer is not from single structure to single structure.

Options

OLDFORMAT = *variate*	Format for values of OLDSTRUCTURES; within the variate, a positive value *n* means take *n* values, −*n* means skip *n* values and a missing value means skip to the next structure; default * i.e. take all the values in turn
NEWFORMAT = *variate*	Format for values of NEWSTRUCTURES; within the variate, a positive value *n* means fill the next *n* positions, −*n* means skip *n* positions and a missing value means skip to the next structure; default * i.e. fill all the positions in turn
FREPRESENTATION = *string token*	How to interpret factor values (labels, levels, ordinals); default leve

Parameters

OLDSTRUCTURES = *identifiers*	Structures whose values are to be transferred; if values of several structures are to be transferred to one item in the NEWSTRUCTURES list, they must be placed in a pointer
NEWSTRUCTURES = *identifiers*	Structures to take each set of transferred values; if several structures are to receive values from one item in the OLDSTRUCTURES list, they must be placed in a pointer

ESTIMATE directive

Estimates parameters in Box-Jenkins models for time series (synonym of TFIT).

Options

PRINT = *string tokens*	What to print (model, summary, estimates, correlations, monitoring); default mode,summ,esti
LIKELIHOOD = *string token*	Method of likelihood calculation (exact, leastsquares, marginal); default exac
CONSTANT = *string token*	How to treat the constant (estimate, fix); default esti
RECYCLE = *string token*	Whether to continue from previous estimation (yes, no); default no
WEIGHTS = *variate*	Weights; default *
MVREPLACE = *string token*	Whether to replace missing values by their estimates (yes, no); default no
FIX = *variate*	Defines constraints on parameters (ordered as in each model, tf models first): zeros fix parameters, parameters with equal numbers are constrained to be equal; default *
METHOD = *string token*	Whether to carry out full iterative estimation, to carry out just one iterative step, to perform no steps but still give parameter standard deviations, or only to initialize for forecasting by regenerating residuals (full, onestep, zerostep, initialize); default full
MAXCYCLE = *scalar*	Maximum number of iterations; default 15
TOLERANCE = *scalar*	Criterion for convergence; default 0.0004
SAVE = *identifier*	To name save structure, or supply save structure with transfer-functions; default * i.e. transfer-functions taken from the latest model

Parameters

SERIES = *variate*	Time series to be modelled (output series)
TSM = *TSM*	Model for output series
BOXCOXMETHOD = *string token*	How to treat transformation parameter in output series (fix, estimate); default fix
RESIDUALS = *variate*	To save residual series

EXAMPLE procedure

Obtains and runs a Genstat example program, PC Windows only (R.W. Payne).

Option

EXECUTE = *string token*	Whether to run the example when Genstat is running interactively (no, yes); default no

Parameters

EXTYPE = *texts*	Types of example
EXNAME = *texts*	Names of example
SOURCE = *texts*	Texts to store the source code of each example
STATEMENT = *texts*	Saves a command to obtain each example (useful if the name and type information has been specified in response to questions from EXAMPLE)

EXECUTE directive

Executes the statements contained within a text.

No options

Parameter

texts	Statements to be executed

EXIT directive

Exits from a control structure.

Options

NTIMES = *scalar*	Number of control structures, *n*, to exit (if *n* exceeds the number of control structures of the specified type that are currently active, the exit is to the end of the outer one; while for *n* negative, the exit is to the end of the −*n*'th structure in order of execution); default 1
CONTROLSTRUCTURE = *string token*	Type of control structure to exit (job, for, if, case, procedure); default for
REPEAT = *string token*	Whether to go to the next set of parameters on exit from a FOR loop or procedure (yes, no); default no
EXPLANATION = *text*	Text to be printed if the exit takes place; default *

Parameter

expression	Logical expression controlling whether or not an exit takes place

EXPORT procedure

Saves data structures in Genstat, Excel, R, Quattro, dBase, SPlus, Gauss, MatLab, SAS, Instat, Image or text files (D.B. Baird).

Options

PRINT = *string token*	What to print (summary); default summ
OUTFILE = *text*	Data file to be written
METHOD = *string token*	Action to take if the file already exists (add, append, concatenate, merge, overwrite, prompt, fail, replace); default prompt in interactive mode, fail in batch mode
PLAINNAMES = *string token*	Whether to leave the column names in the file in plain form rather than decorating them with the column type information i.e. ! for factors, :D for dates etc (yes, no) default no
SHEETNAME = *text*	Name of new sheet to be added to an existing Excel file
NONAMES = *string token*	Whether to suppress column names in output to spreadsheet or text file (yes, no); default no
TITLE = *text*	Description for spreadsheet
READONLY = *string token*	Whether to define the complete sheet as read only (yes, no); default no
ANALYSIS = *text*	Genstat commands to analyse columns in the spreadsheet
ASETUP = *text*	Genstat commands to be run once before the analysis of any columns in the spreadsheet
ADUMMY = *text*	The name of the dummy (if any) used the ANALYSIS commands
CSVOPTIONS = *string tokens*	Options for CSV files (noquotes, pack, round, fixed, align); default pack
HTMLOPTIONS = *string token*	Options for HTML files (allowformats, nogrid, centre, rightjustify); default * i.e. none
COLMATCH = *string token*	How to match columns when appending (name, position); default posi
GROUPS = *factor* or *text*	Identifier for the factor, or text containing the name of the factor, to identify appended sections in the output file
GLABEL = *texts*	Labels for the GROUPS factor for the current appended section, and also for the original section if no previous sections have been appended
MATCH = *texts*, *variates* or *pointers*	Up to four DATA variables to use as keys when METHOD=merge; default * uses the first DATA variable
WITH = *texts*, *variates* or *pointers*	Columns in the file to use as keys when METHOD=merge; default * uses as many columns of the initial columns in the

file as are needed to give a column for each MATCH column

UPDATE = *string token* Whether to use columns with matching names to replace existing columns when concatenating or merging (yes, no); default no changes the names of columns with the same name as existing columns so that they become unique

OUTOPTIONS = *text* Optional output file arguments to be passed to the Dataload.dll

ROWCOLOURS = *factor* The factor to be used for colouring the rows (the factor must have colours defined by the FACCOLOURS parameter)

TABLEFORMAT = *string token* The format to use when displaying tables with two or more classifying factors (page, column); default page

MISSING = *text* String to represent a numerical missing value when writing to a text file (.TXT, .TAB or .CSV) or a spreadsheet file (Excel, Quattro or Open Office); default is to use '*' in .TXT or .TAB files, and leave cells with missing values empty in csv or spreadsheet files

DELETESHEETS = *string token* Whether to delete sheets if you are overwriting a multiple paged file with a single page (always, never, prompt); default prom when running interactively and neve when running in batch

NONASCII = *string token* Specifies how to output non-ASCII characters to text files (utf8, unicode); default utf8

TIMEOUT = *scalar* Number of seconds to wait when a file is open in another process; default 10

Parameters

DATA = *identifiers* The data structures to be written to the file, these must be compatible (i.e. of the same length)

COLUMNS = *texts* Names for the columns to be saved

PROTECT = *scalars* Whether the column is to be defined as read only when option READONLY=no (yes, no); default no

FACCOLOURS = *variates, texts* or *pointers* Specifies background colours for factor columns

FOREGROUND = *variates, texts, scalars* or *pointer* Specifies foreground colours for columns

BACKGROUND = *variates, texts, scalars* or *pointer* Specifies background colours for columns

DECIMALS = *variates* or *scalars* Specifies numbers of decimals for the columns

EXPRESSION directive

Declares one or more expression data structures.

Options

VALUE = *expression* Value for all the expressions; default *

MODIFY = *string token* Whether to modify (instead of redefining) existing structures (yes, no); default no

IPRINT = *string tokens* Information to be used by default to identify the expressions in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers* Identifiers of the expressions

VALUE = *expression structures* Expression data structures providing values for the expressions

EXTRA = *texts* Extra texts associated with the identifiers

EXTERNAL directive

Declares an external function in a DLL for use by the OWN function.

Options

LIBRARY = *text* Name of DLL file containing the function

Parameters

FUNCTION = *text*	Name of the function entry point in the DLL
NAME = *text*	Name for the function to be used in the OWN function; default uses the name set in FUNCTION
RESULTS =*string token*	The type of result returned from the function (summary, transformation); default tran
NPARAMETERS = *scalar*	The number of parameters in the function call; default 0
ERRORS = *scalar* or *variate*	Error codes returned from the function; default * i.e. no error codes
MESSAGES = text	Messages for the corresponding error codes

EXTRABINOMIAL procedure

Fits the models of Williams (1982) to overdispersed proportions (M.S. Ridout & P.W. Goedhart).

Options

PRINT = *string tokens*	What to print if iterative estimation process converges successfully and whether to monitor the iterations (model, summary, accumulated, estimates, correlations, fittedvalues, monitoring); default *
CONSTANT = *string token*	How to treat constant (estimate, omit); default esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default 3
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality); default *
METHOD = *string token*	Which model to fit to take account of the extra variation (II, III); default II
MODIFYMODEL = *string token*	Whether to leave the modified MODEL settings (WEIGHTS and DISPERSION) or whether to restore the original situation (yes, no); default no
WEIGHTS = *variate*	To save estimated weights
PHI = *scalar*	To save estimated overdispersion parameter
MAXCYCLE = *scalar*	Maximum number of iterations; default 10
TOLERANCE = *scalar*	Convergence criterion; default 0.01

Parameter

TERMS = *formula*	Model terms to be fitted; if unset it is assumed that the model consists only of a constant term

FACAMEND procedure

Permutes the levels and labels of a factor (J.T.N.M. Thissen).

Option

DIRECTION = *string token*	Order into which to sort the levels or labels of FACTOR (ascending, descending); default asce

Parameters

FACTOR = *factor*	Factor whose levels or labels are to be permuted
NEWLEVELS = *variate* or *text*	To specify the new order of the factor levels or labels

FACCOMBINATIONS procedure

Forms a factor to indicate observations with identical values of a set of variates, texts or factors (R.W. Payne).

Options

FLABELS = *string token*	When to form labels (always, ifredeclared, only, never); default ifre
SEPARATOR = *text*	Separator to use when constructing labels; default ' '
ISEPARATOR = *text*	Separator to use between identifiers and levels or labels; default ' '
IMETHOD = *string token*	Whether to include identifiers in the labels (include, omit); default omit

Parameters

VECTORS = *pointers*	Pointers containing sets of vectors (variates, and/or factors, and/or texts)
FACTOR = *factors*	Saves a factor for each set of vectors with a level for every different combination of their values

FACDIVIDE procedure

Represents a factor by factorial combinations of a set of factors (R.W. Payne).

Option

OLDFACTOR = *factor*	Factor whose levels are to be represented by the factorial combinations of the NEWFACTORs

Parameters

NEWFACTOR = *factors*	Factors formed to represent OLDFACTOR
LEVELS = *scalars* or *variates*	Levels of the NEWFACTORs

FACEXCLUDEUNUSED procedure

Redefines the levels and labels of a factor to exclude those that are unused (R.W. Payne).

No options

Parameters

FACTOR = *factors*	Factors with unused levels
NEWFACTOR = *factors*	New factors, with levels (and labels) that exclude those that are unused; if unset, the original factor is redefined

FACGETLABELS procedure

Obtains the labels for a factor if it has been defined with labels, or constructs labels from its levels otherwise (V.M. Cave).

Options

PRINT = *string token*	Controls printed output (labels); default *
PREFIX = *text*	Supplies a single line of text to be used as a prefix when constructing labels from the factor levels; default * i.e. none

Parameters

FACTOR = *factors*	Factor whose labels are to be obtained
LABELS =*texts*	Specifies text structures to save the labels of each factor
EXIST = *scalars*	Specifies a scalar for each factor, set to the value 1 if its labels already existed or 0 if they had to be constructed

FACLEVSTANDARDIZE procedure

Redefines a list of factors to coordinate their levels or labels (R.W. Payne).

Options

FREPRESENTATION = *string token*	Whether to coordinate the factors to have the same levels, labels or (ordinal) number of levels (levels, labels, ordinals); default leve
DIRECTION = *string token*	How to sort the levels or labels (ascending, descending, given); default asce
CASE = *string token*	Case to use for labels (given, lower, upper, sentence, title); default give
REMOVEUNUSED = *string token*	Whether to remove unused levels (yes, no); default no

Parameters

FACTOR = *factors*	Factors to be coordinated
NEWFACTOR = *factors*	New factors, redefined to share the same levels or labels; if unset, the original FACTOR is redefined

FACMERGE procedure

Merges levels of factors (S.D. Langton).

Options

PRINT = *string token*	Controls printed output (summary); default * i.e. none
OLDFACTOR = *factor*	Original factor
NEWFACTOR = *factor*	New factor with merged levels

Parameters

MERGE = *variates* or *texts*	Levels to merge
LEVMERGED = *variates*	Level to assign to the merged levels
LABMERGED = *texts*	Label to assign to the merged levels

FACPRODUCT procedure

Forms a factor with a level for every combination of other factors (R.W. Payne).

Options

FLABELS = *string token*	When to form labels (always, ifredeclared, only, never); default ifre
SEPARATOR = *text*	Separator to use when constructing labels; default ' '
LMETHOD = *string token*	Whether to define levels for all combinations or only for those present in the data (all, present); default pres
ISEPARATOR = *text*	separator to use between identifiers and levels or labels; default ' '
IMETHOD = *string token*	Whether to include identifiers in the labels (include, omit); default omit

Parameters

FACTORS = *pointers* or *formulae*	Factors contributing to each product
PRODUCT = *factors*	Factors to be formed

FACROTATE directive

Rotates factor loadings from a principal components, canonical variates or factor analysis.

Options

PRINT = *string tokens*	Printed output required (communalities, loadings, orthogonalrotationmatrix, rotation); default * i.e. no printing
METHOD = *string token*	Criterion (varimax, quartimax); default vari
NROOTS = *scalar*	Sets the number of dimensions to rotate from the original loadings; default * i.e. all

Parameters

OLDLOADINGS = *matrices*	Original loadings
NEWLOADINGS = *matrices*	Rotated loadings for each set of OLDLOADINGS
COMMUNALITIES = *matrices*	Communalities of the variables in each rotation
ROTATION = *matrices*	Saves the orthogonal rotation from the original solution to the rotated space

FACSORT procedure

Sorts the levels of a factor according to an index vector (R.W. Payne).

Options

DIRECTION = *string token*	Direction in which to sort the index (ascending, descending); default asce
SETATTRIBUTES = *string tokens*	Which aspects of each NEWFACTOR to define (levels, labels, values); default * i.e. labels and values if defined for FACTOR, also levels if not the integers 1,2...

Parameters

FACTOR = *factors*	Factors whose levels are to be reordered
INDEX = *variate*, *text* or *one-way table*	Index vectors defining the ordering of the levels of each factor
NEWFACTOR = *factors*	New factors with reordered levels; if unset, the original

FACTOR is redefined

NEWLEVELS = *variates* Saves the (reordered) levels as defined for each NEWFACTOR

FACTOR directive

Declares one or more factor data structures.

Options

NVALUES = *scalar* or *vector* Number of units, or vector of labels; default * takes the setting from the preceding UNITS statement, if any

LEVELS = *scalar* or *vector* Number of levels, or series of numbers which will be used to refer to levels in the program; default *

VALUES = *numbers* Values for all the factors, given as levels; default *

LABELS = *text* Labels for levels, for input and output; default *

MODIFY = *string token* Whether to modify (instead of redefining) existing structures (yes, no); default no

REFERENCELEVEL = *scalar* Defines the reference level used e.g. to define the parameterization of regression models

IPRINT = *string tokens* Information to be used by default to identify the factors in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers* Identifiers of the factors

VALUES = *identifiers* Values for each factor, specified as levels or labels

DECIMALS = *scalars* Number of decimals for printing levels

CHARACTERS = *scalars* Number of characters for printing labels

EXTRA = *texts* Extra text associated with each identifier

DREPRESENTATION = *scalars* or *texts* Default format to use when the contents represent dates and times

FACUNIQUE procedure

Redefines a factor so that its levels and labels are unique (R.W. Payne).

Options

MERGESAME = *string tokens* What must be the same for groups defined by the factor to be merged (levels, labels); default * i.e. no groups are merged

INCREMENT = *scalar* Value to use to modify duplicate levels; default * i.e. a suitable (small) value is determined automatically

ADDTO = *string token* Whether to add the increment to the value or the absolute value of duplicated levels (value, absolutevalue); default abso

Parameters

OLDFACTOR = *factors* Factors whose levels and labels are to be made unique

NEWFACTOR = *factors* New factors with unique levels; if unset, the original OLDFACTOR is redefined

CHANGED = *scalars* Indicates whether the factor has changed

FALIASTERMS procedure

Forms information about aliased model terms in analysis of variance (R.W. Payne).

Options

PRINT = *string tokens* Controls printed output (aovtable, aliasedterms); default alia

TREATMENTSTRUCTURE = *formula* Treatment model for the design; if this is not set, the default is taken from any existing setting defined by the TREATMENTSTRUCTURE directive

BLOCKSTRUCTURE = *formula* Block model for the design; if this is not set, the default is taken from any existing setting defined by the BLOCKSTRUCTURE directive

FACTORIAL = *scalar*	Limit on number of factors in a treatment term; default 3
RESTRICTION = *variate*	Defines a restriction on the units for the analysis; default * i.e. none

Parameters

TERMS = *formula*	Model terms whose aliased terms are to be identified; the default is to take all the terms in the treatment model
ALIASTERMS = *formula* or *pointer*	Saves the aliased terms

FARGUMENTS directive

Forms lists of arguments involved in an expression.

Options

EXPRESSION = *expression structure*	Expression whose arguments are required
NRESULTS = *scalar*	Number of results generated by the expression
NCALCULATIONS = *scalar*	Number of calculations in the expression

Parameters

ICALCULATION = *scalars*	The calculation from which to save the result and arguments
RESULT = *dummies*	Stores the result structure for calculation ICALCULATION
ARGUMENTS = *pointers*	Stores the arguments in calculation ICALCULATION

FAULT directive

Checks whether to issue a diagnostic, i.e. a fault, warning or message.

Options

DIAGNOSTIC = *string token*	Severity of the diagnostic (fault, warning, message); default faul
FAULT = *text*	Diagnostic code; default 'UF 1' for fault, 'UF 2' for warning
EXPLANATION = *text*	Explanatory information
NCALLS = *scalar*	Number of calls from the main procedure (whose name should be used in fault or warning messages); default 0

Parameter

expression	Logical expression to test whether or not to give the diagnostic

FBASICCONTRASTS procedure

Breaks a model term down into its basic contrasts (R.W. Payne).

Options

TERM = *formula*	Model term to split into basic contrasts
PSEUDOFACTORS = *pointer*	Pseudo-factors representing the basic contrasts
NEWTERMS = *formula structure*	Model formula containing the term followed by the pseudofactors

No parameters

FBETWEENGROUPVECTORS procedure

Forms variates and classifying factors containing within-group summaries to use e.g. in a between-group analysis (R.W. Payne).

Options

CLASSIFICATION = *factors*	Factors defining the groups; must be set
COUNTS = *variate*	Saves a variate counting the number of units with each factor combination; default *
WEIGHTS = *variate*	Weights to be used to calculate the within-group summaries; default * indicates that all units have weight 1
METHOD = *string token*	How to summarize the data variates (totals, nobservations, means, minima, maxima, variances, quantiles, sds, skewness, kurtosis, semeans, seskewness, sekurtosis); default mean
PERCENTQUANTILES = *scalar*	Percentage point for quantiles; default 50

OMITEMPTYCELLS = *string token* — Whether to omit units arising from empty cells in the summary table (yes, no); default no

SETLEVELS = *string token* — Whether to redefine the levels of factors (yes, no); default no

Parameters

VECTOR = *variates* and *factors* — Original data vectors

NEWVECTOR = *variates* and *factors* — New vectors containing the within-group summaries

FCA directive

Performs factor analysis.

Options

PRINT = *string tokens* — Printed output required (communalities, loadings, coefficients, scores, residuals, cresiduals, vresiduals, tests); default * i.e. no printing

NDIMENSIONS = *scalar* — Number of factors to fit; no default, must be specified

METHOD = *string token* — Whether to use correlations or variances and covariances (correlation, vcovariance, variancecovariance); default vcov

MAXCYCLE = *scalar* — Maximum number of iterations; default 50

TOLERANCE = *scalar* — Minimum value to assume for the unique component ψ_i^2 of each observed variable; default 10^{-6}

Parameters

DATA = *pointers* or *matrices* or *symmetric matrices* or *SSPMs* — Pointer of variates forming the data matrix, or matrix storing the variate values by columns, or symmetric matrix storing their variances and covariances, or SSPM giving their sums of squares and products

NUNITS = *scalars* — When DATA is set to a symmetric matrix of variances and covariances, NUNITS must specify the number of units from which they were calculated if tests are required

LRV = *LRVs* — Saves the loadings, latent roots and trace from each analysis

SSPM = *SSPMs* — Saves the SSPM formed from a DATA matrix or pointer

COMMUNALITIES = *variates* — Saves the communalities

COEFFICIENTS = *matrices* — Saves the factor score coefficients

SCORES = *matrices* or *pointers* — Saves the factor analysis scores

RESIDUALS = *matrices* or *pointers* — Saves residuals from the dimensions fitted in the analysis

CRESIDUALS = *symmetric matrices* — Saves the residual correlation or covariance matrix

VRESIDUALS = *variates* — Saves the residual variances

FCLASSIFICATION directive

Forms a classification set for each term in a formula, breaks a formula up into separate formulae (one for each term), and applies a limit to the number of factors and variates in the terms of a formula.

Options

FACTORIAL = *scalar* — Limit on the number of factors and variates in each term; default * i.e. no limit

NTERMS = *scalar* — Outputs the number of terms in the formula

CLASSIFICATION = *pointer* — Saves a list of all the factors and variates in the TERMS formula

OUTFORMULA = *formula structure* — Identifier of a formula to store a new formula, omitting terms with too many factors and variates

INCLUDEFUNCTIONS = *string token* — Whether or not to include functions in the formulae saved by the OUTFORMULA option or the OUTTERMS parameter (yes, no); default no

REORDER = *string token* — When to reorder the terms in the model (always, standard, never); default stan

DROPTERMS = *string token* — Whether to include only terms that can be dropped individually from the formula (yes, no); default no

CHECKFUNCTIONS = *scalar* — Indicator, set to one if the TERMS formula contains any

functions, and zero if it contains none

FUNCTIONDEFINITIONS = *pointer* — Saves details of the functions defined for each factor and variate in the TERMS formula

EXCLUDEPSEUDOTERMS = *string token* — Whether to omit pseudo-terms from the numbers of terms and the formulae saved by the OUTFORMULA option and OUTTERMS parameter (yes, no); default no

Parameters

TERMS = *formula* — Formula from which the classification sets, individual model terms and so on are to be formed

CLASSIFICATION = *pointers* — Identifiers for pointers to store the factors and variates composing each model term of the TERMS formula

OUTTERMS = *formula structures* — Identifiers for formulae to store each individual term of the TERMS formula

MAINTERMS = *formula structures* — Identifiers for formulae to store the main term for each individual term of the TERMS formula

FCOMPLEMENT procedure

Forms the complement of an incomplete block design (W. van den Berg).

Option

PRINT = *string token* — Controls whether or not to print a plan of the design (design); default desi

Parameters

TREATMENTS = *factors* — Specifies the treatment factor of the original design

REPLICATES = *factors* — Specifies the replicate factor of the original design when this is a resolvable design

BLOCKS = *factors* — Specifies the block factor of the original design

NEWTREATMENTS = *factors* — Saves the treatment factor of the complement design

NEWREPLICATES = *factors* — Saves the replicate factor of the complement design when this is a resolvable design

NEWBLOCKS = *factors* — Saves the block factor of the complement design

NEWUNITS = *factors* — Saves the treatment factor of the complement design

SEED = *scalars* — Seed for the random-numbers to randomize the design; default 0

FCONTRASTS procedure

Modifies a model formula to contain contrasts of factors (R.W. Payne).

Options

FORMULA = *formula* — Formula to modify to contain contrasts

NEWFORMULA = *formula structure* — Modified formula

FACTORIAL = *scalar* — Limit on the number of variates or factors in terms generated from FORMULA; default 3

Parameters

FACTOR = *factors* — Factors over which to define contrasts

CONTRASTTYPE = *string tokens* — Type of contrast (polynomial, regression); default poly

ORDER = *scalars* — Number of contrasts to define for each FACTOR

XCONTRASTS = *variates* or *matrices* — X-values defining the contrasts for each FACTOR

DEVIATIONS = *string tokens* — Whether to include deviations (yes, no); default no

ORTHOGONALIZE = *string tokens* — Whether to orthogonalize the contrasts (yes, no); default no

SAVECONTRASTS = *pointers* — Pointer to save the contrast variates defined for each FACTOR

FCOPY directive

Makes copies of files.

No options

Parameters

OLD = *texts* — Name of each file to copy

NEW = *texts* — Name for the new copy of each file

OVERWRITE = *string tokens* Whether to overwrite any existing files (yes, no); default no

FCORRELATION procedure

Forms the correlation matrix for a list of variates (R.W. Payne).

Options

PRINT = *string tokens* Printed output (correlations, test); default corr
METHOD = *string token* Type of test to make (against zero) for the correlations (twosided, greater, lessthan); default twos
WEIGHTS = *variate* Provides weights for the units of the variates; default * assumes that they all have weight one
CORRELATIONS = *symmetric matrix* Saves the correlations
PROBABILITIES = *symmetric matrix* Saves the test probabilities
NOBSERVATIONS = *scalars* Saves the number of observations from which the correlations have been calculated

Parameter

DATA = *variates* Variates for which the matrix is to be calculated

FCOVARIOGRAM directive

Forms a covariogram structure containing auto-variograms of individual variates and cross-variograms for pairs from a list of variates.

Options

PRINT = *string token* Controls printed output (statistics, variograms, autovariograms); default stat
METHOD = *string token* Specifies what to do when the measurements are not all made at the same locations (allwithcrossnugget, allnocrossnugget, commonpoints); default comm
COVARIOGRAM = *pointer* Pointer to store the variograms, cross-variograms and associated information for use in MCOVARIOGRAM
MAXLAG = *scalar* Maximum lag in all directions
STEPLENGTHS = *scalar* or *variate* Length of the step or steps in which lag is incremented
DIRECTIONS = *scalar* or *variates* Directions along which to form the variogram, scalar for a single direction in 2 dimensions, variate for several directions in 2 dimensions, and pairs of variates for 3 dimensional data
SEGMENTS = *scalar* Angle subtended by each segment along the DIRECTIONS
COORDSYSTEM = *string token* Coordinate system used for the geometry for discretizing the lag (mathematical, geographical); default math
MAXCONEDIAMETER = *scalar* Diameter at which the segments over which averaging is to be done should cease to expand; default * implies no limit
MINCOUNT = *scalar* Minimum number of points required at a particular lag point for the cross-variogram to be estimated there; default 1
DRIFT = *string token* Mean function (constant, linear, quadratic); default cons

Parameters

DATA = *variates* Measurements as a variate
X1 = *variates* Locations of each set of measurements in the first dimension
X2 = *variates* Locations of each set of measurements in the second dimension (if recorded in more than 1 dimension)
X3 = *variates* Locations of each set of measurements in the third dimension (if recorded in 3 dimensions)

FDELETE directive

Deletes files.

No options

Parameter

NAME = *texts* Names of the files to delete

FDESIGNFILE procedure

Forms a backing-store file of information for AGDESIGN (R.W. Payne).

Option

PRINT = *string tokens*	Controls printed output (catalogue, data, filestructure); default * i.e. none

Parameters

DATAFILE = *texts*	Name of the data file containing the information required to form each backing-store subfile
BSFILE = *texts*	Name of the backing-store file
SUBFILE = *identifiers*	Identifier of the backing-store subfile

FDIALLEL procedure

Forms the components of a diallel model for REML or regression (R.W. Payne).

No options

Parameters

MALEPARENTS = *factors*	Specifies the male parents
FEMALEPARENTS = *factors*	Specifies the female parents
PARENTS = *matrices*	Saves design matrices for the overall parental effects
COMPPARENTS = *matrices*	Saves comparison matrices for overall parental effects
PUREVSCROSS = *factors*	Saves factors to represent the comparison between pure and crossed lines
CROSSPAIR = *factors*	Saves factors to represent the comparison between types of pairs of parent (ignoring the individual genders)

FDISTINCTFACTORS procedure

Checks sets of factors to remove any that define duplicate classifications (R.W. Payne).

No options

Parameters

SET1 = *pointers*	First set of factors
SET2 = *pointers*	Second set of factors
DISTINCTSET = *pointers*	Saves the distinct factors

FDRBONFERRONI procedure

Estimates false discovery rates by a Bonferroni-type procedure (A.I. Glaser).

Options

PRINT = *string token*	Controls printed output (pi0); default pi0
METHOD = *string token*	Controls the method used for calculating π_0 (smoother, bootstrap); default smoo
LOGP = *string token*	Whether to take logs of π_0 when METHOD=smoother (yes, no); default no
DF = *scalar*	Degrees of freedom for smoothing spline; default 3
PLOT = *string token*	Controls plots (phistogram, qhistogram, pi0vslambda, qvsp, tests, expfalsepositive, inference, loginference); default phis, qhis, pi0v, qvsp, test, expf, infe, logi
WINDOW = *scalar*	Window for the graphs; default 1
KEYWINDOW = *scalar*	Window for the key (zero for none); default 2

Parameters

PROBABILITIES = *variates*	Significance values, must lie between 0 and 1
LAMBDA = *scalars* or *variates*	Values of tuning parameter λ, equivalent to significance levels at which to test the PROBABILITIES; default !(0,0.05...0.9)
FDR = *variates*	Saves the False Discovery Rates (i.e. q-values) at the sorted p-values in PROBABILITIES
FRR = *variates*	Saves the False Rejection Rates at the sorted p-values in PROBABILITIES

PI0 = *scalars*	Saves the value of π_0, i.e. the maximum value of the FDR
LOWER = *scalars*	Lower bound of q-values to use with PLOT settings qvsp, tests and expfalsepositive; default 0
UPPER = *scalar*	Upper bound of q-values to to use with PLOT settings qvsp, tests and expfalsepositive; default 1, which indicates maximum q-value

FDRMIXTURE procedure

Estimates false discovery rates using mixture distributions (J.W. McNicol & D.B. Baird).

Options

PRINT = *string token*	What to print (monitoring, estimates); default esti
DISTRIBUTION = *string token*	Which distribution to mix with Uniform (beta, gamma); default beta
INITIAL = *variate*	Initial values for mixing proportion (φ) and Beta or Gamma parameters (*A* and *B*); default !(0.90, 0.30, 2)
LOWER = *variate*	Lower limits for parameters; default !(0.00001, 0.001, 0.001)
UPPER = *variate*	Upper values for parameters; default !(0.99999, 5, 1000)
PLOT = *string token*	What to plot (histogram, density, logdensity, inference, loginference); default hist, dens, logd, infe, logi
WINDOW = *scalar*	Window for graphs; default 1
KEYWINDOW = *scalar*	Key window for Inference plot; default 2
MAXCYCLE = *scalar*	Maximum iteration cycles; default 50
TOLERANCE = *scalar* or *variate*	Tolerance for convergence of parameters; default 0.01 for Beta, and 0.001 for Gamma

Parameters

PROBABILITIES = *variates*	Significance values, must lie between 0 and 1
ESTIMATES = *variates*	Saves the estimates of mixture parameters φ, *A* and *B*
FDR = *variates*	Saves the False Discovery Rates at the *p*-values in PROBABILITIES i.e. *q*-values
FRR = *variates*	Saves the False Rejection Rates at the *p*-values in PROBABILITIES
POWER = *variates*	Saves the power estimates as a function of the *p*-values in PROBABILITIES
POSTHA = *variates*	Saves the Posterior Probability of H_a at the *p*-values in PROBABILITIES
LOGLIKELIHOOD = *scalars*	Value of the loglikelihood at end of the iteration process
NCYCLES = *scalars*	Number of iterations taken to convergence

FEXACT2X2 procedure

Does Fisher's exact test for 2×2 tables (M.S. Ridout & M.W. Patefield).

Option

PRINT = *string tokens*	Controls printed output (probabilities, tables); default prob

Parameters

TABLE = *tables* or *variates*	The numbers in each 2×2 table, ordered row by row or column by column
PROBABILITIES = *variates*	Saves the probabilities for each table in a variate of length 6 (to store in positions 1, 3 and 5 one-tailed, two-tailed calculated as twice the one-tailed probability, and as the sum of the probabilities of all tables with probability less than that of the observed table with the corresponding mid-p values stored in positions 2, 4 and 6)

FFRAME procedure

Forms multiple windows in a plot-matrix for high-resolution graphics (P.W. Goedhart).

Options

PRINT = *string tokens*	Whether to display the layout and numbering of the plot-matrix in a table or in a high-resolution test-graph on the current device (table, testgraph); default *
ARRANGEMENT = *string token*	Type of plot-matrix (rectangle, square, lowersymmetric, uppersymmetric, diagonal); default rectangle
ROWS = *scalar*	Number of rows of plot-matrix; default 3
COLUMNS = *scalar*	Number of columns of plot-matrix; default 3
DIAGONALWINDOWS = *string token*	Whether to include or exclude the diagonal in symmetric plot-matrices (include, exclude); default include
SQUARESHAPES = *string token*	Whether to force the individual windows, excluding margins for annotation, to be square (yes, no); default no
STARTWINDOW = *scalar*	Specifies the number of the first window; default 1
TESTGRAPH = *variate*	Specifies windows to be displayed in a test-graph (if this option is set, only a test-graph is produced and all other settings are ignored); default *
NUMBERING = *string token*	Controls the way in which the individual windows are numbered (rowwise, columnwise); default rowwise
DEFINE = *string token*	Whether to define the windows within the procedure (windows, nothing); default wind
CLEARWINDOW = *scalar* or *variate*	Defines the windows for which the screen should be cleared; i.e. specifies the elements of the SCREEN pointer which are set to the single-values text 'clear', other element of SCREEN are set to 'keep'; default 1
RLOWER = *scalar*	Lowest y device coordinate; default 0
RUPPER = *scalar*	Highest y device coordinate; default 1
CLOWER = *scalar*	Lowest x device coordinate; default 0
CUPPER = *scalar*	Highest x device coordinate; default 1
RSKIP = *scalar*	Space between windows along the y-axis; default 0
CSKIP = *scalar*	Space between windows along the x-axis; default 0
MARGIN = *string tokens*	Sets the size of the margins for labels and titles (xtitle, ytitle, none, small); default *
YMLOWER = *scalar*	Size of bottom margin (x-axis labelling) in each window; default *
YMUPPER = *scalar*	Size of upper margin (overall title) in each window; default *
XMLOWER = *scalar*	Size of left-hand margin (y-axis labelling) in each window; default *
XMUPPER = *scalar*	Size of right-hand margin in each window; default *
RMLOWER = *scalar*	Additional size of bottom margin (x-axis labelling) in windows at the bottom of the plot-matrix; default 0
RMUPPER = *scalar*	Additional size of upper margin (overall title) in windows at the top of the plot-matrix; default 0
CMLOWER = *scalar*	Additional size of left-hand margin (y-axis labelling) windows at the left of the plot-matrix; default 0
CMUPPER = *scalar*	Additional size of right-hand margin in windows at the right of the plot-matrix; default 0
BACKGROUND = *text* or *scalar*	Specifies the colour to be used for the background in each window (where allowed by the graphics device); default 'background'

Parameters

NGRAPHS = *scalar*	To save the number of windows in the plot-matrix
SWINDOW = *pointer*	Pointer to save scalars with window numbers
SYLOWER = *pointer*	Pointer to save scalars with lower y device coordinates for each window

SYUPPER = *pointer*	Pointer to save scalars with upper y device coordinates for each window
SXLOWER = *pointer*	Pointer to save scalars with lower x device coordinates for each window
SXUPPER = *pointer*	Pointer to save scalars with upper x device coordinates for each window
SSCREEN = *pointer*	Pointer to save single-valued texts with value 'clear' or 'keep'; this depends only on the setting of the CLEARWINDOW option
SMYLOWER = *pointer*	Pointer to save scalars with size of bottom margins for each window
SMYUPPER = *pointer*	Pointer to save scalars with size of upper margins for each window
SMXLOWER = *pointer*	Pointer to save scalars with size of left-hand margin for each window
SMXUPPER = *pointer*	Pointer to save scalars with size of right-hand margin for each window

FFREERESPONSEFACTOR procedure

Forms multiple-response factors from free-response data (R.W. Payne).

Options

MRESPONSE = *pointer*	Pointer with a factor for each RESPONSECODE, indicating which of the DATA texts contain that response
RESPONSECODES = *text*	Specifies the codes to look for in the DATA texts
LABELCODES = *text*	Strings to label the factors within the MRESPONSE pointer; default RESPONSECODES
DUPLICATECODES = *factor*	Defines groupings of duplicate or alternative codes within the RESPONSECODES text
EXCLUDENULL = *string token*	Whether to exclude the factor recording which DATA contain none of the RESPONSECODES (yes, no); default no
SUFFIXNULL = *scalars*	Suffix to use to represent the null factor in MRESPONSE; default 0
LABELNULL = *text*	Label to use to represent a the null factor in MRESPONSE; default 'none'
DATAFORMAT = *string token*	Whether the data for the respondents is given line-by-line within the DATA text(s) or whether there is a separate text for each respondent (linebyline, textbytext); default line
CASE = *string token*	Whether to treat the case of letters (small or capital) as significant when searching for the codes (significant, ignored); default igno
MULTISPACES = *string token*	Whether to treat differences between multiple spaces and single spaces as significant, or to treat them all like a single space (significant, ignored); default igno
DISTINCT = *string tokens*	Whether to require each RESPONSECODE to have one or more separators to its left or right within each DATA text (left, right); default left, righ
SEPARATOR = *text*	Characters to use as separators; default ' ,;:.'

Parameter

DATA = *texts*	Information from the respondents

FHADAMARDMATRIX procedure

Forms Hadamard matrices (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (monitoring); default * i.e. none
METHOD = *string token*	Method of construction (firstpaley, secondpaley, stored, sylvestre, tensorproduct, turyn,

williamson); default * i.e. determined automatically

Parameters

NROWS = *scalars*	Number of rows of the matrices
HADAMARDMATRIX = *matrices*	Saves the Hadamard matrices
ERROR = *scalars*	Returns 0 if the matrix has been formed successfully and 1 if not

FHAT procedure

Calculates an estimate of the F nearest-neighbour distribution function (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	What to print (summary); default summ

Parameters

Y1 = *variates*	Vertical coordinates of the first spatial point patterns; no default – this parameter must be set
X1 = *variates*	Horizontal coordinates of the first spatial point patterns; no default – this parameter must be set
Y2 = *variates*	Vertical coordinates of the second spatial point patterns; no default – this parameter must be set
X2 = *variates*	Horizontal coordinates of the second spatial point patterns; no default – this parameter must be set
S = *variates*	Vectors of distances to use; no default – this parameter must be set
FVALUES = *variates*	Variates to receive the estimated F nearest-neighbour distribution functions
NNDISTANCES = *variates*	Variates to receive the nearest-neighbour distances

FIELLER procedure

Calculates effective doses or relative potencies (P.W. Lane).

Options

PRINT = *string token*	What to output (value); default valu
ESTIMATES = *variate*	Parameter estimates; default extracts these with RKEEP
VCOVARIANCE = *symmetric matrix*	Variances and covariances; default extracts these with RKEEP
%LIMIT = *scalar*	Percentage points for limits; default 95, thus giving 95% confidence limits
RELATIVE = *string token*	Whether to calculate relative potencies (no, yes); default no
LINK = *string token*	Which link function to assume when forming effective doses (probit, logit, complementaryloglog); default obtained using RKEEP, if the ESTIMATES or VARIANCES are obtained in that way, otherwise prob
LOGBASE = *string token*	Base of antilog transformation to be applied to value and limits, (ten, e); default * i.e. none
DF = *scalar*	If this has a non-missing value, a t-distribution is used instead of a Normal distribution to calculate the confidence limits; default obtained using RKEEP if the ESTIMATES or VARIANCES are obtained in that way (setting DF to the number of residual d.f. when the dispersion factor is estimated, or a missing value when it is fixed), otherwise the default is a missing value

Parameters

TREATMENT = *variates* or *scalars*	Positions of intercept parameters in list of estimates; default first estimate
SLOPE = *variates* or *scalars*	Positions of slope parameters in list of estimates; default last estimate
%DOSE = *variates* or *scalars*	Percentage dose; default 50, thus giving LD50
VALUE = *variates* or *scalars*	To store estimated values

LOWER = *variates* or *scalars*	To store lower limits
UPPER = *variates* or *scalars*	To store upper limits
SE = *variates* or *scalars*	To store approximate s.e.s of values

FILEREAD procedure

Reads data from a file (P.W. Lane).

Options

PRINT = *string tokens*	What output to display (summary, groups, comments, firstline); default summ, grou, comm, firs
NAME = *text*	External name of the data file; no default in batch mode, name is prompted for in interactive mode
END = *text*	What string terminates data; default ':' (the end of file also terminates data for any setting); the setting END=* is not allowed
MISSING = *text*	What character represents missing values; default '*'
SKIP = *scalar* or *text*	Number of lines to skip at the start of the file, or string to indicate the record before the first record of data; default 0
MAXCATEGORY = *number*	The maximum number of categories for which a structure is defined to be a factor unless otherwise specified by FGROUPS; default 10
COMMENTSYMBOLS = *text*	What characters to treat as introducing comments if found in the first column at the start of the file; default double-quote character (")
IMETHOD = *string token*	How identifiers are to be specified for the data structures to be read (supply, read, none); default supp
ISAVE = *pointer*	To store the identifiers, whether read or supplied, and to provide suffixed identifiers for data with no specified identifiers
SEPARATOR = *text*	What (single) character separates successive values; default is the space character

Parameters

IDENTIFIER = *identifiers*	Names for the data structures that are to be read; these are prompted for if this is unset when running interactively with IMETHOD=supply; identifiers are redefined if they have been used previously
FGROUPS = *string tokens*	Whether to form each data structure into a factor (check, form, leave); default chec, which causes FILEREAD when running interactively to ask about any structure whose number of distinct values is less than or equal to MAXCATEGORY, and when running in batch to define as factors all structures with MAXCATEGORY or fewer distinct values (note: for compatibility with earlier releases, yes and no can be used as synonyms of form and leave)
REPRESENTATION = *string tokens*	What representation to assume for each data structure (numbers, characters); default unset – representation is determined by whether the first value is a number; if set for one structure, this parameter must be set for all structures

FILTER directive

Filters time series by time-series models (synonym of TFILTER).

Option

PRINT = *string tokens*	What to print (series); default *

Parameters

OLDSERIES = *variates*	Time series to be filtered
NEWSERIES = *variates*	To save filtered series
FILTER = *TSMs*	Models to filter with respect to

ARIMA = *TSMs*	ARIMA models for time series

FIT directive

Fits a linear, generalized linear, generalized additive or generalized nonlinear model.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, grid, confidence); default mode, summ, esti or grid if NGRIDLINES is set
CALCULATION = *expression structures*	Calculation of explanatory variates involving nonlinear parameters
OWN = *scalar*	Option setting for OWN directive if this is to be used rather than CALCULATE to calculate explanatory variates
CONSTANT = *string token*	How to treat the constant (estimate, omit, ignore); default esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default as in previous TERMS statement, or 3 if no TERMS given
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
NGRIDLINES = *scalar*	Number of values of each nonlinear parameter for a grid of function evaluations
SELINEAR = *string token*	Whether to calculate s.e.s for linear parameters when nonlinear parameters are also estimated (yes, no); default no
INOWN = *identifiers*	Setting to be used for the IN parameter of OWN if used to calculate explanatory variates
OUTOWN = *identifiers*	Setting to be used for the OUT parameter of OWN if used to calculate explanatory variates
AOVDESCRIPTION = *text*	Description for line in accumulated analysis of variance (or deviance) table when POOL=yes

Parameter

formula	List of explanatory variates and factors, or model formula

FITCURVE directive

Fits a standard nonlinear regression model.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated,

	monitoring); default mode, summ, esti
CURVE = *string token*	Type of curve (exponential, dexponential, cexponential, lexponential, logistic, glogistic, gompertz, ldl, qdl, qdq, fourier, dfourier, gaussian, dgaussian, emax, gemax); default expo
SENSE = *string token*	Sense of curve (right, left); default righ
ORIGIN = *scalar*	Constrained origin; default *
NONLINEAR = *string token*	How to treat nonlinear parameters between groups (common, separate); default comm
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default as in previous TERMS statement, or 3 if no TERMS given
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical); default *
FPROBABILITY = *string token*	Printing of probabilities for variance ratios (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob

Parameter

formula	Explanatory variate, list of variate and factor, or variate*factor

FITINDIVIDUALLY procedure

Fits regression models one term at a time (R.W. Payne).

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence); default mode, summ, esti
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default 3
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions

PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
DEVIANCE = *scalar*	Saves the residual deviance
DF = *scalar*	Saves the residual d.f.
LACKOFFIT = *string token*	Whether to use observations with replicated values of the explanatory variables to split the final residual term into a 'true' residual and lack of fit (estimate, omit); default omit

Parameter

TERMS = *formula*	Terms to be fitted

FITMULTINOMIAL procedure

Fits generalized linear models with multinomial distribution (R.W. Payne).

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence); default mode, summ, esti
RESPONSEFACTOR = *factor*	Factor representing the response categories of the multinomial distribution
CLASSIFICATION = *factors*	Factors classifying the subjects; default uses the factors in TERMS
FACTORIAL = *scalar*	Limit for expansion of model terms from TERMS; default 3
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary (%variance, %ss, adjustedr2, r2, dispersion, %meandeviance, %deviance, aic, bic, sic); default disp
PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
FULL = *string token*	Whether to assign all possible parameters to factors and interactions (yes, no); default no

Parameter

TERMS = *formula*	Terms to be fitted

FITNONLINEAR directive

Fits a nonlinear regression model or optimizes a scalar function.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, grid); default mode, summ, esti or grid if NGRIDLINES is set
CALCULATION = *expression structures*	Calculation of fitted values or of explanatory variates involving nonlinear parameters; default * (valid only if OWN set)
OWN = *scalar*	Option setting for OWN directive if this is to be used rather than CALCULATE; default * requests CALCULATE to be used
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti

FACTORIAL = *scalar* — Limit for expansion of model terms; default as in previous TERMS statement, or 3 if no TERMS given

POOL = *string token* — Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no

DENOMINATOR = *string token* — Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss

NOMESSAGE = *string tokens* — Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df); default *

FPROBABILITY = *string token* — Printing of probabilities for variance and deviance ratios (yes, no); default no

SELECTION = *string tokens* — Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions

NGRIDLINES = *scalar* — Number of values of each parameter for a grid of function evaluations; default *

SELINEAR = *string token* — Whether to calculate s.e.s for linear parameters (yes, no); default no

INOWN = *identifiers* — Setting to be used for the IN parameter of OWN if used in place of CALCULATE; default *

OUTOWN = *identifiers* — Setting to be used for the OUT parameter of OWN if used in place of CALCULATE; default *

Parameter

formula — List of explanatory variates and/or one factor to be used in linear regression, within nonlinear optimization

FKEY directive

Forms design keys for multi-stratum experimental designs, allowing for confounded and aliased treatments.

Options

BASICFACTORS = *factors* — Factors indexing the units of the design

ADDEDFACTORS = *factors* — Factors to be allocated to the units of the design

KEY = *matrix* — Stores the design key (ADDEDFACTORS × BASICFACTORS)

INKEY = *matrix* — Can be used to input existing allocations for some of the added factors

HIERARCHIES = *matrix* — Can be used to specify that some of the factors must be constant within each combination of levels of other factors; the matrix has a row for each added factor and columns first for the basic factors and then for the added factors, ones in the entries where the row factor must be constant within the combinations of the column factors, zero elsewhere

SEED = *scalar* — Can provide a seed to generate a random permutation of the sets of basic effects that may be allocated to each added factor, thus producing design randomly selected from all those that might be possible; default * i.e. no permutation

ROWPRIMES = *variate* — Prime numbers for the rows of the KEY matrix

COLPRIMES = *variate* — Prime numbers for the columns of the KEY matrix

ROWMAPPINGS = *variate* — Mappings from the rows of the KEY to the TREATMENTFACTORS

COLMAPPINGS = *variate* — Mappings from the columns of the KEY to the BLOCKFACTORS

SAVE = *identifier*	Structure to save all the information about the formation of the design; this can then be input later to give a different design (if possible) with the same properties

Parameters

REQUIRED = *formula structures*	Formulae each defining a list of terms that are to be estimated in the analysis
NONNEGLIGIBLE = *formula structures*	Formulae each specifying terms that cannot be ignored in the context of the corresponding REQUIRED formula

FLRV directive

Forms the values of LRV structures.

Options

PRINT = *string tokens*	Printed output required (roots, vectors); default * i.e. no printing
NROOTS = *scalar*	Number of roots or vectors to print; default * i.e. print them all
SMALLEST = *string token*	Whether to print the smallest roots instead of the largest (yes, no); default no
TOLERANCE = *scalar*	Tolerance for detecting zero roots

Parameters

INMATRIX = *matrices* or *symmetric matrices*	Matrices whose latent roots and vectors are to be calculated
LRV = *LRVs*	LRV to store the latent roots and vectors from each INMATRIX
WMATRIX = *symmetric matrices*	(Generalized) within-group sums of squares and products matrix used in forming the two-matrix decomposition; if any of these is omitted, it is taken to be the identity matrix, giving the usual spectral decomposition
ILRV = *LRVs*	LRV to store the imaginary parts of the latent roots and vectors arising from the decomposition of a non-symmetric matrix

FMEGAENVIRONMENTS procedure

Forms mega-environments based on winning genotypes from an AMMI-2 model (D.A. Murray & M. Malosetti).

Option

PRINT = *string tokens*	What to print (summary); default summ

Parameters

DATA = *variates*	Provides the data to be analysed
GENOTYPES = *factors*	Specifies the genotypes
ENVIRONMENTS = *factors*	Specifies the environments (or locations when years are supplied)
YEARS = *factors*	Specifies years within locations
MEGAENVIRONMENTS = *factors*	Saves the mega-environments

FMFACTORS procedure

Forms a pointer of factors representing a multiple-response (R.W. Payne).

Options

MRESPONSE = *pointer*	Pointer with a factor for each code, indicating the units where it occurs in the CODE texts or variates
RESPONSECODES = *text* or *variate*	Saves the set of distinct multiple-response codes
CODENULL = *text* or *variate*	Code(s) used to represent a null value in the CODE texts or variates; default * or ''
EXCLUDENULL = *string token*	Whether to exclude the null factor recording the respondents that made no reply (yes, no); default no
SUFFIXNULL = *scalar*	Suffix to use to represent the null factor in MRESPONSE; default 0
LABELNULL = *text*	Label to use to represent the null factor in MRESPONSE; default 'none'

LDIRECTION = *string token* — How to order the labels from textual codes (ascending, given); default asce

Parameter

CODE = *texts*, *variates* or *factors* — Codes from the respondents

FNCORRELATION procedure

Calculates correlations from variances and covariances, together with their variances and covariances (S.A. Gezan).

Options

PRINT = *string token* — Output required (summary); default summ

IVARIANCES = *variate* — Indexes of the two variances in the ESTIMATES variate; no default – must be set

ICOVARIANCE = *scalar* — Index of the covariance in the ESTIMATES variate; no default – must be set

Parameters

ESTIMATES = *variates* — Estimated values of the variances and covariances

VCOVARIANCE = *symmetric matrices* — Variance-covariance matrix of the variances and covariances

FUNCTIONESTIMATE = *scalars* — Saves the estimated value of the function

SE = *scalars* — Saves the standard error of the function estimate

NEWESTIMATES = *variates* — Saves new vectors of estimates, including the estimated value of the function

NEWVCOVARIANCE = *symmetric matrices* — Saves variance-covariance matrices for the new vectors (including the function estimate)

FNLINEAR procedure

Estimates linear functions of one or more random variables, and calculates their variances and covariances (S.A. Gezan).

Options

PRINT = *string token* — Output required (summary); default summ

CONSTANTVALUE = *scalar* — Constant value for the function; default 0

COEFFICIENTS = *scalar* — Linear coefficients for the random variables in the function; no default – must be set

Parameters

ESTIMATES = *variates* — Estimated values of the random variables

VCOVARIANCE = *symmetric matrices* — Variance-covariance matrix of the random variable estimates

FUNCTIONESTIMATE = *scalars* — Saves the estimated value of the function

SE = *scalars* — Saves the standard error of the function estimate

NEWESTIMATES = *variates* — Saves new vectors of estimates, including the estimated value of the function

NEWVCOVARIANCE = *symmetric matrices* — Saves variance-covariance matrices for the NEWESTIMATES

FNPOWER procedure

Estimates products of powers of two random variables, and calculates their variances and covariances (S.A. Gezan).

Options

PRINT = *string token* — Output required (summary); default summ

CONSTANTVALUE = *scalar* — Constant value for the function; default 0

POWERS = *variate* — Specifies the powers of the two random variables

INDEXES = *variate* — Specifies the locations of the random variables corresponding to the elements of the POWERS variate

CORRECTION = *string token* — Whether to apply an additional correction to the variance of a product, using terms from the second-order approximation; default no

Parameters

ESTIMATES = *variates*	Estimated values of the random variables
VCOVARIANCE = *symmetric matrices*	Variance-covariance matrix of the random variable estimates
FUNCTIONESTIMATE = *scalars*	Saves the estimated value of the function
SE = *scalars*	Saves the standard error of the function estimate
NEWESTIMATES = *variates*	Saves new vectors of estimates, including the estimated value of the function
NEWVCOVARIANCE = *symmetric matrices*	Saves variance-covariance matrices for the new vestors (including the function estimate)

FOCCURRENCES procedure

Counts how often each pair of treatments occurs in the same block (W. van den Berg).

Options

PRINT = *string tokens*	Controls printed output (concurrences, efficiency); default conc, effi
DIAGONAL = *string token*	What to store on the diagonal of the concurrence matrix (missingvalues, replication); default repl

Parameters

TREATMENTS = *factors*	Supplies the treatment factor
REPLICATES = *factors*	Supplies the replicates factor
BLOCKS = *factors*	Supplies the block factor
CONCURRENCES = *symmetric matrices*	Saves the concurrence matrix, recording the number of times each pair of treatments occurs together in a block
EFFICIENCY = *scalars*	Save the efficiency of the design

FOR directive

Introduces a loop; subsequent statements define the contents of the loop, which is terminated by the directive ENDFOR.

Options

NTIMES = *scalar*	Number of times to execute the loop; default is to execute as many times as the length of the first parameter list or once if the first list is null
INDEX = *scalar*	Records the loop index
START = *scalar*	Defines an integer initial value for the loop index; default 1
END = *scalar*	Defines an integer final value for the loop index
STEP = *scalar*	Defines an integer amount by which to increase the index each time the loop is executed; default 1
VALUES = *variate*	Defines a set of values to be taken successively by the loop index (overrides START, END and STEP if these are specified too)

Parameters

	Any number of parameter settings of the form *identifier* = *list of data structures*; the *identifier* is set up as a dummy which is then used within the loop to refer, in turn, to the structures in the list

FORECAST directive

Forecasts future values of a time series (synonym of TFORECAST).

Options

PRINT = *string tokens*	What to print (forecasts, limits, setransform, sfe); default fore,limi
CHANNEL = *scalar*	Channel number for output; default * i.e. current output channel
ORIGIN = *scalar*	Number of known values to be incorporated; default 0
UPDATE = *string token*	Whether to update the forecast origin to the end of the new

observations (yes, no); default no

NEWOBSERVATIONS = *variate* — Variate of length ≥ ORIGIN providing new values of the time series to be incorporated (must be set if ORIGIN > 0)

SFE = *variate* — Saves standardized forecast errors; default *

MAXLEAD = *scalar* — Maximum lead time i.e number of forecasts to be made; default * defines the number as the length of FORECAST variate

FORECAST = *variate* — Variate of length MAXLEAD to save forecasts of output series; default *

SETRANSFORM = *variate* — Saves standard errors of the forecasts (on transformed scale, if defined); default *

LOWER = *variate* — Saves lower confidence limits; default *

UPPER = *variate* — Saves upper confidence limits; default *

PROBABILITY = *scalar* — Probability level for confidence limits; default 0.9

COMPONENTS = *pointer* — Contains variates (of length ORIGIN + MAXLEAD) to save components of the forecast

SAVE = *identifier* — Save structure to supply fitted model; default * i.e. that from last model fitted

Parameters

FUTURE = *variates* — Variates (of length ORIGIN + MAXLEAD) containing future values of input series

METHOD = *string tokens* — How to treat future values of input series (observations, forecasts); default obse

FORMULA directive

Declares one or more formula data structures.

Options

VALUE = *formula* — Value for all the formulae; default *

MODIFY = *string token* — Whether to modify (instead of redefining) existing structures (yes, no); default no

IPRINT = *string tokens* — Information to be used by default to identify the formulae in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers* — Identifiers of the formulae

VALUE = *formula structures* — Value for each formula

EXTRA = *texts* — Extra text associated with each identifier

FOURIER directive

Calculates cosine or Fourier transforms of real or complex series.

Option

PRINT = *string tokens* — What to print (transforms); default *

Parameters

SERIES = *variates* — Real part of each input series

ISERIES = *variates* — Imaginary part of each input series

TRANSFORM = *variates* — To save real part of each output series

ITRANSFORM = *variates* — To save imaginary part of each output series

PERIODOGRAM = *variates* — To save periodogram of each transform

FPARETOSET procedure

Forms the Pareto optimal set of non-dominated groups (W. van den Berg).

Options

PRINT = *string token* — Controls whether to print the groups (groups); default grou

PLOT = *string token* — Controls whether to plot the data, using different coloured points to indicate the groups (data); default data

NGROUPS = *scalar* — Number of groups to form; default 1

GROUPS = *factor*	Saves the group allocations
TITLE = *text*	Title for the plot; default * i.e. none
LABELS = *text*, *variate* or *factor*	Labels for the items; default * i.e. none

Parameters

DATA = *variates*	Data variates, defining the properties of the items
SIGN = *scalars*	Value by which to multiply each DATA variate: for example, this can be set to set to −1 if the variate is to be minimized instead of being maximized; default 1
TITLE = *texts*	Title to use for the axis of each DATA variate in the plot; if unset, its identifier is used

FPLOTNUMBER procedure

Forms plot numbers for a row-by-column design (K. Punyawaew).

Options

FIRSTPLOT = *string token*	Defines the starting location for numbering the plots (lowleft, lowright, upleft, upright); default uple
PLOTORDER = *string token*	Defines the order in which the numbers are allocated (colserpentine, colbycol, rowserpentine, rowbyrow); default rowb

Parameters

NROWS = *scalars*	Number of rows in the design
NCOLUMNS = *scalars*	Number of columns in the design
PLOTNUMBER = *factors*	Saves the plot numbers

FPROJECTIONMATRIX procedure

Forms a projection matrix for a set of model terms (R.W. Payne).

No options

Parameters

TERMS = *formula structure*	Defines the model terms corresponding to the design matrices whose projection matrices are required
PROJECTION = *symmetric matrix*	Saves the projection matrix for each formula structure

FPSEUDOFACTORS directive

Determines patterns of confounding and aliasing from design keys, and extends the treatment model to incorporate the necessary pseudo-factors.

Options

TREATMENTSTRUCTURE = *formula*	Treatment model for the design
BLOCKSTRUCTURE = *formula*	Block model for the design
FACTORIAL = *scalar*	Limit on the number of factors in each treatment term
LROWS = *factors* or *scalars*	Numbers of levels of factors, or factors, corresponding to the rows of the key matrices
LCOLUMNS = *factors* or *scalars*	Numbers of levels of factors, or factors, corresponding to the columns of the key matrices
NEWTREATMENTSTRUCTURE = *identifier*	Store the extended treatment model
PSEUDOFACTORS = *pointer*	Pseudo-factors required for the keys
NPSEUDOFACTORS = *scalar*	Number of pseudo-factors required for the keys
KEYPSEUDOFACTORS = *matrix*	Key to generate the pseudo-factors from the treatment factors
KEYCONTRASTS = *matrix*	Key partitioning the treatment terms into orthogonal sets of contrasts

Parameters

KEY = *matrices*	Design keys
KEYINVERSE = *matrices*	Store the inverses of the design keys
ALIASSETS = *variates*	Stores aliasing information about the orthogonal sets of treatment contrasts
RESOLUTION = *scalars*	Saves the resolution number of the design constructed by each

key

FRAME directive

Defines the positions and appearance of the plotting windows within the frame of a high-resolution graph.

Options

GRID = *string tokens*	Specifies grid lines (xy, xz, yx, yz, zx, zy)
BOXFRAME = *string tokens*	Whether to include a box enclosing the entire frame (include, omit)
BACKGROUND = *scalars* or *texts*	Specifies the colour to be used for the background of the whole frame (where allowed by the graphics device)
RESET = *string token*	Whether to reset the axis definition to the default values (no, yes); default no

Parameters

WINDOW = *scalars*	Window numbers
YLOWER = *scalars*	Lower y device coordinate for each window
YUPPER = *scalars*	Upper y device coordinate for each window
XLOWER = *scalars*	Lower x device coordinate for each window
XUPPER = *scalars*	Upper x device coordinate for each window
YMLOWER = *scalars*	Size of bottom margin (for x-axis labels)
YMUPPER = *scalars*	Size of upper margin (for overall title)
XMLOWER = *scalars*	Size of left-hand margin (for y-axis labels)
XMUPPER = *scalars*	Size of right-hand margin
BACKGROUND = *scalars* or *texts*	Specifies the colour to be used for the background in each window (where allowed by the graphics device)
BOX = *string tokens*	Whether to include a box enclosing the plotted graphic (include, omit)
BOXSURFACE = *string token*	Box to include in a surface plot (full, bounded, omit)
BOXKEY = *string token*	Box to draw around key (full, bounded, omit)
PENTITLE = *scalars*	Pen to use to write the overall title
PENKEY = *scalar*	Pen to use for the key
PENGRID = *scalar*	Pen to use to draw the grid lines
SCALING = *string token*	How to scale the axis in each window (xyequal, xzequal, yzequal, xyzequal)
TPOSITION = *string token*	Position of title (right, left, center, centre)
CINTERIOR = *scalars* or *texts*	Specifies the colour to be used for the interior of each window (where allowed by the graphics device)
CFRAME = *scalars* or *texts*	Specifies the colour to be used for the frame of each window (where allowed by the graphics device)
CTITLE = *scalars* or *texts*	Specifies the colour to be used for the title bar of each window (where allowed by the graphics device)
AXES = *identifiers* or *pointers*	Additional oblique axes to include in each window
SAVE = *pointers*	Saves details of the current settings for the window concerned

FREGULAR procedure

Expands vectors onto a regular two-dimensional grid (R.W. Payne).

Options

ROWS = *factor*	Original row factor
COLUMNS = *factor*	Original column factor
NEWROWS = *factor*	New row factor expanded onto the full grid
NEWCOLUMNS = *factor*	New column factor expanded onto the full grid
SORT = *string token*	Whether to sort the new values into row × column order (yes, no); default no

Parameters

OLDVECTOR = *variates*, *factors* or *texts*	Original data vectors
NEWVECTOR = *variates*, *factors* or *texts*	New vector with values, provided by the VALUES parameter,

	inserted in the units added to complete the grid
VALUES = *variates*, *scalars* or *texts*	Values to insert in the units added to complete the grid; default is to insert missing values

FRENAME directive

Renames files.

No options

Parameters

OLD = *texts*	Name of each file to rename
NEW = *texts*	New name for each file
OVERWRITE = *string tokens*	Whether to overwrite any existing files (yes, no); default no

FRESTRICTEDSET procedure

Forms vectors with the restricted subset of a list of vectors (R.W. Payne).

Options

METHOD = *string token*	Whether to form the new vectors only when the old vectors are restricted or aways (always, whenrestricted); default alwa
RESTRICTED = *scalar*	Scalar set to 1 or 0 according to whether or not the old vectors are found to be restricted
VRESTRICTED = *variate*	Variate with each unit set to 1 or 0 according to whether or not that unit is restricted in any of the OLDVECTORS

Parameters

OLDVECTOR = *factors*, *variates* or *texts*	List of vectors, one or more of which may be restricted
NEWVECTOR = *factors*, *variates* or *texts*	New vectors which will contain only the unrestricted units of the old vectors
SETLEVELS = *string token*	Whether to reform the levels (and labels) of factors to exclude those that do not occur in the restricted subset (yes, no); default no

FRIEDMAN procedure

Performs Friedman's nonparametric analysis of variance (S. Langton).

Options

PRINT = *string tokens*	Output required (test, ranks); default test
TREATMENTS = *factor*	Treatment factor
BLOCKS = *factor*	Block factor

Parameters

DATA = *variates*	Identifier of the variate holding the data values
RANKS = *variates*	Saves the ranks
STATISTIC = *scalars*	Saves the test statistic
DF = *scalars*	Saves the degrees of freedom for the chi-square approximation
PROBABILITY = *scalars*	Saves the probability value for the chi-square statistic

FROWCANONICALMATRIX procedure

Puts a matrix into row canonical, or reduced row echelon, form (C.J. Brien).

Option

PRINT = *string token*	Controls printed output (rowcanonicalmatrix); default * i.e. none

Parameters

MATRIX = *matrices*	Matrix to be put into row canonical form
ROWCANONICALMATRIX = *identifiers*	Matrix in row canonical form

FRQUANTILES directive

Forms regression quantiles.

Options

Y = *variate*	Response variate
DESIGNMATRIX = *matrix*	Design matrix for the regression model
TOLERANCE = *scalar*	Tolerance for the algorithm; default 10^{-12}

Parameters

PRQUANTILE = *scalars*	Values for which to perform the quantile regressions
RESIDUALS = *variates*	Parameter estimates from each quantile regression
ESTIMATES = *variates*	Estimates from each quantile regression
XBARQUANTILES = *variates*	When PRQUANTILE is set to a missing value, saves the sum of the mean of each design column multiplied by its regression quantile for all the quantile solutions
CUMPROBABILITIES = *variates*	When PRQUANTILE is set to a missing value, saves the cumulative probabilitiy values at which the estimated regression quantiles change
EXIT = *scalars*	Saves an exit code, with 0 to indicate success

FRTPRODUCTDESIGNMATRIX procedure

Forms summation, or relationship, matrices for model terms (C.J. Brien).

No options

Parameters

TERM = *formula structures*	Model terms corresponding to design matrices whose summation matrices are required
MATRIX = *symmetric matrices*	Saves the summation or relationship matrix for each term

FSIMILARITY directive

Forms a similarity matrix or a between-group-elements similarity matrix or prints a similarity matrix.

Options

PRINT = *string token*	Printed output required (similarity, summary); default * i.e. no printing
STYLE = *string token*	Print percentage similarities in full or just the 10% digit (full, abbreviated); default full
METHOD = *string token*	Form similarity matrix or rectangular between-group-element similarity matrix (similarity, betweengroupsimilarity); default simi
SIMILARITY = *matrix* or *symmetric matrix*	Input or output matrix of similarities; default *
GROUPS = *factor*	Grouping of units into two groups for between-group-element similarity matrix; default *
PERMUTATION = *variate*	Permutation of units (possibly from HCLUSTER) for order in which units of the similarity matrix are printed; default *
UNITS = *text* or *variate*	Unit names to label the rows of the similarity matrix; default *
MINKOWSKI = *scalar*	Index *t* for use with TEST=minkowski

Parameters

DATA = *variates* or *factors*	The data values
TEST = *string tokens*	Test type, defining how each DATA variate or factor is treated in the calculation of the similarity between each unit (simplematching, jaccard, russellrao, dice, antidice, sneathsokal, rogerstanimoto, cityblock, manhattan, ecological, euclidean, pythagorean, minkowski, divergence, canberra, braycurtis, soergel); default * ignores that variate or factor
RANGE = *scalars*	Range of possible values of each DATA variate or factor; if omitted, the observed range is taken

FSPREADSHEET procedure

Creates a Genstat spreadsheet file (GWB or GSH) from specified data structures, PC Windows only (D.B. Baird).

Options

OUTFILE = *text*	Name of GSH file to store data in
SHEET = *number*	Sequence number of existing sheet, if this is set to 0 the data will be added to the first compatible spreadsheet open in the Windows interface
METHOD = *string token*	What to do with any existing columns with the same names as the new columns (replace, rename); default rena
READONLY = *string token*	Whether to make the complete sheet read-only (yes, no); default no
TITLE = *text*	The title associated with the spreadsheet
POINTER = *pointer* or *text*	A pointer or a name of a pointer to the columns in the spreadsheet
ANALYSIS = *text*	Genstat directives to analyse columns in the spreadsheet
ASETUP = *text*	Genstat directives to be run once before the analysis of any columns in the spreadsheet
ADUMMY = *text*	The name of the dummy (if any) used in the ANALYSIS directives
CURSOR = *variate*	A variate of length 2 giving the active cell position (x,y) when the spreadsheet is first displayed
NOUNITS = *string token*	Whether to stop the inclusion of a units column in the spreadsheet (yes, no); default no
BOOK = *number*	Window number of existing book, if this is set to 0 the sheet will be created in a new book, if to −1 it will be created in the last book formed with BOOK=0, and if set to −2 it will be created in the last book created in the Windows interface.
PAGENAME = *text*	The 32 character text to be displayed on the sheet tab
ROWCOLOURS = *factor*	The factor to be used for colouring the rows (the factor must have colours defined by the FACCOLOURS parameter)
TABLEFORMAT = *string token*	The format to use when displaying tables with two or more classifying factors (page, column); default page
FILEFORMAT = *string token*	The format to use for the spreadsheet file (GWB, GSH); default GWB
MARGINNAME = *text*	The 60 character text to be displayed for the margin labels
FROZENCOLUMNS = *scalar*	The number of columns to freeze on the left hand side of the spreadsheet; default 0 i.e. none

Parameters

DATA = *identifiers*	Data to write to the spreadsheet
PROTECT = *string tokens*	Whether to protect each data column by making it read-only (yes, no); default no
FACCOLOURS = *variates, texts* or *pointers*	Specifies background colours for factor columns
FOREGROUND = *variate, text, scalar* or *pointer*	Specifies foreground colours for columns
BACKGROUND = *variate, text, scalar* or *pointer*	Specifies background colours for columns
HIDDEN = *string tokens*	Whether to hide each DATA column (yes, no); default no

FSSPM directive

Forms the values of SSPM structures.

Options

PRINT = *string tokens*	Printed output required (correlations, wmeans, SSPM); default * i.e. no printing
WEIGHTS = *variate* or *symmetric matrix*	

	Variate of weights for weighted SSP, or symmetric matrix of weights (one row and column for each unit of data); default * i.e. all units with weight one
SEQUENTIAL = *scalar*	Used for sequential formation of SSPMs; a positive value indicates that formation is not yet complete (see READ directive); default * i.e. not sequential

Parameter

SSPMs	Structures to be formed

FSTRING procedure

Forms a single string from a list of strings in a text (R.W. Payne).

No options

Parameters

TEXT = *texts*	Texts containing the lists of strings to put into single strings
STRING = *texts*	Text to store the strings in each TEXT
SEPARATOR = *texts*	Characters to separate all except last two strings of each TEXT; default ', '
LASTSEPARATOR = *texts*	Characters to separate last two strings of each TEXT; default SEPARATOR
PREFIX = *texts*	Characters to insert at the start of each STRING; default '' (i.e. none)
END = *texts*	Characters to put at the end of each STRING; default '' (i.e. none)

FTEXT procedure

Forms a text structure from any Genstat data structure (A. Keen & J.T.N.M. Thissen).

Option

MISSING = *text*	What to print for missing value; default '*'

Parameters

STRUCTURE = *identifiers*	Structure (scalar, variate, factor, text, table, matrix, symmetricmatrix, diagonalmatrix, pointer) from which the text structure is to be formed
TEXT = *texts*	Saves the text structure
DECIMALS = *scalars*	Number of decimals to use when forming the text structure; default * uses the number required to provide 4 significant figures, but unnecessary trailing zeros are ignored
FREPRESENTATION = *string tokens*	How factor values are to be represented in the text structure (labels, levels, ordinals); default is to use labels if available and levels otherwise

FTSM directive

Forms preliminary estimates of parameters in time-series models.

Option

PRINT = *string tokens*	What to print (models); default *

Parameters

TSM = *TSMs*	Models whose parameters are to be estimated
CORRELATIONS = *variates*	Auto- or cross-correlations on which to base estimates for each model
BOXCOXTRANSFORM = *scalars*	Box-Cox transformation parameter
CONSTANTTERM = *scalars*	Constant term
VARIANCE = *scalars*	Variance of ARIMA model, or ratio of input variance to output variance for transfer model

FUNIQUEVALUES procedure

Redefines a variate or text so that its values are unique (R.W. Payne).

Options

INCREMENT = *scalar*	Increment to use to modify duplicated numbers; default * i.e. a suitable (small) value is determined automatically
ADDTO = *string token*	Whether to add the increment to the value or the absolute value of duplicated numbers (value, absolutevalue); default abso

Parameters

OLDVECTOR = *variates* or *texts*	Vectors whose values are to be made unique
NEWVECTOR = *variates* or *texts*	New vectors with unique values; if unset, the values of the corresponding OLDVECTOR are replaced
CHANGED = *scalars*	Indicates whether the values have changed

FVARIOGRAM directive

Forms experimental variograms.

Options

PRINT = *string token*	Controls printed output (statistics); default stat
Y = *variate*	Y positions (needed only for 2-dimensional irregular data)
X = *variate*	X positions or interval (not needed for 2-dimensional regular data i.e. when DATA is a matrix)
YMAX = *scalar*	Maximum lag in the y direction (2-dimensional regular data only)
XMAX = *scalar*	Maximum lag in the x direction
METHOD = *string token*	How to estimate the variogram (moments, cressiehawkins, dowd, genton); default mome
STEPLENGTH = *scalar* or *variate*	Length(s) of the steps in which lag is incremented
DIRECTIONS = *scalar* or *variate*	Directions (degrees) along which to form the variogram (relevant only for 2-dimensional irregular data)
SEGMENTS = *scalar* or *variate*	Angles subtended by the segments (degrees) over which averaging is to be done (relevant only for 2-dimensional irregular data)

Parameters

DATA = *variates* or *matrices*	Measurements as a variate or, for data on a regular grid, as a matrix
VARIOGRAMS = *variates* or *matrices*	Structure to store the sample variogram
COUNTS = *variates* or *matrices*	Numbers of comparisons involved in the calculation of each variogram
DISTANCES = *variates* or *matrices*	Mean lag distances at each step
LAGPOINTS = *pointer*	Saves lag classes, indexes to observations and directions to plot in an h-scattergram

FVCOVARIANCE procedure

Forms the variance-covariance matrix for a list of variates (W. van den Berg).

Options

PRINT = *string tokens*	Printed output (df, vcovariance); default df, vcov
WEIGHTS = *variate*	Provides weights for the units of the variates; default * assumes that they all have weight one
VCOVARIANCE = *symmetric matrix*	Saves the variance-covariance matrix
DF = *scalar*	Saves the number of degrees of freedom of the (co)variances

Parameter

DATA = *variates*	Variates for which the matrix is to be calculated

FVSTRING procedure

Forms a string listing the identifiers of a set of data structures (R.W. Payne).

Options

STRING = *text*	Saves the string
POINTERNAME = *text*	If all the structures are belong to the same pointer, this saves its name
ELEMENTNAMES = *text* or *variate*	Saves the elements of the pointer, in a text if they have labels, otherwise in a variate

Parameter

DATA = *identifiers*	Data structures to be used to form the string

†FWITHINTERMS procedure

Forms factors to define terms representing the effects of one factor within another factor (R.W. Payne).

Options

LEVNULL = *scalar*	Numerical value to represent the null level assigned to units not involved in the comparison of the levels of one of the factors within a particular level of the other factor; default 0
LABNULL = *text*	String to label the null level; default '-'

Parameters

F1 = *factors*	First factor
F2 = *factors*	Second factor
F1WITHINF2 = *pointers*	Pointer containing a factor for each level of the second factor, used to estimate the effects of the first factor within that level
F2WITHINF1 = *pointers*	Pointer containing a factor for each level of the first factor, used to estimate the effects of the second factor within that level

FZERO procedure

Gives the F function expectation under complete spatial randomness (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	What to print (summary); default summ

Parameters

DENSITY = *scalars*	Densities to use i.e. numbers of points per unit area; no default – this parameter must be set
S = *variates*	Vectors of distances to use; no default – this parameter must be set
FVALUES = *variates*	Variates to receive the expected values of the F nearest-neighbour distribution function under CSR

F2DRESIDUALVARIOGRAM procedure

Calculates and plots a 2-dimensional variogram from a 2-dimensional array of residuals (S.J. Welham).

Options

PLOT = *string token*	What to plot (surface); default surf
ROWS = *factor*	Factor defining the rows of the grid
COLUMNS = *factor*	Factor defining the columns of the grid
REPLICATES = *factor*	Factor defining the replicate grids (if any)
RMAX = *scalar*	Maximum lag to include in variogram in row direction (default determined by procedure)
CMAX = *scalar*	Maximum lag to include in variogram in column direction (default determined by procedure)
RSCALE = *scalar*	Actual distance represented by 1 unit in row direction (default 1)
CSCALE = *scalar*	Actual distance represented by 1 unit in column direction

(default 1)

MINREP = *scalar* — Minimum replication required for position to be included in variogram (default 30)

TITLE = *text* — Title for surface/graph; default * i.e. none

WINDOW = *scalar* — Graphics window to be used for plotting; default 1

SCREEN = *string token* — Whether to keep or clear screen before plotting variogram (clear, keep); default clear

METHOD = *text* — Whether to use Fortran DLL or Genstat code to calculate variogram (dll, genstat); default dll

SCALEPLOT = *string token* — Whether to scale variogram to 0-1 (i.e. unit) scale for plotting (unit, none); default unit

Parameters

RESIDUALS = *variates* — Variate of residuals to form variogram

VARIOGRAM = *matrices* — Calculated variogram (trimmed)

FULLVARIOGRAM = *matrices* — Calculated variogram (all values)

COUNTS = *matrices* — Number of comparisons contributing to each variogram position

COMPONENTS = *pointers* — Components used to calculate variogram (only available when METHOD=genstat)

GALOIS procedure

Forms addition and multiplication tables for a Galois finite field (I. Wakeling & R.W. Payne).

Option

METHOD = *string token* — Whether to choose the primitive polynomial to generate the Galois field with the least number of higher terms or whether to make a random choice (minimal, random); default rand

Parameters

ORDER = *scalars* — Order of the required Galois field

ADDITION = *symmetric matrices* — Saves the addition table of the field

MULTIPLICATION = *symmetric matrices* — Saves the field's multiplication table

PRIMITIVE = *variates* — Saves the primitive irreducible polynomial

ERROR = *scalars* — Returns 0 or 1 according to whether or not the tables have been formed successfully

GBGRIDCONVERSION procedure

Converts GB grid references to or from latitudes and longitudes or to or from UTM coordinates (R.W. Payne).

Options

INPUTSOURCE = *string token* — Which of the coordinate systems if acting as input for conversion to either of the other two systems (gridreference, geographical, utm); default geog

GRIDREFERENCES = *texts* — Grid references

LATITUDES = *scalar*s or *variates* — Latitudes

LONGITUDES = *scalars* or *variates* — Longitudes

EASTINGS = *scalars* or *variates* — UTM easting references

NORTHINGS = *scalars* or *variates* — UTM northing references

GRIDACCURACY = *string token* — The accuracy for saving grid references (kilometres, hectometres, dekametres, metres); default hect

No parameters

GEE procedure

Fits models to longitudinal data by generalized estimating equations (D.M. Smith & M.G.Kenward).

Options

PRINT = *string token* — What to display (estimates, correlations, scalefactor, wald, monitoring); default esti, corr,

	`scal`
`DISTRIBUTION` = *string token*	Distribution of response (`normal`, `Poisson`, `binomial`, `gamma`, `inversenormal`, `negativebinomial`); default `*`
`LINK` = *string token*	Link function (`identity`, `logarithm`, `logit`, `reciprocal`, `power`, `squareroot`, `probit`, `complementaryloglog`, `logratio`); default `*`
`EXPONENT` = *scalar*	Exponent for power link; default −2
`TERMS` = *formula*	Explanatory variates, factors etc
`CONSTANT` = *string token*	How to treat constant (`estimate`, `omit`); default `esti`
`FACTORIAL` = *scalar*	Limit for expansion of model terms; default 3
`AGGREGATION` = *scalar*	Fixed parameter for negative binomial distribution (parameter k as in variance function var = mean + mean^2/k); default 1
`KLOGRATIO` = *scalar*	Parameter for logratio link, in form log(mean / (mean + k)); default as set in `AGGREGATION` option
`QUADESTIMATION` = *string token*	Whether to use quadratic estimation (`used`, `notused`); default `used`
`SCALEFACTOR` = *string token*	How to calculate the scale factor (`fixed`, `constant`, `varytime`); default varies with distribution, `fixed` for Poisson and binomial, `constant` for rest
`SFVALUE` = *scalar*	Value for scale factor when `SCALEFACTOR=fixed`; default 1.0 for Poisson and binomial, missing for rest
`CRTYPE` = *string token*	Form of correlation matrix (`independence`, `unstructured`, `exchangeable`, `autoregressive`, `dependence`, `antedependence`); default `*`
`ORDER` = *scalar*	Order in dependence and ante-dependence form of correlation matrix; default 1
`TIMEDEPENDENT` = *string token*	Whether correlation in dependence model changes with time (`no`, `yes`); default `no`

Parameters

`Y` = *variates*	Response variate for each analysis
`NBINOMIAL` = *variates* or *scalars*	Denominator in binomial
`FITTEDVALUES` = *variates*	To store fitted values
`RESIDUALS` = *variates*	To store residuals
`SUBJECT` = *factors*	Identifier of subjects
`OUTCOME` = *factors*	Identifier of outcomes
`COUNT` = *variates*	Variate of counts of no. outcomes
`TIME` = *factors*	Times of repeated measures variate
`WEIGHT` = *variates*	Weight variate
`OFFSET` = *variates*	Offset variate
`SAVE` = *pointers*	Structure to save output variables

`GENERATE` directive

Generates factor values for designed experiments: with no options set, factor values are generated in standard order; the options allow treatment factors to be generated using the design-key method, or pseudo-factors to be generated to describe the confounding in a partially balanced experimental design.

Options

`TREATMENTS` = *formula*	Model term for which pseudo-factors are to be generated; default `*`
`REPLICATES` = *formula*	Factors defining replicates of the design; default `*`
`BLOCKS` = *formula*	Block formula (for design-key generation) or term (for generation of pseudo-factors); default `*`
`KEY` = *matrix*	Key matrix (number of factors in the parameter list by number of factors in the `BLOCKS` formula) to generate the factors by the design key method; default `*`
`BASEVECTOR` = *variate*	Base vector for design key generation; default `*`

Parameter

factors	Factors whose values are to be generated

GENPROCRUSTES procedure

Performs a generalized Procrustes analysis (G.M. Arnold & R.W. Payne).

Options

PRINT = *string tokens*	Printed output required (analysis, centroid, column, individual, monitoring); default anal, cent
SCALING = *string token*	Type of scaling to use (none, isotropic, separate); default none
METHOD = *string token*	Method to be used (Gower, TenBerge); default Gowe
NROOTS = *scalar*	Number of roots (i.e. dimensions) to print for the output configurations, consensus and rotation matrices, and number of dimensions to save with the XOUTPUT, CONSENSUS and ROTATIONS paramaters if their matrices have alread not been defined; default is to print and save all the dimensions
PLOT = *string tokens*	Controls which graphs to display (consensus, individuals, projections); default * i.e. none
NDROOTS = *scalar*	Number of dimensions to display in the consensus and individuals plots; default 3
TOLERANCE = *scalar*	The algorithm is assumed to have converged when (last residual sum of squares) − (current residual sum of squares) < TOLERANCE × (number of configurations); default 0.00001
MAXCYCLE = *scalar*	Limit on number of iterations; default 50

Parameters

XINPUT = *pointers*	Each pointer points to a set of matrices holding the original input configurations
XOUTPUT = *pointers*	Each pointer points to a set of matrices to store a set of final (output) configurations
CONSENSUS = *matrices*	Stores the final consensus configuration from each analysis
ROTATIONS = *pointers*	Each pointer points to a set of matrices to store the rotations required to transform each set of XINPUT configurations to their final (scaled) XOUTPUT configurations
RESIDUALS = *pointers*	Each pointer points to a set of matrices to store the distances of a set of scaled XINPUT configurations from its consensus
RSS = *scalars*	Stores the residual sum of squares from each analysis
ROOTS = *diagonal matrices*	Stores the latent roots from referring the centroid configuration to its principal axis form (consensus) for each analysis
WSS = *scalars*	Stores the initial within-configuration sum of squares from each analysis
SCALINGFACTOR = *variates*	Stores the isotropic scaling factors for configurations from each analysis
PROJECTIONS = *pointers*	Each pointer points to a set of matrices to store a set of projection matrices

GESTABILITY procedure

Calculates stability coefficients for genotype-by-environment data (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (means, stability, sortedstability, quantiles); default stab, quan
METHOD = *string tokens*	Methods to use to calculate stability (superiority, static, wricke, ranks); default supe
BESTMETHOD = *string token*	How to define the best genotype (minimum, maximum); default maxi
PLOT = *string tokens*	What graphs to plot (stability); default * i.e. none
NBEST = *string tokens*	Number of best genotypes to print in tables of sorted stability

coefficients; default * i.e. print all of them

DIRECTION = *string token* — Direction to sort tables of sorted stability coefficients (ascending, descending); default asce

PERCENTQUANTILES = *scalar* or *variate* — Percentage points for which quantiles are required; default !(50,5,1,0.1)

NTIMES = *scalar* — Number of permutations to make; default 999

BLOCKSTRUCTURE = *formula* — Model formula defining any blocking to consider during the permutation test; default none

EXCLUDE = *factors* — Factors in the block formula whose levels are not to be randomized in the permutation test

Parameters

Y = *variates* — Yields (or other measurements) made on the genotypes in the environments

GENOTYPES = *factors* — Genotype corresponding to each yield

ENVIRONMENTS = *factors* — Environment where each yield was recorded

SEED = *scalar* — Seed for the random number generator used to make the permutations; default 0 continues from the previous generation or (if none) initializes the seed automatically

STABILITY = *tables* or *pointers* — Saves stability coefficients

QUANTILES = *tables* or *pointers* — Saves quantiles of the stability coefficients

TITLE = *texts* — Overall title for the graphs; default * i.e. none

GET directive

Accesses details of the "environment" of a Genstat job.

Options

†ENVIRONMENT = *pointer* — Pointer given unit labels 'inprint', 'outprint', 'diagnostic', 'errors', 'pause', 'prompt', 'newline', 'case', 'run', 'wordlength', 'captions', 'typeset', 'cmethod', 'dataspace', 'algorithms', 'actionafterfault', 'unsetdummy', 'language', 'year2digitbreak' and 'timewithseconds' to save the current settings of those options of SET; default *

SPECIAL = *pointer* — Pointer given unit labels 'units', 'blockstructure', 'treatmentstructure', 'covariate', 'asave', 'dsave', 'rsave', 'tsave', 'vsave' and 'vcomponents', used to save the current settings of those options of SET; default *

LAST = *text* — To save the last input statement; default *

FAULT = *text* — To save the last fault code; default *

FIELDWIDTH = *scalar* — Saves the fieldwidth currently defined as the default minimum for PRINT and other output commands

SIGNIFICANTFIGURES = *scalar* — Saves the minimum number of significant figures currently to be supplied in the default formats determined by PRINT and other output commands

SEEDS = *pointer* — Saves a pointer to variates defining the seeds currently used as defaults by random-number functions, the RANDOMIZE directive, and internally by various other directives

EPS = *scalar* — To obtain the value of the smallest x (on this computer) such that $1+x > 1$; default *

NJOB = *scalar* — Number of the current job within the program; default *

VERSION = *pointer* — Information about the version of Genstat that is being used; default *

PID = *scalar* — Gets an integer value unique in the current job to use, for example, in names of temporary files

WORKINGDIRECTORY = *text* — Saves the name of the current working directory

No parameters

GETATTRIBUTE directive

Accesses attributes of structures.

Option

ATTRIBUTE = *string tokens*	Which attributes to access (nvalues, nlevels, nrows, ncolumns, type {type number}, levels, labels {of a factor or pointer}, nmv, present, identifier, refnumber {structure number}, extra, decimals, characters, minimum, maximum, restriction, mode {integer code 1 - 5 denoting type of values: double real, real, integer, character and word}, maxline {of a text or factor}, rows, columns, classification, margins {of a table}, associatedidentifier {of a table}, unknown {cell of a table}, suffixes {of a pointer}, owner, terms {of an SSPM}, groups {of an SSPM}, weights {of an SSPM}, SSPMauxiliary, SSPrst, tsmmodel, rstat {of an RSAVE}, stype {type as a character string}, referencelevel {of a factor}, drepresentation, unitlabels {of a vector}, iprint, datavariate {of a table}, summarytype {of a table}, percentquantile {of a table of quantiles}, %margin {of a table of percentages}, coding {of a text}); default * i.e. none

Parameters

STRUCTURE = *identifiers*	Structures whose attributes are to be accessed
SAVE = *pointers*	Pointer to store copies of the attributes of each structure; these are labelled by the ATTRIBUTE strings

GETLOCATIONS directive

Finds locations of an identifier within a pointer, or a string within a factor or text, or a number within any numerical data structure.

Options

CASE = *string token*	Whether to treat the case of letters (small or capital) as significant when searching for a string (significant, ignored); default sign
TOLERANCE = *scalar*	Tolerance for comparing numbers
SUBSTITUTE = *string token*	Whether to substitute dummies within pointers in DATA or FIND (yes, no); default no

Parameters

DATA = *identifiers*	Variates, scalars, matrices, tables, factors, texts or pointers to be searched
FIND = *scalars*, *texts* or *pointers*	Numbers, strings or identifiers to be located in DATA
NLOCATIONS = *scalars*	Saves the number of times that FIND occurs in DATA
LOCATIONS = *variates* or *pointers*	Saves the locations where FIND occurs as one of the values in DATA, in a variate if DATA is a one-dimensional data structure like a variate or text, or in a pointer containing a variate for each dimension if DATA is a multi-dimensional data structure like a matrix or table
CLASSIFICATION = *pointers*	Saves the classifying factors for a DATA table, in the same order as the corresponding variates in the LOCATIONS and LEVELS pointers
LEVELS = *pointers*	Saves the levels of the classifying factors where FIND occurs as one of the values of a DATA table, the information is saved in a pointer containing a variate for each factor

GETNAME procedure

Forms the name of a structure according to its IPRINT attribute (A.R.G. McLachlan).

No options

Parameters

STRUCTURE = *identifiers*	Structures whose names are to be obtained
NAME = *texts*	Saves the names of the structures
IDENTIFIER = *texts*	Saves the identifiers of the structures
EXTRA = *texts*	Saves the extra texts of the structures
IPRINT = *texts*	Saves (or forms) IPRINT attributes

GETRGB procedure

Gets the RGB values of the standard graphics colours (R.W. Payne).

No options

Parameters

COLOUR = *scalars* or *variates*	Colour numbers
RGB = *scalars* or *variates*	RGB values
NAME = *texts*	Names of nearest colours

GETTEMPFOLDER procedure

Gets gets the location of the folder used by Genstat for temporary files (R.W. Payne).

Option

PRINT = *string token*	Controls printed output (tempfolder); default temp

Parameter

TEMPFOLDER = text	Saves the name of the temporary folder

GGEBIPLOT procedure

Plots displays to assess genotype + genotype-by-environment variation (A.I. Glaser).

Options

PRINT = *string tokens*	What to print (variation); default * i.e. nothing
DIMENSIONS = *scalars*	Which dimensions to display; default 1,2
PLOT = *string token*	Type of plot (scatter, ranking, compare, joint, centred); default scat
METHOD = *string token*	Whether the names in LEV1 (and LEV2) are from the ENVIRONMENTS or GENOTYPES factor (environments, genotypes); default envi
SCPLOT = *string token*	Features to add to a scatter plot (hull, sector, megaenvironment, vector, linear); default * i.e. none
SCALING = *string tokens*	What scaling to use (genotype, environment, symmetric); default envi
NORMALIZE = *string token*	Whether to scale the data using the within-environment standard deviation (yes, no); default no
CULL = *variate* or *text*	Specifies environments at which to examine the performance of the genotypes in order to decide which genotypes to cull
QUANTILE = *scalar*	Proportion at which to calculate quantile for CULL; default 0.5.
DIVISIONS = *scalar*	Number of parallel lines or concentric circles to use when ranking genotypes or environments; default 10
RANKINGLINES = *string token*	Whether the ranking lines drawn with PLOT setting ranking or joint are perpendicular to the biplot axis or projected onto the axis (perpendicular, projection); default perp
GENREVERSE = *string token*	Whether to reverse the order of the genotype scores (yes, no); default no
ENVREVERSE = *string token*	Whether to reverse the order of the environment scores (yes, no); default no
WINDOW = *scalar*	Which graphical window to use; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2

Parameters

DATA = *variates* or *tables*	Provides the data to be analysed
GENOTYPES = *factors*	Specifies the genotypes
ENVIRONMENTS = *factors*	Specifies the environments
LEV1 = *texts* or *scalars*	First environment (or genotype) to use with PLOT settings centred, compare, joint or ranking, or with scatter when SCPLOT=linear
LEV2 = *texts* or *scalars*	Second environment (or genotype) to use with PLOT settings centred, compare or joint
LABGENOTYPES = *texts*	Labels for genotypes
LABENVIRONMENTS = *texts*	Labels for environments
TITLE = *texts*	Titles for the plots; if this is unset, an appropriate title is formed auomatically
MEGAGROUPS = *variates* or *texts*	Specifies or saves the groupings to use for the plot produced by SCPLOT=megaenvironment

GHAT procedure

Calculates an estimate of the G nearest-neighbour distribution function (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	What to print (summary); default summ

Parameters

Y = *variates*	Vertical coordinates of each spatial point pattern; no default – this parameter must be set
X = *variates*	Horizontal coordinates of each spatial point pattern; no default – this parameter must be set
S = *variates*	Vectors of distances to use with each pattern; no default – this parameter must be set
GVALUES = *variates*	Variates to receive the estimated G nearest-neighbour distribution functions
NNDISTANCES = *variates*	Variates to receive the nearest-neighbour distances
NNUNITS = *variates*	Variates to receive the unit numbers of the nearest neighbours

GINVERSE procedure

Calculates the generalized inverse of a matrix (S.K. Haywood).

Options

PRINT = *string token*	Printed output from the procedure (inverse); default *, i.e. no printing
METHOD = *string token*	Method to be used to invert symmetric matrices (svd, lrv); default lrv
TOLERANCE = *scalar*	How close a number must be be to zero before it is recognised as zero; default 1.0^{-6}

Parameters

INMATRIX = *matrices*	The matrix whose inverse is to be calculated
INVERSE = *matrices*	Matrix to save the generalized inverse

†GLDISPLAY procedure

Displays further output from a GLMM analysis (R.W. Payne).

Options

PRINT = *string token*	What output to display (model, components, effects, fittedvalues, means, backmeans, vcovariance, waldtests, missingvalues, covariancemodels, deviance); default *
PTERMS = *formula*	Formula specifying fixed terms for which means or back-transformed means are to be printed; default * prints all the fixed model terms

PSE = *string token* — Standard errors to print with tables of means (se, sesummary, sed, sedsummary, vcovariance, differences, estimates, alldifferences, allestimates); default seds

OFFSET = *scalar* — Offset value to use when calculating predicted means; default 0

RMETHOD = *string token* — Which random terms to use when calculating RESIDUALS (final, all); default fina

CFORMAT = *string token* — Whether printed output for covariance models gives the variance matrices or the parameters (variancematrices, parameters); default vari

FMETHOD = *string token* — Controls whether and how to calculate F-statistics for fixed terms (automatic, none, algebraic, numerical); default auto

GLSAVE = *pointer* — Save structure from the GLMM analysis

No parameters

†GLKEEP procedure

Saves results from a GLMM analysis (R.W. Payne).

FACTORIAL = *scalar* — Limit on number of factors in the model terms generated from the TERMS parameter; default 3

RESIDUALS = *variate* — Residuals from the analysis

FITTEDVALUES = *variate* — Fitted values from the analysis

DISPERSION = *scalar* — Dispersion component

VCOVARIANCE = *symmetric matrix* — Variance-covariance matrix for the estimates of the variance components

VESTIMATES = *variate* — Saves a vector of all parameters in the variance model

VARESTIMATES = *symmetric matrix* — Variance-covariance matrix for the parameters in the variance model (as saved by VESTIMATES)

VLABELS = *text* — Vector of text labels for the VESTIMATES and VARESTIMATES structures

MVESTIMATES = *variate* — Estimates of missing values

MVSE = *variate* — Standard errors of missing-value estimates

MVUNITS = *variate* — Unit numbers of missing values

DEVIANCE = *scalar* — Saves the deviance

MODEL = *pointer* — Information defining the mode;

RMETHOD = *string token* — Which random terms to use when calculating RESIDUALS (final, all); default all

DFFIXED = *scalar* — Number of degrees of freedom in the fixed model

DFRANDOM = *scalar* — Number of degrees of freedom in the random model

FMETHOD = *string token* — Controls how to calculate F-statistics for fixed terms (automatic, none, algebraic, numerical); default auto

WMETHOD = *string token* — Controls which Wald statistics are saved (add, drop); default drop

OFFSET = *scalar* — Offset value to use when calculating predicted means; default 0

ITERATIVEWEIGHTS = *variate* — Saves the iterative weights from the generalized linear model fitting

LINEARPREDICTOR = *variate* — Linear predictor from a generalized linear model

YADJUSTED = *variate* — Adjusted response variate

ZADJUSTED = *variate* — Adjusted dependent variate on the linear predictor scale

LPRESIDUALS = *variate* — Residuals from the fit on the linear predictor scale

SELPRESIDUALS = *variate* — Standard errors for the residuals from the fit on the linear predictor scale

EXIT = *scalar* — Exit status of the fit (0 if successful)

GLSAVE = *pointer* — Save structure from the GLMM analysis

Parameters

TERMS = *formula* Model terms for which information is required
COMPONENTS = *scalar* or *pointer to scalars*
Estimated variance components
MEANS = *table* or *pointer* to *tables* Predicted means for each term
BACKMEANS = *table* or *pointer* to *tables* Back-transformed means
SEDMEANS = *symmetric matrix* or *pointer* to *symmetric matrices*
Standard errors of differences between means
VARMEANS = *symmetric matrix* or *pointer* to *symmetric matrices*
Variance-covariance matrix for the means
EFFECTS = *table* or *pointer* to *tables* Effects for each term
SEDEFFECTS = *symmetric matrix* or *pointer* to *symmetric matrices*
Standard errors of differences between effects
VAREFFECTS = *symmetric matrix* or *pointer* to *symmetric matrices*
Variance-covariance matrix for the effects
CADJUSTMENT = *scalar* or *pointer to scalars*
For a term involving covariates, saves the adjustment made to its values during the analysis
WALD = *scalar* or *pointer to scalars* Wald statistic (fixed terms only)
FSTATISTIC = *scalar* or *pointer to scalars*
F statistics (fixed terms only)
NDF = *scalar* or *pointer to scalars* Numerator d.f. (fixed terms only)
DDF = *scalar* or *pointer to scalars* Denominator d.f. (fixed terms only)

GLM procedure

Analyses non-standard generalized linear models (P.W. Lane).

Options

PRINT = *string tokens* What to display (deviance, estimates, correlations, monitoring); default devi, esti
DISTRIBUTION = *string token* Distribution of response (Normal, Poisson, binomial, gamma, inversenormal); default * indicates calculations supplied for non-standard distribution via procedure GLMDISTRIBUTION (see the details of the procedures called by GLM)
LINK = *string token* Link function (identity, logarithm, logit, reciprocal, power, squareroot, probit, complementaryloglog); default * indicates calculations supplied for non-standard link via procedure GLMLINK (see Method)
EXPONENT = *scalar* Exponent for power link; default −2
TERMS = *list* or *formula* Explanatory variates, factors, and interactions specified as for the standard regression directives; default null model
CONSTANT = *string token* Whether to include constant term (estimate, omit); default esti
INITIALLINEAR = *variate* Initial guess at linear predictor, if specifying own link function and not defining procedure GLMINITIAL

Parameters

Y = *variates* Response variate; this parameter must be set
NBINOMIAL = *variates* Totals for use when DISTRIBUTION=binomial; must then be set
FITTEDVALUES = *variates* To store correct fitted values

GLMM procedure

Fits a generalized linear mixed model (S.J. Welham).

Options

†PRINT = *string token* What output to display (model, components, effects, fittedvalues, means, backmeans, monitoring, vcovariance, waldtests, missingvalues,

	covariancemodels, deviance); default mode, comp, effe, mean, back, moni, vcov, cova
DISTRIBUTION = *string token*	Error distribution (binomial, poisson, normal, gamma, negativebinomial); default bino
LINK = *string token*	Link function (identity, logarithm, logit, reciprocal, probit, complementaryloglog, logratio); default * gives the canonical link
DISPERSION = *scalar*	Value at which to fix the residual variance, if missing the variance is estimated; default 1 for binomial, Poisson and negative binomial distributions, a missing value otherwise
RANDOM = *formula*	Random model *excluding* bottom stratum; this must be set
FIXED = *formula*	Fixed model; default *
ABSORB = *factor*	Absorbing factor to be used at the REML step of the iterations
CONSTANT = *string token*	Whether to estimate or omit constant term in fixed model (omit, estimate); default esti
FACTORIAL = *scalar*	Limit on number of factors/covariates in a model term; default 3
PTERMS = *formula*	Formula specifying fixed terms for which means or back-transformed means are to be printed; default * prints all the fixed model terms
†PSE = *string token*	Standard errors to print with tables of means (se, sesummary, sed, sedsummary, vcovariance, differences, estimates, alldifferences, allestimates); default seds
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default * i.e. omit units with missing values in either explanatory factors or variates or y-variates
MAXCYCLE = *scalar*	Maximum number of iterations of the GLMM algorithm; default 20
TOLERANCE = *scalar*	Convergence criterion for iterative procedure; default 0.0001
†FMETHODGLMM = *string token*	Specifies fitting method (all, fixed): all indicates the method of Schall (1991); fixed indicates the marginal method of Breslow & Clayton (1993) ; default all
OFFSET = *variate*	Variate holding values to be used as an offset on the linear predictor scale; default *
CADJUST = *string token*	What adjustment to make to covariates for the REML analysis (mean, none); default mean
AGGREGATION = *scalar*	Fixed parameter for negative binomial distribution (parameter k as in variance function var = mean + mean^2/k); default 1
KLOGRATIO = *scalar*	Parameter k for logratio link, in form log(mean / (mean + k)); default as set in AGGREGATION option
OWNDIST = *text*	For non-standard distributions only: text specifying the variance function to be used with dummy variable DUM, e.g. OWNDIST='DUM'
OWNLINK = *text*	For non-standard link functions only: text specifying 3 functions using dummy variable DUM – the link function, its inverse and its derivative, e.g. OWNLINK = !T('log(DUM)','exp(DUM)','1/DUM')
CDEFINITIONS = *text*	Statements to execute to define correlation models; default * i.e. none
CVECTORS = *pointer*	Data structures involved in the correlation models
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm; default 1
VCONSTRAINTS = *string token*	Whether to constrain variance components to be positive (none, positive); default posi

†VMETHOD = *string token* — Indicates whether to use the standard Fisher-scoring algorithm or the new AI algorithm with sparse matrix methods (Fisher, AI); default AI

†VMAXCYCLE = *scalar* — Limit on the number of iterations; default 30

Parameters

Y = *variates* — Dependent variates

NBINOMIAL = *scalars* or *variates* — Number of binomial trials for each unit (must be set if DISTRIBUTION=binomial)

FITTEDVALUES = *variates* — Variates to save fitted values

COMPONENTS = *variates* — Variate to save estimated variance components

VCOVARIANCE = *symmetric matrices* — Variance-covariance matrix for the variance components

MEANS = *pointers* — Pointer to save tables of means for each Y variate

VARMEANS = *pointers* — Pointer to save covariance matrices of tables of means for each Y variate

BACKMEANS = *pointers* — Pointer to save tables of back-transformed means for each Y variate

ITERATIVEWEIGHTS = *variates* — Saves the iterative weights from the generalized linear model fitting

INITIALFITTEDVALUES = *variates* — Defines initial values for the fitted values; if unset, these are formed automatically

†EXIT = *scalar* — Exit status for the fit of the GLMM (0 if successful)

SAVE = *REML save structures* — Saves details of the REML analysis used to fit the model

†GLSAVE = *pointer* — Saves details of the GLMM analysis

†GLPERMTEST procedure

Does random permutation tests for generalized linear mixed models (R.W. Payne).

Options

PRINT = *string tokens* — Controls printed output (prwald, criticalwald, ownstatistics, monitoring); default prwa, crit

NTIMES = *scalar* — Number of permutations to make; default 99

NRETRIES = *scalar* — Maximum number of extra samples to take when some analyses fail to converge; default NTIMES

BLOCKSTRUCTURE = *formula* — Model formula defining any blocking to consider during the randomization; default none

EXCLUDE = *factors* — Factors in the block formula whose levels are not to be randomized

SEED = *scalar* — Seed for the random number generator used to make the permutations; default 0 continues from the previous generation or (if none) initializes the seed automatically

BINMETHOD = *string token* — How to permute binomial data (individuals, units; default indi

WMETHOD = *string token* — Controls which Wald statistics are used (add, drop); default add

OWNMETHOD = *string token* — Type of test required for own statistics (twosided, greaterthan, lessthan); default twos

CIPROBABILITY = *scalar* — Probability level for the confidence interval for own statistics; default 0.95

Parameters

GLSAVE = *pointers* — Save structure of the original analysis from GLMM; default * uses the save structure from the most recent GLMM analysis

WALD = *pointers* — Saves a pointer with a variate for each of the fixed terms containing the Wald statistics from the permuted data sets

PRWALD = *pointers* — Saves a pointer with a scalar for each of the fixed terms, containing the test probability obtained from the position of its Wald statistic within those from the permuted data sets

CRITICALWALD = *pointers* — Saves a pointer with variates for the 5%, 1% and 0.1% significance levels containing the corresponding critical values

	for the fixed terms, obtained from the quantiles of the Wald statistics from the permuted data sets
NNOTCONVERGED = *scalars*	Saves the number of permuted data sets whose analyses failed to converge
OWNDATA = *pointers*	Data required to calculate own statistics
OWNOBSERVEDVALUES = *variates*	Saves observed values of the own statistics
OWNPROBABILITIES = *variates*	Saves bootstrap probabilities for the own statistics
OWNESTIMATES = *variates*	Saves bootstrap estimates for the own statistics
OWNSES = *variates*	Saves bootstrap standard errors for the own statistics
OWNLOWERCIS = *variates*	Saves bootstrap lower values of the confidence intervals for the own statistics
OWNUPPERCIS = *variates*	Saves bootstrap upper values of the confidence intervals for the own statistics
OWNSTATISTICS = *pointers*	Saves the own statistics obtained from the permuted data sets, in a pointer with a variate for each statistic

†GLPLOT procedure

Plots residuals from a GLMM analysis (R.W. Payne).

Options

RMETHOD = *string token*	Which random terms to use when calculating the residuals (final, all); default all
BACKTRANSFORM = *string token*	Whether to plot residuals on the natural scale (calculated using back-transformed fitted values) or standardized residuals on the linear-predictor scale (link, none); default none
INDEX = *variate* or *factor*	X-variable for an index plot; default !(1,2...)
OFFSET = *scalar*	Value of offset to use when calculating the residuals; default 0
GRAPHICS = *string token*	What type of graphics to use (lineprinter, highresolution); default high
TITLE = *text*	Overall title for the plots; the default is to form a title displaying the identifier of the y-variate and the type of residual
GLSAVE = *pointer*	Save structure from the GLMM analysis; default * uses the GLSAVE structure from the most recent GLMM analysis

Parameters

METHOD = *string tokens*	Type of residual plot (fittedvalues, normal, halfnormal, histogram, absresidual, index); default fitt, norm, half, hist
PEN = *scalars*, *variates* or *factors*	Pen(s) to use for each plot

†GLPREDICT procedure

Forms predictions from a GLMM analysis (R.W. Payne).

Options

PRINT = *string tokens*	What to print (description, predictions, backpredictions, se, sesummary, sed, sedsummary, vcovariance); default desc, pred, back, seds
MODEL = *formula*	Indicates which model terms (fixed and/or random) are to be used in forming the predictions; default * includes all the fixed terms and relevant random terms
OMITTERMS = *formula*	Specifies terms to be excluded from the MODEL; default * i.e. none
FACTORIAL = *scalar*	Limit on the number of factors or variates in each term in the models specified by MODEL or OMITTERMS; default 3
PRESENTCOMBINATIONS = *factors*	Lists factors for which averages should be taken across combinations that are present
WEIGHTS = *tables*	One-way tables of weights classified by factors in the model; default *

OFFSET = *scalar*	Value of offset on which to base predictions; default 0
NBINOMIAL = *scalar*	Supplies the total number of trials to be used for prediction with a binomial distribution (providing a value *n* greater than one allows predictions to be made of the number of "successes" out of *n*, whereas the value one predicts the proportion of successes); default 1
PREDICTIONS = *table* or *scalar*	To save the predictions; default *
BACKPREDICTIONS = *table* or *scalar*	To save back-transformed predictions; default *
SE = *table* or *scalar*	To save standard errors of predictions; default *
SED = *symmetric matrix*	To save standard errors of differences between predictions; default *
VCOVARIANCE = *symmetric matrix*	To save variances and covariances of predictions; default *
GLSAVE = *pointer*	Save structure from the GLMM analysis; default * uses the SAVE structure from the most recent GLMM analysis

Parameters

CLASSIFY = *vectors*	Variates and/or factors to classify table of predictions
LEVELS = *variates*, *scalars* or *texts*	To specify values of variates and/or levels of factors for which predictions are calculated
PARALLEL = *identifiers*	For each vector in the CLASSIFY list, allows you to specify another vector in the CLASSIFY list with which the values of this vector should change in parallel (you then obtain just one dimension in the table of predictions for these vectors)
NEWFACTOR = *identifiers*	Identifiers for new factors that are defined when LEVELS are specified

†GLRTEST procedure

Calculates likelihood tests to assess the random terms in a generalized linear mixed model (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (tests); default test
SELECTION = *string tokens*	Specifies information to print with the tests (aic, sic, bic, critical); default crit
CRITICAL = *variate*	Saves the critical values
GLSAVE = *pointer*	Save structure of the original analysis from GLMM; default * uses the save structure from the most recent GLMM analysis

Parameters

TERMS = *formula*	Random terms to be tested; default is to test them all
TESTSTATISTIC = *scalar* or *pointer* to *scalars*	Test statistics for each term
DF = *scalar* or *pointer* to *scalars*	Degrees of freedom of the test statistics
AIC = *scalar* or *pointer* to *scalars*	Akaike information coefficients for each term
SIC = *scalar* or *pointer* to *scalars*	Schwarz (Bayesian) information coefficients for each term

GPREDICTION procedure

Produces genomic predictions (breeding values) using phenotypic and molecular marker information (M. Malosetti, M.P. Boer & S.J. Welham).

Options

PRINT = *string token*	What to print (summary); default summ
PLOT = *string token*	What to plot (scatterplot, pco); default scat, pco
MODELTYPE = *string token*	Model to use to obtain the predictions (gblup, gaussian, exponential); default gblu
THETA = *variate*	Values to use for the tuning parameter θ when the model is Gaussian or exponential
SIMILARITY = *symmetric matrix*	Similarity matrix between individuals of the whole population

Parameters

TRAIT = *variates*	Quantitative trait to be analysed; must be set

GENOTYPES = *factors*	Genotype factor; must be set
MKSCORES = *pointers*	Marker scores
IDMGENOTYPES = *texts*	Labels of the tested and untested genotypes
PREDICTIONS = *variates*	Saves the predictions
NEWGENOTYPES = *factors*	Factor to index the predictions
TESTED = *factors*	Factor that classifies NEWGENOTYPES as part of the tested or the untested set
SAVE = *pointers*	Pointer to REML save structures to save details of the analyses

GRANDOM procedure

Generates pseudo-random numbers from probability distributions (D.M. Roberts & P.W. Lane).

Options

DISTRIBUTION = *string token*	Type of distribution required (beta, chisquare, exponential, F, gamma, logNormal, Normal, t, uniform, Weibull, binomial, hypergeometric, Poisson); default Norm
NVALUES = *scalar*	Number of values to generate; default 1
SEED = *scalar*	Seed to start random number generation; default set by CALCULATE or continued from previous generation
MEAN = *scalar*	Mean for distribution, except for Weibull or hypergeometric); default 0 for Normal distribution and 1 for Poisson and exponential, otherwise *
VARIANCE = *scalar*	Variance for distribution, except for the Weibull or hypergeometric; must be positive; default *, except for Normal when default is 1
LOWER = *scalar*	Lower bound for the uniform or beta distribution; default 0
UPPER = *scalar*	Upper bound for the uniform or beta distribution; default 1
LOCATION = *scalar*	Location parameter for the log-Normal, gamma or Weibull distribution; default 0
SCALE = *scalar*	Scale parameter for the Weibull distribution; must be positive; default 1
SHAPE = *scalar*	Shape parameter for the Weibull distribution; must be positive; default 1
ABETA = *scalar*	First shape parameter for the beta distribution; must be positive; default 1
BBETA = *scalar*	Second shape parameter for the beta distribution; must be positive; default 1
AGAMMA = *scalar*	Location-scale parameter for the gamma distribution, must be positive, usually denoted by alpha or theta; default 1
BGAMMA = *scalar*	Shape parameter for the gamma distribution, must be positive, usually denoted by beta or kappa; default 1
DF = *scalar*	Number of degrees of freedom for the t or chi distribution, must be 1 or greater; default 1
DFNUMERATOR = *scalar*	Number of degrees of freedom of the numerator for the F distribution, must be 1.0 or greater; default 1
DFDENOMINATOR = *scalar*	Number of degrees of freedom of the denominator for the F distribution, must be 1.0 or greater; default 1
NBINOMIAL = *scalar*	Number of binomial trials for the binomial distribution, must be positive; default 1
PROBABILITY = *scalar*	probability of success for the binomial or hypergeometric distribution, must be positive and not greater than 1; default 0.5
NHYPERGEOMETRIC = *scalar*	Number of elements for the hypergeometric distribution, must be positive; default 1
SSHYPERGEOMETRIC = *scalar*	Sample size for the hypergeometric distribution, must be positive and less than NHYPERGEOMETRIC; default 1

Parameter

NUMBERS = *scalar* or *variate*	The generated numbers are returned here; if the length of the supplied structure is defined, it must equal the setting of the NVALUES option

GRAPH directive

Produces scatter and line graphs on the terminal or line printer.

This directive was replaced in Release 10 by the directive LPGRAPH (with exactly the same options and parameters). It is currently retained as a synonym of LPGRAPH, but may be removed in a future release.

GRCSR procedure

Generates completely spatially random points in a polygon (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	What to print (summary); default summ

Parameters

YPOLYGON = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
XPOLYGON = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
NPOINTS = *scalars*	How many points to generate in each polygon; no default – this parameter must be set
YCSR = *variates*	Variates to receive the vertical coordinates of the points that have been generated
XCSR = *variates*	Variates to receive the horizontal coordinates of the points that have been generated
SEED = *scalars*	Seeds for the random numbers used to generate the points; default 0

GREJECTIONSAMPLE procedure

Generates random samples using rejection sampling (W. van den Berg).

Options

PLOT = *string tokens*	What to plot (density, sample); default dens, samp
NVALUES = *scalar*	Size of each random sample; no default, must be set
PRDENSITY = *expression structure*	Calculation defining the probability density function f(x) to sample; no default, must be set
X = *identifier*	Data structure used inside PRDENSITY for the x-coefficient of the density function f(x) no default, must be set
XLOWER = *scalar*	Lower bound of the region in which f(x) is non-negligible; default –10
XUPPER = *scalar*	Upper bound of the region in which f(x) is non-negligible; default 10
PRENVELOPE = *expression structure*	Calculation defining the probability density function g(x) used to generate the sample; default !e(PRT(X; 60))
GRENVELOPE = *expression structure*	Calculation to sample from the probability density g(x) used to generate the sample (note, PRENVELOPE and GRENVELOPE must either be both set, or both unset); default !e(GRT(NTRIES; 60))
MULTIPLIER = *scalar*	Multiplier M used in the definition of the envelope $M \times$ g(x) that must always be greater than f(x); default 10
NTRIES = *scalar*	Number of random samples to take in each sampling step; default * i.e. determined automatically

Parameters

NUMBERS = *variates*	Saves each random sample
SEED = *scalars*	Seed to use for the random numbers used to generate each random sample; default 0

GRIBIMPORT procedure

Reads data from a GRIB2 meteorological data file, and loads it or converts it to a spreadsheet file (D.B. Baird).

PRINT = *string token*	What information to print (catalogue); default cata
OUTTYPE = *string token*	Output file type (GEN, GSH, GWB, XLS, XLSX, TXT, CSV, RECORDS); default GWB
METHOD = *string token*	Whether to load data into the Genstat server after creating the file, or merely to create the file (create, load); default load
SERIAL = *string token*	Whether to store the records in series, in a single column, instead of in parallel columns (no, yes); default no
LONGITUDERANGE = *string token*	What range to use for longitude (negative, positive); default posi
MISSING = *scalar*	What value represents a missing value; default -999
GRID = *variate*	Specifies limits on the longitude and latitude for the data to be read; default * i.e. read all grid points
ENDTIME = *string token*	Whether to keep the end time for each period when SERIAL = yes (yes, no); default no
SCOPE = *string token*	Whether to create the data locally in a procedure that is using GRIBIMPORT, or globally in the whole program (local, global); default loca

Parameters

FILE = *texts*	Input file or URL to be read
OUTFILE = *texts*	Name of the output file to be created; if this is not provided a temporary file will be created, and then deleted if the data are loaded
RECORDS = *scalars* or *variates*	The numbers of the records to read; default is to read all the records in the file
MATCH = *texts*	Text strings to match in the record descriptions; default * requests all the records selected by RECORDS
COLUMNS = *texts*	Names and/or type codes for the columns that are read (the type of column can be forced by ending the column name, if supplied, with the code **!** for a factor, **#** for a variate, and **$** for a text), using a name of '*' will cause a column to be dropped
ISAVE = *pointers*	Saves the identifiers of the columns

GRLABEL procedure

Randomly labels two or more spatial point patterns (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Options

PRINT = *string token*	What to print (summary); default summ
SEED = *scalar*	Seed for the random numbers used to create the random labellings; default 0

Parameters

OLDY = *variates*	Vertical coordinates of two or more spatial point patterns; no default – this parameter must be set
OLDX = *variates*	Horizontal coordinates of two or more spatial point patterns; no default – this parameter must be set
NEWY = *variates*	Variates to receive the vertical coordinates of the spatial point patterns created by random labelling
NEWX = *variates*	Variates to receive the horizontal coordinates of the spatial point patterns created by random labelling

GRMNOMIAL procedure

Generates multinomial pseudo-random numbers (D.B. Baird).

Options

NVALUES = *scalar*	Number of values to generate
SEED = *scalar*	Seed to generate the random numbers; default 0 continues an existing sequence or initializes the sequence automatically if no random numbers have been generated in this job

Parameters

PROBABILITIES = *variates* or *tables*	Probabilities for the categories
NUMBERS = *factors*	Saves the random numbers
COUNTS = *tables*	Saves counts of the numbers generated in each category

GRMULTINORMAL procedure

Generates multivariate Normal pseudo-random numbers (P.W. Goedhart & K.L. Moore).

Options

NVALUES = *scalar*	Number of values to generate; default 1
MEANS = *variate*	The mean for the multivariate Normal distribution; default is a variate with values all equal to 0
VCOVARIANCE = *symmetric matrix*	The variance/covariance matrix for the multivariate Normal distribution; default is to use an identity matrix
SEED = *scalar*	Seed to generate the random numbers; default 0 continues an existing sequence or initializes the sequence automatically if no random numbers have been generated in this job

Parameters

NUMBERS = *pointers* or *matrices*	Saves the random numbers as either a pointer to a set of variates or a matrix

GROUPS directive

Forms a factor (or grouping variable) from a variate or text, together with the set of distinct values that occur.

Options

PRINT = *string token*	Printed output required (summary); default * i.e. no printing
NGROUPS = *scalar*	Number of groups to form when LIMITS is not specified; if NGROUPS is also unspecified, each distinct value (allowing for rounding) defines a group; default *
LMETHOD = *string token*	Defines how to form the levels variate if the setting of the VECTOR parameter is a variate, or the labels if it is a text; if LMETHOD=* no levels/labels are formed, and existing levels (for a variate VECTOR) or labels (for a text VECTOR) of an already declared FACTOR will be retained if still appropriate (given, minimum, median, maximum, limit); default medi
DECIMALS = *scalar*	Number of decimal places to which to round the VECTOR before forming the groups; default * i.e. no rounding
BOUNDARIES = *string token*	Whether to interpret the LIMITS as upper or lower boundaries (upper, lower); default lowe
REDEFINE = *string token*	Whether to allow a structure in the FACTOR list that has already been declared (e.g. as a variate or text) to be redefined (yes, no); default no
CASE = *string token*	Whether the case of letters (small and capital) in text should be regarded as significant or ignored (significant, ignored); default sign
LDIRECTION = *string token*	How to define the levels (for a variate VECTOR) or labels (for a text VECTOR) when LMETHOD = minimum, median or maximum (ascending, given); default asce
OMITUNBOUNDED = *string token*	Whether to omit the (unbounded) group that occurs below the lowest limit when BOUNDARIES=lower, or above the final

limit when BOUNDARIES=upper (yes, no); default no

Parameters

VECTOR = *variates* or *texts*	Vectors whose values are to define the groups
FACTOR = *factors*	Structures to be defined as factors to save details of the groups; default * will, if REDEFINE=yes, cause the corresponding VECTOR itself to be defined as a factor
LIMITS = *variates* or *texts*	Limits to define the groups
LEVELS = *variates*	Variate to define the levels of each FACTOR if LMETHOD=give, or to save them otherwise
LABELS = *texts*	Text to define the labels of each FACTOR if LMETHOD=give, or to save them otherwise

GRTHIN procedure

Randomly thins a spatial point pattern (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	What to print (summary); default summ

Parameters

OLDY = *variates*	Vertical coordinates of each spatial point pattern; no default – this parameter must be set
OLDX = *variates*	Horizontal coordinates of each spatial point pattern; no default – this parameter must be set
NPOINTS = *scalars*	How many points to return from each pattern; no default – this parameter must be set
NEWY = *variates*	Variates to receive the vertical coordinates of the randomly thinned patterns
NEWX = *variates*	Variates to receive the horizontal coordinates of the randomly thinned patterns
SEED = *scalars*	Seeds for the random numbers used to select the thinned points; default 0

GRTORSHIFT procedure

Performs a random toroidal shift on a spatial point pattern (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	What to print (summary); default summ

Parameters

OLDY = *variates*	Vertical coordinates of each spatial point pattern; no default – this parameter must be set
OLDX = *variates*	Horizontal coordinates of each spatial point pattern; no default – this parameter must be set
YBOX = *variates*	Vertical coordinates of the toroidal regions
XBOX = *variates*	Horizontal coordinates of the toroidal regions
NEWY = *variates*	Variates to receive the vertical coordinates of the randomly shifted patterns
NEWX = *variates*	Variates to receive the horizontal coordinates of the randomly shifted patterns
SEED = *scalars*	Seeds for the random numbers used to perform the shifts; default 0

GSTATISTIC procedure

Calculates the gamma statistic of agreement for ordinal data (A.W. Gordon).

Options

PRINT = *string token*	Whether to print the statistic with its associated information and the resulting test (test); default test
METHOD = *string token*	Type of test required (twosided, positive, negative);

default `twos`

Parameters

`DATA` = *tables*	Tables of data each classified by the two variables (factors) of interest
`STATISTIC` = *scalars*	Save the value of gamma for each data table
`VARIANCE` = *scalars*	Save the corresponding variances

G2AEXPORT procedure

Forms a dbase file to transfer `ANOVA` output to Agronomix Generation II (R.W. Payne).

Options

`PRINT` = *strings*	Controls printed output (`columns`); default `*` i.e. none
`REPLICATETERMS` = *formula*	Specifies the term or terms that define the replication in the design
`METHOD` = *string*	How to form the means (`lowestratum`, `combined`); default `lowe`
`ALPHALEVEL` = *scalar*	Alpha value to use when calculating least significant differences; default 0.05
`TAIL` = *scalar*	Number of tails in the calculation of least significant differences (1, 2); default 1
`SAVE` = *ANOVA save structure*	Save structure for the analysis from which the means &c are to be saved; default * takes the information from the most recent `ANOVA` analysis

Parameters

`MEANTERM` = *formula*	Defines the treatment term whose means are to be saved; no default (must be specified)
`OUTFILE` = *text*	Name of the output file (dbf) to form; default `*` i.e. file not formed

G2AFACTORS procedure

Redefines block and treatment variables as factors (R.W. Payne).

No options

Parameter

`FACTOR` = *variates* or *texts*	Other variates or texts to convert into factors (if required)

G2VEXPORT procedure

Forms a dbase file to transfer `REML` output to Agronomix Generation II (R.W. Payne).

Options

`PRINT` = *strings*	Controls printed output (`columns`); default `*` i.e. none
`REPLICATETERMS` = *formula*	Specifies the term or terms that define the replication in the design
`MODEL` = *formula*	Indicates which model terms (fixed and/or random) are to be used in forming the predictions; default `*` includes all the fixed terms and relevant random terms
`OMITTERMS` = *formula*	Specifies terms to be excluded from the `MODEL`; default `*` i.e. none
`FACTORIAL` = *scalar*	Limit on the number of factors or variates in each term in the models specified by `MODEL` or `OMITTERMS`; default 3
`PRESENT` = *identifiers*	Lists factors for which averages should be taken across combinations that are present
`WEIGHTS` = *tables*	One-way tables of weights classified by factors in the model; default `*`
`ALPHALEVEL` = *scalar*	Alpha value to use when calculating least significant differences; default 0.05
`TAIL` = *scalar*	Number of tails in the calculation of least significant differences (1, 2); default 1
`SAVE` = *REML save structure*	Save structure for the analysis from which the means &c are to

	be saved; default * takes the information from the most recent REML analysis

Parameters

MEANTERM = *formula*	Defines the treatment term whose means are to be saved; no default (must be specified)
OUTFILE = *text*	Name of the output file (dbf) to form; default * i.e. file not formed

HANOVA procedure

Does hierarchical analysis of variance/covariance for unbalanced data (P.W. Lane).

Options

PRINT = *string token*	Which analyses to print (all, some, none); default all
INCHANNEL = *scalar*	Channel from which to read data; default * specifies that the data values are already stored in the factors and variates specified by the parameters of HANOVA
FORMAT = *variate*	Format for reading data; default * requests free format
ANALYSIS = *symmetric matrix*	For PRINT=some, this indicates which analyses to print
SSPM = *SSPM*	Stores the corrected sums of squares and products; default *
COEFFICIENT = *matrix*	Stores the estimated variance and co-variance components; default *

Parameters

VARIATES = *pointers*	Variates to be analysed
FACTORS = *pointers*	Factors defining the hierarchy, the first factor of the pointer defining the first stratum, and so on

HBOOTSTRAP directive

Performs bootstrap analyses to assess the reliability of clusters from hierarchical cluster analysis (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (clusters, dendrograms; default * i.e. none
METHOD = *string token*	Criterion for forming clusters (singlelink, nearestneighbour, completelink, furthestneighbour, averagelink, mediansort, groupaverage); default sing
CLIMIT = *scalar*	Similarity value below which clusters are not recorded; default 0
UNITS = *text* or *variate*	Names to label the units of the clusters when they are printed; default *
MINKOWSKI = *scalar*	Index *t* for use with TEST=minkowski
CLUSTERS = *pointer*	Specifies or saves the clusters
REPLICATION = *variate*	Saves the replication of the clusters in the bootstrap samples
NDATASAMPLE = *scalar*	Number of DATA vectors to take in each sample; default takes the same number as supplied by the DATA parameter
NTIMES = *scalar*	Number of times to resample; default 100
SEED = *scalar*	Seed for random number generator; default continue from previous generation or use system clock

Parameters

DATA = *variates* or *factors*	The characteristics of the units to be clustered
TEST = *string tokens*	Test type, defining how each DATA variate or factor is treated in the calculation of the similarity between each unit (simplematching, jaccard, russellrao, dice, antidice, sneathsokal, rogerstanimoto, cityblock, manhattan, ecological, euclidean, pythagorean, minkowski, divergence, canberra, braycurtis, soergel); default * ignores that variate or factor

RANGE = *scalars*	Range of possible values of each DATA variate or factor; if omitted, the observed range is taken

HCLUSTER directive

Performs hierarchical cluster analysis.

Options

PRINT = *string tokens*	Printed output required (dendrogram, amalgamations); default * i.e. no printing
METHOD = *string token*	Criterion for forming clusters (singlelink, nearestneighbour, completelink, furthestneighbour, averagelink, mediansort, groupaverage); default sing
CTHRESHOLD = *scalar*	Clustering threshold at which to print formation of clusters; default * i.e. determined automatically

Parameters

SIMILARITY = *symmetric matrices*	Input similarity matrix for each cluster analysis
GTHRESHOLD = *scalars*	Grouping threshold where groups are formed from the dendrogram
GROUPS = *factors*	Stores the groups formed
PERMUTATION = *variates*	Permutation order of the units on the dendrogram
AMALGAMATIONS = *matrices*	To store linked list of amalgamations

HCOMPAREGROUPINGS procedure

Compares groupings generated, for example, from cluster analyses (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (indexes, tests); default inde
PLOT = *string*	What to plot (histogram); default *
METHOD = *string tokens*	Which indexes to calculate (arand, jaccard, rand); default arand
NTIMES = *scalar*	Number of permutations to make for the tests; default 999

Parameters

FIRSTGROUPING = *factors*	First set of groupings
SECONDGROUPING = *factors*	Second set of groupings
ESTIMATES = *pointers*	Saves the values of the indexes calculated from the original data set
SEED = *scalars*	Seed for the random number generator used to make the permutations; default 0 continues from the previous generation or (if none) initializes the seed automatically
PERMUTATIONESTIMATES = *pointers*	Saves the values of the indexes calculated from the permuted data sets

HDISPLAY directive

Displays results ancillary to hierarchical cluster analyses: matrix of mean similarities between and within groups, a set of nearest neighbours for each unit, a minimum spanning tree, and the most typical elements from each group.

Option

PRINT = *string tokens*	Printed output required (neighbours, tree, typicalelements, gsimilarities); default tree

Parameters

SIMILARITY = *symmetric matrices*	Input similarity matrix for each cluster analysis
NNEIGHBOURS = *scalars*	Number of nearest neighbours to be printed
NEIGHBOURS = *matrices*	Matrix to store nearest neighbours of each unit
GROUPS = *factors*	Indicates the groupings of the units (for calculating typical elements and mean similarities between groups)
TREE = *matrices*	To store the minimum spanning tree (as a series of links and corresponding lengths)

GSIMILARITY = *symmetric matrices* To store similarities between groups

HEATUNITS procedure

Calculates accumulated heat units of a temperature dependent process (R.J. Reader, R.A. Sutherland & K. Phelps).

Options

METHOD = *string token* Temperature/time relationship to be used (sawtooth, cosine, linsine, expsine); default sawt

LATITUDE = *scalar* Latitude at which temperatures were measured; default 52.205 N {Wellesbourne, U.K.}

RATE = *variate* Value of rate relationship at cardinal temperatures

TEMPERATURE = *variate* Cardinal temperatures

PARAMETERS = *variate* Parameters *a*, *b*, *c* (*a*, *c* in hours) for the expsine method

Parameters

MINTEMPERATURE = *variates* Minimum temperature on each day

MAXTEMPERATURE = *variates* Maximum temperature on each day

FIRSTDAY = *scalars* Day of year of first temperature recorded

HEATUNITS = *variates* Development on each day

HELP directive

Provides help information about Genstat commands and functions.

No options

Parameter

TOPIC = *texts* Single-valued texts indicating the command or function about which the information is required

HFAMALGAMATIONS procedure

Forms an amalgamations matrix from a minimum spanning tree (R.W. Payne).

No options

Parameters

TREE = *matrices* Minimum spanning tree

AMALGAMATIONS = *matrices* Saves the amalgamation matrices formed from the minimum spanning trees

HFCLUSTERS procedure

Forms a set of clusters from an amalgamations matrix (R.W. Payne).

Options

CLIMIT = *scalar* Similarity value below which clusters are not formed; default 0

ORDERING = *string token* How to order the clusters (join, lexicographic); default lexi

NCLUSTERS = *scalar* Saves the number of clusters that have been formed

Parameters

AMALGAMATIONS = *matrices* Amalgamation matrices

CLUSTERS = *pointers* Saves the clusters

SIMILARITIES = *variates* Saves the similarity values at which the clusters are formed

HGANALYSE procedure

Analyses data using a hierarchical or double hierarchical generalized linear model (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

PRINT = *string tokens* Controls printed output (model, fixedestimates, randomestimates, dispersionestimates, likelihoodstatistics, deviance, waldtests, fittedvalues, monitoring, dhgmonitoring); default mode, fixe, disp, like, devi, moni

LMETHOD = *string token* — Whether to use exact likelihood or extended quasi likelihood to obtain the y-variate and weights for the dispersion model (exact, eql); default exac

SEMETHOD = *string token* — Method to use to calculate the se's for the dispersion estimates (approximate, profilelikelihood); default appr

DMETHOD = *string token* — Method to use for the adjusted profile likelihood when calculating the likelihood statistics (automatic, choleski, lrv); default auto

EMETHOD = *string token* — Extrapolation method to use (aitken, adjustedaitken); default aitk

MLAPLACEORDER = *scalar* — Order of Laplace approximation to use in the estimation of the mean model (0 or 1); default 0

DLAPLACEORDER = *scalar* — Order of Laplace approximation to use in the estimation of the dispersion components (0, 1 or 2); default 0

MAXCYCLE = *scalars* — Maximum number of iterations of the hierarchical generalized linear model fits, and maximum number of iterations in the fitting of the mean and dispersion models; default 99,50

EXIT = *scalar* — Exit status (0 for success, 1 for failure to converge)

TOLERANCE = *scalar* — Criterion for convergence; default 0.0005

ETOLERANCE = *scalar* — Maximum size of ratio of the original to the new estimates allowed in Aitken extrapolation; default 7.5

GROUPTERM = *formula* — Random term to use as groups when fitting the augmented mean model; default * i.e. none

Parameters

Y = *variate* — Response variate (must be one only)

NBINOMIAL = *variate* — Total numbers for binomial data

RESIDUALS = *variate* — Saves the residuals

FITTEDVALUES = *variate* — Saves the fitted values

SAVE = *pointer* — Saves details of the analysis for use in subsequent HGDISPLAY, HGKEEP, HGPLOT or HGPREDICT statements

HGDISPLAY procedure

Displays results from a hierarchical or double hierarchical generalized linear model analysis (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

PRINT = *string tokens* — Controls printed output (model, fixedestimates, randomestimates, dispersionestimates, likelihoodstatistics, deviance, waldtests, fittedvalues); default *

SEMETHOD = *string token* — Method to use to calculate the se's for the dispersion estimates (approximate, profilelikelihood); default appr

DMETHOD = *string token* — Method to use for the adjusted profile likelihood when calculating the likelihood statistics (automatic, choleski, lrv); default auto

DISPERSIONTERM = *formula* — Model term for output from a dispersion analysis

SAVE = *pointer* — Save structure (from HGANALYSE) to provide details of the analysis; if omitted, output is from the most recent analysis

No parameters

HGDRANDOMMODEL procedure

Defines the random model in a hierarchical generalized linear model for the dispersion in a double hierarchical generalized linear model (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

DISTRIBUTION = *string token* — Distribution for the random model (beta, normal, gamma, inversegamma); default norm

LINK = *string token* — Link for the random model (identity, logarithm, logit,

	`reciprocal`); default `iden`
`RANDOMTERM` = *formula*	Random term whose dispersion is being modelled; if unset, the model is assumed to be for the residual dispersion parameter (phi)
`PHIMETHOD` = *string token*	Whether to fix or estimate the residual dispersion parameter in the dispersion HGLM (`fix`, `estimate`); default `fix`

Parameters

`TERMS` = *formula*	Random model
`DLINK` = *string tokens*	Link for the dispersion model for each random term (`logarithm`, `reciprocal`); default `loga`
`DFORMULA` = *formula structures*	Dispersion model for each random term; default `*` i.e. none
`DOFFSET` = *variates*	Offset variate for dispersion model for each random term; default `*` i.e. none
`LMATRIX` = *matrices*	Linear transformation to apply to design matrix **Z** of each random term, in order to define correlations between its effects; default `*` i.e. none
`DDISPERSION` = *scalar*	Dispersion parameter to use in the dispersion model for each random term; default 1
`FDISPERSION` = *scalar*	Fixed value for the dispersion parameter of each random term; default `!s(*)` i.e. dispersion is estimated

`HGFIXEDMODEL` procedure

Defines the fixed model for a hierarchical or double hierarchical generalized linear model (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

`DISTRIBUTION` = *string token*	Distribution of the data (`binomial`, `poisson`, `normal`, `gamma`); default `norm`
`LINK` = *string token*	Link for the fixed model (`identity`, `logarithm`, `logit`, `reciprocal`, `probit`, `complementaryloglog`); default `iden`
`DLINK` = *string token*	Link for the dispersion model (`logarithm`, `reciprocal`); default `loga`
`DISPERSION` = *scalar*	Value of dispersion parameter in calculation of s.e.s etc; default `*` for `DIST=norm` or `gamm`, and 1 for `DIST=pois` or `bino`
`DTERMS` = *formula*	Dispersion model; default `*` i.e. none
`CONSTANT` = *string token*	How to treat the constant (`estimate`, `omit`) default `esti`
`FACTORIAL` = *scalar*	Limit on number of variates and/or factors in a fixed model term; default 3
`WEIGHTS` = *variate*	Prior weights; default `*` i.e. 1
`OFFSET` = *variate*	Offset variate; default `*` i.e. none
`DOFFSET` = *variate*	Offset variate for dispersion model; default `*` i.e. none
`DDISPERSION` = *scalar*	Dispersion parameter to use in a dispersion model for the residual dispersion parameter phi; default 1
`IDISPERSION` = *scalar*	Initial value for the residual dispersion parameter phi; default `*` i.e. formed automatically

Parameter

`TERMS` = *formula*	Fixed model

`HGFTEST` procedure

Calculates likelihood tests for fixed terms in a hierarchical generalized linear model (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

`PRINT` = *string token*	Controls printed output (`tests`); default `test`
`FACTORIAL` = *scalar*	Limit on number of factors in the model terms generated from the `TERMS` parameter

LMETHOD = *string token*	Whether to use exact likelihood or extended quasi likelihood to obtain the y-variate and weights for the dispersion model (exact, eql); default is to use the same setting as in the original analysis
DMETHOD = *string token*	Method to use for the adjusted profile likelihood when calculating the likelihood statistics (automatic, choleski, lrv); default auto
EMETHOD = *string token*	Extrapolation method to use (aitken, adjustedaitken); default is to use the same setting as in the original analysis
MLAPLACEORDER = *scalar*	Order of Laplace approximation to use in the estimation of the mean model (0 or 1); default is to use the same setting as in the original analysis
DLAPLACEORDER = *scalar*	Order of Laplace approximation to use in the estimation of the dispersion components (0, 1 or 2); default is to use the same setting as in the original analysis
MAXCYCLE = *scalars*	Maximum number of iterations of the hierarchical generalized linear model fits, and maximum number of iterations in the fitting of the mean and dispersion models; default 99,50
EXIT = *scalar*	Exit status (0 for success, 1 for failure to converge with any of the fixed terms)
TOLERANCE = *scalar*	Criterion for convergence; default is to use the same setting as in the original analysis
ETOLERANCE = *scalar*	Maximum size of ratio of the original to the new estimates allowed in Aitken extrapolation; default is to use the same setting as in the original analysis
SAVE = *pointer*	Save structure from the original analysis

Parameters

TERMS = *formula*	Terms to test
TESTSTATISTIC = *pointer* or *scalar*	Saves the test statistics
DF = *pointer* or *scalar*	Saves the degrees of freedom

HGGRAPH procedure

Draws a graph to display the fit of an HGLM or DHGLM analysis (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

GRAPHICS = *string token*	Type of graphics to use (lineprinter, highresolution); default high
TITLE = *text*	Title for the graph; default * sets an appropriate title automatically
WINDOW = *number*	Which high-resolution graphics window to use; default 4 (redefined if necessary to fill the frame)
SCREEN = *string token*	Whether to clear the graphics screen before plotting (clear, keep); default clea
BACKTRANSFORM = *string token*	What back-transformation to make (link, none, axis); default none
OMITRESPONSE = *string token*	Whether to omit the adjusted response values (no, yes); default no
SAVE = *pointer*	Specifies the save structure (from HGANALYSE) of the analysis from which to predict; default uses the most recent analysis

Parameters

INDEX = *variates* or *factors*	Which variate or factor to display along the x-axis; default * if GROUPS is set, otherwise INDEX is set to the first variate in the fixed model
GROUPS = *factors*	Factor to define groups of points to display; default * if INDEX is set, otherwise GROUPS is set to the first factor in the fixed model

HGKEEP procedure

Saves information from a hierarchical or double hierarchical generalized linear model analysis (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

MODELTYPE = *string token*	Type of model from which to save information (mean, dispersion); default mean
RMETHOD = *string token*	Type of residuals to save using the RESIDUALS parameter (deviance, Pearson, simple); default devi
DMETHOD = *string token*	Method to use for the adjusted profile likelihood when calculating the likelihood statistics (automatic, choleski, lrv); default auto
IGNOREFAILURE = *string token*	Whether to save information even if the fitting of the HGLM failed to converge (yes, no); default no
SAVE = *pointer*	Save structure (from HGANALYSE) to provide details of the analysis; if omitted, information is saved from the most recent analysis

Parameters

RANDOMTERM = *formula*	Random model terms from whose analysis the information is to be saved
DHGRANDOMTERM = *formula*	Random model terms in a DHGLM from whose (HGLM) analysis the information is to be saved
RESIDUALS = *variates*	Residuals
FITTEDVALUES = *variates*	Fitted values
LEVERAGES = *variates*	Leverages
ESTIMATES = *variates*	Estimates of parameters
SE = *variates*	Standard errors of the estimates
VCOVARIANCE = *symmetric matrices*	Variance-covariance matrix of each set of estimates
DEVIANCE = *scalars* or *tables*	Scaled deviances (in a table) for a mean model, or residual deviance (in a scalar) for a dispersion model
DF = *scalars* or *tables*	Residual degrees of freedom
ITERATIVEWEIGHTS = *variates*	Iterative weights
LINEARPREDICTOR = *variates*	Linear predictors
YADJUSTED = *variates*	Adjusted responses
LIKELIHOODSTATISTICS = *variates*	Likelihood statistics
LDF = *variates*	Numbers of fixed and random parameters in the mean and dispersion models

HGNONLINEAR procedure

Defines nonlinear parameters for the fixed model of a hierarchical generalized linear model (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

CALCULATION = *expression structures*	Calculation of explanatory variates involving nonlinear parameters
METHOD = *string token*	Algorithm for fitting the nonlinear model (GaussNewton, NewtonRaphson, FletcherPowell); default Gaus
VECTORS = *variates*	Vectors involved in the calculations (data vectors or factors or derived vectors that appear in the fixed model)

Parameters

PARAMETER = *scalars*	Nonlinear parameters in the model
LOWER = *scalars*	Lower bound for each parameter
UPPER = *scalars*	Upper bound for each parameter
STEPLENGTH = *scalars*	Initial step length for each parameter
INITIAL = *scalars*	Initial value for each parameter
DELTA = *scalars*	Parameter increment to use when calculating numerical derivatives

HGPLOT procedure

Produces model-checking plots for a hierarchical or double hierarchical generalized linear model analysis (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

MODELTYPE = *string token*	Type of model for which plots are required (mean, dispersion); default mean
RANDOMTERM = *formula*	Random term whose residuals are to be plotted; default * i.e. the residuals from the full model
DHGRANDOMTERM = *formula*	Random model term in a DHGLM whose residuals are to be plotted; default *
RMETHOD = *string token*	Type of residual to use (deviance, Pearson, simple); default devi
INDEX = *variate* or *factor*	X-values to use for an index plot; default !(1,2...)
GRAPHICS = *string token*	What type of graphics to use (lineprinter, highresolution); default high
TITLE = *text*	Overall title for the plots; if unset, the identifier of the y-variate is used
SAVE = *pointer*	Specifies the analysis (by HGANALYSE) from which the residuals and fitted values are to be taken; by default they are taken from the most recent analysis

Parameters

METHOD = *string tokens*	Types of graph (up to four out of the six possible) to be plotted (histogram, fittedvalues, absresidual, normal, halfnormal, index); default hist, fitt, norm, absr
PEN = *scalars*, *variates* or *factors*	Pen(s) to use for each plot

HGPREDICT procedure

Forms predictions from a hierarchical or double hierarchical generalized linear model analysis (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

PRINT = *string token*	What to print (description, predictions, se, sed, vcovariance); default desc, pred, se
COMBINATIONS = *string token*	Which combinations of factors in the current model to include (full, present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment (marginal, equal); default marg
WEIGHTS = *table*	Weights classified by some or all of the factors in the model; default *
OFFSET = *scalar*	Value of offset on which to base predictions; default mean of offset variate
METHOD = *string token*	Method of forming margin (mean, total); default mean
ALIASING = *string token*	How to deal with aliased parameters (fault, ignore); default faul
BACKTRANSFORM = *string token*	What back-transformation to apply to the values on the linear scale, before calculating the predicted means (link, none); default none
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, nonlinear); default *
NBINOMIAL = *scalar*	Supplies the total number of trials to be used for prediction with a binomial distribution (providing a value *n* greater than one allows predictions to be made of the number of "successes" out of *n*, whereas the value 1 predicts the proportion of successes); default 1
PREDICTIONS = *table* or *scalar*	To save the predictions; default *
SE = *table* or *scalar*	To save standard errors of predictions; default *
SED = *symmetric matrix*	To save matrices of standard errors of differences between

predictions; default *

VCOVARIANCE = *symmetric matrix* — To save variance-covariance matrices of predictions; default *

SAVE = *pointer* — Specifies the save structure (from HGANALYSE) of the analysis from which to predict; default uses the most recent analysis

Parameters

CLASSIFY = *vectors* — Variates and/or factors to classify table of predictions

LEVELS = *variates* or *scalars* — To specify values of variates, levels of factors

NEWFACTOR = *identifiers* — Identifiers for new factors that are defined when LEVELS are specified

HGRANDOMMODEL procedure

Defines the random model for a hierarchical or double hierarchical generalized linear model (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

DISTRIBUTION = *string token* — Distribution for the random model (beta, normal, gamma, inversegamma); default norm

LINK = *string token* — Link for the random model (identity, logarithm, logit, reciprocal); default iden

Parameters

TERMS = *formula* — Random model

DLINK = *string tokens* — Link for the dispersion model for each random term (logarithm, reciprocal); default loga

DFORMULA = *formula structure* — Dispersion model for each random term; default * i.e. none

DOFFSET = *variates* — Offset variate for dispersion model for each random term; default * i.e. none

LMATRIX = *matrices* — Linear transformation to apply to design matrix **Z** of each random term, in order to define correlations between its effects; default * i.e. none

DDISPERSION = *scalar* — Dispersion parameter to use in the dispersion model for each random term; default 1

FDISPERSION = *scalar* — Fixed value for the dispersion parameter of each random term; default !s(*) i.e. dispersion is estimated

IDISPERSION = *scalar* — Initial value for the dispersion parameter for each random term; default * i.e. formed automatically

HGRTEST procedure

Calculates likelihood tests for random terms in a hierarchical generalized linear model (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

PRINT = *string token* — Controls printed output (tests); default test

LMETHOD = *string token* — Whether to use exact likelihood or extended quasi likelihood to obtain the y-variate and weights for the dispersion model (exact, eql); default is to use the same setting as in the original analysis

DMETHOD = *string token* — Method to use for the adjusted profile likelihood when calculating the likelihood statistics (automatic, choleski, lrv); default auto

EMETHOD = *string token* — Extrapolation method to use (aitken, adjustedaitken); default is to use the same setting as in the original analysis

MLAPLACEORDER = *scalar* — Order of Laplace approximation to use in the estimation of the mean model (0 or 1); default is to use the same setting as in the original analysis

DLAPLACEORDER = *scalar* — Order of Laplace approximation to use in the estimation of the dispersion components (0, 1 or 2); default is to use the same setting as in the original analysis

MAXCYCLE = *scalars* — Maximum number of iterations of the hierarchical generalized

	linear model fits, and maximum number of iterations in the fitting of the mean and dispersion models; default 99,50
EXIT = *scalar*	Exit status (0 for success, 1 for failure to converge for any of the random terms)
TOLERANCE = *scalar*	Criterion for convergence; default is to use the same setting as in the original analysis
ETOLERANCE = *scalar*	Maximum size of ratio of the original to the new estimates allowed in Aitken extrapolation; default is to use the same setting as in the original analysis
GROUPTERM = *formula*	Random term to use as groups when fitting the augmented mean model; default is to use the same setting as in the original analysis
SAVE = *pointer*	Save structure from the original analysis

Parameters

TERMS = *formula*	Terms to test
TESTSTATISTIC = *pointer* or *scalar*	Saves the test statistics
DF = *pointer* or *scalar*	Saves the degrees of freedom

HGSTATUS procedure

Displays the current HGLM model definitions (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Option

SAVE = *pointer*	Save structure (from HGANALYSE) to provide details of the HGLM; if omitted, information is printed for the most recently defined or fitted HGLM

No parameters

HGWALD procedure

Prints or saves Wald tests for fixed terms in an HGLM (R.W. Payne, Y. Lee, J.A. Nelder & M. Noh).

Options

PRINT = *string token*	Controls printed output (waldtests); default wald
FACTORIAL = *scalar*	Limit on number of factors in the model terms generated from the TERMS parameter; default 3
SAVE = *pointer*	Specifies the save structure (from HGANALYSE) of the analysis from which to calculate the tests; default uses the most recent analysis

Parameters

TERMS = *formula*	Model terms for which tests are required
WALDSTATISTIC = *scalar* or *pointer* to *scalars*	Saves Wald statistics
DF = *scalar* or *pointer* to *scalars*	Saves d.f. of Wald statistics

HISTOGRAM directive

Produces histograms of data on the terminal or line printer.

This directive was replaced in Release 10 by the directive LPHISTOGRAM (with exactly the same options and parameters). It is currently retained as a synonym of LPHISTOGRAM, but may be removed in a future release.

HLIST directive

Lists the data matrix in abbreviated form.

Options

GROUPS = *factor*	Defines groupings of the units; used to split the printed table at appropriate places and to label the groups; default *
UNITS = *text* or *variate*	Names for the rows (i.e. units) of the table; default *

Parameters

DATA = *variates* or *factors*	The data variables

TEST = *string tokens*	Test type, defining how each variable is treated in the calculation of the similarity between each unit (simplematching, jaccard, russellrao, dice, antidice, sneathsokal, rogerstanimoto, cityblock, manhattan, ecological, euclidean, pythagorean, minkowski, divergence, canberra, braycurtis, soergel); default * ignores that variable
RANGE = *scalars*	Range of possible values of each variable; if omitted, the observed range is taken

HPCLUSTERS procedure

Prints a set of clusters (R.W. Payne).

Option

UNITS = *variate* or *text*	Names to use for the units in the clusters

Parameters

CLUSTERS = *pointers*	Clusters to print
EXTRA = *pointers*	Extra information to print

HREDUCE directive

Forms a reduced similarity matrix (referring to the GROUPS instead of the original units).

Options

PRINT = *string token*	Printed output required (similarities); default * i.e. no printing
METHOD = *string token*	Method used to form the reduced similarity matrix (first, last, mean, minimum, maximum, zigzag); default firs

Parameters

SIMILARITY = *symmetric matrices*	Input similarity matrix
REDUCEDSIMILARITY = *symmetric matrices*	Output (reduced) similarity matrix
GROUPS = *factors*	Factor defining the groups
PERMUTATION = *variates*	Permutation order of units (for METHOD = firs, last or zigz)

HSUMMARIZE directive

Forms and prints a group by levels table for each test together with appropriate summary statistics for each group.

Option

GROUPS = *factor*	Factor defining the groups; no default i.e. this option must be specified

Parameters

DATA = *variates* or *factors*	The data variables
TEST = *string tokens*	Test type, defining how each variable is treated in the calculation of the similarity between each unit (simplematching, jaccard, russellrao, dice, antidice, sneathsokal, rogerstanimoto, cityblock, manhattan, ecological, euclidean, pythagorean, minkowski, divergence, canberra, braycurtis, soergel); default * ignores that variable
RANGE = *scalars*	Range of possible values of each variable; if omitted, the observed range is taken

IDENTIFY procedure

Identifies an unknown specimen from a defined set of objects (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (identification, transcript);

default iden, tran

METHOD = *string token* — Type of run (batch, interactive); if this is not set IDENTIFY checks whether the run of Genstat itself is batch or interactive

TAXA = *text* or *factor* — Names for the taxa (i.e. the objects); default uses the positive integers 1, 2...

NMISTAKE = *scalar* — Number of mistakes to allow for; default 0

IDENTIFICATION = *text* — Saves the names of the taxa that are identified; default * i.e. not saved

DIFFERENCES = *variate* — Saves the number of differences between the observed character states and those that can be displayed by each taxon; default * i.e. not saved

Parameters

CHARACTER = *factors* or *tables* — Define the characteristics of the taxa; must be set

OBSERVATION = *scalars* or *texts* — Can define an observation for each character; default * i.e. none

COST = *scalars* — Costs of observing each character; default 1

IF directive

Introduces a block-if control structure.

No options

Parameter

expression — Logical expression, indicating whether or not to execute the first set of statements.

IFUNCTION procedure

Estimates implicit and/or explicit functions of parameters (W.M. Patefield).

Options

PRINT = *string token* — What to print (estimates, correlations, monitoring); default esti

NOMESSAGE = *string token* — Which warning messages to suppress (parameter, convergence); default *

NPARAMETER = *scalar* — Number of parameters; default zero

MAXCYCLE = *scalar* — Maximum number of iterations; default 20

STRINGENCY = *scalar* — Stringency of tests for convergence, 0,1,2...etc; default 5

EXITCONTROL = *string token* — Control for exit on fault detection (job, procedure); default job for batch jobs, proc for interactive

ZCALCULATION = *expression structures* — Specify the calculation of ZERO and DZBIMPLICIT

DZPCALCULATION = *expression structures* — Specify the calculation of DZBPARAMETER

ECALCULATION = *expression structures* — Specify the calculation of EXPLICIT, DEBPARAMETER and DEBIMPLICIT

Parameters

IMPLICIT = *variate* or *pointer to scalars* — Implicit functions

INITIAL = *variate* — Initial values for IMPLICIT functions

LOWER = *variate* — Lower bounds to IMPLICIT functions; default -10^{10}

UPPER = *variate* — Upper bounds to IMPLICIT functions; default $+10^{10}$

VCOVARIANCE = *symmetric matrix* — Variance-covariance matrix of parameter estimates

ZERO = *variate* — Equations defining implicit functions (values calculated by ZCALCULATION)

DZBIMPLICIT = *matrix* — First derivatives of equations ZERO with respect to implicit functions IMPLICIT (values calculated by ZCALCULATION); rows correspond to ZERO, columns correspond to IMPLICIT

DZBPARAMETER = *matrix* — First derivatives of equations ZERO with respect to parameters (must not be set for NPARAMETER=0; values calculated by

	DZPCALCULATION); rows correspond to ZERO, columns to parameters
DIBPARAMETER = *matrix*	First derivatives of IMPLICIT functions with respect to parameters (must not be set for NPARAMETER=0); rows correspond to IMPLICIT, columns correspond to parameters
EXPLICIT = *variate* or *pointer to scalars*	Explicit functions of parameters and/or implicit functions (values calculated by ECALCULATION)
DEBPARAMETER = *matrix*	First partial derivatives of EXPLICIT functions with respect to parameters (values calculated by ECALCULATION); rows correspond to EXPLICIT, columns correspond to parameters
DEBIMPLICIT = *matrix*	First partial derivatives of EXPLICIT functions with respect to IMPLICIT functions (values calculated by ECALCULATION); rows correspond to EXPLICIT, columns correspond to IMPLICIT
DFBPARAMETER = *matrix*	First derivatives of ESTIMATES with respect to parameters; rows correspond to ESTIMATES, columns correspond to parameters
ESTIMATES = *variate*	Estimates of IMPLICIT and EXPLICIT functions
SE = *variate*	Standard errors of ESTIMATES
CORRELATIONS = *symmetric matrix*	Correlation matrix of ESTIMATES
FCOVARIANCE = *symmetric matrix*	Variance-covariance matrix of ESTIMATES

IMPORT procedure

Reads data from a foreign file format and loads it or converts it to a spreadsheet file (D.B. Baird).

Options

PRINT = *string token*	What to print (catalogue, summary); default cata
OUTTYPE = *string token*	Output file type (GEN, GSH, GWB, XLS, XLSX, TXT, CSV, SHEETS); default GWB
METHOD = *string token*	Whether to load data into the Genstat server after creating the file, or merely to create the file (create, load); default load
IMETHOD = *string token*	How identifiers are to be specified for the columns (read, supply, none, overlay); default supply if COLUMNS is set (and specifies names rather than just types), otherwise read
ENDSTATEMENT = *string token*	Ending statement for a type GEN output file (return, endbreak); default retu
SPSSMV = *string token*	What to do with SPSS missing value codes (ignore, convert); default conv
MISSING = *text*	What labels represent missing values in Excel, Quattro or Lotus files; default '*'
FORDER = *string token*	The order in which to define the labels or levels of a factor (sorted, unsorted); default sort
TEXTCONVERSION = *string token*	How to convert text to numbers for the columns (strict, single, common, standard, lax); default stan
KEEPEMPTY = *string tokens*	Whether to retain any empty rows or columns found in the data (rows, columns, none); default none
NAMEROW = *scalar*	The row number within an Excel or Quattro spreadsheet which contains the column names (IMETHOD must be unset or set to read); default, the first row in CELLRANGE
EMETHOD = *string token*	Whether to read column descriptions/extra from Excel, SigmaPlot or Quattro spreadsheets (read, none); default none
EXTRAROW = *scalar*	The row number within an Excel or Quattro spreadsheet which contains the column descriptions (EMETHOD must be set to read); default, the second row in CELLRANGE
PREFIX = *text*	The string with which to prefix numerical column names;

default '%'

TEMPMISSING = *string token* — Whether to read temporarily missing values as missing (yes, no); default no

INOPTIONS = *text* — Optional input file arguments to be passed to the Dataload.dll

OUTOPTIONS = *text* — Optional output file arguments to be passed to the Dataload.dll

RGBMETHOD = *string token* — How to read colour values (combined, separate, matrix); default sepa

SEPARATORS = *text* — Alternative separators to use in text or csv files

SCOPE = *string token* — Whether to create the data locally in a procedure that is using IMPORT, or globally in the whole program (local, global); default loca

IPREFIX = *text* — Prefix to use with unnamed columns, default 'C'

TRANSPOSE = *string token* — Whether to transpose the rows and columns of the input file (yes, no); default no

UNICODE = *string token* — What to do with Unicode characters found e.g. in Excel XLSX input files (utf8, typeset, ascii, remove); default utf8

COLUNICODENAMES = *string token* — How to convert Unicode column names (suffix, extra, ignore) default suff

UNINAME = *text* — Name of the pointer for Unicode column names used as suffixes; default 'C'

†XLSCONTENT = *string tokens* — What content to read from an Excel XLSX file (values, formulae, forecolour, backcolour, fontname, style, size); default valu

Parameters

FILE = *texts* — Input file or URL to be read

OUTFILE = *texts* — Name of the output file to be created; if this is not provided a temporary file will be created, and then deleted if the data are loaded

SHEETNAME = *texts* or *scalars* — Name of a spreadsheet worksheet or named range, or number of a worksheet within the file; default is the first sheet in the file

CELLRANGE = *texts* — Cell range within a worksheet, giving the top left and bottom right cell in the format XXNN:XXNN where XX = A - IV, NN = 1 - 64384; default * requests all data on the sheet

COLUMNS = *texts* — Names and/or type codes for the columns read (the type of column can be forced by ending the column name, if supplied, with the code **!** for a factor, **#** for a variate, and **$** for a text), using a name of '*' will cause a column to be dropped

ISAVE = *pointers* — Saves the identifiers of the columns

START = *texts* — Contents of a cell in a spreadsheet file or a line in a text file from which to start reading

END = *texts* — Contents of a cell in a spreadsheet file or a line in a text file at which to end reading

ANCILLARY = *texts* — Extra information returned by some file formats (currently only population type from QTL location files)

ROWSELECTION = *variates* — Numbers of the rows to import; if unset, all rows are imported

COLSELECTION = *variates* or *texts* — Numbers or names of the columns to import; if unset, all the columns are imported

INPUT directive

Specifies the input file from which to take further statements.

Options

PRINT = *string tokens* — What output to generate from the statements in the file (statements, macros, procedures, unchanged);

default stat

REWIND = *string token* Whether to rewind the file (yes, no); default no

Parameter

scalar Channel number of input file

INSIDE procedure

Determines whether points lie within a specified polygon (S.A. Harding).

Option

TOLERANCE = *scalar* Value used for testing against zero; default 10^{-4}

Parameters

Y = *variates* Y coordinates of points

X = *variates* X coordinates of points

YPOLYGON = *variates* Y coordinates of polygon

XPOLYGON = *variates* X coordinates of polygon

INSIDE = *variates* Indicate whether points are inside (1) the polygon, outside (−1) or on an edge (0)

INTERPOLATE directive

Interpolates values at intermediate points.

Options

CURVE = *string token* Type of curve to be fitted to calculate the interpolated value (linear, cubic); default line

METHOD = *string token* Type of interpolation required (interval, value, missing): for METHOD=valu, values are interpolated for each point in the NEWINTERVAL variate and stored in the NEWVALUE variate; for METHOD=inte, points are estimated in the NEWINTERVAL variate for the observations in the NEWVALUE variate; while for METHOD=miss, the NEWVALUE and NEWINTERVAL lists are irrelevant, INTERPOLATE now interpolates for missing values in the OLDVALUE and OLDINTERVAL variates (except those missing in both variates); default inte

Parameters

OLDVALUES = *variates* Observations from which interpolation is to be done

NEWVALUES = *variates* Results of each interpolation

OLDINTERVALS = *variates* Points at which each set of OLDVALUES was observed

NEWINTERVALS = *variates* Points for each set of NEWVALUES

IRREDUNDANT directive

Forms irredundant test sets for the efficient identification of a set of objects.

Options

PRINT = *string tokens* Controls printed output (numbers, diagram, notdistinguished, messages); default numb, diag, notd, mess

BESTSET = *pointer* Saves the best set

SETS = *matrix* Saves details of the available sets

NOTDISTINGUISHED = *matrix* Saves details of the objects that cannot be distinguished

METHOD = *string token* Algorithm to use (exact, sequential); default exac

TAXONNAMES = *text*, *variate* or *factor*

Defines labels for the objects (or *taxa*) to be identified; default uses the unit labels vector of the CHARACTER factors

GROUPS = *factor* Defines groupings of the objects so that the sets are constructed to distinguish only between the objects that belong to different groups; default constructs sets to distinguish between individual objects

OBJECT = *scalar* or *text* If this is specified, sets are constructed just to distinguish the

specified object (or *taxon*) from the other objects

NDISTINCTIONS = *scalar* Number of factors required in each set to distinguish between each pair of objects; default 1

MAXPREFERENCE = *scalar* Maximum preference of the factors to be included in the sets

MAXSIZE = *scalar* Limit on number of factors in a set (sets containing more than this are discarded); default * i.e. none

NPRINT = *scalar* Number of sets to print (a positive number specifies the number to print, a negative number sets a tolerance on the difference between the sizes of the sets printed and the size of the best set); default * prints them all

NSAVE = *scalar* Number of sets to save in the SETS matrix; default * saves them all

LIMSETS = *variate* Variate containing two numbers n_1 and n_2, if this is specified then every time that there are more than n_1 sets under construction using the exact method, the sets are arranged in order of increasing size and all sets containing more factors than set n_2 are deleted

DISTINCTIONS = *string token* Whether or not to store the distinctions or recalculate them at every stage in the exact algorithm (store, calculate); default stor

CRITERION = *string token* Function to be use to select factors by the sequential method (ndistinctions, weightedndistinctions); default ndis

MAXCYCLE = *scalar* Maximum number of improvement cycles to perform during the sequential method; default 20

EQUIVALENCE = *scalar* Value for determining equivalence of the selection criteria of tests selected during the sequential method

Parameters

CHARACTER = *identifiers* Factors, and/or tables classified by a single factor, defining the properties of the objects to to be identified

COST = *scalars* Cost associated with each factor; default 1

PREFERENCE = *scalar* Preference rating for each factor (1 representing most preferred etc.); default 1

VARIABLE = *scalar* or *text* Factor level used to represent variable information; default is to use a missing value

INAPPLICABLE = *scalar* or *text* Factor level used to indicate that the information provided by that factor is inapplicable for a particular object

JACKKNIFE procedure

Produces Jackknife estimates and standard errors (R.W. Payne).

Options

PRINT = *string token* Controls printed output (estimates, vcovariance); default esti

DATA = *variates*, *factors* or *texts* Data vectors from which the statistics are to be calculated

ANCILLARY = *any type* Other relevant information needed to calculate the statistics

VCOVARIANCE = *symmetric matrix* Saves the variance-covariance matrix for the statistics

Parameters

LABEL = *texts* Texts, each containing a single line, to label the statistics

ESTIMATE = *scalars* Saves the Jackknife estimate for each statistic

SE = *scalars* Saves Jackknife estimates of the standard errors

PSEUDOVALUES = *variates* Saves the Jackknife pseudo-values

ACCELERATION = *scalars* Saves the acceleration parameter for bias-corrected and accelerated bootstrap confidence intervals

JOB directive

Starts a Genstat job.

Options

INPRINT = *string tokens*	Printing of input as in PRINT option of INPUT (statements, macros, procedures, unchanged); default unch
OUTPRINT = *string tokens*	Additions to output as in PRINT option of OUTPUT (dots, page, unchanged); default unch
DIAGNOSTIC = *string tokens*	Defines the least serious class of Genstat diagnostic which should still be generated (messages, warnings, faults, extra, unchanged); default unch
ERRORS = *scalar*	Limit on number of error diagnostics that may occur before the job is abandoned; default * i.e. no change
PROMPT = *text*	Characters to be printed for the input prompt
WORDLENGTH = *string token*	Length of word (8 or 32 characters) to check in identifiers, directives, options, parameters and procedures (long, short); default * i.e. no change

Parameter

text	Name to identify the job

JOIN procedure

Joins or merges two sets of vectors together, based on the values of sets of classifying keys (C.F. Johnston & D.B. Baird).

Options

NINDEX = *scalar*	Number of index vectors in structures (up to 10); default 1
METHOD = *string token*	Type of join (inner, left, right, full); default full
REPEATS = *string token*	How to handle repeats of matches (combinations, single); default sing outputs one row per match
INCLUDE = *string token*	How to handle restrictions on the input vectors (all, nonrestricted); default all uses all the data rows
SORT = *string token*	Whether NEWVECTORS should be sorted on the index vectors (ascending, descending, unsorted); default unsorted keeps the same ordering as the input sets

Parameters

LEFTVECTORS = *pointer*	Pointer to a list of vectors in left set (keys and variables)
RIGHTVECTORS = *pointer*	Pointer to a list of vectors in right set (keys and variables)
NEWVECTORS = *pointer*	Pointer to a list of output vectors (keys and variables)

KALMAN procedure

Calculates estimates from the Kalman filter (A.I. Glaser).

Option

PRINT = *string tokens*	Controls printed output (xpredicted, xfiltered, deviance, residuals, gain, varpredictions, varfiltered, varresiduals); default *

Parameters

Y = *variates*, *matrices* or *pointers*	Time series data
YTRANSITIONMATRIX = *scalars*, *matrices* or *pointers*	Observation transition matrix, mapping the relationship between the current value of the state vector and the observation
YVCOVARIANCE = *scalars*, *symmetricmatrices* or *pointers*	Observation error covariance matrix
XSTATETRANSITIONMATRIX = *scalars*, *matrices* or *pointers*	State transition matrix, mapping the relationship between the current value of the state vector and its previous value
BXVCOVARIANCE = *scalars*, *matrices* or *pointers*	State noise coefficient matrix

XVCOVARIANCE = *scalars*, *symmetricmatrices* or *pointers*
State error covariance matrix
MEANINITIAL = *scalars*, *variates* or *matrices*
Initial value of the mean of the state vector
VARINITIAL = *scalars* or *symmetricmatrices*
Initial value of the variance-covariance matrix of the state vector
DEVIANCE = *scalars* To save the deviance of the model
XPREDICTED = *matrices* Saves the predicted (a priori) state estimate matrix
XFILTERED = *matrices* Saves the filtered (a posteriori) state estimate matrix
RESIDUALS = *matrices* Saves the matrix of residuals
GAIN = *pointers* Saves the Kalman gain matrix at each iteration
VARPREDICTIONS = *pointers* Saves the variances of the predicted state estimate matrix at each iteration
VARFILTERED = *pointers* Saves the variances of the filtered state estimate matrix at each iteration
VARRESIDUALS= *pointers* Saves the variances of the residuals at each iteration
SAVE = *pointers* Save structure which provides information for use in DKALMAN

KAPLANMEIER procedure

Calculates the Kaplan-Meier estimate of the survivor function (J.T.N.M. Thissen).

Options

PRINT = *string tokens* Whether to print the estimates or to display the Kaplan-Meier estimate in a graph (estimate, mean, quantiles, summary, graph); default esti, grap
GRAPHICS = *string token* Type of graphics to use (lineprinter, highresolution); default high
TITLE = *text* General title for the graph; default *
WINDOW = *scalar* Window number for the high-resolution graph; default 1
KEYWINDOW = *scalar* Window number for the key (zero for no key); default 2
SCREEN = *string token* Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
PROBABILITY = *scalar* Probability level of the confidence interval for the Kaplan-Meier estimates; default 0.95
XLOWER = *scalar* Lower bound for x-axis; default 0
XUPPER = *scalar* Upper bound for x-axis; default * i.e. a value slightly larger than the maximum of the TIME parameter (or EVENT parameter if TIME is not set) is used
PLOT = *string tokens* What additional plotting features to include (referenceline, censored); default * i.e. none
PERCENTILES = *variate* or *scalar* Percentiles at which to estimate quantiles of survival times; default 25,50,75

Parameters

TIME = *variates* Observed timepoints
CENSORED = *variates* Variate specifying whether the corresponding element of TIME is censored (1) or not (0); default is to assume no censoring
GROUPS = *factors* Factor specifying the different groups for which the survivor function is estimated
EVENT = *variates* Saves the distinct TIME values when TIME is set; otherwise supplies an input variate specifying the endpoint of each interval
NDEATH = *variates* Saves the number of deaths at each EVENT when TIME is set; otherwise supplies an input variate specifying the number of deaths in each interval
NATRISK = *variates* Saves the number of units at risk at each EVENT when TIME is set; otherwise supplies an input variate with the number at risk

	in each interval
ESTIMATE = *variates*	Saves the Kaplan-Meier estimates of the survivor function
NEWGROUPS = *factors*	Saves the grouping of the EVENT, NDEATH, NATRISK and ESTIMATE variates when TIME is set

KAPPA procedure

Calculates a kappa coefficient of agreement for nominally scaled data (A.J. Rook).

Option

PRINT = *string token*	Whether to print kappa and its associated information (test); default test

Parameters

DATA = *tables*	Data sets, each consisting of an object × category table whose entries are the number of judges assigning the *i*th object to the *j*th category
STATISTIC = *scalars*	Save the value of kappa for each data table
VARIANCE = *scalars*	Save the corresponding variances

KCONCORDANCE procedure

Calculates Kendall's Coefficient of Concordance; synonym CONCORD (S.J. Welham, N.M. Maclaren & H.R. Simpson).

Options

PRINT = *string tokens*	Output required (test, ranks): test produces the relevant test statistics, ranks produces the vector of mean ranks and the ranks for each sample; default test
GROUPS = *factor*	Defines the variable stored in each unit if only one variate is specified by DATA
STATISTIC = *scalar*	Scalar to save the coefficient of concordance
CHISQUARE = *scalar*	Scalar to save the chi-square approximation to the coefficient (calculated only if the sample size is at least 8)
MEANRANKS = *variate*	Variate to save the mean ranks for individuals over variables
DF = *scalar*	Scalar to save the degrees of freedom for CHISQUARE

Parameters

DATA = *variates*	List of variables to be compared, or a single variate containing the data for all the variables (the GROUPS option must then be set to indicate the variable recorded in each unit belongs)
RANKS = *variates*	Save the ranks of the variables

KCROSSVALIDATION procedure

Computes cross validation statistics for punctual kriging (D.A. Murray & R. Webster).

Options

PRINT = *string tokens*	Controls printed output (statistics, correlation); default stat
PLOT = *string token*	Whether to produce a scatter plot of the predicted against the true values (scatter); default * i.e. none
Y = *variate* or *scalar*	Y positions or interval (not needed for 2D regular data i.e. when DATA is a matrix)
X = *variate*	X positions (needed only for 2D irregular data)
YOUTER = *variate*	Variate containing 2 values to define the Y-bounds of the region to be examined (bottom then top); by default the whole region is used
XOUTER = *variate*	Variate containing 2 values to define the X-bounds of the region to be examined (bottom then top); by default the whole region is used
RADIUS = *scalar*	Maximum distance between target point and usable data
SEARCH = *string token*	Type of search (isotropic, anisotropic); default isot
MINPOINTS = *scalar*	Minimum number of data points from which to compute

elements; default 7

MAXPOINTS = *scalar* — Maximum number of data points from which to compute elements; default 20

DRIFT = *string token* — Amount of drift (constant, linear, quadratic); default cons

YXRATIO = *scalar* — Ratio of Y interval to X interval

SAVE = *pointer* — Pointer containing model estimates saved from MVARIOGRAM

Parameters

DATA = *variates* or *matrices* — Observed measurements as a variate or, for data on a regular grid, as a matrix

ISOTROPY = *string tokens* — Form of variogram (isotropic, Burgess, geometrical); default isot

MODELTYPE = *string tokens* — Model fitted to the variogram (power, boundedlinear, circular, spherical, doublespherical, pentaspherical, exponential, besselk1, gaussian, cubic, stable, cardinalsine, matern); default *

NUGGET = *scalars* — The nugget variance

SILLVARIANCES = *scalars* or *variates* — Sill variances of the spatially dependent component

RANGES = *scalar* or *variates* — Ranges of the spatially dependent component

GRADIENT = *scalars* or *variates* — Slope of the unbounded component

EXPONENT = *scalars* or *variates* — Power of the unbounded component or power for the stable model

SMOOTHNESS = *scalar* — Value of ν parameter for the Matern model

PHI = *scalars* or *variates* — Phi parameters in anisotropic model (ISOTROPY = burg or geom)

RMAX = *scalars* or *variates* — Maximum gradient of an anisotropic model

RMIN = *scalars* or *variates* — Minimum gradient of an anisotropic model

MEASUREMENTERROR = *scalars* — Variance of measurement error

PREDICTIONS = *variates* or *matrices* — Saves the kriged estimates in matrices for 2D Regular data, otherwise in variates

VARIANCES = *variates* or *matrices* — Saves the estimation variances in matrices for 2D Regular data, otherwise in variates

STATISTICS = *variates* — Saves the cross validation statistics

KCSRENVELOPES procedure

Simulates K function bounds under complete spatial randomness (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string tokens* — What to print (summary, monitoring); default summ, moni

Parameters

YPOLYGON = *variates* — Vertical coordinates of each polygon; no default – this parameter must be set

XPOLYGON = *variates* — Horizontal coordinates of each polygon; no default – this parameter must be set

NPOINTS = *scalars* — How many points to generate in each simulation; no default – this parameter must be set

NSIMULATIONS = *scalars* — How many simulations of CSR to use; no default – this parameter must be set

S = *variates* — Vectors of distances to use; no default – this parameter must be set

KLOWER = *variates* — Variates to receive the values of the lower bound of the K function

KUPPER = *variates* — Variates to receive the values of the upper bound of the K function

SEED = *scalars* — Seeds for the random numbers used in the simulations; default 0

KERNELDENSITY procedure

Uses kernel density estimation to estimate the underlying density of a sample (P.W. Goedhart).

Options

PRINT = *string token*	What to print (integral, summary, monitoring, graph); default inte
METHOD = *string token*	Which automatic bandwidth selection method should be used when the BANDWIDTH option is not set (s1, s2, s3, sj); default sj
BANDWIDTH = *scalar* or *variate*	Which bandwidth value or values are to be used; default *
NGRIDEXPONENT2 = *scalar*	Defines the number of grid points as 2**NGRIDEXPONENT2; default 11
SAVEGRIDEXTENT = *scalar*	Defines the lower and upper limit of the interval on which the kernel density is saved; the default value of 4 uses the full interval on which the kernel density is calculated
NFOURIER = *scalar*	Defines the upper limit of the sample size for which the kernel density is calculated directly (when the sample size exceeds the setting of this option, the fast Fourier transform is used to calculate the kernel density); default 100
PROPORTION = *variate*	Proportions at which to calculate quantiles of the kernel density estimate; default !(0.025, 0.25, 0.5, 0.75, 0.975)
WINDOW = *scalar* or *variate*	Window number(s) for the graph(s); default 1
SCREEN = *string token*	Whether to clear the screen before plotting into the first window, or whether to or continue plotting on the old screen (clear, keep); default clea

Parameters

SAMPLE = *variates*	The sample for which to calculate the kernel density estimate
GRID = *variates*	Saves the grid of equidistant points at which the kernel density is calculated
DENSITY = *variates* or *pointers*	Saves the kernel density estimate
CUMULATIVE = *variates* or *pointers*	Saves the estimated cumulative distribution
QUANTILE = *variates* or *pointers*	Saves the quantiles calculated from the estimated cumulative distribution
SAVEBANDWIDTH = *scalars*	Saves the automatically selected bandwidths as specified by the METHOD option

KHAT procedure

Calculates an estimate of the K function (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	What to print (summary); default summ

Parameters

Y = *variates*	Vertical coordinates of each spatial point pattern; no default – this parameter must be set
X = *variates*	Horizontal coordinates of each spatial point pattern; no default – this parameter must be set
YPOLYGON = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
XPOLYGON = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
S = *variates*	Vectors of distances to use; no default – this parameter must be set
KVALUES = *variates*	Variates to receive the estimated values of the K function

KLABENVELOPES procedure

Gives bounds for K function differences under random labelling (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Options

PRINT = *string tokens*	What to print (summary, monitoring); default summ, moni

Parameters

Y1 = *variates*	Vertical coordinates of the first spatial point patterns; no default – this parameter must be set
X1 = *variates*	Horizontal coordinates of the first spatial point patterns; no default – this parameter must be set
Y2 = *variates*	Vertical coordinates of the second spatial point patterns; no default – this parameter must be set
X2 = *variates*	Horizontal coordinates of the second spatial point patterns; no default – this parameter must be set
YPOLYGON = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
XPOLYGON = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
NSIMULATIONS = *scalars*	How many simulations of random labelling to use; no default – this parameter must be set
S = *variates*	Vectors of distances to use; no default – this parameter must be set
KLOWER = *variates*	Variates to receive the values of the lower bound of the difference between the K functions
KUPPER = *variates*	Variates to receive the values of the upper bound of the difference between the K functions
SEED = *scalars*	Seeds for the random numbers used to generate the random labellings; default 0

KNEARESTNEIGHBOURS procedure

Classifies items or predicts their responses by examining their *k* nearest neighbours (R.W. Payne).

Options

PRINT = *string tokens*	Printed output required (neighbours, predictions); default pred
SIMILARITY = *matrix* or *symmetric matrix*	Provides the similarities between the training and prediction sets of items
NEIGHBOURS = *pointer*	Pointer with a variate for each prediction item to save the numbers of its nearest neighbours in the training set
GROUPS = *factor*	Defines groupings to identify the training and prediction sets of items when SIMILARITY is a symmetric matrix
LEVTRAINING = *scalar* or *text*	Identifies the level of GROUPS or dimension of SIMILARITY that represents the training set; default 1
LEVPREDICTION = *scalar* or *text*	Identifies the level of GROUPS or dimension of SIMILARITY that represents the prediction set; default 2
METHOD = *string token*	How to calculate the prediction from a DATA variate (mean, median); default medi
MINSIMILARITY = *scalar*	Cut-off minimum value of the similarity for items to be regarded as neighbours; default 0.75
MINNEIGHBOURS = *scalar*	Minimum number of nearest neighbours to use; default 5
MAXNEIGHBOURS = *scalar*	Maximum number of nearest neighbours to use; default 10
SEED = *scalar*	Seed for the random numbers used to select neighbours when more than MAXNEIGHBOURS are available; default 0

Parameters

DATA = *variates* or *factors*	Data values for the items in the training set
PREDICTIONS = *variates* or *factors*	Saves the predictions

KOLMOG2 procedure

Performs a Kolmogorov-Smirnoff two-sample test (S.J. Welham, N.M. Maclaren & H.R. Simpson).

Options

PRINT = *string tokens*	Output required (test, differences, ranks): test gives the test statistic, differences gives signed differences, and ranks produces the ranks for each sample; default test
GROUPS = *factor*	Defines the groups for a two-sample test if only the Y1 parameter is specified

Parameters

Y1 = *variates*	Identifier of the variate holding the first sample
Y2 = *variates*	Identifier of the variate holding the second sample
R1 = *variates*	Saves the ranks of the first sample
R2 = *variates*	Saves the ranks of the second sample
STATISTIC = *scalars*	Scalar to save the test statistic (the maximum absolute difference between the cumulative distribution functions)
CHISQUARE = *scalars*	Scalar to save the chi-square approximation to the test statistic
DIFFERENCES = *variates*	Variate to save the signed differences between the cumulative distribution functions

KRIGE directive

Calculates kriged estimates using a model fitted to the sample variogram.

Options

PRINT = *string token*	Controls printed output (description, search, weights, monitor, data); default desc
Y = *variate*	Y positions (not needed for 2-dimensional regular data i.e. when DATA is a matrix)
X = *variate*	X positions (needed only for 2-dimensional irregular data)
YOUTER = *variate*	Variate containing 2 values to define the Y-bounds of the region to be examined (bottom then top); by default the whole region is used
XOUTER = *variate*	Variate containing 2 values to define the X-bounds of the region to be examined (left then right); by default the whole region is used
YINNER = *variate*	Variate containing 2 values to define the Y-bounds of the interpolated region (bottom then top); no default
XINNER = *variate*	Variate containing 2 values to define the X-bounds of the interpolated region (left then right); no default
BLOCK = *variate*	Dimensions (length and height) of block; default !(0, 0) i.e. punctual kriging
RADIUS = *scalar*	Maximum distance between target point in block and usable data
SEARCH = *string token*	Type of search (isotropic, anisotropic); default isot
MINPOINTS = *scalar*	Minimum number of data points from which to compute elements; default 7
MAXPOINTS = *scalar*	Maximum number of data points from which to compute elements (2 < MINPOINTS ≤ MAXPOINTS < 41); default 20
NSTEP = *scalar*	Number of steps for numerical integration; (3 < NSTEP < 11); default 8
DRIFT = *string token*	Amount of drift (constant, linear, quadratic); default cons
YXRATIO = *scalar*	Ratio of Y interval to X interval; default 1.0
INTERVAL = *scalar*	Distance between successive interpolations; default 1.0

Parameters

DATA = *variates* or *matrices*	Observed measurements as a variate or, for data on a regular grid, as a matrix

ISOTROPY = *string tokens*	Form of variogram (isotropic, Burgess, geometrical); default isot
MODELTYPE = *string tokens*	Model fitted to the variogram (power, boundedlinear, circular, spherical, doublespherical, pentaspherical, exponential, besselk1, gaussian, cubic, stable, cardinalsine, matern); default powe
NUGGET = *scalars*	The nugget variance
SILLVARIANCES = *variates*	Sill variances of the spatially dependent component; default none
RANGES = *variates*	Ranges of the spatially dependent component; default none
GRADIENT = *variates*	Slope of the unbounded component; default none
EXPONENT = *variates*	Power of the unbounded component or power for the stable model; default none
SMOOTHNESS = *scalar*	Value of ν parameter for the Matern model; defalt none
PHI = *variates*	Phi parameters of an anistropic model (ISOTROPY = Burg or geom)
RMAX = *variates*	Maximum gradient or distance parameter of an anistropic model
RMIN = *variates*	Minimum gradient or distance parameter of an anistropic model
PREDICTIONS = *matrices*	Kriged estimates
VARIANCES = *matrices*	Estimation variances
LAGRANGEMULTIPLIER = *matrices* or *pointers*	Saves the Lagrange multipliers from each kriging solution
MEASUREMENTERROR = *scalar*	Specifies the variance of the measurement error
SAVE = *pointers*	Supplies the model name and estimates, as saved from MVARIOGRAM

KRUSKAL procedure

Carries out a Kruskal-Wallis one-way analysis of variance (S.J. Welham, N.M. Maclaren & H.R. Simpson).

Options

PRINT = *string tokens*	Output required (test, ranks): test produces the relevant test statistics, ranks produces a vector of ranks for each sample relative to the whole data set; default test
GROUPS = *factor*	Defines the sample membership if only one variate is specified by DATA
STATISTIC = *scalar*	Scalar to save the Kruskal-Wallis test statistic
MEANRANKS = *variate*	Variate to save the mean ranks of the samples
DF = *scalar*	Scalar to save the degrees of freedom for the statistic

Parameters

DATA = *variates*	List of variates containing the data for each sample, or a single variate containing the data from all the samples (the GROUPS option must then be set to indicate the sample to which each unit belongs)
RANKS = *variates*	Allow the ranks to be saved (relative to the combined data)

KSED procedure

Calculates the standard error for K function differences under random labelling (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	Controls printed output (summary); default summ

Parameters

Y1 = *variates*	Vertical coordinates of the first spatial point patterns; no default – this parameter must be set
X1 = *variates*	Horizontal coordinates of the first spatial point patterns; no

	default – this parameter must be set
`Y2` = *variates*	Vertical coordinates of the second spatial point patterns; no default – this parameter must be set
`X2` = *variates*	Horizontal coordinates of the second spatial point patterns; no default – this parameter must be set
`YPOLYGON` = *variates*	Vertical coordinates of the polygons; no default – this parameter must be set
`XPOLYGON` = *variates*	Horizontal coordinates of the polygons; no default – this parameter must be set
`S` = *variates*	Vectors of distances to use; no default – this parameter must be set
`KSED` = *variates*	Variates to receive the values of the standard error of the difference between the K functions for the two patterns under random labelling
`VCOVARIANCE` = *symmetric matrices*	Saves the variance-covariance matrix
`VK1` = *variates*	Saves the variance of Khat for first spatial point pattern
`VK2` = *variates*	Saves the variance of Khat for second spatial point pattern
`VK12` = *variates*	Saves the covariance of Khat for the two samples

`KSTHAT` procedure

Calculates an estimate of the K function in space, time and space-time (D.A. Murray, P.J. Diggle & B.S. Rowlingson).

Option

`PRINT` = *string token*	Controls printed output (`summary`); default summ

Parameters

`Y` = *variates*	Vertical coordinates of the spatial point patterns; no default – this parameter must be set
`X` = *variates*	Horizontal coordinates of the spatial point patterns; no default – this parameter must be set
`TIMES` = *variates*	Times for each event
`YPOLYGON` = *variates*	Vertical coordinates of the polygons; no default – this parameter must be set
`XPOLYGON` = *variates*	Horizontal coordinates of the polygons; no default – this parameter must be set
`S` = *variates*	Vectors of distances to use; no default – this parameter must be set
`TVALUES` = *variates*	Time scales for the analysis
`TLOWER` = *variates*	Lower temporal domain
`TUPPER` = *variates*	Upper temporal domain
`KS` = *variates*	Saves the spatial K function estimates
`KT` = *variates*	Saves the spatial K function estimates
`KST` = *variates*	Saves the space-time K function estimates

`KSTMCTEST` procedure

Performs a Monte-Carlo test for space-time interaction (D.A. Murray, P.J. Diggle & B.S. Rowlingson).

Options

`PRINT` = *string token*	Controls printed output (`statistic`, `rank`); default `stat`, `rank`
`PLOT` = *string token*	Whether to produce a plot of the test statistic (`histogram`); default `hist`
`NTIMES` = *scalar*	Number of simulations for Monte-Carlo test; default 49
`SEED` = *scalar*	Seed for random number generator; default 0 continues from previous generation or uses system clock

Parameters

`Y` = *variates*	Vertical coordinates of the first spatial point patterns; no

default – this parameter must be set

`X` = *variates* Horizontal coordinates of the first spatial point patterns; no default – this parameter must be set

`TIMES` = *variates* Times for each event

`YPOLYGON` = *variates* Vertical coordinates of the polygons; no default – this parameter must be set

`XPOLYGON` = *variates* Horizontal coordinates of the polygons; no default – this parameter must be set

`S` = *variates* Vectors of distances to use; no default – this parameter must be set

`TVALUES` = *variates* Time scales for the analysis

`TLOWER` = *variates* Lower temporal domain

`TUPPER` = *variates* Upper temporal domain

`STATISTIC` = *scalars* Saves the Monte-Carlo statistic

`KSTSE` procedure

Calculates the standard error for the space-time K function (D.A. Murray, P.J. Diggle & B.S. Rowlingson).

Option

`PRINT` = *string token* Controls printed output (`summary`); default summ

Parameters

`Y` = *variates* Vertical coordinates of the spatial point patterns; no default – this parameter must be set

`X` = *variates* Horizontal coordinates of the spatial point patterns; no default – this parameter must be set

`TIMES` = *variates* Times for each event

`YPOLYGON` = *variates* Vertical coordinates of the polygons; no default – this parameter must be set

`XPOLYGON` = *variates* Horizontal coordinates of the polygons; no default – this parameter must be set

`S` = *variates* Vectors of distances to use; no default – this parameter must be set

`TVALUES` = *variates* Time scales for the analysis

`TLOWER` = *variates* Lower temporal domain

`TUPPER` = *variates* Upper temporal domain

`SE` = *variates* Saves the standard errors

`KTAU` procedure

Calculates Kendall's rank correlation coefficient τ (R.W. Payne & D.B. Baird).

Options

`PRINT` = *string tokens* Output required (`correlations`, `probabilities`); default `corr`, `prob`

`GROUPS` = factor Defines the sample membership if only one variate is specified by `DATA`

`CORRELATIONS` = *scalar* or *symmetric matrix* Scalar to save the rank correlation coefficient if there are two samples, or symmetric matrix to save the coefficients between all pairs of samples if there are several

`PROBABILITIES` = *scalar* or *symmetric matrix* Scalar to save the probability for the correlation coefficient if there are two samples, or symmetric matrix to save the probabilities for all pairs of samples if there are several

`NORMAL` = *scalar* or *symmetric matrix* Scalar to save a transformation of tau that approximately follows a Normal distribution with mean zero and variance if there are two samples, or symmetric matrix to save the transformation for all pairs of samples if there are several

Parameter

DATA = *variates*	List of variates containing the data for each sample, or a single variate containing the data from all the samples (the GROUPS option must then be set to indicate the sample to which each unit belongs)

KTORENVELOPES procedure

Gives bounds for the bivariate K function under independence (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string tokens*	What to print (summary, monitoring); default summ, moni

Parameters

Y1 = *variates*	Vertical coordinates of the first spatial point patterns; no default – this parameter must be set
X1 = *variates*	Horizontal coordinates of the first spatial point patterns; no default – this parameter must be set
Y2 = *variates*	Vertical coordinates of the second spatial point patterns; no default – this parameter must be set
X2 = *variates*	Horizontal coordinates of the second spatial point patterns; no default – this parameter must be set
YPOLYGON = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
XPOLYGON = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
NSIMULATIONS = *scalars*	How many simulations of independence to use; no default – this parameter must be set
S = *variates*	Vectors of distances to use; no default – this parameter must be set
KLOWER = *variates*	Variates to receive the values of the lower bound of the bivariate K function
KUPPER = *variates*	Variates to receive the values of the upper bound of the bivariate K function
SEED = *scalars*	Seeds for the random numbers used to generate the random shifts; default 0

K12HAT procedure

Calculates an estimate of the bivariate K function (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	What to print (summary); default summ

Parameters

Y1 = *variates*	Vertical coordinates of the first spatial point patterns; no default – this parameter must be set
X1 = *variates*	Horizontal coordinates of the first spatial point patterns; no default – this parameter must be set
Y2 = *variates*	Vertical coordinates of the second spatial point patterns; no default – this parameter must be set
X2 = *variates*	Horizontal coordinates of the second spatial point patterns; no default – this parameter must be set
YPOLYGON = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
XPOLYGON = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
S = *variates*	Vectors of distances to use; no default – this parameter must be set

KVALUES = *variates*	Variates to receive the estimated values of the bivariate K functions

LCONCORDANCE procedure

Calculates Lin's concordance correlation coefficient (R.W. Payne & M.S. Dhanoa).

Options

PRINT = *string token*	Controls printed output (concordance); default conc
GROUPS = *factor*	Defines the sets of measurements when they are all supplied in a single DATA variate
CONCORDANCE = *scalar* or *variate*	Saves Lin's the concordance coefficient
LOWER = *scalar* or *variate*	Saves the lower confidence limit for the coefficient
UPPER = *scalar* or *variate*	Saves the upper confidence limit for the coefficient
CORRELATION = *scalar* or *variate*	Saves the correlation coefficient
CB = *scalar* or *variate*	Saves the bias correction factor
ZTRANSFORMATION = *scalar* or *variate*	Saves the Z transformation of the coefficient
ZSD = *scalar* or *variate*	Saves the standard deviation of the Z transformation
CIPROBABILITY = *scalar*	Defines the size of the confidence interval; default 0.95 i.e. 95%
REFERENCELEVEL = *scalar* or *text*	Defines the set of measurements to be used as the control if there are more than two variates or groups; default 1

Parameter

DATA = *variates*	List of variates specifying the sets of measurements to be compared, or a single variate containing all the measurements (the GROUPS option must then be set to indicate the set to which each unit belongs)

LIBEXAMPLE procedure

Accesses examples and source code of library procedures (R.W. Payne).

No options

Parameters

PROCEDURE = *texts*	Single-valued texts indicating the procedures about which the information is required
EXAMPLE = *texts*	Identifiers of text structures to store the example for each procedure
SOURCE = *texts*	Identifiers of text structures to store the source code of each procedure

LIBFILENAME procedure

Supplies the names of information files for library procedures (R.W. Payne).

No options

Parameters

FILENAME = *texts*	Text in which to store the name of the backing-store file containing the required information
CONTENTS = *string tokens*	Indicates which file is required (procedures, adesign, afraction, acyclic, agenerator); default proc

LIBHELP procedure

Provides help information about library procedures (R.W. Payne).

No options

Parameter

PROCEDURE = *texts*	Single-valued texts indicating the procedures about which the information is required; if this is not set, information is given about LIBHELP itself

LIBSOURCE procedure

Obtains the source code of a Genstat procedure, PC Windows only (R.W. Payne).

No options

Parameters

PROCEDURE = *texts*	Procedure names
SOURCE = *texts*	Texts to store the source code of each procedure
STATEMENT = *texts*	Saves a command to obtain the source of each procedure (useful if the name has been specified in response to questions from PROCEDURE)

LIBVERSION procedure

Provides the name of the current Genstat Procedure Library (R.W. Payne).

Option

PRINT = *string token*	Controls printed output (release); default rele

Parameter

RELEASENAME = *text*	Text in which to store the name of the currently available release of the Genstat 5 Procedure Library

LINDEPENDENCE procedure

Finds the linear relations associated with matrix singularities (J.H. Maindonald).

Option

PRINT = *string tokens*	Printed output (dependent, coefficients); default depe

Parameters

DATA = *symmetric matrices*	Specifies the positive semi-definite matrix for which the information is required
COEFFICIENTS = *matrices*	Stores the coefficients of the linear dependencies

LIST directive

Lists details of the data structures currently available within Genstat.

Options

PRINT = *string tokens*	What to print (identifier, attributes); default iden, attr
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
SYSTEM = *string token*	Whether to include "system" structures with prefix _ (yes, no); default no
SCOPE = *string token*	When used within a procedure, this allows the listing of structures in the program that called the procedure (SCOPE=external), or in the main program itself (SCOPE=global), rather than those within the procedure (local, external, global); default loca
NSTRUCTURES = *scalar*	Saves the number of structures of the requested types
SAVE = *pointer*	Saves a pointer containing the structures of the requested types

Parameter

strings	Types of structure to list (all, diagonal, dummy, expression, factor, formula, lrv, matrix, pointer, scalar, sspm, symmetric, table, text, tsm, variate); default all

LORENZ procedure

Plots the Lorenz curve and calculates the Gini and asymmetry coefficients (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (gini, lorenz, asymmetry); default gini, lore, asym
PLOT = *string token*	Controls graphical output (curve); default curv
TITLE = *string*	Title for the graph; default uses the identifier of the DATA

variate

NBOOT = *scalar* — Number of samples to make to construct the bootstrap confidence intervals; default 100

SEED = *scalar* — Seed for the random number generator used to construct the bootstrap samples; default 0 i.e. continue an existing sequence of random numbers or, if none, initialize the generator automatically

CIPROBABILITY = *scalar* — Probability for the bootstrap confidence interval; default 0.95

Parameters

DATA = *variates* — Specifies sets of data values

GINI = *scalars* — Saves the Gini coefficient for each DATA variate

ASYMMETRY = *scalars* — Saves the asymmetry coefficient for each DATA variate

LPCONTOUR directive

Produces contour maps of two-way arrays of numbers using character (i.e. line-printer) graphics.

Options

CHANNEL = *scalar* — Channel number of output file; default is current output file

INTERVAL = *scalar* — Contour interval for scaling; default * i.e. determined automatically

TITLE = *text* — General title; default *

YTITLE = *text* — Title for y-axis; default *

XTITLE = *text* — Title for x-axis; default *

YLOWER = *scalar* — Lower bound for y-axis; default 0

YUPPER = *scalar* — Upper bound for y-axis; default 1

XLOWER = *scalar* — Lower bound for x-axis; default 0

XUPPER = *scalar* — Upper bound for x-axis; default 1

YINTEGER = *string token* — Whether y-labels integral (yes, no); default no

XINTEGER = *string token* — Whether x-labels integral (yes, no); default no

LOWERCUTOFF = *scalar* — Lower cut-off for array values; default *

UPPERCUTOFF = *scalar* — Upper cut-off for array values; default *

Parameters

GRID = *identifiers* — Pointers (of variates representing the columns of a data matrix), matrices or two-way tables specifying values on a regular grid

DESCRIPTION = *texts* — Annotation for key

LPGRAPH directive

Produces point and line graphs using character (i.e. line-printer) graphics.

Options

CHANNEL = *scalar* — Channel number of output file; default is current output file

TITLE = *text* — General title; default *

YTITLE = *text* — Title for y-axis; default *

XTITLE = *text* — Title for x-axis; default *

YLOWER = *scalar* — Lower bound for y-axis; default *

YUPPER = *scalar* — Upper bound for y-axis; default *

XLOWER = *scalar* — Lower bound for x-axis; default *

XUPPER = *scalar* — Upper bound for x-axis; default *

MULTIPLE = *variate* — Numbers of plots per frame; default * i.e. all plots are on a single frame

JOIN = *string token* — Order in which to join points (ascending, given); default asce

EQUAL = *string tokens* — Whether/how to make bounds equal (no, scale, lower, upper); default no

NROWS = *scalar* — Number of rows in the frame; default * i.e. determined automatically

NCOLUMNS = *scalar* — Number of columns in the frame; default * i.e. determined

	automatically
YINTEGER = *string token*	Whether y-labels integral (yes, no); default no
XINTEGER = *string token*	Whether x-labels integral (yes, no); default no
Parameters	
Y = *identifiers*	Y-coordinates
X = *identifiers*	X-coordinates
METHOD = *string tokens*	Type of each graph (point, line, curve, text); if unspecified, poin is assumed
SYMBOLS = *factors* or *texts*	For factor SYMBOLS, the labels (if defined), or else the levels, define plotting symbols for each unit, whereas a text defines textual information to be placed within the frame for METHOD=text or the symbol to be used for each plot for other METHOD settings; if unspecified, * is used for points, with integers 1-9 to indicate coincident points, ' and . are used for lines and curves
DESCRIPTION = *texts*	Annotation for key

LPHISTOGRAM directive

Produces histograms using character (i.e. line-printer) graphics.

Options	
CHANNEL = *scalar*	Channel number of output file; default is the current output file
TITLE = *text*	General title; default *
LIMITS = *variate*	Variate of group limits for classifying variates into groups; default *
NGROUPS = *scalar*	When LIMITS is not specified, this defines the number of groups into which a data variate is to be classified; default is the integer value nearest to the square root of the number of values in the variate
LABELS = *text*	Group labels
SCALE = *scalar*	Number of units represented by each character; default 1
Parameters	
DATA = *identifiers*	Data for the histograms; these can be either a factor indicating the group to which each unit belongs, a variate whose values are to be grouped, or a one-way table giving the number of units in each group
NOBSERVATIONS = *tables*	One-way table to save numbers in the groups
GROUPS = *factors*	Factor to save groups defined from a variate
SYMBOLS = *texts*	Characters to be used to represent the bars of each histogram
DESCRIPTION = *texts*	Annotation for key

LRIDGE procedure

Does logistic ridge regression (A.I. Glaser).

Options	
PRINT = *string token*	What output to print (correlation, crossvalidation, ridge, scaledridge, standarderrors); default corr
PLOT = *string tokens*	What graphs to plot (correlation, ridgetrace, buildup); default * i.e. none
LINK = *string token*	Link function (logit, probit, complementaryloglog); default logi
DISPERSION = *scalar*	Value of the the dispersion parameter; default 1
TERMS = *formula*	Explanatory model
FACTORIAL = *scalar*	Limit on number of factors/covariates in a model term; default 3
LAMBDA = *variate* or *scalar*	Values for the ridge parameter lambda
CROSSVALIDATION = *string token*	Whether to use cross-validation to find an optimal value of lambda (yes, no); default no

NCROSSVALIDATIONGROUPS = *scalar*	Number of groups for cross-validation; default 10
CVMETHOD = *string token*	Which method to use for cross-validation (deviance, squarederror, countingerror); default devi
SEED = *scalar*	Seed for random numbers to use in cross-validation; default 0

Parameters

Y = *variates*	Response variate
NBINOMIAL = *scalars* or *variates*	Number of binomial trials for each unit; default 1
YVALIDATION = *variates*	Response variate for validation
XVALIDATION = *pointers*	Explanatory variables for validation
XDATA = *pointers*	Pointer containing the original explanatory variables in the same order as in XVALIDATION; default takes the variables in the order in which they occur in TERMS
NVALIDATION = *variates* or *scalars*	Number of binomial trials for the units of each YVALIDATION variate; default 1
BESTLAMBDA = *scalars*	Saves the optimal lambda value from cross-validation
CVSTATISTICS = *matrices*	Saves the cross-validation statistics
RESIDUALS = *variates*	Saves residuals when LAMBDA is a scalar
FITTEDVALUES = *variates*	Saves fitted values when LAMBDA is a scalar
ESTIMATES = *variates*	Saves parameter estimates when LAMBDA is a scalar
SE = *variates*	Saves standard errors of the parameter estimates when LAMBDA is a scalar
DEVIANCE = *scalars*	Saves the residual deviance when LAMBDA is a scalar
LINEARPREDICTOR = *variates*	Saves the linear predictor when LAMBDA is a scalar

LRV directive

Declares one or more LRV data structures.

Options

ROWS = *scalar*, *vector* or *pointer*	Number of rows, or row labels, for the matrix; default *
COLUMNS = *scalar*, *vector* or *pointer*	Number of columns, or column labels, for matrix and diagonal matrix; default *

Parameters

IDENTIFIER = *identifiers*	Identifiers of the LRVs
VECTORS = *matrices*	Matrix to contain the latent vectors for each LRV
ROOTS = *diagonal matrices*	Diagonal matrix to contain the latent roots for each LRV
TRACE = *scalars*	Trace of the matrix

LRVSCREE procedure

Prints a scree diagram and/or a difference table of latent roots (P.G.N. Digby).

Option

PRINT = *string tokens*	Printed output (scree, differences); default scre
PLOT = *string token*	What to plot in high-resolution graphics (scree); default scre
TITLE = *text*	Title for the graph; default * i.e. none
WINDOW = *scalar*	Window to use for the graph; default 1

Parameters

ROOTS = *LRVs* or *any numerical structures*	Latent roots to be displayed; if an LRV is supplied the trace will also be extracted from it
TRACE = *scalars*	Supplies or saves the total of the latent roots
DIFFERENCES = *pointers*	Contains 3 variates to save the difference table

LSIPLOT procedure

Plots least significant intervals, saved from SEDLSI (M.C. Hannah).

Options

WINDOW = *scalar*	Window in which to plot the graph
TITLE = *text*	Title for the graph; default 'Estimates with LSIs by

	`Treatment'`
`YTITLE` = *text*	Title for the y-axis; default `'Estimates'`

Parameters

`LSI` = *pointers*	Defines the least significant intervals
†`SYMBOL` = *texts* or *scalars*	Symbol to use to plot each set of estimates
†`CSYMBOL` = *texts* or *scalars*	Colour for each symbol
†`SMSYMBOL` = *scalars*	Multiplier to use in the calculation of the size of each symbol
†`SMLABEL` = *scalars*	Multiplier to use in the calculation of the size of the labels in each plot

`LSPLINE` procedure

Calculates design matrices to fit a natural polynomial or trigonometric L-spline as a linear mixed model (S.J. Welham).

Options

`KMETHOD` = *string token*	Method for constructing the set of knots (`equal`, `quantile`, `given`); default equa
`NSEGMENTS` = *scalar*	Specifies the number of segments between boundaries; default `*` obtains a value automatically
`INKNOTS` = *variate*	Provides the set of internal knots when `KMETHOD=given`
`CORE` = *string token*	The form of core function to use; (`cossin`, `intcossin`, `lincossin`, `intercept`, `linear`, `quadratic`) default `linc`
`PERIOD` = *scalar*	Defines the period for trigonometric functions (not required for polynomial splines)
`LOWER` = *scalar*	Specifies the lower boundary when `KMETHOD=equal`; default takes the minimum value in `X`
`UPPER` = *scalar*	Specifies the upper boundary when `KMETHOD=equal`; default takes the maximum value in `X`
`ORTHOGONALIZETO` = *variate*	Variate to use to get an orthogonalized basis; default `*` i.e. orthogonalization with respect to `X`
`SCALING` = *scalar*	Scaling of the `XRANDOM` terms (`automatic`, `none`); default `auto`

Parameters

`X` = *variates*	The explanatory variate for which the spline values are required
`XFIXED` = *matrices*	Saves the design matrix to define the fixed terms (excluding the constant) for fitting the L-spline
`XRANDOM` = *matrices*	Saves the design matrix to define the random terms for fitting the L-spline
`KNOTS` = *variates*	Saves the internal knots and boundaries used to form the basis for the spline
`PX` = *variates*	Specifies x-values at which predictions are required
`PFIXED` = *matrices*	Saves the design matrix for the fixed terms (excluding the constant) for the spline at the prediction points
`PRANDOM` = *matrices*	Saves the design matrix for the random terms for the spline at the prediction points

`LVARMODEL` procedure

Analyses a field trial using the Linear Variance Neighbour model (D.B. Baird).

Options

`PRINT` = *string tokens*	Controls printed output (`data`, `effects`, `sed`, `residuals`, `variances`); default `effe`, `sed`, `vari`
`METHOD` = *string token*	Indicates which version of the LV model to use (`full`, `reduced`); default `full`
`LAMBDA` = *scalar*	Number between 0 and 1 which defines the value for the variance parameter λ (if `METHOD=full` and `LAMBDA=0`, the value is estimated by REML); default 0

VARMETHOD = *string token*	Specifies which estimator of residual variance to use to calculate the sed's of treatment effects (RMS2, GLS) default RMS2
TOLERANCE = *scalar*	Defines the precision to which the variance parameter λ should be estimated; default 0.01

Parameters

Y = *variates*	Y-values (usually plot yields) row by row
TREATMENTS = *factors*	Plot treatments for each y-variate
BLOCKS = *factors*	Block factor, defining groups of plots to be de-trended independently
UNITS = *factors*	Unit-within-block factor, defining the order of plots within each block
EFFECTS = *tables*	To save the estimated treatment effects from each analysis
SED = *matrices* or *symmetric matrices*	To save the estimated standard errors of differences between treatments
WNOISE = *variates*	To save the estimated white noise component
TREND = *variates*	To save the estimated trend component
COMPONENTS = *variates*	To save the estimated variance components: the tuning parameter λ, and either the variance of the random walk innovations ($\lambda<0.9$) or the white noise variance ($\lambda\geq0.9$)

MAANOVA procedure

Does analysis of variance for a single-channel microarray design (R.W. Payne & D.B. Baird).

Options

PRINT = *string tokens*	Controls printed output (summary, monitoring); default * i.e. none
TREATMENTSTRUCTURE = *formula*	Treatment formula for the analysis; if this is not set, the default is taken from the setting (which must already have been defined) of the TREATMENTSTRUCTURE directive
BLOCKSTRUCTURE = *formula*	Block formula for the analysis; if this is not set, the default is taken from any existing setting specified by the BLOCKSTRUCTURE directive and if neither has been set the design is assumed to be unstratified (i.e. to have a single error term)
COVARIATE = *variates*	Defines any covariates
FACTORIAL = *scalar*	Limit on the number of factors in a treatment term
SAVETERMS = *formula*	Treatment terms for which to save information; if this is not set, information is saved for all the treatment terms
REPLICATION = *pointer*	Pointer to tables saving the replication of the SAVETERMS
SPREADSHEET = *string tokens*	What results to save in spreadsheets (aov, means, vcmeans, effects, vareffects, seeffects, teffects, preffects, contrasts, secontrasts, tcontrasts, prcontrasts); default * i.e. none
CONTRASTSLIMIT = *scalar*	Limit on the order of a contrast of a treatment term; default 4
DEVIATIONSLIMIT = *scalar*	Limit on the number of factors in a treatment term for the deviations from its fitted contrasts to be retained in the model; default 9

Parameters

Y = *variates* or *pointers*	Y-variates for each analysis
PROBES = *factors* or *texts*	Defines the probe information for each analysis
SLIDES = *factors* or *texts*	Defines the slide information for each analysis
CHECK = *texts* or *variates*	Slide ID's that can be compared with the labels or levels of the SLIDES factor to ensure that the slide order is correct in each analysis
IDS = *texts*	Saves the probes names that have been generated to label the rows of the output structures from each analysis
RESIDUALS = *matrices*	Saves the residuals

FITTEDVALUES = *matrices*	Saves the fitted values
MEANS = *pointers*	Pointer to a matrix for each of the SAVETERMS, saving the means from each analysis
VCMEANS = *pointers*	Pointer to matrices saving variances and covariances for the means
EFFECTS = *pointers*	Pointer to matrices saving effects
VAREFFECTS = *pointers*	Pointer to variates saving unit variances for effects
SEEFFECTS = *pointers*	Pointer to variates saving standard errors of effects
TEFFECTS = *pointers*	Pointer to variates saving t-statistics of effects
PREFFECTS = *pointers*	Pointer to variates saving probabilities for t -statistics of effects
DF = *pointers*	Pointer to variates saving degrees of freedom
SS = *pointers*	Pointer to variates saving sums of squares
MS = *pointers*	Pointer to variates saving mean squares
RDF = *pointers*	Pointer to variates saving degrees of freedom for the residual corresponding to each of the SAVETERMS
RSS = *pointers*	Pointer to variates saving residual sums of squares
RMS = *pointers*	Pointer to variates saving residual mean squares
VR = *pointers*	Pointer to variates saving variance ratios
PRVR = *pointers*	Pointer to variates saving probabilities for the variance ratios
CONTRASTS = *pointers*	Pointer to matrices saving estimates of contrasts
SECONTRASTS = *pointers*	Pointer to matrices saving standard errors of contrasts
TCONTRASTS = *pointers*	Pointer to matrices saving t-statistics for contrasts
PRCONTRASTS = *pointers*	Pointer to matrices saving probabilities for t-statistics of contrasts

MABGCORRECT procedure

Performs background correction of Affymetrix slides (D.B. Baird).

Options

PRINT = *string token*	What to print (quantiles); default quan
METHOD = *string token*	Method of establishing grid background (mean, quantile); default mean
WEIGHTING = *string token*	Weighting method to use (affymetrix, distance); default affy
POWER = *scalar*	Power applied to distance; default 2 i.e. square
SMOOTH = *scalar*	Smoothing parameter applied to weights; default 100

Parameters

DATA = *variates* or *pointers*	Data values
SLIDES = *factors* or *texts*	Defines the slides
ROWS = *factors*	Defines the rows within each slide
COLUMNS = *factors*	Defines the columns within each slide
NEWDATA = *variates* or *pointers*	Saves the corrected values; if unset, they replace the original DATA values

MACALCULATE procedure

Corrects and transforms two-colour microarray differential expressions (D.B. Baird).

Options

PRINT = *string token*	What to print (summary); default summ
BMETHOD = *string token*	How to correct for spot foreground for background values (subtract, smooth, none); default subtracts REDBACKGROUND and GREENBACKGROUND if set
TRANSFORMATION = *string token*	Type of transformation to apply to the red/green ratios (log, glog); default log
MINIMUM = *scalar*	Minimum value per channel; if RSDBACKGROUND and GSDBACKGROUND are supplied, this is the multiplier of these per spot, default 0

PERSPOTMINIMUM = *string token* — Use a single minimum value per spot rather than per slide (yes, no); default no
CONSTANTVALUE = *scalar* — Constant to add to red and green foreground values; default 0
DF = *scalar* — Degrees of freedom to use for loess smoothing of background; default 20

Parameters

RFOREGROUND = *variates* or *pointers* — Red foreground values per spot
GFOREGROUND = *variates* or *pointers* — Green foreground values per spot
RBACKGROUND = *variates* or *pointers* — Red background values per spot
GBACKGROUND = *variates* or *pointers* — Green background values per spot
RSDBACKGROUND = *variates* or *pointers* — Standard deviation of red background
GSDBACKGROUND = *variates* or *pointers* — Standard deviation of green background
SLIDES = *factors* or *texts* — Defines the slide to which each spot belongs for smoothing, or per slide minima
ROWS = *factors* — Defines the row position of each spot for background smoothing
COLUMNS = *factors* — Defines the column position of each spot for background smoothing
LOGRATIOS = *variates* or *pointers* — Saves the differential expression per spot
INTENSITIES = *variates* or *pointers* — Saves the intensity of each spot
RCORRECTED = *variates* or *pointers* — Saves the corrected red values per spot
GCORRECTED = *variates* or *pointers* — Saves the corrected green values per spot

MADESIGN procedure

Assesses the efficiency of a two-colour microarray design (D.B. Baird).

Options

PRINT = *string tokens* — What to print (design, sed, secontrasts, vcovariance, summary); default desi, sed, seco, vcov, summ
DYEBIASMETHOD = *string token* — Whether to estimate dye bias effects (estimate, omit); default esti
SPREADSHEET = *string tokens* — What results to put in spreadsheets (sed, secontrasts, vcovariance); default sed, seco

Parameters

RED = *factors* — Targets on red dye
GREEN = *factors* — Targets on green dye
XCONTRASTS = *matrices* — Contrasts to estimate
SED = *symmetric matrices* — Saves standard errors of differences
VCOVARIANCE = *symmetric matrices* — Saves variance and covariances of treatments
SECONTRASTS = *symmetric matrices* — Saves standard errors of contrasts specified in XCONTRASTS

MAEBAYES procedure

Modifies t-values by an empirical Bayes method (D.B. Baird).

Options

PRINT = *string tokens* — What to print (estimates); default esti
PLOT = *string tokens* — What to plot (phistograms, thistograms, pvalues, tvalues); default * i.e. nothing
DATATYPE = *string token* — Type of data specified by the DATA parameter when it is a variate (means, tvalues); default tval
METHOD = *string token* — Type of test to use to form probability values (twosided, greaterthan, lessthan); default twos
DEVICE = *scalar* — Device number on which to plot the graphs
GRAPHICSFILE = *text* — What graphics filename template to use to save the graphs; default *

Parameters

DATA = *pointers* or *variates* — Pointers of variates or variates of means or t-values to be summarized

SD = *variates*	Supplies standard deviations of the data when DATA is a variate of means or t-values
DF = *variates*	or scalars Supplies degrees of freedom when DATA is a variate of means or t-values
SD0 = *scalars*	Saves the estimated prior standard deviation
DF0 = *scalars*	Saves the estimated number of degrees of freedom assigned to the prior standard deviation
TMODIFIED = *variates*	Saves the modified t-values
SDMODIFIED = *variates*	Saves the shrunken SD values
PMODIFIED = *variates*	Saves the modified probability values

MAESTIMATE procedure

Estimates treatment effects from a two-colour microarray design (D.B. Baird).

Options

PRINT = *string tokens*	What to print (design, summary, monitoring); default desi, summ, moni
DYEBIASMETHOD = *string token*	Whether to estimate dye bias effects (estimate, omit); default esti
SPREADSHEET = *string tokens*	What results to put in spreadsheets (estimates, df, rsd, dyebias, seestimates, tvalues, probabilities, contrasts, secontrasts, tcontrasts, prcontrasts); default esti, df, rsd, dyeb, sees, tval, prob, cont, seco, tcon, prco

Parameters

LOGRATIOS = *variates* or *pointers*	Log-ratios
PROBES = *factors* or *texts*	Probes for the log-ratios
SLIDES = *factors* or *texts*	Slides for the log-ratios
REDTREATMENTS = *factors*	Targets on red dye for slides
GREENTREATMENTS = *factors*	Targets on green dye for slides
CHECK = *texts* or *variates*	Slide ID's of the red and green treatments for a check matching the slide order with the labels or levels of SLIDE
XCONTRASTS = *matrices*	Contrasts to estimate
IDPROBES = *texts*	Saves the probe names for each output row
DF = *variates*	Saves degrees of freedom for t-values
RSD = *variates*	Saves the residual standard deviation
DYEBIAS = *variates*	Saves estimated dye swap bias effects
ESTIMATES = *pointers*	Saves the estimates
SEESTIMATES = *pointers*	Saves the standard errors of the estimates
TVALUES = *pointers*	Saves t-values of the estimates
PROBABILITIES = *pointers*	Saves probabilities for the t-values
CONTRASTS = *pointers*	Saves estimates of the contrasts
SECONTRASTS = *pointers*	Saves the standard errors of the contrasts
TCONTRASTS = *pointers*	Saves t-values for the contrasts
PRCONTRASTS = *pointers*	Saves probabilities for the contrasts

MAHISTOGRAM procedure

Plots histograms of microarray data (D.B. Baird).

Options

SLIDES = *factor* or *text*	Defines the slides when the DATA variate contains data from more then one slide
SLIST = *variate* or *text*	Subset of slides to plot; default * i.e. all
NGROUPS = *scalar*	Number of groups into which to classify the DATA units; default 100
COLOUR = *text* or *scalar*	Colour to use for the bars of the histogram; default 'red'
TRANSFORMATION = *string token*	Whether to transform data to logarithms base 2 (log2, none); default none

SCALING = *string token*	Whether to use a common scale when not using Trellis plots (common, none); default comm
NROWS = *scalar*	Number of rows on a page in a trellis plot
NCOLUMNS = *scalar*	Number of columns on a page in a trellis plot
TITLE = *text*	Title for the graph
YTITLE = *text*	Title for the y-axis
XTITLE = *text*	Title for the x-axis
ARRANGEMENT = *string token*	Whether to use trellis or single plots when the DATA variate contains data from more then one slide (single, trellis); default trel
WINDOW = *scalar*	Window number for the graphs; default 3
DEVICE = *scalar*	Device number on which to plot the graphs
GRAPHICSFILE = *text*	What graphics filename template to use to save the graphs; default *
YMINIMUM = *scalar*	Minimum value on the y-axis of the histogram
YMAXIMUM = *scalar*	Maximum value on the y-axis of the histogram
XMINIMUM = *scalar*	Minimum value on the x-axis of the histogram
XMAXIMUM = *scalar*	Maximum value on the x-axis of the histogram
Parameter	
DATA = *variates* or *pointers*	Data values to plot

MANNWHITNEY procedure

Performs a Mann-Whitney U test (S.J. Welham, N.M. Maclaren & H.R. Simpson).

Options

†PRINT = *string tokens*	Output required (test, ranks, hodgeslehmann, confidence); default test
METHOD = *string token*	Type of test required (twosided, greaterthan, lessthan); default twos
GROUPS = *factor*	Defines the samples for a two-sample test if the Y2 parameter is not set
CIPROBABILITY = *scalar*	Probability for the confidence interval for the median difference between the samples; default 0.95
CONTROL = *scalar* or *text*	Identifies the control group against which to make comparisons if GROUPS is set; default uses the reference level of GROUPS

Parameters

Y1 = *variates*	Identifier of the variate holding the first sample if Y2 is set, or both samples if Y2 is unset (the GROUPS option must then also be set)
Y2 = *variates*	Identifier of the variate holding the second sample
R1 = *variates*	Saves the ranks of the first sample if Y2 is set, or both samples if Y2 is unset
R2 = *variates*	Saves the ranks of the second sample if Y2 is set
STATISTIC = *scalars* or *tables*	Saves the test statistics *U*
PROBABILITY = *scalars* or *tables*	Probability values for the test statistics
SIGN = *scalars* or *tables*	Saves indicators: 1 if the first sample scores the highest ranks on average, 0 otherwise
†HODGESLEHMANN = *scalars* or *tables*	Saves the Hodges-Lehmann estimates for the differences in location of the two samples (i.e. the median differences between the samples)
LOWER = *scalars* or *tables*	Saves lower confidence values for median differences between the samples
UPPER = *scalars* or *tables*	Saves upper confidence values for median differences between the samples

MANOVA procedure

Performs multivariate analysis of variance and covariance (R.W. Payne & G.M. Arnold).

Options

PRINT = *string tokens*	Printed output required from the multivariate analysis of covariance (ssp, tests); default test
APRINT = *string tokens*	Printed output from the univariate analyses of variance of the y-variates (as for the ANOVA PRINT option); default *
UPRINT = *string tokens*	Printed output from the univariate unadjusted analyses of variance of the y-variates (as for the ANOVA UPRINT option); default *
CPRINT = *string tokens*	Printed output from the univariate analyses of variance of the covariates (as for the ANOVA CPRINT option); default *
TREATMENTSTRUCTURE = *formula*	Treatment formula for the analysis; if this is not set, the default is taken from the setting (which must already have been defined) by the TREATMENTSTRUCTURE directive
BLOCKSTRUCTURE = *formula*	Block formula for the analysis; if this is not set, the default is taken from any existing setting specified by the BLOCKSTRUCTURE directive and if neither has been set the design is assumed to be unstratified (i.e. to have a single error term)
COVARIATES = *variates*	Covariates for the analysis; by default MANOVA uses those listed by a previous COVARIATE directive (if any)
FACTORIAL = *scalar*	Limit on the number of factors in a treatment term
LRV = *pointer*	Contains elements first for the treatment terms and then the covariate term (if any), allowing the LRV's to be saved from one of the analyses; if a term is estimated in more than one stratum, the LRV is taken from the lowest stratum in which it is estimated
FPROBABILITY = *string token*	Printing of probabilities for F statistics and Chi-square variables (no, yes); default no
SELECTION = *string tokens*	Which test statistics to print when PRINT=test (lawleyhotellingtrace, pillaibartletttrace, roysmaximumroot, wilkslambda}; default lawl, pill, roys, wilk
NTIMES = *scalar*	Number of permutations to make when PRINT=perm; default 999
EXCLUDE = *factors*	Factors in the block model of the design whose levels are not to be randomized
SEED = *scalar*	Seed for the random number generator used to make the permutations; default 0 continues from the previous generation or (if none) initializes the seed automatically

Parameter

Y = *variates*	Y-variates for an analysis

MANTEL procedure

Assesses the association between similarity matrices (J.W. McNicol, E.I. Duff & D.A. Elston).

Options

PRINT = *string token*	Controls printed output (test); default * i.e. none
METHOD = *string token*	The type of metric by which to compare the distance matrices (correlation, rankcorrelation, mantel); default corr
NPERMUTATIONS = *scalar*	The number of permutations of the units in the second distance matrix X on which the significance of the correlation between Y and X is to be based; default 100

Parameters

Y = *symmetric matrices*	The first distance or similarity matrix: the order of the units of this matrix is held fixed

X = *symmetric matrices*	The second distance or similarity matrix: the rows of X are permuted to allow the significance of the correlation between Y and X to be assessed
SEED = *scalars*	Random number seed for the permutations; default set by RANDOMIZE
M = *scalars*	Association between Y and X
MPERMUTED = *variates*	Associations between Y and the permuted X's
CUPROB = *scalars*	The proportion of MPERMUTED values greater than or equal to M
YOFFDIAGONAL = *variates*	Variate to save the off-diagonal elements of the distance/similarity matrix Y
XOFFDIAGONAL = *variates*	Variate to save the off-diagonal elements of the distance/similarity matrix X

MAPCLUSTER procedure

Clusters probes or genes with microarray data (D.B. Baird).

Options

PRINT = *string tokens*	What to print (cluster, groups, summary); default clus
PLOT = *string tokens*	What to plot (dendrogram, groups, meangroups); default dend, grou
METHOD = *string token*	Type of clustering to use (hierarchical, kmeans); default hier
DMETHOD = *string token*	Distance method to use for hierarchical clustering (euclidean, cityblock); default eucl
LMETHOD = *string token*	What type of link to use in hierarchal clustering (singlelink, nearestneighbour, completelink, furthestneighbour, averagelink, mediansort, groupaverage); default aver
CRITERION = *string token*	Criterion to use in forming groups when LMETHOD=kmeans (sums, predictive, within, Mahalanobis); default sums
NGROUPS = *scalar*	Number of groups to form when LMETHOD=kmeans
GTHRESHOLD = *scalar*	Grouping threshold for forming groups from the dendrogram; default *
PERCENT = *scalar*	Percentage of the probes/genes to use; default 100
DTITLE = *text*	Title for the dendrogram
GTITLE = *text*	Title for the groups plot
ARRANGEMENT = *string token*	Whether to use a trellis or single plot (single, trellis); default trel
WINDOW = *scalar*	Window number for the graphs; default 3
DEVICE = *scalar*	Device number on which to plot the graphs
GRAPHICSFILE = *text*	What graphics filename template to use to save the graphs; default *
SPREADSHEET = *string token*	What results to put in spreadsheets (top%probes); default * i.e. none

Parameters

DATA = *variates* or *pointers*	Data values (i.e. log-ratios)
SLIDES = *factors*, *texts* or *variates*	Identifies the slides
PROBES = *factors*, *texts* or *variates*	Identifies the probes or genes
SIMILARITY = *symmetric matrices*	Saves the pair-wise similarities between probes or genes when METHOD=hier
GROUPS = *factors*	Saves the group membership for each probe
AMALGAMATIONS = *matrices*	Saves the probe or gene amalgamation data when METHOD=hier

MAPLOT procedure

Produces two-dimensional plots of microarray data (D.B. Baird).

Options

SLIDES = *factor* or *text*	Defines the slides when the X and Y variates contain data from more than one slide
SLIST = *variate* or *text*	Subset of slides to plot; default * i.e. all
GROUPS = *factor*	Specifies groups within slides
COLOURS = *text*, *scalar* or *variate*	Colours to use for the plots
SYMBOLS = *scalar* or *variate*	Symbols to use for the plots
REFERENCELINECHOICE = *string token*	Reference line to include (identity, zero, none); default none
TRANSFORMATION = *string token*	Whether to transform data to logarithms base 2 (log2, none); default none
SCALING = *string token*	Whether to use a common scale when not using Trellis plots (common, none); default comm
BANDS = *string token*	Whether to plot approximate confidence bands (confidence, none); default none
SMOOTHEDMEAN = *string token*	Whether to plot spline smooth of mean (yes, no); default no
NROWS = *scalar*	Number of rows on a page in a trellis plot
NCOLUMNS = *scalar*	Number of columns on a page in a trellis plot
TITLE = *text*	Title for the graph
YTITLE = *text*	Title for the y-axis
XTITLE = *text*	Title for the x-axis
ARRANGEMENT = *string token*	Whether to use trellis, single or multiple plots when the X and Y variates contain data from more than one slide (separate, overlaid, trellis); default trel
WINDOW = *scalar*	Window number for the graphs; default 3
KEYWINDOW = *scalar*	Window number for the key; default 0
DEVICE = *scalar*	Device number on which to plot the graphs
GRAPHICSFILE = *text*	What graphics filename template to use to save the graphs; default *

Parameters

Y = *variates* or *pointers*	Y-coordinates
X = *variates* or *pointers*	X-coordinates

MAREGRESSION procedure

Does regressions for single-channel microarray data (P. Brain, R.W. Payne & D.B. Baird).

Options

PRINT = *string tokens*	Controls printed output (model, summary); default * i.e. none
TERMS = *formula*	Defines the regression model over the slides
WEIGHTS = *variate*	Weights for the regression; default 1
OFFSET = *variate*	Offset; default * i.e. none
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default 3
FULL = *string token*	Whether to assign all possible parameters to factors and interactions (yes, no); default no
POOL = *string token*	Whether to pool the information on each term in the analysis of variance (yes, no); default no
RMETHOD = *string token*	Type of residuals to form (deviance, Pearson, simple); default devi
SPREADSHEET = *string tokens*	What results to save in a book of spreadsheets (aov, residuals, fittedvalues, estimates, se, testimates, prestimates); default * i.e. none

Parameters

Y = *variates* or *pointers*	Y-values for each set of analyses

PROBES = *factors* or *texts*	Defines the probe information for each analysis
SLIDES = *factors* or *texts*	Defines the slide information for each analysis
CHECK= *texts* or *variates*	Slide ID's that can be compared with the labels or levels of the SLIDES factor to ensure that the slide order is correct in each analysis
IDS = *texts*	Saves the probes names that have been generated to label the rows of the output structures from each analysis
RESIDUALS = *matrices*	Saves residuals from each set of analyses
FITTEDVALUES = *matrices*	Saves fitted values from each set of analyses
ESTIMATES = *matrices*	Saves estimates from each set of analyses
SE = *matrices*	Saves s.e.'s of estimates
TESTIMATES = *matrices*	Saves t-statistics of estimates
PRESTIMATES = *matrices*	Saves t-probabilities of estimates
DF = *pointers*	Saves degrees of freedom for the model terms or variates in each analysis of variance
SS = *pointers* or *variates*	Saves sums of squares for the model terms in each analysis of variance
MS = *pointers* or *variates*	Saves mean squares for the model terms in each analysis of variance
RDF = *variates*	Saves degrees of freedom from the "residual" lines in each analysis of variance
RSS = *variates*	Saves sums of squares from the "residual" lines
RMS = *variates*	Saves mean squares from the "residual" lines
TDF = *variates*	Saves degrees of freedom from the "total" lines in each analysis of variance
TSS = *variates*	Saves sums of squares from the "total" lines
TMS = *variates*	Saves mean squares from the "total" lines
VR = *pointers* or *variates*	Saves variance ratios for the model terms in each analysis of variance
PRVR = *pointers* or *variates*	Saves probabilities of the variance ratios

MARGIN directive

Forms and calculates marginal values for tables.

Option

CLASSIFICATION = *factors*	Factors classifying the margins to be formed; default * requests all margins to be formed

Parameters

OLDTABLE = *tables*	Tables from which the margins are to be taken or calculated
NEWTABLE = *tables*	New tables formed with margins
METHOD = *string tokens*	Way in which the margins are to be formed for each table (totals, means, minima, maxima, variances, medians, deletion, or a null string to indicate that the marginal values are all to be set to the missing value); default tota

MARMA procedure

Calculates Affymetrix expression values (D.B. Baird).

Options

PRINT = *string token*	What to print (estimates, monitoring); default esti
METHOD = *string token*	Method of establishing grid background (rma, rma2); default rma
NORMALIZED = *string token*	Whether slides have been normalized (yes, no); default no

Parameters

DATA = *variates* or *pointers*	Perfect-match data
SLIDES = *factors* or *texts*	Defines the slides
NEWDATA = *variates* or *pointers*	Saves the corrected values; if this is unset, they replace the

	original values in DATA
ESTIMATES = *variates*	Saves the estimated parameters of the model

MAROBUSTMEANS procedure

Does a robust means analysis for Affymetrix slides (D.B. Baird).

Options

TRANSFORMATION = *string token*	How to transform the data (log2, none); default none
MAXCYCLE = *scalar*	Maximum number of iterations; default 50
TOLERANCE = *scalar*	Tolerance for convergence; default 0.0001

Parameters

DATA = *variates* or *pointers*	Expression data to be summarized
SLIDES = *factors* or *texts*	Defines the slides
PROBES = *factors*	Defines the probes
IDPROBES = *factors*	Saves the probe IDs
MEDIANS = *variates* or *pointers*	Saves the robust means
SEM = *variates* or *pointers*	Saves approximate standard errors of the robust means

MASCLUSTER procedure

Clusters microarray slides (D.B. Baird).

Options

PRINT = *string tokens*	What to print (cluster, pco, correlations, distances); default clus, pco, corr, dist
PLOT = *string tokens*	What to plot (dendrogram, mst); default dend, mst
DMETHOD = *string token*	What distance method to use to form the similarity matrix (correlation, euclidean, cityblock); default corr
PERCENT = *scalar*	Percentage of the probes/genes to use to calculate correlations; default 100
DTITLE = *text*	Title for the dendrogram
MTITLE = *text*	Title for the minimum spanning tree
WINDOW = *scalar*	Window number for the graphs; default 3
DEVICE = *scalar*	Device number on which to plot the graphs
GRAPHICSFILE = *text*	What graphics filename template to use to save the graphs; default *

Parameters

DATA = *variates* or *pointers*	Data values (i.e. log-ratios)
SLIDES = *factors*, *texts* or *variates*	Identifies the slides
PROBES = *factors*, *texts* or *variates*	Identifies the probes or genes
CORRELATION = *symmetric matrices*	Saves the correlation matrix
DISTANCE = *symmetric matrices*	Saves the distance matrix

MASHADE procedure

Produces shade plots to display spatial variation of microarray data (D.B. Baird).

Options

SLIDES = *factor* or *text*	Defines the slides when the DATA variate contains data from more than one slide
SLIST = *variate* or *text*	Subset of slides to plot; default * i.e. all
ROWS = *factor* or *variate*	Row to which each DATA unit belongs
COLUMNS = *factor* or *variate*	Column to which each DATA unit belongs
COLOURS = *text*, *scalar* or *variate*	Colours to use for the plots; default !t(blue,red)
SHADING = *string token*	Shading scale (natural, percentiles); default natu
TITLE = *text*	Title for the graph
YTITLE = *text*	Title for the y-axis
XTITLE = *text*	Title for the x-axis
WINDOW = *scalar*	Window number for the graphs; default 3
DEVICE = *scalar*	Device number on which to plot the graphs
GRAPHICSFILE = *text*	What graphics filename template to use to save the graphs;

default *

Parameter

DATA = *variates* or *pointers* — Values for each shade plot

MATRIX directive

Declares one or more matrix data structures.

Options

ROWS = *scalar*, *vector*, *pointer* or *text* — Number of rows, or labels for rows; default *

COLUMNS = *scalar*, *vector*, *pointer* or *text* — Number of columns, or labels for columns; default *

VALUES = *numbers* — Values for all the matrices; default *

MODIFY = *string token* — Whether to modify (instead of redefining) existing structures (yes, no); default no

IPRINT = *string tokens* — Information to be used by default to identify the matrices in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers* — Identifiers of the matrices

VALUES = *identifiers* — Values for each matrix

DECIMALS = *scalars* — Number of decimal places for printing

EXTRA = *texts* — Extra text associated with each identifier

MINIMUM = *scalars* — Minimum value for the contents of each structure

MAXIMUM = *scalars* — Maximum value for the contents of each structure

DREPRESENTATION = *scalars* or *texts* — Default format to use when the contents represent dates and times

MAVDIFFERENCE procedure

Applies the average difference algorithm to Affymetrix data (D.B. Baird).

Options

PRINT = *string token* — Whether to print monitoring information (monitoring); default *

SDLIMIT = *scalar* — Maximum number of iterations; default 50

Parameters

DATA = *variates* or *pointers* — Data values

GROUPS = *factors* — Groupings of the data values

MEANS = *variates* — Saves the means

SE = *variates* — Saves standard errors

MAVOLCANO procedure

Produces volcano plots of microarray data (D.B. Baird).

Options

NGROUPS = *scalar* — Number of groupings for a Z variate; default 10

COLOURS = *text*, *scalar* or *variate* — Colours to use for the plots; default !t(blue,red)

SYMBOL = *scalar* — Symbol to use for the points; default 1

TRANSFORMATION = *string token* — Whether to transform data to logarithms base 2 (log10, none); default log10

TITLE = *text* — Title for the graph

YTITLE = *text* — Title for the y-axis

XTITLE = *text* — Title for the x-axis

WINDOW = *scalar* — Window number for the graphs; default 3

KEYWINDOW = *scalar* — Window number for the graphs; default 0

DEVICE = *scalar* — Device number on which to plot the graphs

GRAPHICSFILE = *text* — What graphics filename template to use to save the graphs; default *

Parameters

X = *variates* — X-coordinates

Y = *variates* or *factors*	Y-coordinates
Z = *variates* or *factors*	Z-coordinates

MA2CLUSTER procedure

Performs a two-way clustering of microarray data by probes (or genes) and slides (D.B. Baird).

Options

PRINT = *string tokens*	What to print (cluster, groups, summary); default clus
PLOT = *string tokens*	What to plot (dendrogram, shade, meanshade); default dend, shad
METHOD = *string token*	Type of clustering to use (hierarchical, kmeans); default hier
DMETHOD = *string token*	Distance method to use for hierarchical clustering (euclidean, cityblock); default eucl
LMETHOD = *string token*	What type of link to use in hierarchal clustering (singlelink, nearestneighbour, completelink, furthestneighbour, averagelink, mediansort, groupaverage); default aver
CRITERION = *string token*	Criterion to use in forming groups when LMETHOD=kmeans (sums, predictive, within, Mahalanobis); default sums
PNGROUPS = *scalar*	Number of probe groups to form when LMETHOD=kmeans
SNGROUPS = *scalar*	Number of target (slide) groups to form when LMETHOD=kmeans
GTHRESHOLD = *scalar*	Grouping threshold for forming probe groups from the dendrogram; default *
SGTHRESHOLD = *scalar*	Grouping threshold for forming target (slide) groups from the dendrogram; default *
MINOBSERVATIONS = *scalar*	Smallest number of observations before probes are dropped; default *
PERCENT = *scalar*	Percentage of the probes/genes to use; default 100
STANDARDIZE = *string token*	Allows you to centre the values by slide and probe (centre); default * i.e. no centring
COLOURS = *text*, *scalar* or *variate*	Colours to use for shade plot; default !t(blue,red)
DTITLE = *text*	Title for the dendrogram
STITLE = *text*	Title for the shade plot
WINDOW = *scalar*	Window number for the graphs; default 3
DEVICE = *scalar*	Device number on which to plot the graphs
GRAPHICSFILE = *text*	What graphics filename template to use to save the graphs; default *
SPREADSHEET = *string token*	What results to put in spreadsheets (top%probes); default * i.e. none

Parameters

DATA = *variates* or *pointers*	Data values (i.e. log-ratios)
SLIDES = *factors*, *texts* or *variates*	Identifies the slides
PROBES = *factors*, *texts* or *variates*	Identifies the probes or genes
GMEANS = *matrices*	Saves the tabulation of the data by probe groups and target groups, as a two-way matrix
PGROUPS = *factors*	Saves the group membership for each probe (or gene)
SGROUPS = *factors*	Saves the group membership for each slide (or target)
PAMALGAMATIONS = *matrices*	Saves the probe (or gene) amalgamation data when METHOD=hier
SAMALGAMATIONS = *matrices*	Saves the slide (or target) amalgamation data when METHOD=hier

MCNEMAR procedure

Performs McNemar's test for the significance of changes (R.W. Payne & D.A. Murray).

Options

PRINT = *string tokens*	Controls printed output (test, table); default test

METHOD = *string token* — Type of test required (twosided, greaterthan, lessthan); default twos

Parameters

Y1 = *factors* or *tables* — Factor containing the responses obtained before the treatment (with 1 indicating a positive response) or two-by-two table (classified by factors representing the two occasions of testing) summarizing the responses before and after treatment

Y2 = *factors* — Factor containing the responses obtained after the treatment (need not be specified if Y1 is a table)

STATISTIC = *scalars* — Saves the test statistic

PROBABILITY = *scalars* — Saves the probability value

MCOMPARISON procedure

Performs pairwise multiple comparison tests within a table of means (D.M. Smith).

Options

PRINT = *string tokens* — Controls printed output (comparisons, critical, description, lines, letters, plot, mplot, pplot); default lett

METHOD = *string token* — Test to be performed (flsd, bonferroni, sidak); default flsd

DIRECTION = *string token* — How to sort means (ascending, descending); default asce

PROBABILITY = *scalar* — The required significance level; default 0.05

STUDENTIZE = *string token* — Whether to use the alternative LSD test where the Studentized Range statistic is used instead of Student's t (yes, no); default no

Parameters

MEANS = *tables* — Means to be compared

SED = *symmetric matrix* or *scalar* — Standard errors of differences of the means

DF = *symmetric matrix* or *scalar* — Degrees of freedom for the standard errors of differences

VMEANS = *pointer* or *variate* — Saves the means in a variate, sorted as requested by the DIRECTION option

DIFFERENCES = *symmetric matrix* — Saves differences between the (sorted) means

LABELS = *text* — Saves labels for the (sorted) means

LETTERS = *text* — Saves letters indicating groups of means that do not differ significantly

SIGNIFICANCE = *symmetric matrix* — Indicators to show significant comparisons between (sorted) means

CIWIDTH = *symmetric matrix* — Saves the width of the confidence interval for the absolute differences between the (sorted) means

TERMNAME = *texts* — Name of the term, to use to annotate the graphs

MCORANALYSIS procedure

Does multiple correspondence analysis (A.I. Glaser).

Options

PRINT = *string tokens* — Printed output from the analysis (roots, rowscores, rowinertias, rowchisquare, rowmass, rowquality, colscores, colinertias, colchisquare, colmass, colquality); default * i.e. no output

ROWMETHOD = *string token* — Analysis method for rows i.e. units (indicator); default indi

COLMETHOD = *string token* — Analysis method for columns i.e. factors (adjusted, burt, indicator); default adju

NROOTS = *scalar* — Number of latent roots for printed output; default * requests them all to be printed

%METHOD = *string token* — How to represent proportions or %s in quality statistics (permills, percentages, proportions); default prop

NDIMENSIONS = *scalar* — Number of dimensions for which quality statistics are required;

default 2

TOLERANCE = *scalar* Tolerance criteria for zero eigenvalues; default 10^{-6}

Parameters

DATA = *pointers* Data to be analysed

ROOTS = *diagonal matrices* Saves the squared singular values from each analysis

ROWSCORES = *matrices* Saves the scores for the rows of the data

COLSCORES = *matrices* Saves the scores for the columns of the data

ROWINERTIAS = *matrices* Saves the total inertias for the rows of the data

COLINERTIAS = *matrices* Saves the total inertias for the columns of the data

ROWQUALITY = *matrices* Saves the quality statistics for rows of the data

COLQUALITY = *matrices* Saves the quality statistics for columns of the data

SUBINERTIAS = *matrices* Saves the inertias of the subtables of the Burt matrices

FREQUENCY = *variates* Frequencies for elements of DATA

SAVE = *pointers* Saves details of the analysis for use by CABIPLOT

MCOVARIOGRAM directive

Fits models to sets of variograms and cross-variograms.

Options

PRINT = *string tokens* Controls printed output from the fit (model, summary, estimates, fittedvalues, monitoring); default mode, summ, esti

WEIGHTING = *string token* Method to be used for weighting (counts, equal); default coun

MAXLAG = *scalar* Maximum lag distance of points to be included in the modelling

MINCOUNT = *scalar* Minimum number of points required at a particular lag point for a pair of variables for this to be used to model their cross-variogram; default 30 for equal weighting and 10 for counts

MAXCYCLE = scalar Maximum number of iterations for model fitting; default 30

TOLERANCES = *variate* Tolerances for model fitting; default * i.e. appropriate default values

COORDSYSTEM = *string token* Coordinate system used for the geometry for discretizing the lag (mathematical, geographical); default math

COVARIOGRAM = *pointers* Experimental variograms, cross-variograms and associated information defining the data for fitting the model

Parameters

MODELTYPE = *string tokens* Defines the model structures to be fitted (nugget, power, boundedlinear, circular, spherical, pentaspherical, cubic, stable, besselk1, cardinalsine, dampenedcosine); no default i.e. must be specified

INITIAL = *scalars* or *variates* Scalar defining the initial distance parameter for fitting an isotropic model structure or a variate defining initial values for an anisotropic ellipse or ellipsoid for fitting an geometrical anisotropic model

ISOTROPY = *string tokens* Specifies the zonal anisotropy to be used for model structure (isotropic, x, y, z, xy, xz, yz); default isot

ESTIMATES = *pointers* Structures to store the estimated non-linear parameters and sill values

LOWER = *scalars* Lower bound for each non-linear distance parameter

UPPER = *scalars* Upper bound for each non-linear distance parameter

STEPLENGTH = *scalars* Initial step length for each non-linear distance parameter

SMOOTHNESS = *scalars* Value of exponent parameter for the power and stable models, or theta parameter for the dampened-cosine model

MCROSSPECTRUM procedure

Performs a spectral analysis of a multiple time series (G. Tunnicliffe Wilson & R.P. Littlejohn).

Options

PRINT = *string token*	Controls printed output (description); default desc
PLOT = *string tokens*	Variables for which to plot the analysis (explanatory, response); default expl, resp
CORRECT = *string token*	Whether to mean or trend correct the series (mean, linear, quadratic, none); default mean
BANDWIDTH = *scalar*	Bandwidth for smoothing, must be between 0 and 0.5; if unset, a default is calculated automatically
MAXLAG = *scalar*	Maximum lag for the time domain outputs; if unset, a default is calculated automatically
PROBABILITY = *scalar*	Probability value for confidence limits; default 0.95
TAPER = *scalar*	The proportion of data to be tapered using a cosine bell window; default 0
YLOG = *string token*	Whether to plot the univariate spectra with a $\log_{10}$-transformed y-axis (yes, no); default no

Parameters

Y = *variates*	Response time series
X = *variates or pointers*	Explanatory time series
ALIGN = *variates*	Shifts to apply to the explanatory series; default none
SPECTRUM = *pointers*	Saves autospectra, co-spectra and quad-spectra
FREQUENCY = *variate*	Saves the frequency values at which the spectra are calculated
VARSPECTRUM = *pointers*	Saves information about the variation of the spectrum: coefficient of variation, degrees of freedom, and lower and upper multiplicative limits for the univariate spectra
MULTICOHERENCYSQUARED = *pointers*	Saves estimates, significance limits, lower and upper confidence limits for the squared multiple coherency between the response and explanatory series
PARTIALCOHERENCYSQUARED = *pointers*	Saves estimates, significance limits, lower and upper confidence limits for the squared partial coherency of the response series with each explanatory series
GAIN = *pointers*	Saves estimates, lower and upper limits for the estimated gain of response series from each of the explanatory series
PHASE = *pointers*	Saves estimates, lower and upper limits for the estimated phase of response series from each of the explanatory series
NOISESPECTRUM = *variates*	Saves the estimated spectrum of the noise process
IMPULSERESPONSE = *pointers*	Saves the impulse response from −maxlag to +maxlag: estimates and significance limit
LAGS = *variates*	Saves the lags for the impulse response
ACFNOISE = *variates*	Saves the ACF of the noise process

MC1PSTATIONARY procedure

Gives the stationary probabilities for a 1st-order Markov chain (R.P. Littlejohn).

Option

PRINT = *string token*	What to print (transitions, pstationary); default psta

Parameters

DATA = *matrices* or *factors*	Specifies the Markov chain as a factor, or matrix of transitions
STATES = *texts*	Labels for the states
PSTATIONARY = *variates*	Saves the stationary probabilities
TRANSITIONS = *matrices*	Saves the transition matrices

MDS directive

Performs non-metric multidimensional scaling.

Options

PRINT = *string tokens*	Printed output required (coordinates, roots, distances, fitteddistances, stress, monitoring); default * i.e. no printing
DATA = *symmetric matrix*	Distances amongst a set of units
METHOD = *string token*	Whether to use non-metric scaling, or metric scaling with linear regression of the fitted distances to the actual distances (nonmetric, linear); default nonm
SCALING = *string token*	Whether least-squares, least-squares-squared, or log-stress scaling is to be used (ls, lss, logstress); default ls
TIES = *string token*	Treatment of tied data values (primary, secondary, tertiary); default prim
WEIGHTS = *symmetric matrix*	Weights for each distance value; default * i.e. all distances with weight one
INITIAL = *matrix*	Initial configuration; default * i.e. a principal coordinate solution is used
NSTARTS = *scalar*	Number of starting configurations to be used, by making random perturbations to the initial configuration; default 10
SEED = *scalar*	Seed for the random-number generator; default 0
MAXCYCLE = *scalar*	Maximum number of iterations; default 30

Parameters

NDIMENSIONS = *scalars*	Number of dimensions for each solution
COORDINATES = *matrices*	To store the coordinates of the units for each solution
STRESS = *scalars*	To store the stress value for each solution
DISTANCES = *symmetric matrices*	To store the distances amongst the points for the units in the fitted number of dimensions
FITTEDDISTANCES = *symmetric matrices*	To store the fitted distances from the monotonic (METHOD=nonmetric) or linear (METHOD=linear) regression

MEDIANTETRAD procedure

Gives robust identification of multiple outliers in 2-way tables (J.K.M. Brown).

Options

PRINT = *string tokens*	Printed output required (graph, table); default grap, tabl
GRAPHICS = *string tokens*	Type of graph required (highresolution, lineprinter); default high
SORT = *string tokens*	Sorting of printed output, in order of absolute value of median tetrad (ascending, descending, none); default none

Parameters

TABLE = *tables*	Specifies the two-way table of data
ROWS = *factors*	Saves the factor classifying the table rows
COLUMNS = *factors*	Saves the factor classifying the table columns
DATA = *variates*	Saves the data values in the body of the table
MEDIANTETRADS = *variates*	Saves median tetrads for each cell in the table
RANKS = *variates*	Saves ranks of absolute values of median tetrads
HALFNORMALSCORES = *variates*	Saves half-Normal scores of absolute values of median tetrads
TESTOUTLIERS = *scalars*	Specifies the number of cells, with the highest absolute median tetrads, to be set to their predicted values before re-running the analysis

META procedure

Combines estimates from individual trials (R.W. Payne & S. Senn).

Options

PRINT = *string tokens*	Controls output (estimates, overalltest, heterogeneity, confidenceplot, galbraithplot, monitoring); default esti, over, hete, conf
SELECTION = *string tokens*	Which combined estimates to include in the output (fixed, random); default fixe, rand
RMETHOD = *string token*	How to form the random estimate (maxlikelihood, maxremllikelihood, moments, reml); default reml
XLABEL = *text*	Label for the x-axis of the confidence plot; default 'treatment effect'
SMETHOD = *string token*	How to set the sizes of symbols on the confidence plot (equal, inversese); default inve
CIPROBABILITY = *scalar*	Probability level to use for the confidence intervals; default 0.95
CIMETHOD = *string token*	Method to use for calculating the confidence interval for random estimates formed by maximum likelihood or REML (approximate, profile); default prof
PRMETHOD = *string token*	Type of test to use for the overall probability values (greaterthan, lessthan, twosided); default grea
MAXCYCLE = *scalar*	Maximum number of iterations to use with RMETHOD settings maxlikelihood and maxremllikelihood; default 100
TOLERANCE = *scalar*	Convergence criterion to use with RMETHOD settings maxlikelihood and maxremllikelihood; default 10^{-6}

Parameters

ESTIMATES = *variates*	Supplies the estimates to combine
SEESTIMATES = *variates*	Specifies the standard errors of the estimates
LABELS = *texts*	Labels to use for each variate of ESTIMATES in the output
FIXEDESTIMATE = *scalars*	Saves the combined estimate for each variate of ESTIMATES, treating them as fixed effects
SEFIXEDESTIMATE = *scalars*	Saves the standard error of the combined estimate for each variate of ESTIMATES, treating them as fixed effects
PRFIXEDESTIMATE = *scalars*	Saves the probability of the combined estimate for each variate of ESTIMATES, treating them as fixed effects
RANDOMESTIMATE = *scalars*	Saves the combined estimate for each variate of ESTIMATES, treating them as random effects
SERANDOMESTIMATE = *scalars*	Saves the standard error of the combined estimate for each variate of ESTIMATES, treating them as random effects
PRRANDOMESTIMATE = *scalars*	Saves the probability of the combined estimate for each variate of ESTIMATES, treating them as random effects
QSTATISTIC = *scalars*	Saves the statistic *Q* for the test of heterogeneity across trials
QDF = *scalars*	Saves the degrees of freedom of the statistic *Q*
RVARIANCE = *scalars*	Saves the random effect variance
LOWER = *variates*	Saves lower values of the confidence interval
UPPER = *variates*	Saves upper values of the confidence interval

MICHAELISMENTEN procedure

Fits the Michaelis-Menten equation for substrate concentration versus time data (M.C. Hannah).

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, monitoring); default mode, summ, esti
PLOT = *string tokens*	What to plot (concentration, rate); default conc
WINDOW = *scalar*	Window in which to plot the graphs; default 1
TITLE = *text*	Title for the graphs; default 'Michaelis-Menten

process'

TTIMES = *text* — Title for the times axis; if this is unset, the identifier of the TIMES variate is used

TCONCENTRATIONS = *text* — Title for the concentrations axis; if this is unset, the identifier of the CONCENTRATIONS variate is used if available, otherwise 'Concentration'

TRATES = *text* — Title for the rates axis; if this is unset, the identifier of the RATES variate is used if available, otherwise 'Rate'

WEIGHTS = *variate* — Weights for the observations, to use in the fit, if required; default * i.e. all observations with weight one

Parameters

TIMES = *variates* — Times at which substrate concentration data were measured

CONCENTRATIONS = *variates* — Substrate concentration data

STEPLENGTHS = *variates* — Variate with four values defining initial step lengths for the parameters S_0, V_{max}, K_m and K_1 (in that order)

INITIAL = *variates* — Variate containing initial values for the parameters, similarly to STEPLENGTHS

RESIDUALS = *variates* — Saves the residuals from each fit

FITTEDVALUES = *variates* — Saves the fitted concentration values

ESTIMATES = *variates* — Saves the parameter estimates

SE = *variates* — Saves the standard errors of the estimates

VCOVARIANCE = *symmetric matrix* — Saves the variance-covariance matrix of the estimates

OBSRATES = *variates* — Saves reaction rates, calculated from the observed concentrations

FITRATE = *variates* — Saves fitted reaction rates

MINFIELDWIDTH procedure

Calculates minimum field widths for printing data structures (R.W. Payne).

Option

IPRINT = *string tokens* — What identifier and/or text to print for the structure (identifier, extra); default is to take the IPRINT setting of each STRUCTURE

Parameters

STRUCTURE = *identifiers* — Data structures to be printed

FIELDWIDTH = *scalars* — Saves the minimum field widths

DECIMALS = *scalars* — Number of decimal places to be used for numerical data structures; if unset, a default is obtained using the DECIMALS procedure

SKIP = *scalars* — Number of spaces to leave before each value of the structure; default 1

FREPRESENTATION = *string tokens* — How to represent factor values (labels, levels, ordinals); default is to use labels if available, otherwise levels

MINIMIZE procedure

Finds the minimum of a function calculated by a procedure (R.W. Payne).

Options

PRINT = *string tokens* — What output to produce (minimum, monitoring); default mini

FUNCTIONVALUE = *scalar* — Saves the minimum function value

DATA = any type — Data to be used with procedure _MINFUNCTION

MAXCYCLE = *scalar* — Maximum number of iterations; default 2000

NSTARTS = *scalar* — Maximum number of restarts; default 4

STEPADJUSTMENT = *scalar* — Adjustment to step lengths at each restart; default 0.1

EXIT = *scalar* — Indicates whether there has been convergence (zero) or non-convergence (non-zero)

TOLERANCE = *scalar* — Convergence criterion; default 0.0001

METHOD = *string token*	Algorithm for fitting nonlinear model (GaussNewton, NewtonRaphson, FletcherPowell); default Newt

Parameters

PARAMETER = *scalars*	Parameters to be estimated
LOWER = *scalars*	Lower bound for each parameter
UPPER = *scalars*	Upper bound for each parameter
STEPLENGTH = *scalars*	Step length for each parameter
INITIAL = *scalars*	Initial value for each parameter

MIN1DIMENSION procedure

Finds the minimum of a function in one dimension (R.W. Payne).

Options

PRINT = *string tokens*	What output to produce (minimum, monitoring, plot); default mini
CALCULATION = *expression structures*	Expressions to calculate the target function
FUNCTIONVALUE = *scalars*	Identifier of the scalar, calculated by CALCULATION, whose value is to be minimized
DATA = any type	Data to be used with procedure _MIN1DFUNCTION
CRITERION = *string token*	Criterion for convergence (function, parameters); default func
MAXCYCLE = *scalars*	Maximum number of iterations; default 250
EXIT = *scalars*	Indicates whether there has been convergence (0) or non-convergence (1)
TOLERANCE = *scalars*	Convergence criterion; default 10^{-6} or variate

Parameters

PARAMETER = *scalars*	Parameters to be estimated
LOWER = *scalars*	Lower bound for each parameter
UPPER = *scalars*	Upper bound for each parameter
STEPLENGTH = *scalars*	Step length for each parameter
INITIAL = *scalars*	Initial value for each parameter

MERGE directive

Copies subfiles from backing-store files into a single file.

Options

PRINT = *string token*	What to print (catalogue); default *
OUTCHANNEL = *scalar*	Channel number of the backing-store file where the subfiles are to be stored; default 0, i.e. the workfile
METHOD = *string token*	How to append subfiles to the OUT file (add, overwrite, replace); default add, i.e. clashes in subfile identifiers cause a fault (note: replace overwrites the complete file)
PASSWORD = *text*	Password to be checked against that stored with the file; default *

Parameters

SUBFILE = *identifiers*	Identifiers of the subfiles
INCHANNEL = *scalars*	Channel number of the backing-store file containing each subfile
NEWSUBFILE = *identifiers*	Identifier to be used for each subfile in the new file

MMPREDICT procedure

Predicts the Michaelis-Menten curve for a particular set of parameter values (M.C. Hannah).

Options

PLOT = *string tokens*	What to plot (concentration, rate); default conc
WINDOW = *scalar*	Window in which to plot the graphs; default 1
TITLE = *text*	Title for the graphs; default 'Michaelis-Menten process'
TTIMES = *text*	Title for the times axis; if this is unset, the identifier of the

	TIMES variate is used
TCONCENTRATIONS = *text*	Title for the concentrations axis; if this is unset, the identifier of the CONCENTRATIONS variate is used if available, otherwise 'Concentration'
TRATES = *text*	Title for the rates axis; if this is unset, the identifier of the RATES variate is used if available, otherwise 'Rate'

Parameters

PARAMETERS = *variates*	Variate with four values specifying the values of the parameters S_0, V_{max}, K_m and K to use to form the predictions
TIMES = *variates*	Times at which to make predictions
CONCENTRATIONS = *variates*	Saves the predicted substrate concentrations
RATES = *variates*	Saves the predicted reaction rates

MNORMALIZE procedure

Normalizes two-colour microarray data (D.B. Baird).

Options

PRINT = *string tokens*	What to print (summary, slidesummary, monitoring); default summ, slid, moni
PLOT = *string tokens*	What plots to produce (pineffects, roweffects, columneffects, intensityeffects, rowxcoleffects, ma, standardizedma, spatialresiduals); default * i.e. none
METHOD = *string token*	What type of model components to fit (spline, loess); default spli
MODELTERMS = *string tokens*	What model components to fit (pins, rows, columns, intensity, pinxintensity, ar1, rowxcolumn, pinxrow, pinxcolumn); default pins, rows, colu, inte
DFINTENSITY = *scalar*	Degrees of freedom for intensity cubic spline; default 24
DFROWXCOLUMN = *scalar*	Degrees of freedom for row × col thinplate spline; default 49
POORFLAGS = *text* or *variate*	Levels of FLAGS that are poor quality spots
BADFLAGS = *text* or *variate*	Levels of FLAGS that are bad spots
ARRANGEMENT = *string token*	Whether to use trellis or single plots (single, trellis); default trel
WINDOW = *scalar*	Window number for the graphs; default 3
DEVICE = *scalar*	Device number on which to plot the graphs
GRAPHICSFILE = *text*	What graphics filename template to use to save the graphs; default *

Parameters

LOGRATIOS = *variates* or *pointers*	Log-ratios
INTENSITIES = *variates* or *pointers*	Spot intensities
SLIDES = *factors* or *texts*	Slides
PINS = *factors*	Pins
SROWS = *factors*	Rows across whole slide
SCOLUMNS = *factors*	Columns across whole slide
PROWS = *factors*	Rows within pins
PCOLUMNS = *factors*	Columns within pins
FLAGS = *factors* or *pointers*	Quality flags
CLOGRATIOS = *variates* or *pointers*	Save corrected log-ratios
SLOGRATIOS = *variates* or *pointers*	Save standardized log-ratios
SDSMOOTH = *variates* or *pointers*	Save smoothed deviations
PINEFFECTS = *tables*	Save estimated pin effects
ROWEFFECTS = *tables*	Save estimated row effects
COLEFFECTS = *tables*	Save estimated column effects
INTEFFECTS = *variates* or *pointers*	Save estimated intensity effects
CLRED = *variates* or *pointers*	Save corrected log2 red values
CLGREEN = *variates* or *pointers*	Save corrected log2 green values

VAREXPLAINED = *variates*	Save the variance explained by slide

MODEL directive

Defines the response variate(s) and the type of model to be fitted for linear, generalized linear, generalized additive, and nonlinear models.

Options

DISTRIBUTION = *string token*	Distribution of the response variable (normal, poisson, binomial, gamma, inversenormal, multinomial, calculated, negativebinomial, geometric, exponential, bernoulli); default norm
LINK = *string token*	Link function (canonical, identity, logarithm, logit, reciprocal, power, squareroot, probit, complementaryloglog, calculated, logratio); default cano (i.e. iden for DIST=norm or calc; loga for DIST=pois; logi for DIST=bino, bern or mult; reci for DIST=gamm or expo; powe for DIST=inve; logr for DIST=nega or geom)
EXPONENT = *scalar*	Exponent for power link; default -2
AGGREGATION = *scalar*	Fixed parameter for negative binomial distribution (parameter k as in variance function Var = mean + mean$^2/k$); default 1
KLOGRATIO = *scalar*	Parameter for logratio link, in form log(mean/(mean+k)); default as set in AGGREGATION option
DISPERSION = *scalar*	Value of dispersion parameter in calculation of s.e.s etc; default * for DIST=norm, gamm, inve or calc, and 1 for DIST=pois, bino, mult, nega, geom, expo or bern
WEIGHTS = *variate* or *symmetric matrix*	Variate of weights for weighted regression, or symmetric matrix of weights (one row and column for each unit of data) for generalized least squares; default *
OFFSET = *variate*	Offset variate to be included in model; default *
GROUPS = *factor*	Absorbing factor defining the groups for within-groups linear or generalized linear regression; default *
RMETHOD = *string token*	Type of residuals to form, if any, after each model is fitted (deviance, Pearson, simple); default devi
DMETHOD = *string token*	Basis of estimate of dispersion, if not fixed by DISPERSION option (deviance, Pearson); default devi
FUNCTIONVALUE = *scalar*	Scalar whose value is to be minimized by calculation; default *
YRELATION = *string token*	Whether to analyse the y-variates separately, as in ordinary regression, or to analyse them cumulatively as counts in successive categories of a multinomial distribution (separate, cumulative); default sepa
DCALCULATION = *expression structures*	Calculations to define the deviance contributions and variance function for a non-standard distribution; must be specified when DIST=calc
LCALCULATION = *expression structures*	Calculations to define the fitted values and link derivative for a non-standard link; must be specified when LINK=calc
DFDISPERSION = *scalar*	Allows you to specify the number of degrees of freedom for a dispersion parameter specified by the DISPERSION option; if this is not set, the supplied dispersion is assumed to be known exactly
SAVE = *identifier*	To name regression save structure; default *

Parameters

Y = *variates*	Response variates; only the first is used in nonlinear models and in generalized linear models except when DIST=mult, when they specify the numbers in each category of an ordinal response model

NBINOMIAL = *variate* or *scalar*	Total numbers for DIST=bino
RESIDUALS = *variates*	To save residuals for each y variate after fitting a model
FITTEDVALUES = *variates*	To save fitted values, and provide fitted values if no terms are given in FITNONLINEAR
LINEARPREDICTOR = *variate*	Specifies the identifier of the variate to hold the linear predictor
DERIVATIVE = *variate*	Specifies the identifier of the variate to hold the derivative of the link function at each unit
DEVIANCE = *variate*	Specifies the identifier of the variate to hold the contribution to the deviance from each unit
VFUNCTION = *variate*	Specifies the identifier of the variate to hold the value of the variance function at each unit

MONOTONIC directive

Fits an increasing monotonic regression of y on x.

No options

Parameters

Y = *variates*	Y-values of the data points
X = *variates*	X-values of the data points; default is to assume that the x-values are monotonically increasing
RESIDUALS = *variates*	Variate to save the residuals from each fit
FITTEDVALUES = *variates*	Variate to save the fitted values from each fit

MOVINGAVERAGE procedure

Calculates and plots the moving average of a time series (R.P. Littlejohn, G. Tunnicliffe Wilson & D.B. Baird).

Options

PRINT = *string token*	What to print (parameters); default * i.e. nothing
NSAMPLES = *scalar*	Number of samples used to calculate each moving average
METHOD = *string token*	How to calculate the averages (past, centred, exponential, filter, holtwinters) default past
ORDER = *scalars*	Order for polynomial smoothing (0, 1, 2, 3, 4); default 0 i.e. ordinary moving-averages calculated from means
TRIM = *string token*	Whether to trim transients with METHOD settings past or centre when ORDER=0 (yes, no); default no
PLOT = *string token*	What to plot (components, movingaverages); default * i.e. nothing
ALPHA = *scalar*	Allows the smoothing parameter for the contribution of the last value in the series to the moving average to be specified for the exponential or Holt-Winters methods
BETA = *scalar*	Allows the smoothing parameter for the trend to be specified for the Holt-Winters method
GAMMA = *scalar*	Allows the smoothing parameter for the seasonal component to be specified for the Holt-Winters method
MULTIPLICATIVE = *string token*	Controls whether the seasonal component is multiplicative in the Holt-Winters method (yes, no); default no

Parameters

SERIES = *variates*	Time series whose moving averages are required
MASERIES = *pointers*	Saves the moving averages for the defined ORDER settings
TITLE = *texts*	Title for the graph
SEASONAL = *factors*	Factor for seasonal adjustment
SAVE = *pointers*	Saves results from the Holt-Winters method or from seasonal adjustment

MPOLISH procedure

Performs a median polish of two-way data (D.B. Baird).

Options

MAXCYCLE = *scalar*	Maximum number of iterations; default 50
TOLERANCE = *scalar*	Tolerance for convergence; default 0.0001

Parameters

DATA = *variates* or *pointers* or *matrices* or *tables*	Two-way data to be polished
ROWS = *factors*	Row definitions for a DATA variate
COLUMNS = *factors*	Column definitions for a DATA variate
ROWEFFECTS = *variate*	Row effects removed from polished results
COLEFFECTS = *variate*	Column effects removed from polished results
POLISH = *variates* or *pointers* or *matrices* or *tables*	Polished result in same format as DATA
CENTRE = *scalars*	Estimate of overall centre point

MPOWER procedure

Forms integer powers of a square matrix (P.W. Lane).

No options

Parameters

MATRIX = *matrices*, *symmetric matrices* or *diagonal matrices*	Matrix from which to form the power
POWER = *scalars*	Power to which each matrix is to be raised
RESULT = *identifiers*	Structure to store the result

MSEKERNEL2D procedure

Estimates the mean square error for a kernel smoothing (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	What to print (summary); default summ

Parameters

Y = *variates*	Vertical coordinates of each spatial point pattern; no default – this parameter must be set
X = *variates*	Horizontal coordinates of each spatial point pattern; no default – this parameter must be set
YPOLYGON = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
XPOLYGON = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
NSTEP = *scalars*	How many values of the kernel width to use; no default – this parameter must be set
HMAX = *scalars*	Maximum values for the kernel width; no default – this parameter must be set
HVALUES = *variates*	Variates to receive the values of the kernel width
MSE = *variates*	Variates to receive the estimated mean square error for each value of the kernel width

MTABULATE procedure

Forms tables classified by multiple-response factors (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (counts, totals, nobservations, means, minima, maxima, variances, quantiles, sds, skewness, kurtosis, semeans, seskewness, sekurtosis); default * i.e. none
CLASSIFICATION = *factors*	Non multiple-response factors classifying the tables
MRESPONSE = *pointers*	Pointers to factors defining the multiple-responses for the

tables

MRFACTOR = *identifiers* — Identifier of factors to index the sets of multiple responses in the tables

COUNTS = *table* — Saves a table counting the number of units with each factor combination; default *

MARGINS = *string token* — Whether the tables should be given margins (yes, no); default no

WEIGHTS = *variate* — Weights to be used in the tabulations; default * indicates that all units have weight 1

PERCENTQUANTILES = *scalar* or *variate* — Percentages for which quantiles are required; default 50 i.e. median

Parameters

DATA = *variates* — Data values to be tabulated

TOTALS = *tables* — Tables to contain totals

NOBSERVATIONS = *tables* — Tables containing the numbers of non-missing values in each cell

MEANS = *tables* — Tables of means

MINIMA = *tables* — Tables of minimum values in each cell

MAXIMA = *tables* — Tables of maximum values in each cell

VARIANCES = *tables* — Tables of cell variances

QUANTILES = *tables* or *pointers* — Table to contain quantiles at a single PERCENTQUANTILE, or pointer of pointers to tables for several PERCENTQUANTILES

SDS = *tables* — Tables of standard deviations

SKEWNESS = *tables* — Tables of skewness coefficients

KURTOSIS = *tables* — Tables of kurtosis coefficients

SEMEANS = *tables* — Tables of standard errors of means

SESKEWNESS = *tables* — Tables of standard errors of skewness

SEKURTOSIS = *tables* — Tables of standard errors of kurtosis

MULTMISSING procedure

Estimates missing values for units in a multivariate data set (H.R. Simpson & R.P. White).

Option

MAXCYCLE = *scalar* — Defines the maximum allowed number of iterations; default 10

Parameters

DATA = *pointers* — Each pointer contains a set of variates whose missing values are to be estimated; these will be overwritten by the estimates unless the OUT parameter is specified

OUT = *pointers* — Each pointer contains a set of variates to hold the results

MVAOD procedure

Does an analysis of distance of multivariate data (R.W. Payne & R.P. White).

Options

PRINT = *string tokens* — Controls printed output (aodtable, permutationtest); default aodt

TERMS = *formula* — Model terms to fit in the analysis; must be specified

FACTORIAL = *scalar* — Limit on the number of factors or variates in a term for it to be included in the analysis; default 3

NTIMES = *scalar* — Number of permutations to use in the permutation test; default 999

SEED = *scalar* — Seed for the random number generator used to make the permutations; default 0 continues from the previous generation or (if none) initializes the seed automatically

Parameters

DATA = *symmetric matrices* — Supplies the squared distances between the data points

SSD = *variates* — Saves the sums of squared distances

DF = *variates*	Saves the numbers of degrees of freedom
PRPERMUTATION = *variates*	Saves probabilities from the permutation test
DISTANCES = *pointers*	Contains a symmetric matrix of distances for each model term

MVARIOGRAM procedure

Fits models to an experimental variogram (S.A. Harding & R. Webster).

Options

PRINT = *string tokens*	Controls printed output from the fit (model, summary, estimates, correlations, fittedvalues, monitoring); default mode, summ, esti
MODELTYPE = *string token*	Defines which model to fit (power, boundedlinear, circular, spherical, doublespherical, pentaspherical, exponential, besselk1, gaussian, affinepower, linear, cubic, stable, cardinalsine, matern); default powe
WEIGHTING = *string token*	Method to be used for weighting (counts, cbyvar, equal); default coun
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
SMOOTHNESS = *scalar*	Value of power parameter for the stable model, or ν parameter for the Matern model; default * i.e. estimate
ISOTROPY = *string token*	Defines whether to fit an isotropic or geometrical anisotropic model (isotropic, geometrical); default isot
WINDOW = *scalar*	Window in which to plot a graph; default 0 i.e. no graph
TITLE = *text*	Title for the graph
XUPPER = *scalar*	Upper limit for the *x*-axis in the graph
PENDATA = *scalar*	Pen to be used to plot the data; default 1
PENMODEL = *scalar*	Pen to be used to plot the model; default 2

Parameters

VARIOGRAM = *variates* or *matrices*	Experimental variogram to which the model is to be fitted, as a variate if in only one direction or as a matrix if there are several
COUNTS = *variates* or *matrices*	Counts for the points in each variogram (not required if WEIGHTING=equal)
DISTANCE = *variates* or *matrices*	Mean lag distances for the points in each variogram
DIRECTION = *variates*	Directions in which each variogram was computed
INITIAL = *scalars* or *variates*	Scalar defining initial distance parameter for an isotropic model, or variate with two values for a double-spherical isotropic model, or a variate with three values for a geometrical anisotropic model
ESTIMATES = *variates*	Estimated parameter values
FITTEDVALUES = *variates*	Fitted values
EXIT = *scalars*	Exit status from the nonlinear fitting
SAVE = *pointers*	Saves the model name and estimates in a pointer that can be used in KRIGE

MVFILL procedure

Replaces missing values in a vector with the previous non-missing value in that vector (J.T.N.M. Thissen).

No options

Parameter

VECTORS = *vectors*	Variates, texts or factors whose missing values are replaced by the previous non-missing value of that vector

NAG directive

Calls an algorithm from the NAG Library.

Options

PRINT = *string token*	Controls printed output (algorithms, monitoring); default * i.e. none
NAME = *string token*	Name of the algorithm to call; default * i.e. none
ZDZ = *string token*	Value to be given to zero divided by zero in Genstat expressions defined in the ARGUMENTS (missing, zero); default miss
TOLERANCE = *scalar*	If the scalar is non missing, this defines the smallest non-zero number for use in Genstat expressions defined in the ARGUMENTS; otherwise it accesses the default value, which is defined automatically for the computer concerned
SEED = *scalar*	Seed to use for any random number generation in Genstat expressions defined in the ARGUMENTS; default 0
INDEX = *scalar*	If a Genstat expression defined in the ARGUMENTS has a list of structures before the assignment operator (=), the scalar indicates the position within the list of the structure currently being evaluated

Parameters

ARGUMENTS = *pointer*	Arguments for the call
RESULT = *scalar*	Stores the result for algorithms that take the form of a function rather than a subroutine

NCONVERT procedure

Converts integers between base 10 and other bases (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (number); default numb
METHOD = *string token*	Whether to convert NUMBER to DIGITS or vice versa (tobase, frombase); default toba
BASE = *scalars*	Base to which to convert number; default 2

Parameters

NUMBER = *scalars*	Number in base 10
DIGITS = *pointers*	Digits of the NUMBER in the base specified by the BASE option
SIGN = *scalars*	Sign of the NUMBER

NCSPLINE procedure

Calculates natural cubic spline basis functions for use e.g. in REML (S.J. Welham).

Options

INKNOTS = *variate*	Defines a set of knots to use to construct the spline
METHOD = *string token*	Whether to produce a basis suitable for use with independent or correlated random effects; (independent, correlated); default inde
ORTHOGONALIZETO = *variate*	Variate to use to get an orthogonalized basis; default * i.e. orthogonalization with respect to KNOTS

Parameters

X = *variates*	Values for which the basis functions are calculated
BASIS = *pointers*	Non-linear part of spline basis for use as design matrix for random effects in REML analysis
DBASIS = *pointers*	First derivative of BASIS functions
D2BASIS = *pointers*	Second derivative of BASIS functions
INVCOVARIANCE = *symmetric matrices*	Inverse covariance matrix for use with correlated spline random effects
SECONDDIFFERENCES = *matrices*	Scaled second divided difference matrix associated with KNOTS
KNOTS = *variates*	Knots used in construction of basis

DISTANCES = *variates*	Inter-knot distances used in construction of basis
SCALE = *scalars*	Saves the appropriate value for scaling design matrix

NLAR1 procedure

Fits curves with an AR1 or a power-distance correlation model (R.W. Payne).

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, cparameter, cmonitoring, cplot); default mode, summ, esti, cpar
CURVE = *string token*	Which standard curve to fit (exponential, dexponential, cexponential, lexponential, logistic, glogistic, gompertz, ldl, qdl, qdq, fourier, dfourier, gaussian, dgaussian); default expo
SENSE = *string token*	Sense of a standard curve (right, left); default righ
ORIGIN = *scalars*	Constrained origin for a standard curve; default * i.e. not constrained
NONLINEAR = *string token*	How to treat nonlinear parameters between groups in standard curves (common, separate); default comm
CALCULATION = *expression structures*	Define a nonlinear model involving explanatory variates and nonlinear parameters; default * implies that a standard curve is fitted
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalars*	Limit for expansion of model terms; default 3
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob
SELINEAR = *string token*	Whether to calculate s.e.s for linear parameters when nonlinear parameters are also estimated (yes, no); default no
WEIGHTS = *variate*	Prior weights for the units
CPARAMETER = *scalars*	Correlation parameter
CPOSITIONS = *variate*	Correlation positions
CGROUPS = *factor*	Groupings of correlation positions
MAXCYCLE = *scalars*	Maximum number of iterations; default 100
TOLERANCE = *scalars*	Convergence criterion; default 10^{-5}

Parameter

TERMS = *formula*	Terms to be fitted

NLCONTRASTS procedure

Fits nonlinear contrasts to quantitative factors in ANOVA (R.C. Butler).

Options

PRINT = *string tokens*	Printed output required (aovtable, information, covariates, effects, residuals, contrasts, means, %cv, missingvalues); default aovt, info, cova, mean, miss

CURVE = *string token*	Curve (as in FITCURVE) to use for nonlinear regression (exponential, dexponential, cexponential, lexponential, logistic, glogistic, gompertz, ldl, qdl, qdq); default expo
FPROBABILITY = *string token*	Printing of probabilities for variance ratios (yes, no); default no
PSE = *string token*	Standard errors to print with means tables (differences, means); default diff
WEIGHT = *variate*	Variate of weights for each unit; default * (no weights)

Parameters

Y = *variates*	Data to be analysed
XFACTOR = *factors*	Factor with quantitative levels for which contrasts are to be found
XLEVELS = *variates*	Variate of values to use for the levels of XFACTOR; if unset, the factor levels themselves are used
GROUPFACTOR = *factors*	Factor whose interaction with XFACTOR is to be assessed
CONTRASTS = *pointers*	Structures to hold the estimates of the fitted contrasts: CONTRASTS[1] is a pointer with two values, labelled 'Curve' (parameter estimates for a single fitted curve) and 'Deviations' (the differences between this curve and the means for XFACTOR); CONTRASTS[2] has three values, labelled 'Common NonLin' (parameter estimates for curves fitted with common nonlinear parameters for all levels of GROUPFACTOR), 'Separate Curves' (parameter estimates for curves fitted with all parameters varying with the levels of GROUPFACTOR) and 'Deviations' (differences between the treatment means and the Separate Curves); the order of the parameters is as in the output of the procedure, the variates of estimated contrasts are labelled by the parameter names as used in the printed output, while the 'Deviations' are both tables, labelled by the relevant factors
SECONTRASTS = *pointers*	Structures to save the standard errors for the contrast estimates, including 'deviations'; the pointer has the same form as the CONTRASTS pointer
DFCONTRASTS = *pointers*	Structures to save the degrees of freedom for the contrast estimates; the pointer has the same form as the CONTRASTS pointer, except that the variates and tables are replaced by scalars

NNDISPLAY directive

Displays output from a multi-layer perceptron neural network fitted by NNFIT.

Option

PRINT = *string tokens*	Controls fitted output (description, estimates, fittedvalues, summary); default desc, esti, summ

Parameter

pointers	Save structure with details of the network and the estimated parameters

NNFIT directive

Fits a multi-layer perceptron neural network.

Options

PRINT = *string tokens*	Controls fitted output (description, estimates, fittedvalues, summary); default desc, esti, summ
NHIDDEN = *scalar*	Number of functions in the hidden layer; no default, must be set
HIDDENMETHOD = *string token*	Type of activation function in the hidden layer (logistic,

	`hyperbolictangent`); default `logi`
`OUTPUTMETHOD` = *string token*	Type of activation function in the output layer (`linear`, `logistic`, `hyperbolictangent`); default `line`
`GAIN` = *scalar*	Multiplicative constant to use in the functions; default 1
`NTRIES` = *scalar*	Number of times to search for a good initial starting point for the optimization; default 5
`NSTARTITERATIONS` = *scalar*	Number of iterations to use to find a good starting point for the optimization; default 30
`VALIDATIONOPTIONS` = *variate*	Variate containing three integers to control validation for early stopping; default `*` i.e. no early stopping; default `!(10,4,16)`
`SEED` = *scalar*	Seed for random numbers to generate initial values for the free parameters; default 0
`MAXCYCLE` = *scalar*	Maximum number of iterations of the conjugate-gradient algorithm; default 50

Parameters

`Y` = *variates*	Response variates
`X` = *pointers*	Input variates
`YVALIDATION` = *variates*	Validation data for the dependent variates
`XVALIDATION` = *pointers*	Validation data for the independent variates
`FITTEDVALUES` = *variates*	Fitted values generated for each y-variate by the neural network
`NCOMPLETED` = *scalars*	Number of completed iterations of the conjugate-gradient algorithm
`EXIT` = *scalars*	Saves the exit code
`SAVE` = *pointers*	Saves details of the network and the estimated parameters

`NNPREDICT` directive

Forms predictions from a multi-layer perceptron neural network fitted by `NNFIT`.

Option

`PRINT` = *string tokens*	Controls fitted output (`description`, `predictions`); default `desc`, `pred`

Parameters

`X` = *pointers*	Input variates
`PREDICTIONS` = *variates*	Predictions
`SAVE` = *pointers*	Details of the network

`NORMTEST` procedure

Performs tests of univariate and/or multivariate normality (M.S. Ridout).

Option

`PRINT` = *string tokens*	Allows the required printed output to be selected: test statistics, tables of critical values and the flagging of significant values with stars (`marginal`, `bivariateangle`, `radius`, `critical`, `stars`); default `marg`, `biva`, `radi`

Parameter

`DATA` = *variates* or *pointers*	Variates whose univariate normality is to be tested or pointers, each to a set of variates whose normality and/or multivariate normality are to be tested

`NOTICE` procedure

Provides news and other information about Genstat (R.W. Payne).

Option

`PRINT` = *string tokens*	Indicates what information is required (`news`, `release`, `errors`, `instructions`); default `news`

No parameters

OPEN directive

Opens files.

No options

Parameters

NAME = *texts*	External names of the files
CHANNEL = *scalars*	Channel number to be used to refer to each file in other statements (numbers for each type of file are independent); if this is set to a scalar containing a missing value, the first available channel of the specified type is opened and the scalar is set to the channel number
FILETYPE = *string tokens*	Type of each file (input, output, unformatted, backingstore, procedurelibrary, graphics); default inpu
WIDTH = *scalars*	Maximum width of a record in each file; default 80
INDENTATION = *scalar*	Number of spaces to leave at the start of each line; default 0
PAGE = *scalars*	Number of lines per page (relevant only for output files)
ACCESS = *string token*	Allowed type of access (readonly, writeonly, both); default both
STYLE = *string token*	Style in which to write to an output file (plaintext, html, latex, rtf); default plai
HTMLHEAD = *texts*	Text structures containing custom content for the header of an HTML document

OPLS procedure

Performs orthogonal partial least squares regression (V. M. Cave).

Options

PRINT = *string tokens*	Printed output required (data, xloadings, yloadings, ploadings, scores, leverages, xerrors, yerrors, scree, xpercent, ypercent, predictions, groups, estimates, fittedvalues, summary); default esti, xper, yper, scor, xloa, yloa, ploa, summ
PCPRINT = *string tokens*	Controls printed output from principal components analysis of orthogonal X matrix (loadings, roots, scores, tests); default root
PLOT = *string token*	What graphs to plot (pcplot); default * (i.e. none)
NORTHOGONALROOTS = *scalar*	Number of orthogonal components to extract; default 1
NROOTS = *scalar*	Number of predictive (i.e. PLS) components to extract; default 1
STANDARDIZE = *string tokens*	Whether to standardize the Y, X and filtered X variables to unit variance and zero mean (Y, X, filteredX); default * (i.e. no standardizing)
NGROUPS = *scalar*	Number of cross-validation groups used by PLS; default 1 (i.e. no cross-validation performed)
SEED = *scalar* or *factor*	A scalar indicating the seed value used for dividing the data randomly into NGROUPS groups for cross-validation by PLS, or a factor indicating a specific set of groupings to use for cross-validation by PLS; default 0
LABELS = *text*	Sample labels for X and Y to use in output; default uses the integers 1...*n* where *n* is the length of the variates in X and Y
PLABELS = *text*	Labels for XPREDICTIONS; default uses P1, P2 etc.
PCMETHOD = *string tokens*	Method used by PCP to perform principal components analysis on the orthogonal X matrix (ssp, correlation, vcovariance, variancecovariance); default * (i.e. principal components analysis not performed)
WINDOW = *scalar*	Window to use for graph (available only when NORTHOGONALROOTS = 1); default 3

Parameters

Y = *pointers*	Pointer to variates containing the dependent variable(s) for each analysis
X = pointers	Pointer to variates containing the independent variables for each analysis
YLOADINGS = *pointers*	Pointer to variates containing the Y component loadings, for the predictive (i.e. PLS) dimensions, extracted from the filtered X matrix
XLOADINGS = *pointers*	Pointer to variates containing the component loading weights for the predictive dimensions, extracted from the filtered X matrix
PLOADINGS = *pointers*	Pointer to variates containing the bilinear model loadings for the predictive dimensions, extracted from the filtered X matrix
YSCORES = *pointers*	Pointer to variates containing the Y component scores, for each predictive dimension extracted from the filtered X matrix
XSCORES =*pointers*	Pointer to variates containing the component scores for each predictive dimension, extracted from the filtered X matrix
B = *diagonal matrices*	Saves the regression coefficients of YSCORES on XSCORES, for the predictive dimensions, extracted from the filtered X matrix
YPREDICTIONS = *pointer*	Pointer to variates used to store predicted y-values for samples in the prediction set
XPREDICTIONS = *pointer*	Pointer to variates containing data for the independent variables in the prediction set
ESTIMATES = *matrices*	An n_X+1 by n_Y matrix (where n_X and n_Y are the number of variates contained in X and Y, respectively) to store the PLS regression coefficients
FITTEDVALUES = *pointers*	Pointer to variates used to store the fitted values for the Y variates
LEVERAGES = *variates*	Variate to store the leverage that each sample has on the PLS model
PRESS = *variates*	Variate used to store the Predictive Residual Error Sum of Squares for each dimension in the PLS model, available only if cross-validation has been selected
RSS = *variates*	Variate to save residual sums of squares
YRESIDUALS = *pointers*	Pointer to variates containing the residuals from the Y block after NROOTS predictive dimensions have been extracted, uncorrected for any scaling applied using STANDARDIZE
XRESIDUALS = *pointers*	Pointer to variates containing the residuals from the X block after NROOTS predictive dimensions have been extracted, uncorrected for any scaling applied using STANDARDIZE
PCSCORES = *matrices*	Matrix to save principal component scores
PCSAVE = *pointers*	Pointer to save structures from the principal component analysis (by PCP) of the orthogonal X matrix
SAVE = *pointers*	Pointer to save structures from the orthogonal projection

OPTION directive

Defines the options of a Genstat procedure with information to allow them to be checked when the procedure is executed.

No options

Parameters

NAME = *texts*	Names of the options
MODE = *string tokens*	Mode of each option (e, f, p, t, v, as for unnamed structures); default p
NVALUES = *scalars* or *variates*	Specifies allowed numbers of values
VALUES = *variates* or *texts*	Defines the allowed values for a structure of type variate or text

DEFAULT = *identifiers*	Default values for each option
SET = *string tokens*	Indicates whether or not each option must be set (yes, no); default no
DECLARED = *string tokens*	Indicates whether or not the setting of each option must have been declared (yes, no); default no
TYPE = *texts*	Text for each option, whose values indicate the types allowed (ASAVE, datamatrix {i.e. pointer to variates of equal lengths as required in multivariate analysis}, diagonalmatrix, dummy, expression, factor, formula, LRV, matrix, pointer, RSAVE, scalar, SSPM, symmetricmatrix, table, text, tree, TSAVE, TSM, variate, VSAVE); default * meaning no limitation
COMPATIBLE = *texts*	Defines aspects to check for compatibility with the first parameter of the directive or procedure (nvalues, nlevels, nrows, ncolumns, type, levels, labels {of factors or pointers}, mode, rows, columns, classification, margins, associatedidentifier, suffixes {of pointers}, restriction)
PRESENT = *string tokens*	Indicates whether or not each structure must have values (yes, no); default no
LIST = *string tokens*	Whether to allow a list of identifiers (MODE=p) or of values (MODE=v or t) instead of just one (yes, no); default no
INPUT = *string token*	Whether the option only supplies input information to the procedure (yes, no); default no

OR directive

Introduces a set of alternative statements in a "multiple-selection" control structure.

No options or parameters

ORTHPOLYNOMIAL procedure

Calculates orthogonal polynomials (P.W. Lane).

Options

MAXDEGREE = *scalar*	Maximum degree of polynomial to be calculated; default is the number of identifiers in the pointer specified by the POLYNOMIAL parameter
WEIGHTS = *variate*	Weights to be used in orthogonalization; default * gives an equal weight to each unit

Parameters

X = *variates*	Values from which to calculate the polynomials; no default – this parameter must be set
POLYNOMIAL = *pointers*	Identifiers of variates to store results; no default – this parameter must be set

OUTPUT directive

Defines where output is to be stored or displayed.

Options

PRINT = *string tokens*	Additions to output (dots, page, unchanged); default dots,page
DIAGNOSTIC = *string tokens*	What diagnostic printing is required (messages, warnings, faults, extra, unchanged); default faul,mess,warn
WIDTH = *scalar*	Limit on number of characters per record; default width of output file
INDENTATION = *scalar*	Number of spaces to leave at the start of each line; default 0
PAGE = *scalar*	Number of lines per page
STYLE = *string token*	Style for future output to the channel (plaintext, formatted); default * i.e. unchanged

Parameter

scalar	Channel number of output file

OWN directive

Does work specified in Fortran subprograms linked into Genstat by the user.

Option

SELECT = *scalar*	Sets a switch, designed to allow OWN to be used for many applications; standard set-up assumes a scalar in the range 0-9; default 0

Parameters

IN = *identifiers*	Supplies input structures, which must have values, needed by the auxiliary subprograms
OUT = *identifiers*	Supplies output structures whose values or attributes are to be defined by the auxiliary subprograms

PAGE directive

Moves to the top of the next page of an output file.

Option

CHANNEL = *scalar*	Channel number of file; default * i.e. current output file

No parameters

PAIRTEST procedure

Performs t-tests for pairwise differences (P.W. Goedhart).

Options

PRINT = *string tokens*	What to print (differences, sed, tvalues, tprobabilities); default diff, sed, tval
DF = *scalar*	Degrees of freedom for calculation of TPROBABILITIES from TVALUES; default 10000, approximates to the normal distribution
SORT = *string token*	Whether ESTIMATES (and other output) are sorted in ascending order (yes, no); default no

Parameters

ESTIMATES = *variates*	Estimates to be compared
VCOVARIANCE = *symmetric matrices*	Symmetric matrix containing the variance-covariance matrix of the estimates
LABELS = *texts*	Text vector naming the elements of ESTIMATES; if unset, the numbers 1, 2... are used as labels
DIFFERENCES = *symmetric matrices*	To save the pairwise differences (ESTIMATES on the diagonal)
SED = *symmetric matrices*	To save the standard errors of the pairwise differences (missing values on the diagonal)
TVALUES = *symmetric matrices*	To save the t-values (missing values on the diagonal)
TPROBABILITIES = *symmetric matrices*	To save the t-probabilities (missing values on the diagonal)

PARAMETER directive

Defines the parameters of a Genstat procedure with information to allow them to be checked when the procedure is executed.

No options

Parameters

NAME = *texts*	Names of the parameters
MODE = *string tokens*	Mode of each parameter (e, f, p, t, v, as for unnamed structures); default p
NVALUES = *scalars* or *variates*	Specifies allowed numbers of values
VALUES = *variates* or *texts*	Defines the allowed values for a structure of type variate or text
DEFAULT = *identifiers*	Default values for each parameter

SET = *string tokens*	Indicates whether or not each parameter must be set (yes, no); default no
DECLARED = *string tokens*	Indicates whether or not the setting of each parameter must have been declared (yes, no); default no
TYPE = *texts*	Text for each option, whose values indicate the types allowed (ASAVE, datamatrix {i.e. pointer to variates of equal lengths as required in multivariate analysis}, diagonalmatrix, dummy, expression, factor, formula, LRV, matrix, pointer, RSAVE, scalar, SSPM, symmetricmatrix, table, text, TSAVE, TSM, variate); default * meaning no limitation
COMPATIBLE = *texts*	Defines aspects to check for compatibility with the first parameter of the directive or procedure (nvalues, nlevels, nrows, ncolumns, type, levels, labels {of factors or pointers}, mode, rows, columns, classification, margins, associatedidentifier, suffixes {of pointers}, restriction)
PRESENT = *string tokens*	Indicates whether or not each structure must have values (yes, no); default no
INPUT = *string token*	Whether the parameter only supplies input information to the procedure (yes, no); default no

PARTIALCORRELATIONS procedure

Calculates partial correlations for a list of variates (S. Langton).

Options

PRINT = *string token*	Output required (correlations); default corre
CORRELATIONS = *symmetric matrix*	Saves the partial correlations
WEIGHTS = *variate*	Supplies weights for the units; default * i.e. all 1

Parameters

DATA = *variates*	Set of variates whose partial correlations are to be calculated

PASS directive

Performs tasks specified in subprograms supplied by the user, but not linked into Genstat; this directive may not be available on some computers.

Option

NAME = *text*	Filename of external executable program; default 'GNPASS'

Parameter

pointers	Structures whose values are to be passed to the external program, and returned

PCO directive

Performs principal coordinates analysis, also principal components and canonical variates analysis (but with different weighting from that used in CVA) as special cases.

Options

PRINT = *string tokens*	Printed output required (roots, scores, loadings, residuals, centroid, distances); default * i.e. no printing
NROOTS = *scalar*	Number of latent roots for printed output; default * requests them all to be printed
SMALLEST = *string token*	Whether to print the smallest roots instead of the largest (yes, no); default no

Parameters

DATA = *identifiers*	These can be specified either as a symmetric matrix of similarities or transformed distances or, for the canonical variates analysis, as an SSPM containing within-group sums of squares and products etc or, for principal components analysis, either as a pointer containing the variates of the data matrix or

	as a matrix storing the variates by columns
LRV = *LRVs*	Latent vectors (i.e. coordinates or scores), roots, and trace from each analysis
CENTROID = *diagonal matrices*	Squared distances of the units from their centroid
RESIDUALS = *matrices* or *variates*	Distances of the units from the fitted space
LOADINGS = *matrices*	Principal component loadings, or canonical variate loadings
DISTANCES = *symmetric matrices*	Computed inter-unit distances calculated from the variates of a data matrix, or inter-group Mahalanobis distances calculated from a within-group SSPM
SAVE = *pointers*	Saves details of the analysis; if unset, an unnamed save structure is saved automatically (and this can be accessed using the GET directive)

PCOPROCRUSTES procedure

Performs a multiple Procrustes analysis (P.G.N. Digby).

Options

PROTATE = *string tokens*	Printed output required from each Procrustes rotation (rotations, coordinates, residuals, sums); default * i.e. no output
PPCO = *string tokens*	Printed output required from the PCO analysis (roots, scores, centroid); default root, score, cent
SCALING = *string token*	Whether isotropic scaling should be used for the Procrustes rotations (no, yes); default no
STANDARDIZE = *string tokens*	Whether to centre the configurations and/or normalize them to unit sums-of-squares for the Procrustes rotations (centre, normalize); default cent, norm

Parameters

DATA = *pointers*	Each pointer points to a set of matrices holding the original input configurations
LRV = *LRVs*	Stores the latent vectors (i.e. coordinates), roots and trace from the PCO analysis
CENTROID = *diagonal matrices*	Stores the squared distances of the points representing the input configurations from their overall centroid from the PCO analysis
DISTANCES = *symmetric matrices*	Stores the residual sums-of-squares from the Procrustes rotations

PCORELATE directive

Relates the observed values on a set of variates or factors to the results of a principal coordinates analysis.

Options

COORDINATES = *matrix*	Points in reduced space; no default i.e. this option must be specified
NROOTS = *scalar*	Number of latent roots for printed output; default * requests them all to be printed

Parameters

DATA = *variates* or *factors*	The data variables
TEST = *string tokens*	Test type, defining how each variable is treated in the calculation of the similarity between each unit (simplematching, jaccard, russellrao, dice, antidice, sneathsokal, rogerstanimoto, cityblock, manhattan, ecological, euclidean, pythagorean, minkowski, divergence, canberra, braycurtis, soergel); default * ignores that variable
RANGE = *scalars*	Range of possible values of each variable; if omitted, the observed range is taken

PCP directive

Performs principal components analysis.

Options

PRINT = *string tokens*	Printed output required (loadings, roots, residuals, scores, tests); default * i.e. no printing
NROOTS = *scalar*	Number of latent roots for printed output; default * requests them all to be printed
SMALLEST = *string token*	Whether to print the smallest roots instead of the largest (yes, no); default no
METHOD = *string token*	Whether to use sums of squares, correlations or variances and covariances (ssp, correlation, vcovariance, variancecovariance); default ssp

Parameters

DATA = *pointers* or *matrices* or *SSPMs*	Pointer of variates forming the data matrix, or matrix storing the variate values by columns, or SSPM giving their sums of squares and products (or correlations) etc
LRV = *LRVs*	To store the principal component loadings, roots, and trace from each analysis
SSPM = *SSPMs*	To store the computed sum-of-squares-and-products or correlation matrix
SCORES = *matrices*	To store the principal component scores
RESIDUALS = *matrices* or *variates*	To store residuals from the dimensions fitted in the analysis (i.e. number of columns of the SCORES matrix, or as defined by the NROOTS option)
SAVE = *pointers*	Saves details of the analysis; if unset, an unnamed save structure is saved automatically (and this can be accessed using the GET directive)

PDESIGN procedure

Prints or stores treatment combinations tabulated by the block factors (R.W. Payne).

Options

PRINT = *string token*	Controls the printing of the design (design); default desi
BLOCKSTRUCTURE = *formula*	Defines the block factors for the design; the default is to take those specified by the BLOCKSTRUCTURE directive
TREATMENTSTRUCTURE = *formula*	Defines the treatment factors for each design; the default is to take those specified by the TREATMENTSTRUCTURE directive
TABLES = *pointer*	Contains tables to store the tabulated factor values for printing outside the procedure in some other format
FREPRESENTATION = *string token*	How to represent the factor values (labels, levels); default leve

No parameters

PDUPLICATE procedure

Duplicates a pointer, with all its components (R.W. Payne).

No options

Parameters

OLDPOINTER = *pointers*	Pointers to duplicate
NEWPOINTER = *pointers*	Duplicated pointers

PEAKFINDER procedure

Finds the locations of peaks in an observed series (D.B. Baird).

Options

PRINT = *string token*	Controls printed output (peaks); default peak
CURVE = *string token*	Shape of curve to fit to peaks (normal, exponential); default norm

PLOT = *string tokens* — What to plot (peaks, trace); default peak
METHOD = *string token* — The method for finding the peaks (additive, local); default addi
BANDWIDTH = *scalar* — Width of window to use when fitting peaks locally, or the number of low points at the edge of each zone when fitting peaks additively; default takes the number of points divided by ten, or six if this is greater
MINPEAK = *scalar* — Minimum height of a peak; no default (must be set)
MINGAP = *scalar* — Minimum number of points between two peaks when METHOD=additive; default 5
MINFALL = *scalar* — Minimum fall around a peak before a new peak will be found when METHOD=additive; default MINPEAK/10
MINCOHERENCY = *scalar* — Minimum coherency (i.e. proportion of variation explained) for a peak to be selected when METHOD=local; default 0.1
MAXSIGMA = *scalar* — The maximum value of sigma for peaks when METHOD=local; default 4*BANDWIDTH
MAXRESIDUAL = *scalar* — Limit on the absolute size of any residual for the adding of peaks to stop when METHOD=additive; default MINPEAK/3
WINDOW = *scalar* — Window number for the plots; default 3
SCREEN = *string token* — Whether to clear the screen before plotting or continue plotting on the old screen (clear, keep); default clea

Parameters

Y = *variates* — Series to search for peaks
X = *variates* — X-coordinates for the series; default !(1...n) where n is the number of Y values
YPEAKS = *variates* — Saves the y-values of the peaks
XPEAKS = *variates* — Saves the positions of the peaks
FITTEDYPEAKS = *variates* — Saves the heights of the peaks predicted by the fitted models
SIGMA = *variates* — Saves the sigma values of the fitted Normal or exponential models, which provide a measure of the widths of the peaks
COHERENCY = *variates* — Saves the coherency (i.e. the proportion of variation accounted for) of the model fitted to identify each peak model
TITLE = *texts* — Titles for the plots

PEN directive

Defines the properties of "pens" for high-resolution graphics.

Option

RESET = *string token* — Whether to reset the pen definitions to their default values (yes, no); default no
BOXUNITS = *string token* — Units to use for text boxes (characters, distance); the default is to retain the existing setting

Parameters

NUMBER = *scalars* — Numbers associated with the pens
COLOUR = *texts* or *scalars* — Colour to use with each pen unless otherwise specified by the CSYMBOL, CLINE, CFILL or CAREA parameters
LINESTYLE = *texts* or *scalars* — Style for line used by each pen when joining points
METHOD = *string tokens* — Method for determining line (point, line, monotonic, closed, open, fill, spline, polygon)
SYMBOL = *texts*, *scalars*, *pointers* or *matrices* — Defines the plotting symbol for each pen, by a text or scalar for a pre-defined symbol, a pointer for a user-defined symbol, or a matrix to supply a bitmap
LABELS = *texts* or *factors* — Define labels that will be printed alongside the plotting symbols
ROTATION = *scalars* or *variates* — Rotation required for the plotting symbols and labels (in degrees)

JOIN = *string tokens*	Order in which points are to be joined by each pen (ascending, given)
BRUSH = *scalars*	Number of the type of area filling used with each pen when drawing pie charts or histograms
FONT = *texts* or *scalars*	Font to be used for any text written by each pen
THICKNESS = *scalars*	Thickness with which any lines are drawn by each pen
SIZEMULTIPLIER = *scalars* or *variates*	Multiplier to used in the calculation of the size in which to draw symbols and labels by each pen unless otherwise specified by SMSYMBOL or SMLABEL
CSYMBOL = *texts* or *scalars*	Colour to use with each pen when drawing symbols
CLINE = *texts* or *scalars*	Colour to use with each pen when drawing lines
CFILL = *texts* or *scalars*	Colour to use with each pen when filling areas inside hollow symbols
CAREA = *texts* or *scalars*	Colour to use with each pen when filling areas inside polygons and bars of histograms
SMSYMBOL = *scalars* or *variates*	Multiplier used in the calculation of the size in which to draw symbols by each pen
SMLABEL = *scalars* or *variates*	Multiplier used in the calculation of the size in which to draw labels by each pen
DFSPLINE = *scalars*	Number of degrees of freedom to use when METHOD=spline
YMISSING = *string token*	How to treat missing y-values when METHOD=spline (break, interpolate)
XMISSING = *string token*	How to treat missing x-values when METHOD=spline (break, ignore)
YLPOSITION = *string token*	How to position labels in the y-direction with respect to the points (above, centre, below, automatic)
XLPOSITION = *string token*	How to position labels in the x-direction with respect to the points (left, centre, right, automatic)
YLSIZE = *scalars* or *variates*	Widths in the y-direction of the text boxes into which to plot labels
XLSIZE = *scalars* or *variates*	Widths in the x-direction of the text boxes
YLOFFSET = *scalars* or *variates*	Offsets in the y-direction of the text boxes
XLOFFSET = *scalars* or *variates*	Offsets in the x-direction of the text boxes
BARTHICKNESS = *scalars*	Thickness with which any error bars are drawn by each pen
BARCAPWIDTH = *scalars*	Width of the cap drawn by each pen at the top and bottom of any error bars
DESCRIPTION = *texts*	Description for points plotted by the pen, to be used by the Data Information tool in the Graphics Viewer
TSYMBOL = *scalars*	Defines the transparency of symbols drawn by each pen, on a scale of 0 (opaque) to 255 (completely transparent)
TLINE = *scalars*	Defines the transparency of lines drawn by each pen
TFILL = *scalars*	Defines the transparency to use when filling areas inside hollow symbols with each pen
TAREA = *scalars*	Defines the transparency to use when filling areas inside polygons and bars of histograms with each pen
SAVE = *pointers*	Saves details of the current settings for the pen concerned

PENSPLINE procedure

Calculates design matrices to fit a penalized spline as a linear mixed model (S.J. Welham).

Options

KMETHOD = *string token*	Method for constructing the set of knots (equal, quantile, given); default equa
NSEGMENTS = *scalar*	Specifies the number of segments between boundaries; default * obtains a value automatically
INKNOTS = *variate*	Provides the set of knots when KMETHOD=given
DEGREE = *scalar*	Degree of polynomial used to form the underlying spline basis

functions; default 1

LOWER = *scalar* — Specifies the lower boundary when KMETHOD=equal; default takes the minimum value in X

UPPER = *scalar* — Specifies the upper boundary when KMETHOD=equal; default takes the maximum value in X

ORTHOGONALIZETO = *variate* — Variate to use to get an orthogonalized basis; default * i.e. orthogonalization with respect to X

SCALING = *scalar* — Scaling of the XRANDOM terms (automatic, none); default auto

Parameters

X = *variates* — The explanatory variate for which the spline values are required

XFIXED = *matrices* — Saves the design matrix to define the fixed terms (excluding the constant) for fitting the penalized spline

XRANDOM = *matrices* — Saves the design matrix to define the random terms for fitting the penalized spline

KNOTS = *variates* — Saves the internal knots and boundaries used to form the basis for the spline

PX = *variates* — Specifies x-values at which predictions are required

PFIXED = *matrices* — Saves the design matrix for the fixed terms (excluding the constant) for the spline at the prediction points

PRANDOM = *matrices* — Saves the design matrix for the random terms for the spline at the prediction points

PERCENT procedure

Expresses the body of a table as percentages of one of its margins (R.W. Payne).

Options

CLASSIFICATION = *factors* — Factors classifying the margin over which the percentages are to be calculated; if this is not set, the percentages are over the final margin (grand mean or grand total etc.)

METHOD = *string token* — Method to use to calculate the margin if not already present (totals, means, minima, maxima, variances, medians); default tota

HUNDRED = *string token* — Whether to put 100% values into the margin instead of the original values (no, yes); default no

Parameters

OLDTABLE = *tables* — Tables containing the original values

NEWTABLE = *tables* — Tables to store the percentage values; if any of these is unset, the new values replace those in the original table

PERIODTEST procedure

Gives periodogram-based tests for white noise in time series (R.P. Littlejohn).

Option

LENGTH = *scalar* or *variate* — Scalar specifying that the first N units of the series are to be used, or a variate specifying the first and last units of the series to be used

Parameters

SERIES = *variates* — Specify the time series to be analysed

PERIODOGRAM = *variates* — Save periodograms of the time series

PERMUTE procedure

Forms all possible permutations of the integers 1...*n* (J.W. McNicol & R.W. Payne).

Option

SORT = *string token* — Whether or not to sort the permutations (no, yes); default no

Parameters

NVALUES = *scalars* — Specifies the final number, *n*, in the sequence of integers 1...*n*

to be permuted

PERMUTATIONS = *pointers* — Pointer to a set of variates of length NVALUES storing the permutations

PFACLEVELS procedure

Prints levels and labels of factors (R.W. Payne).

No options

Parameter

FACTOR = *factors* — Factors whose levels and labels are to be printed

PLINK procedure

Prints a link to a graphics file into an HTML file (D.A. Murray).

Options

CHANNEL = *scalar* — Output channel number of file; default current output channel

EXCLUDEPATH = *string token* — Whether to remove path information when printing the link (yes, no); default no

Parameter

FILENAME = *texts* — Name of the graphics file to be linked within the html file

PLS procedure

Fits a partial least squares regression model (Ian Wakeling & Nick Bratchell).

Options

PRINT = *string tokens* — Printed output required (data, xloadings, yloadings, ploadings, scores, leverages, xerrors, yerrors, scree, xpercent, ypercent, predictions, groups, estimates, fittedvalues); default esti, xper, yper, scor, xloa, yloa, ploa

NROOTS = *scalar* — Number of PLS dimensions to be extracted

YSCALING = *string token* — Whether to scale the Y variates to unit variance; (yes, no); default no

XSCALING = *string token* — Whether to scale the X variates to unit variance; (yes, no); default no

NGROUPS = *scalar* — Number of cross-validation groups into which to divide the data; default 1 (i.e. no cross-validation performed)

SEED = *scalar* or *factor* — A scalar indicating the seed value to use when dividing the data randomly into NGROUPS groups for the cross-validation or a factor to indicate a specific set of groupings to use for the cross-validation; default 0

LABELS = *text* — Sample labels for X and Y that are to be used in the printed output; defaults to the integers 1...*n* where *n* is the length of the variates in X and Y

PLABELS = *text* — Sample labels for XPREDICTIONS that are to be used in the printed output; default uses the integers 1, 2 ...

Parameters

Y = *pointers* — Pointer to variates containing the dependent variables

X = *pointers* — Pointer to variates containing the independent variables

YLOADINGS = *pointers* — Pointer to variates used to store the *Y* component loadings for each dimension extracted

XLOADINGS = *pointers* — Pointer to variates used to store the *X* component loadings for each dimension extracted

PLOADINGS = *pointers* — Pointer to variates used to store the loadings for the bilinear model for the *X* block

YSCORES = *pointers* — Pointer to variates used to store the *Y* component scores for each dimension extracted

XSCORES = *pointers* — Pointer to variates used to store the *X* component scores for each dimension extracted

B = *matrices*	A diagonal matrix containing the regression coefficients of YSCORES on XSCORES for each dimension
YPREDICTIONS = *pointers*	A pointer to variates used to store predicted *Y* values for samples in the prediction set
XPREDICTIONS = *pointers*	A pointer to variates containing data for the independent variables in the prediction set
ESTIMATES = *matrices*	An n_X+1 by n_Y matrix (where n_X and n_Y are the numbers of variates contained in X and Y respectively) used to store the PLS regression coefficients for a PLS model with NROOTS dimensions
FITTEDVALUES = *pointers*	Pointer to variates used to store the fitted values for each Y variate
LEVERAGES = *variates*	Variate used to store the leverage that each sample has on the PLS model
PRESS = *variates*	Variate used to contain the Predictive Residual Error Sum of Squares for each dimension in the PLS model, available only if cross-validation has been selected
RSS = *variates*	Variate used to store the Residual Sum of Squares for each dimension extracted
YRESIDUALS = *pointers*	Pointer to variates used to store the residuals from the *Y* block after NROOTS dimensions have been extracted, uncorrected for any scaling applied using YSCALING
XRESIDUALS = *pointers*	Pointer to variates used to store the residuals from the *X* block after NROOTS dimensions have been extracted, uncorrected for any scaling applied using XSCALING
XPRESIDUALS = *pointers*	Pointer to variates used to store the residuals from the XPREDICTIONS block after NROOTS dimensions have been extracted
†FTEST = *pointers*	Pointer to save the results from the Osten F test (when NGROUPS > 1)

PNTEST procedure

Calculates one- and two-sample Poisson tests (D.A. Murray).

Options

PRINT = *string tokens*	Controls printed output (test, summary, confidence); default test, summ, conf
METHOD = *string token*	Type of test required (twosided, greaterthan, lessthan); default twos
TEST = *string token*	Form of the test for one-sample test (exact, normalapproximation); default norm
S1 = *scalar*	Sample size for sample 1; default 1
S2 = *scalar*	Sample size for sample 2; default 1
CIPROBABILITY = *scalar*	The probability level for the confidence interval; default 0.95
NULL = *scalar*	The value of the probability of success under the null hypothesis for the one-sample test

Parameters

MU1 = *scalars*	Number recorded in the first sample
MU2 = *scalars*	Number recorded in the second sample
R2 = *scalars*	Sample size of the second sample
NORMAL = *scalars*	Saves the Normal approximation
PROBABILITY = *scalars*	Saves the probability value from the one-sample or two-sample tests
LOWER = *scalars*	Saves the lower limit of the confidence interval
UPPER = *scalars*	Saves the upper limit of the confidence interval

POINTER directive

Declares one or more pointer data structures.

Options

NVALUES = *scalar* or *text*	Number of values, or labels for values; default *
VALUES = *identifiers*	Values for all the pointers; default *
SUFFIXES = *variate* or *scalar*	Defines an integer number for each of the suffixes; default * indicates that the numbers 1,2,... are to be used
CASE = *string token*	Whether to distinguish upper and lower case in the labels of the pointers (significant, ignored); default sign
ABBREVIATE = *string token*	Whether or not to allow the labels to be abbreviated (yes, no); default no
FIXNVALUES = *string token*	Whether or not to prohibit automatic extension of the pointers (yes, no); default no
RENAME = *string token*	Whether to reset the default names of elements of the pointer if they do not have their own identifiers (yes, no); default no
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used by default to identify the pointers in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output
EXTEND = *string token*	Whether to extend (instead of redefining) an existing pointer (yes, no); default no

Parameters

IDENTIFIER = *identifiers*	Identifiers of the pointers
VALUES = *pointers*	Values for each pointer
EXTRA = *texts*	Extra text associated with each identifier

POSSEMIDEFINITE procedure

Calculates a positive semi-definite approximation of a non-positive semi-definite symmetric matrix (L.C.P Keizer, M. Malosetti & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	Controls printed output (approximation, eigenvalues, epsilon); default * i.e. none
EPSILON = *scalar*	Specifies the lowest eigenvalue for the positive semi-definite matrix; default 0.0001

Parameters

OLDSYMMETRICMATRIX = *symmetric matrices*
: Symmetric matrices to approximate

NEWSYMMETRICMATRIX = *symmetric matrices*
: Positive semi-definite approximations to the old symmetric matrices

PPAIR procedure

Displays results of t-tests for pairwise differences in compact diagrams (P.W. Goedhart, H. van der Voet & D.C. van der Werf).

PRINT = *string token*
: What to print (items, groups); default grou

PROBABILITY = *scalar* or *symmetric matrix*
: Level of significance of pairwise comparison tests; default 0.05

Parameters

TPROBABILITIES = *symmetric matrices*
: Probabilities of tests of pairwise comparisons

DIFFERENCES = *symmetric matrices*, *variates* or *tables*
: What to print alongside the labels of TPROBABILITIES; default *

LABELS = *texts*
: Text vector labelling the output; if unset the row labels of

TPROBABILITIES and the diagonal of DIFFERENCES (if set) are used

ITEMLETTERS = *texts* — Saves the letters showing the items not significantly different from each item

GROUPLETTERS = *texts* — Saves the letters showing groups of items not significantly different from each other

PRCORRELATION procedure

Calculates probabilities for product moment correlations (R.W. Payne).

Option

NOBSERVATIONS = *scalar* — Number of observations from which the correlation(s) were calculated

Parameters

DATA = *scalars*, *variates*, *tables*, *matrices*, *diagonal matrices* or *symmetric matrices* — Correlations for calculating probabilities or cumulative lower probabilities for calculating equivalent deviates

CLPROBABILITY = *scalars*, *variates*, *tables*, *matrices*, *diagonal matrices* or *symmetric matrices* — Saves cumulative lower probabilities

CUPROBABILITY = *scalars*, *variates*, *tables*, *matrices*, *diagonal matrices* or *symmetric matrices* — Saves cumulative upper probabilities

PROBABILITY = *scalars*, *variates*, *tables*, *matrices*, *diagonal matrices* or *symmetric matrices* — Saves probability densities

CORRELATION = *scalars*, *variates*, *tables*, *matrices*, *diagonal matrices* or *symmetric matrices* — Saves correlations

PRDOUBLEPOISSON procedure

Calculates the probability density for the double Poisson distribution (V.M. Cave).

Options

PRINT = *string tokens* — Controls printed output (probability, summary); default prob

PLOT = *string token* — Whether to plot the *k* terms used to approximate the normalizing constant by the kpartialsum method (yes, no); default no

METHOD = *string token* — How to approximate the normalizing constant (kpartialsum, edgeworth); default kpar

LOCATION = *scalar* or *variate* — Location parameter; no default, must be set

SHAPE = *scalar* or *variate* — Shape parameter; default 1

MAXCYCLE = *scalar* or *variate* — Limits the number of terms, *k*, used to approximate the normalizing constant by the kpartialsum method; default MAX(1000, 2*LOCATION)

TOLERANCE = *scalar* — Convergence criterion used when approximating the normalizing constant by the kpartialsum method; default 1E-12

Parameters

DATA = *scalar* or *variate* — Non-negative integer values for which the double Poisson probabilities are to be calculated

DECIMALS = *scalars* — Number of decimal places for printing; default *

PROBABILITY = *variate* — Saves the probabilities

PREDICT directive

Forms predictions from a linear or generalized linear model.

Options

PRINT = *string token* — What to print (description, lsd, predictions, se, sed, vcovariance); default desc,pred,se

CHANNEL = *scalar* — Channel number for output; default * i.e. current output channel

COMBINATIONS = *string token*	Which combinations of factors in the current model to include (`full, present, estimable`); default `esti`
ADJUSTMENT = *string token*	Type of adjustment (`marginal, equal`); default `marg`
WEIGHTS = *table*	Weights classified by some or all of the factors in the model; default *
OFFSET = *scalar*	Value of offset on which to base predictions; default mean of offset variate
METHOD = *string token*	Method of forming margin (`mean, total`); default `mean`
ALIASING = *string token*	How to deal with aliased parameters (`fault, ignore`); default `faul`
BACKTRANSFORM = *string token*	What back-transformation to apply to the values on the linear scale, before calculating the predicted means (`link, none`); default `link`
SCOPE = *string token*	Controls whether the variance of predictions is calculated on the basis of forecasting new observations rather than summarizing the data to which the model has been fitted (`data, new`); default `data`
NOMESSAGE = *string tokens*	Which warning messages to suppress (`dispersion, nonlinear`); default *
DISPERSION = *scalar*	Value of dispersion parameter in calculation of s.e.s; default is as set in the `MODEL` statement
DMETHOD = *string token*	Basis of estimate of dispersion, if not fixed by `DISPERSION` option (`deviance, Pearson`); default is as set in the `MODEL` statement
NBINOMIAL = *scalar*	Supplies the total number of trials to be used for prediction with a binomial distribution (providing a value *n* greater than one allows predictions to be made of the number of "successes" out of *n*, whereas the value one predicts the proportion of successes); default 1
PREDICTIONS = *tables* or *scalars*	Saves predictions for each y variate; default *
SE = *tables* or *scalars*	Saves standard errors of predictions for each y variate; default *
SED = *symmetric matrices*	Saves standard errors of differences between predictions for each y variate; default *
LSD = *symmetric matrices*	Saves least significant differences between predictions for each y variate (models with Normal errors only); default *
LSDLEVEL = *scalar*	Significance level (%) to use in the calculation of least significant differences; default 5
VCOVARIANCE = *symmetric matrices*	Saves variance-covariance matrices of predictions for each y variate; default *
SAVE = *identifier*	Specifies save structure of model to display; default * i.e. that from latest model fitted

Parameters

CLASSIFY = *vectors*	Variates and/or factors to classify table of predictions
LEVELS = *variates*, *scalars* or *texts*	To specify values of variates, levels of factors
PARALLEL = *identifiers*	For each vector in the `CLASSIFY` list, allows you to specify another vector in the `CLASSIFY` list with which the values of this vector should change in parallel (you then obtain just one dimension in the table of predictions for these vectors)
NEWFACTOR = *identifiers*	Identifiers for new factors that are defined when `LEVELS` are specified

PREWHITEN procedure

Filters a time series before spectral analysis (A.W.A. Murray).

Option

PHI = *scalar*	Specifies the value of the parameter used in filtering; default

0.99

Parameters

SERIES = *variates*	Input series
FILTERED = *variates*	Output series

PRIMEPOWER procedure

Decomposes a positive integer into its constituent prime powers (I. Wakeling & R.W. Payne).

Option

PRINT = *string token*	Controls printed output (decomposition); default *

Parameters

NUMBER = *scalars*	Number to be decomposed
PRIMES = *pointers*	Prime factors of NUMBER
POWERS = *pointers*	Powers of the prime factors in NUMBER

PRINT directive

Prints data in tabular format in an output file, unformatted file or text.

Options

CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
SERIAL = *string token*	Whether structures are to be printed in serial order, i.e. all values of the first structure, then all of the second, and so on (yes, no); default no, i.e. values in parallel
IPRINT = *string tokens*	What identifier and/or text to print for the structure (identifier, extra, associatedidentifier), for a table associatedidentifier prints the identifier of the variate from which the table was formed (e.g. by TABULATE), IPRINT=* suppresses the identifier altogether; default iden
RLPRINT = *string tokens*	What row labels to print (labels, integers, identifiers), RLPRINT=* suppresses row labels altogether; default labe, iden
CLPRINT = *string tokens*	What column labels to print (labels, integers, identifiers), CLPRINT=* suppresses column labels altogether; default labe, iden
RLWIDTH = *scalar*	Field width for row labels; default 13
INDENTATION = *scalar*	Number of spaces to leave before the first character in the line; default 0
WIDTH = *scalar*	Last allowed position for characters in the line; default width of current output file
SQUASH = *string token*	Whether to omit blank lines in the layout of values (yes, no); default no
MISSING = *text*	What to print for missing value; default uses '*' for numbers and blanks in texts
ORIENTATION = *string token*	How to print vectors or pointers (down, across); default down, i.e. down the page
ACROSS = *scalar* or *factors*	Number of factors or list of factors to be printed across the page when printing tables; default for a table with two or more classifying factors prints the final factor in the classifying set and the notional factor indexing a parallel list of tables across the page, for a one-way table only the notional factor is printed across the page
DOWN = *scalar* or *factors*	Number of factors or list of factors to be printed down the page when printing tables; default is to print all other factors down the page
WAFER = *scalar* or *factors*	Number of factors or list of factors to classify the separate "wafers" (or slices) used to print the tables; default 0
PUNKNOWN = *string token*	When to print unknown cells of tables (present, always,

	zero, missing, never); default pres
UNFORMATTED = *string token*	Whether file is unformatted (yes, no); default no
REWIND = *string token*	Whether to rewind unformatted file before printing (yes, no); default no
WRAP = *string token*	Whether to wrap output that is too long for one line onto subsequent lines, rather than putting it into a subsequent "block" (yes, no); default no
STYLE = *string token*	Style to use for an output file (plaintext, formatted); default * uses the current style of the channel
PMARGIN = *string tokens*	Which margins to print for tables (full, columns, rows, wafers); default full
OMITMISSINGROWS = *string token*	Whether to omit rows of tables that contain only missing values (yes, no); default no
VSPECIAL = *scalar* or *variate*	Special values to be modified in the output
TSPECIAL = *text*	Strings to be used for the special values; must be set if VSPECIAL is set

Parameters

STRUCTURE = *identifiers*	Structures to be printed
FIELDWIDTH = *scalars*	Field width in which to print the values of each structure (a negative value -*n* prints numbers in E-format in width *n*); if omitted, a default is determined (for numbers, this is usually 12; for text, the width is one more character than the longest line)
DECIMALS = *structures*	Number of decimal places for numerical data structures, a scalar if the same number of decimals is to be used for all values of the structure, or a data structure of the same type and size to use different numbers of decimals for each value; if omitted or set to a missing value, a default is determined which prints the mean absolute value to 4 significant figures
CHARACTERS = *scalars*	Number of characters to print in strings
SKIP = *scalars* or *variates*	Number of spaces to leave before each value of a structure (* means a new line before structure)
FREPRESENTATION = *string tokens*	How to represent factor values (labels, levels, ordinals); default is to use labels if available, otherwise levels
JUSTIFICATION = *string tokens*	How to position values within the field (right, left, center, centre); if omitted, right is assumed
MNAME = *string tokens*	Name to print for table margins (margin, total, nobservd, mean, minimum, maximum, variance, count, median, quantile); if omitted, "Margin" is printed
DREPRESENTATION = *scalars* or *texts*	Format to use for dates and times (stored in numerical structures)
HEADING = *texts*	Heading to be used for vectors printed in columns down the page; default is to use the information requested by the IPRINT option
TLABELS = *texts*	If this is specified for a table STRUCTURE, the values of the table are interpreted as references to lines within the TLABELS text that are to be printed instead of the values of the table itself

PRKTAU procedure

Calculates probabilities for Kendall's rank correlation coefficient τ (D.B. Baird).

No options

Parameters

N = *scalars*	Sizes of the first groups of observations
TAU = *scalars*	Values of Kendall's τ statistic
CLPROBABILITY = *scalars*	Cumulative lower probability of TAU

CUPROBABILITY = *scalars*	Cumulative upper probability of TAU
PROBABILITY = *scalars*	Probability density of TAU
LPROBABILITIES = *variates*	Probability densities of -1...TAU
LTAU = *variates*	Values of Tau at corresponding values of LPROBABILITIES

PRMANNWHITNEYU procedure

Calculates probabilities for the Mann-Whitney U statistic (D.B. Baird).

No options

Parameters

N1 = *scalars*	Sizes of the first groups of observations
N2 = *scalars*	Sizes of the second groups of observations
U = *scalars*	Values of the U statistic
TIES = *scalars*	Number of tied observations; default 0
CLPROBABILITY = *scalars*	Cumulative lower probability of U
CUPROBABILITY = *scalars*	Cumulative upper probability of U
PROBABILITY = *scalars*	Probability density of U
LPROBABILITIES = *variates*	Probability densities of 0...U
EXIT = *scalars*	Set to 1 if it has not been possible to calculate the probabilities when there are ties, otherwise 0

PROBITANALYSIS procedure

Fits probit models allowing for natural mortality and immunity (R.W. Payne).

Options

PRINT = *string tokens*	Printed output required (model, summary, estimates, correlations, fittedvalues, monitoring, effectivedoses); default mode, summ, esti, fitt
TRANSFORMATION = *string token*	Transformation to be used (probit, logit, complementaryloglog); default prob
MORTALITY = *string token*	Whether to estimate natural mortality (omit, estimate); default omit
IMMUNITY = *string token*	Whether to estimate natural immunity (omit, estimate); default omit
GROUPS = *factor*	Defines groups for an analysis of parallelism; default * i.e. no groups
SEPARATE = *string tokens*	Which parameters (apart from intercept) should be estimated separately for different groups (slope, mortality, immunity, notintercept); default * i.e. none
LD = *scalar* or *variate*	Effective, or lethal, doses to be estimated, other than 50
CIPROBABILITY = *scalar*	Probability level for the confidence interval of effective doses; default 0.95, i.e. a 95% confidence interval
LOGBASE = *string token*	Base of antilog transformation to be applied to LD's (ten, e); default * i.e. none
DISPERSION = *scalar*	Controls the use of a heterogeneity factor in the calculation of s.e.s etc; with the default of 1 no factor is used, a missing value * estimates the heterogeneity from the residual deviance
FITMETHOD = *string token*	Method to use to fit the model (generalizednonlinear, nonlinear) default nonl for Wadley's problem, otherwise gene
MAXCYCLE = *scalar*	Maximum number of iterations for fitting the model; default 30

Parameters

Y = *variates*	Number of subjects responding in each batch
DOSE = *variates*	Dose received by each batch of subjects
NBINOMIAL = *variates*, *scalars* or *factors*	Variate specifying the number of subjects in each batch, or factor specifying groupings of the observations assumed to

have equal expected total numbers of subjects in Wadley's problem; if omitted, assumes Wadleys's problem with all observations having the same expected total number of subjects

`INITIAL` = *variates* Initial values for parameters

`STEPLENGTHS` = *variates* Step lengths for parameters

`LDESTIMATES` = *variates* Saves estimates of the effective, or lethal, doses

`LDLOWER` = *variates* Saves lower values of the confidence intervals for the estimates of the effective, or lethal, doses (for `FITMETHOD=gene` only)

`LDUPPER` = *variates* Saves upper values of the confidence interval values for the estimates of the effective, or lethal, doses (for `FITMETHOD=gene` only)

`PROCEDURE` directive

Introduces a Genstat procedure.

Options

`PARAMETER` = *string token* Whether to process the structures in each parameter list of the procedure sequentially using a dummy to store each one in turn, or whether to put them all into a pointer so that the procedure is called only once (`dummy, pointer`); default `dumm`

`RESTORE` = *string tokens* Which aspects of the Genstat environment to store at the start of the procedure and restore at the end (`inprint`, `outprint`, `outstyle`, `diagnostic`, `errors`, `pause`, `prompt`, `newline`, `case`, `run`, `units`, `blockstructure`, `treatmentstructure`, `covariate`, `asave`, `dsave`, `msave`, `rsave`, `tsave`, `vsave`, `vcomponents`, `seeds`, `captions`, `cmethod`, `actionafterfault`, `unsetdummy`, `all`); default `*`

`SAVE` = *text* Text to save the contents of the procedure (omitting comments and some spaces)

`WORDLENGTH` = *string token* Length of word (8 or 32 characters) to check in identifiers, directives, options, parameters and procedures within the procedure (`long`, `short`); default `*` i.e. no change

Parameter

text Name of the procedure

`PRSPEARMAN` procedure

Calculates probabilities for Spearman's rank correlation statistic (D.B. Baird).

No options

Parameters

`N` = *scalars* Numbers of pairs of observations

`CORRELATION` = *scalars* Values of the signed rank statistic

`CLPROBABILITY` = *scalars* Cumulative lower probability of `CORRELATION`

`CUPROBABILITY` = *scalars* Cumulative upper probability of `CORRELATION`

`PROBABILITY` = *scalars* Probability density of `CORRELATION`

`UPROBABILITIES` = *variates* Probability densities of `CORRELATION`...1

`UCORRELATION` = *variates* Values of `CORRELATION` at corresponding elements of `UPROBABILITIES`

`PRWILCOXON` procedure

Calculates probabilities for the Wilcoxon signed-rank statistic (D.B. Baird).

No options

Parameters

`N` = *scalars* Sizes of the first groups of observations

SIGNEDRANK = *scalars*	Values of the signed rank statistic
CLPROBABILITY = *scalars*	Cumulative lower probability of SIGNEDRANK
CUPROBABILITY = *scalars*	Cumulative upper probability of SIGNEDRANK
PROBABILITY = *scalars*	Probability density of SIGNEDRANK
LPROBABILITIES = *variates*	Probability densities of 0...SIGNEDRANK

PSPLINE procedure

Calculates design matrices to fit a P-spline as a linear mixed model (S.J. Welham).

Options

NSEGMENTS = *scalar*	Specifies the number of segments between boundaries; default * obtains a value automatically
DEGREE = *scalar*	Degree of polynomial used to form the underlying spline basis functions; default 3
DIFFORDER = *scalar*	Differencing order for penalty; default 2
LOWER = *scalar*	Specifies the lower boundary; default takes the minimum value in X
UPPER = *scalar*	Specifies the upper boundary; default takes the maximum value in X
ORTHOGONALIZETO = *variate*	Variate to use to get an orthogonalized basis; default * i.e. orthogonalization with respect to X
SCALING = *scalar*	Scaling of the XRANDOM terms; (automatic, none); default auto

Parameters

X = *variates*	The explanatory variate for which the basis functions are required
XFIXED = *matrices*	Saves the design matrix to define the fixed terms (excluding the constant) for fitting the P-spline
XRANDOM = *matrices*	Saves the design matrix to define the random terms for fitting the P-spline
KNOTS = *variates*	Saves the internal knots and boundaries used to form the basis functions
PX = *variates*	Specifies x-values at which predictions are required
PFIXED = *matrices*	Saves the design matrix for the fixed terms (excluding the constant) for the spline at the prediction points
PRANDOM = *matrices*	Saves the design matrix for the random terms for the spline at the prediction points

PTAREAPOLYGON procedure

Calculates the area of a polygon (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	What to print (summary); default summ

Parameters

YPOLYGON = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
XPOLYGON = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
AREA = *scalars*	Scalars to receive the areas of the polygons

PTBOX procedure

Generates a bounding or surrounding box for a spatial point pattern (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Options

PRINT = *string token*	What to print (summary); default summ
METHOD = *string token*	Type of box to form (bounding, surrounding); default boun

Parameters

`Y` = *variates*	Vertical coordinates of each spatial point pattern; no default – this parameter must be set
`X` = *variates*	Horizontal coordinates of each spatial point pattern; no default – this parameter must be set
`YBOX` = *variates*	Variates to receive the vertical coordinates of the bounding or surrounding boxes
`XBOX` = *variates*	Variates to receive the horizontal coordinates of the bounding or surrounding boxes
`YFRACTION` = *scalars*	How much to extend the extremes of the vertical coordinates of each surrounding box as a fraction of the range of the vertical coordinates; default 0.1
`XFRACTION` = *scalars*	How much to extend the extremes of the horizontal coordinates of each surrounding box as a fraction of the range of the horizontal coordinates; default 0.1

`PTCLOSEPOLYGON` procedure

Closes open polygons (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

`PRINT` = *string token*	What to print (`summary`); default `summ`

Parameters

`OLDYPOLYGON` = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
`OLDXPOLYGON` = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
`NEWYPOLYGON` = *variates*	Vertical coordinates of the closed polygons
`NEWXPOLYGON` = *variates*	Horizontal coordinates of the closed polygons

`PTDESCRIBE` procedure

Gives summary and second order statistics for a point process (R.P. Littlejohn & R.C. Butler).

Options

`PRINT` = *string token*	Whether to print (`statistics`); default `stat`
`SELECTION` = *string tokens*	What to print (`interval`, `trend`, `poisson`, `icorrelation`, `ispectrum`, `cspectrum`, `cintensity`, `vtcurve`, `all`); default `inte`
`REPRESENTATION` = *string token*	How the point process is represented in the `DATA` variate (`time`, `interval`, `zeroone`); default `time`
`GRAPHICS` = *string token*	Style of graphical output, or `GRAPHICS=*` to avoid any graphs (`lineprinter`, `highresolution`); default `high`

Parameters

`DATA` = *variates*	Variate containing point process to be analysed
`START` = *scalars*	Initial time (if `REPRESENTATION=time`); default 0
`LENGTH` = *scalars*	Length of time over which process is observed; default takes the time of the last event
`CITAU` = *scalars*	Window width for calculating count intensity; default 0.5 × mean interval length
`VTTAU` = *scalars*	Window width for calculating variance-time curve; default 0.5 × mean interval length
`SAVE` = *pointers*	Pointer to save calculated values

`PTGRID` procedure

Generates a grid of points in a polygon (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

`PRINT` = *string token*	What to print (`summary`); default `summ`

Parameters

YPOLYGON = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
XPOLYGON = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
NPOINTS = *scalars*	How many points to generate
YSTEP = *scalars*	Spacings to use between columns of the grid
XSTEP = *scalars*	Spacings to use between rows of the grid
YGRID = *variates*	Variates to receive the vertical coordinates of the points in the grid
XGRID = *variates*	Variates to receive the horizontal coordinates of the points in the grid

PTINTENSITY procedure

Calculates the overall density for a spatial point pattern (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	What to print (summary); default summ

Parameters

Y = *variates*	Vertical coordinates of each spatial point pattern; no default – this parameter must be set
X = *variates*	Horizontal coordinates of each spatial point pattern; no default – this parameter must be set
YPOLYGON = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
XPOLYGON = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
DENSITY = *scalars*	Scalars to receive the density of the spatial point patterns, i.e. the number of points per unit area

PTKERNEL2D procedure

Performs kernel smoothing of a spatial point pattern (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string tokens*	What to print (grid, monitoring); default grid, moni

Parameters

Y = *variates*	Vertical coordinates of each spatial point pattern; no default – this parameter must be set
X = *variates*	Horizontal coordinates of each spatial point pattern; no default – this parameter must be set
YPOLYGON = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
XPOLYGON = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
HZERO = *scalars*	What kernel width to use for each pattern; no default – this parameter must be set
NY = *scalars*	Numbers of rows to use in the grid of kernel density estimates; default 20
NX = *scalars*	Numbers of columns to use in the grid of kernel density estimates; default 20
YGRID = *variates*	Variates to receive the vertical coordinates at which each kernel function has been evaluated
XGRID = *variates*	Variates to receive the horizontal coordinates at which each kernel function has been evaluated
ZGRID = *matrices*	Matrices of dimension NY by NX to receive the grid of density estimates

PTK3D procedure

Performs kernel smoothing of space-time data (D.A. Murray, P.J. Diggle & B.S. Rowlingson).

Option

PRINT = *string token*	Controls printed output (grid, monitoring); default grid

Parameters

Y = *variates*	Vertical coordinates of the spatial point pattern
X = *variates*	Horizontal coordinates of the spatial point pattern
TIMES = *variates*	Times for each event
XGRID = *variates*	The values of x to compute kernel function
YGRID = *variates*	The values of y to compute kernel function
ZGRID = *variates*	The values of z, or time dimension, to compute kernel function
HXY = *scalars*	What quartic kernel width to use in the XY direction
HZ = *scalars*	What quartic kernel width to use in the Z or time direction
GRID = *pointers*	Pointer to matrices containing the kernel smoothed values

PTREMOVE procedure

Removes points interactively from a spatial point pattern (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Options

PRINT = *string token*	What to print (summary, monitoring); default summ, moni
WINDOW = *scalar*	Which graphics window to use for the plot; default 1

Parameters

OLDY = *variates*	Vertical coordinates of each spatial point pattern; no default – this parameter must be set
OLDX = *variates*	Horizontal coordinates of each spatial point pattern; no default – this parameter must be set
NEWY = *variates*	Variates to receive the vertical coordinates of the original points minus the deleted points of each pattern
NEWX = *variates*	Variates to receive the horizontal coordinates of the original points minus the deleted points of each pattern

PTROTATE procedure

Rotates a point pattern (W. van den Berg).

Options

ANGLE = *scalar*	Angle, in degrees over which the point pattern is to be rotated; no default – must be set
HUB = *string token*	Whether the point pattern is to be rotated around the origin or around the centroid (origin, centroid); default orig

Parameters

OLDY = *variates*	Vertical coordinates of each spatial point pattern
OLDX = *variates*	Horizontal coordinates of each spatial point pattern
NEWY = *variates*	Save the vertical coordinates of the rotated point patterns; if this unset, these replace the original values in OLDY
NEWX = *variates*	Save the horizontal coordinates of the rotated point patterns; if this unset, these replace the original values in OLDX
ROTATION = *matrices*	Save the rotation matrices

PTSINPOLYGON procedure

Returns points inside or outside a polygon (M.A. Mugglestone, S.A. Harding, B.Y.Y. Lee, P.J. Diggle & B.S. Rowlingson).

Options

PRINT = *string token*	What to print (summary); default summ
METHOD = *string token*	Whether to select points inside or outside the polygon (inside, outside); default insi

Parameters

OLDY = *variates*	Vertical coordinates of each spatial point pattern; no default – this parameter must be set
OLDX = *variates*	Horizontal coordinates of each spatial point pattern; no default – this parameter must be set
YPOLYGON = *variates*	Vertical coordinates of each polygon; no default – this parameter must be set
XPOLYGON = *variates*	Horizontal coordinates of each polygon; no default – this parameter must be set
NEWY = *variates*	Variates to receive the vertical coordinates of points inside (or outside) the polygons
NEWX = *variates*	Variates to receive the horizontal coordinates of points inside (or outside) the polygons

QBESTGENOTYPES procedure

Sorts individuals of a segregating population by their genetic similarity with a defined target genotype, using the identity by descent (IBD) information at QTL positions for one or more traits (M. Malosetti & F.A. van Eeuwijk).

Options

PRINT = *string tokens*	What to print (summary); default summ
PLOT = *string tokens*	What to plot (haplotypes); default hapl
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL); default F2
IBDWINDOW = *scalar*	Size of the window around the QTL position to use to construct the haplotypes; default 10
TRAITS = *text*	Names of the traits whose QTL information is to be used; default is to use all the traits
SELECTION = *variate*	Indicator variate with values defining whether each trait should be maximized (1), minimized (–1) or remain unchanged (0); if unset, the default is to maximize every trait
%BESTGENOTYPES = *scalar*	Specifies the percentage of the best genotypes to display in the output and plots; default 10

Parameters

GENFILENAME = *texts*	Name of a Flapjack genotype file
MAPFILENAME = *texts*	Name of a Flapjack map file
FJQTLFILENAME = *texts*	Name of a file to supply the QTL results
QTRAITS = *texts*	Names of the traits affected by each QTL
QCHROMOSOMES = *factors*	Factor defining the linkage group of each QTL
QPOSITIONS = *variates*	Position of each QTL within the linkage group
QNAMES = *texts*	Name of each QTL
QEFFECTS = *variates*	Individual QTL effects
QBESTSAVE = *pointers*	Saves similarities with the target genotype, and their ranks, across and per trait

QCANDIDATES procedure

Selects QTLs on the basis of a test statistic profile along the genome (M.P. Boer & J.T.N.M. Thissen).

Options

PRINT = *string token*	What to print (summary); default summ
THRESHOLD = *scalar*	Threshold for the test statistic; default 0
QTLWINDOW = *scalar*	Minimum distance in cM between two peaks to be selected as two QTLs; default 10

Parameters

STATISTICS = *variates*	Test statistic along the genome; must be set
CHROMOSOMES = *factors*	Chromosome for each locus; must be set
POSITIONS = *variates*	Position on the chromosome for each locus; must be set
IDLOCI = *texts*	Labels for the loci

QTLCANDIDATES = variates	Saves the index numbers of the selected QTLs

QCOCHRAN procedure

Performs Cochran's *Q* test for differences between related samples (D.A. Murray).

Options

PRINT = *string token*	Controls printed output (test); default test
METHOD = *string token*	Form of the test (exact, chisquare); default exac for small samples, otherwise chis
GROUPS = *factor*	Defines the groups if there only one variable supplied for the DATA
STATISTIC = *scalar*	Scalar to save the Q value
PROBABILITY = *scalar*	Scalar to save the probability for the Q Test
MAXTIME = *scalar*	Defines a limit for the maximum time for calculating the exact test; default * i.e. no limit.

Parameter

DATA = *variates*	List of related samples, or variate containing all the samples (the GROUPS option must then be set to indicate the variable recorded in each unit belongs)

QDESCRIBE procedure

Calculates descriptive statistics of molecular markers (M.P. Boer & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (chromosomes, genome); default chro
DISTANCE = *scalar*	Distance between chromosomes (for plotting purposes); default 10

Parameters

CHROMOSOMES = *factors*	Chromosome for each locus; must be set
POSITIONS = *variates*	Position on the chromosome for each locus; must be set
IDLOCI = *texts*	Labels for the loci
CUMPOSITIONS = *variates*	Saves the cumulative positions of the loci along the genome
NLOCI = *variates*	Saves the number of loci on each chromosome
FIRST = *variates*	Saves the index number of the first locus of each chromosome
LAST = *variates*	Saves the index number of the last locus of each chromosome
LENGTHS = *variates*	Saves the lengths of the chromosomes
MIDDLEPOSITIONS = *variates*	Saves the middle positions of the chromosomes (as cumulative positions)
SEPARATION = *variates*	Saves the positions of the gaps between chromosomes (as cumulative positions)
GENOMELENGTH = *scalars*	Saves the length of the genome
TOTLENGTH = *scalars*	Saves the total length of the genome, including added gaps between chomosomes

QDIALOG procedure

Produces a modal dialog box to obtain a response from the user.

Options

DIALOG = *string token*	Type of dialog box (checkbox, pushbutton, radiobutton, text, integer, real, variable, query, message); no default, must be specified
TITLE = *text*	Title for the dialog box; default * i.e. none
PREAMBLE = *text*	Informative text that appears above any controls on the dialog; default * i.e. none
LABEL = *text*	Label for the data entry field; default * i.e. none
RESPONSE = *identifier*	Structure to store the response
STATUS = *scalar*	Stores the exit status as 1 for OK, 2 for cancel, 3 for no, or 4 for yes
DEFAULT = *identifier*	Default setting or settings to appear in the menu; default * i.e.

none

LIST = *string token* — Whether an interger, real or variable entry field can contain a list of settings (yes, no); default no

HELP = *texts* — Help on the menu, to be displayed in a pop-up window; default * i.e. none

ICON = *string token* — Type of icon to display in the dialog box (information, warning, error, query); default * i.e. none

TIMEOUT = *scalar* — Permits the dialog to continue and return a default value after a specified period (in seconds); default * i.e. no timeout

MINIMUM = *scalar* — Minimum value for numerical input fields; default * i.e. none

MAXIMUM = *scalar* — Minimum value for numerical input fields; default * i.e. none

Parameters

BOXLABEL = *texts* — Label for each checkbox or radio button

BOXRESPONSE = *scalars* — Indicates the selection status of each checkbox or radio button

QDISCRIMINATE procedure

Performs quadratic discrimination between groups i.e. allowing for different variance-covariance matrices (D.B. Baird).

Options

PRINT = *string tokens* — Printed output from the analysis (allocation, counts, distance, probabilities, specificity, summary, table, validation, vcovariance); default spec, summ, vali

VALIDATIONMETHOD = *string token* — Validation method to use to calculate error rates (bootstrap, crossvalidation, jackknife, prediction); default cros

NSIMULATIONS = *scalar* — Number of bootstraps or cross-validation sets; default 50

NCROSSVALIDATIONGROUPS = *scalar* — Number of groups for cross-validation, default 10

Parameters

DATA = *pointers* — Each pointer contains a training set of variates to be used to form a quadratic discrimination

GROUPS = *factors* — Define groupings for the units in each training set

PRIORPROBABILITIES = *variates* — Prior probabilities of group membership; default * i.e. equal

SEED = *scalars* — Seed for the random numbers used in bootstrapping or cross-validation; default 0 continues from the previous generation or (if none) initializes the seed automatically

ERRORRATE = *scalars* — Saves the validation error rate

SPECIFICITY = *matrices* — Saves the specificity table

ALLOCATION = *factors* — Saves the groups allocated by the discriminant rule

PROBABILITIES = *matrices* or *pointers* — Save posterior probabilities of membership of the groups (in the columns of a matrix or the variates in a pointer) for the units in the training set (in the rows)

QEIGENANALYSIS procedure

Uses principal components analysis and the Tracy-Widom statistic to find the number of significant principal components to represent a set of variables (M. Malosetti & J.T.N.M. Thissen).

Options

PRINT = *string tokens* — What to print (summary, scores); default summ

NROOTS = *scalar* — Number of principal components to retain; default saves the significant components

PLOT = *string tokens* — What to plot (eigenvalues, %variance); default eige, %var

PROBABILITY = *scalar* — Specifies the significance level; default 0.05

SCALING = *string token* — Whether to scale the principal component scores by the square roots of their singular values (singularvalues, none); default none

STANDARDIZE = *string token*	How to standardize the DATA variates (frequency, none); default freq
TITLE = *text*	General title for the plots

Parameters

DATA = *pointers*	Data variates; must be set
SCORES = *pointers*	Pointer of variates to store the scores of the significant axes for each set of DATA variates
EVALUES = *variates*	Saves the eigenvalues of the significant principal components
NEFFECTIVE = *scalars*	Saves the effective number of columns of the marker data matrix
%VARIANCE = *variates*	Saves the percentage variances explained by the significant principal components
CUM%VARIANCE = *variates*	Saves the cumulative percentage variances explained by the significant principal components

QEXPORT procedure

Exports genotypic and phenotypic data for QTL analysis (D.A. Murray).

Options

OUTFILENAME = *text*	Name of the file to receive the data
MAPFILENAME = *text*	Name of the associated map file for Flapjack or MapQTL(R)
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP, AMP); must be set
NGENERATIONS = *scalar*	Number of generations for a RIL population
NAME = *text*	Name for the header in a .loc file
MISSING = *text*	Character to represent a missing genotype in Flapjack or R/QTL format; default '-'
SEPARATOR = *text*	Character to separate data values in Flapjack format; default separates them by tabs
ASEPARATOR = *text*	Character to separate allele values in Flapjack format; default '/'
FJROWS = *string token*	Specifies whether the genotypes or markers are to be stored on the rows in Flapjack format (genotypes, markers); default geno

Parameters

MKSCORES = *pointers*	Genotype codes for each marker
CHROMOSOMES= *factors*	Linkage groups for the markers
POSITIONS = *variates*	Positions within the linkage groups of markers
MKNAMES = *texts*	Marker names
MKSETS = *factors*	Marker sets
IDMGENOTYPES = *texts*	Labels for genotypes
PARENTS = *pointers*	Parent information
IDPARENTS = *texts*	Labels used to identify the parents

QFACTOR procedure

Allows the user to decide to convert texts or variates to factors (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (replication, summary); default summ
MAXCATEGORY = *scalar*	Maximum number of distinct values that a VECTOR may contain if it is to be converted; default 10
QUERY = *string token*	Whether to ask the user if each VECTOR with no more than MAXCATEGORY distinct values is to be converted

Parameter

VECTOR = *variates* or *texts*	Vectors to be converted into factors

QFLAPJACK procedure

Creates a Flapjack project file from genotypic and phenotypic data (D.A. Murray).

Options

WORKDIRECTORY = *text*	Working directory to use for files; default current Genstat working directory
FJPATH = *text*	Path specifying the location of Flapjack; by default QFLAPJACK searches for a version of Flapjack installed within C:\program files (x86)\Flapjack or C:\program files\Flapjack
DECIMALSYMBOL = *string token*	Controls whether to use the locale (automatic) or English (dot) representation of decimal marks (automatic, dot); default auto

Parameters

FJFILENAME = *texts*	Name of the Flapjack project file to create
TRAITS = *pointers*	Pointer to variates containing the phenotypic trait data
GENOTYPES = *factors*	Genotype factor associated with the traits
ENVIRONMENTS = *factors*	Environment factor
GENFILENAME = *texts*	Name of a Flapjack genotype file
MAPFILENAME = *texts*	Name of a Flapjack map file
FJTRAITFILENAME = *texts*	Name of a file to supply the trait data, or to save them if the TRAITS and GENOTYPES parameters are also set
FJQTLFILENAME = *texts*	Name of a file to supply the QTL results, or to save them if the QSAVE parameter is also set
QSAVE = *pointers*	Information and results saved from an earlier QTL analysis

QGSELECT procedure

Obtains a representative selection of genotypes by means of genetic distance sampling or genetic distance optimization (J. Jansen & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, monitoring); default summ
NCLUSTERS = *scalar*	The number of genotypes to be selected; must be set
METHOD = *string token*	Method to be used (sampling, optimization); default samp

Parameters

GENOTYPES = *factors*	Genotype factor; must be set
SIMILARITY = *symmetric matices*	Input similarity matrix for each selection; must be set
PRIORGROUPS = *factors*	Defines prior groupings of the genotypes
SELECTED = *variates*	Logical variate indicating whether a genotype is selected (1) as cluster centre or not (0)
NEIGHBOURS = *variates*	Saves the nearest cluster centres of the genotypes
DISTANCES = *variates*	Saves the distances of the genotypes to the nearest cluster centre
SEED = *scalars*	Seed for randomization at the start; default 0

QIBDPROBABILITIES procedure

Reads molecular marker data and calculates IBD probabilities (M.P. Boer & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, loci); default summ
STEPSIZE = *scalar*	Maximum stepsize along the genome; default 10^6, i.e. the IBD probabilities are calculated only at the marker positions
METHOD = *string token*	Method of calculation for IBD probabilities of RIL populations (approximate, exact); default appr
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set
NGENERATIONS = *scalar*	Number of generations of selfing for a RIL population

NBACKCROSSES = *scalar*	Number of backcrosses for a BCxSy population
NSELFINGS = *scalar*	Number of selfings for a BCxSy population
MAPPINGFUNCTION = *string token*	Mapping function (haldane, kosambi); default hald

Parameters

MKSCORES = *pointers*	Genotype codes for each marker; must be set
CHROMOSOMES = *factors*	The chromosome where each marker is located; must be set
POSITIONS = *variates*	The position on the chromosome of each marker; must be set
MKNAMES = *texts*	Marker names; must be set
IDMGENOTYPES = *texts*	Labels for the genotypes
PARENTS = *pointers*	Parent information; must be set
IDPARENTS = *texts*	Labels used to identify the parents; must be set
PEDIGREE = *pointers*	Defines the parents of the offspring
ADDITIVEPREDICTORS = *pointers*	Saves the additive genetic predictors
ADD2PREDICTORS = *pointers*	Saves the second (paternal) additive genetic predictors if POPULATIONTYPE is CP
DOMINANCEPREDICTORS = *pointers*	Saves the dominance genetic predictors if POPULATIONTYPE is F2, RIL, BCxSy or CP
SCHROMOSOMES = *factors*	Saves the chromosome where each locus is located
SPOSITIONS = *variates*	Saves the position on the chromosome of each locus
LOCI = *variates*	Saves the index number of each locus
IDLOCI = *texts*	Saves the locus labels
MKLOCI = *variates*	Saves a logical variate indicating whether each locus is a marker
NLOCI = *scalars*	Saves the number of loci
NGENOTYPES = *scalars*	Saves the number of genotypes
APROBABILITIES = *pointers*	Saves probabilities of the genotypes being equal to parent A
BPROBABILITIES = *pointers*	Saves probabilities of the genotypes being equal to parent B
HPROBABILITIES = *pointers*	Saves the probabilities of the genotypes being heterozygous
ACPROBABILITIES = *pointers*	Saves the probabilities of the genotypes being AC when POPULATIONTYPE is CP
ADPROBABILITIES = *pointers*	Saves the probabilities of the genotypes being AD when POPULATIONTYPE is CP
BCPROBABILITIES = *pointers*	Saves the probabilities of the genotypes being BC when POPULATIONTYPE is CP
BDPROBABILITIES = *pointers*	Saves the probabilities of the genotypes being BD when POPULATIONTYPE is CP
OUTFILENAME = *texts*	Name of the Genstat workbook file (*.gwb) to be created

QIMPORT procedure

Imports genotypic and phenotypic data for QTL analysis (D.A. Murray).

Options

PRINT = *string token*	What to print (catalogue, errorreport); default cata, erro
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP, AMP); must be set
MISSING = *text*	Character representing a missing genotype in Flapjack or R/QTL format; default '-'
SEPARATOR = *text*	Character separating data values in Flapjack format; default separates them by tabs
ASEPARATOR = *text*	Character separating allele values in Flapjack format; default '/'
FJROWS = *string token*	Specifies whether the genotypes or markers are stored on the rows in Flapjack format (genotypes, markers); default geno
NPARENTS = *scalar*	Number of parents in Flapjack file; default 0 for population AMP, 4 for CP, and 2 otherwise

Parameters

FILENAME = *texts*	Name of the file for import
MAPFILENAME = *texts*	Name of the map file (Flapjack or MapQTL(R))
PHEFILENAME = *texts*	Name of the phenotypic file (MapQTL(R))
MKSCORES = *pointers*	Saves the genotype codes for each marker
TRAITS = *pointers*	Saves the trait data from the phenotypic file
CHROMOSOMES = *factors*	Saves linkage groups for each marker
POSITIONS = *variates*	Saves positions of the markers within linkage groups
MKNAMES = *texts*	Saves the marker names
MKSETS = *factors*	Saves marker sets
IDMGENOTYPES = *texts*	Labels for genotypes
PARENTS = *pointers*	Saves the parent information
IDPARENTS = *texts*	Saves the labels used to identify the parents
IDFILENAME = *texts*	Specifies a file containing genotype labels for MapQTL(R) files; if unset, they are assumed to be in the .loc file

QKINSHIPMATRIX procedure

Forms a kinship matrix from molecular markers (L.C.P. Keizer & J.T.N.M. Thissen).

Options

PRINT = *string token*	What to print (summary); default summ
METHOD = *string token*	Method to use for the calculation (correlation, dice); default dice

Parameters

MKSCORES = *pointers*	Pointer with the marker scores; must be set
IDMGENOTYPES = *texts*	Labels for the genotypes
KMATRIX = *symmetric matrices*	Saves the kinship matrix
OUTFILENAME = *texts*	Name of the file to receive the kinship matrix

QLDDECAY procedure

Estimates linkage disequilibrium (LD) decay along a chromosome (M. Malosetti & J.T.N.M. Thissen).

Options

PRINT = *string token*	What to print (progress); default *
PLOT = *string tokens*	What to plot (ldmatrix, lddecay); default ldde
RELATIONSHIPMODEL = *string token*	What model to use to account for genetic relatedness (eigenanalysis, subpopulations, null); default eige
SCORES = pointer	Provides the scores of significant principal components, obtained from an eigenvalue analysis
SUBPOPULATIONS = *factor*	Defines groupings of genotypes into subpopulations
CHRANALYSE = *scalar*	Defines which chromosome to analyse, using a level of the CHROMOSOMES factor
MAX%MISSING = *scalar*	Markers with more than the specified % of missing values will be excluded from the LD calculations; default 20
MAXDISTANCE = *scalar*	Defines the maximum distance between markers to show in LD plots; default 30
TITLE = *text*	General title for the plots
YTITLE = *text*	Title for the y-axis
XTITLE = *text*	Title for the x-axis

Parameters

MKSCORES = *pointers*	Genotype codes for each marker; must be set
CHROMOSOMES= *factors*	Linkage groups for the markers; must be set
POSITIONS = *variates*	Positions within the linkage groups of markers; must be set
DISTANCES = *symmetric matrices*	Saves the distances between markers
R2 = *symmetric matrices*	Saves the value of r^2 between markers

QLINKAGEGROUPS procedure

Forms linkage groups using marker data from experimental populations (J. Jansen, J.T.N.M. Thissen & M.P. Boer).

Options

PRINT = *string token*	What to print (summary); default summ
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, CP); must be set
USEPENALTY = *string token*	Whether to increase the number of recombinations by 0.5 recombination per informative meiosis for each missing marker score (yes, no); default no
THRESHOLD = *scalar* or *variate*	Threshold for the recombination frequency at which markers are said to be linked; default 0.2

Parameters

MKSCORES = *pointers*	Marker scores for each marker; must be set
CHROMOSOMES = *factors* or *pointers*	Saves the linkage groups of the markers
MKNAMES = *texts*	Names of the markers; must be set
PARENTS = *pointers*	Marker scores of the parents; must be set
SMKSCORES = *pointers*	Saves the marker scores factors according to the SMKNAMES parameter
SCHROMOSOMES = *factors* or *pointers*	Saves the sorted linkage groups
SMKNAMES = *texts* or *pointers*	Saves the names of the markers according to the SCHROMOSOMES parameter
SPARENTS = *pointers*	Saves the parent information according to the SMKNAMES parameter when POPULATIONTYPE=CP

QLIST procedure

Gets the user to select a response interactively from a list (R.W. Payne).

Option

HELP = *text*	Help information for the QUESTION

Parameters

ALTERNATIVES = *texts*	Alternatives from which each choice is to be made
CODES = *texts*	Codes to use to represent each set of alternatives
PREAMBLE = *texts*	Preamble for the question used to select from each set of alternatives
CHOICE = *texts*	Alternative chosen from each set
NCHOICE = *scalars*	Numbers of the chosen alternatives (0 if exit has been chosen instead)

QMAP procedure

Constructs genetic linkage maps using marker data from experimental populations (J. Jansen, J.T.N.M. Thissen & M.P. Boer).

Options

PRINT = *string token*	What to print (map, monitoring, summary); default summ
PLOT = *string token*	What to plot (frequiencies, map); default map
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, CP); must be set
USEPENALTY = *string token*	Whether to increase the number of recombinations by 0.5 recombination per informative meiosis for each missing marker score (yes, no); default no
SPATIALMETHOD = *string token*	Which method to use for clustering (sampling, optimization, none); default opti for population CP, samp otherwise
NGROUPS = *scalar*	Number of groups for clustering; default 10
MAPCHROMOSOMES = *variate*, *text* or *scalar*	Allows a subset of chromosomes to be mapped; default * i.e. all the chromosomes
LINKAGEPHASES = *string token*	Controls estimation of linkage phases for population type CP (estimate, omit); default esti

TITLE = *text*	General title for the graph
OUTFILENAME = *text*	Name (without extension) of the Flapjack files to be created

Parameters

MKSCORES = *pointers*	Marker scores for each marker; must be set
CHROMOSOMES = *factors*	Factor defining the linkage groups
POSITIONS = *variates*	Saves the positions of markers
MKNAMES = *texts*	Names of the markers; must be set
IDMGENOTYPES = *texts*	Names of the genotypes
PARENTS = *pointers*	Marker scores of the parents; must be set
IDPARENTS = *texts*	Labels to identify the parents
SMKSCORES = *pointers*	Saves the scores of the markers, sorted according to the markers in the SCHROMOSOMES factor (if CHROMOSOMES is set) and the SPOSITIONS variate
SCHROMOSOMES = *factors*	Saves the sorted linkage groups
SPOSITIONS = *variates*	Saves the sorted positions of markers (within the sorted linkage groups if CHROMOSOMES is set)
SMKNAMES = *texts*	Saves the names of the markers, sorted according to the SCHROMOSOMES factor (if CHROMOSOMES is set) and the SPOSITIONS variate
SPARENTS = *pointers*	Saves the marker scores of the parents, sorted according to the markers in the SCHROMOSOMES factor (if CHROMOSOMES is set) and the SPOSITIONS variate
SEED = *scalars*	Seed for the random numbers used for spatial sampling; default 0

QMASSOCIATION procedure

Performs multi-environment marker-trait association analysis in a genetically diverse population using bi-allelic and multi-allelic markers (M. Malosetti & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, progress); default summ
PLOT = *string tokens*	What to plot (profile, map); default prof, map
RELATIONSHIPMODEL = *string token*	What model to use to account for genetic relatedness (eigenanalysis, subpopulations, null); default eige
VCMODEL = *string token*	Specifies the variance-covariance model for the set of environments (identity, diagonal, cs, hcs, outside, fa, unstructured, best); default best
CRITERION = *string token*	Defines which criterion is used to compare the different covariance structures (aic, sic); default sic
MINORALLELE = *scalar*	Frequency of minor alleles; default 0.05
THRESHOLD = *scalar*	Threshold value for significant LD, on the −log10 scale; default 2
SUBPOPULATIONS = *factor*	Defines groupings of genotypes into subpopulations
MODELPART = *string token*	Defines which part of the model should include SUBPOPULATIONS if RELATIONSHIPMODEL is set to subpopulations, or the principal components scores if RELATIONSHIPMODEL is set to eigenanalysis (fixed, random); default rand
SCALING = *string token*	Whether to scale the scores by the square roots of their singular values if RELEATIONSHIPMODEL is set to eigenanalysis (singularvalues, none); default sing
STANDARDIZE = *string token*	Whether to standardize the marker scores according to their frequencies (frequency, none); default freq
TITLE = *text*	General title for the plots
YTITLE = *text*	Title for the y-axis
XTITLE = *text*	Title for the x-axis

Parameters

TRAIT = *variates*	Phenotypic trait to analyse; must be set
GENOTYPES = *factors*	Genotype factor; must be set
ENVIRONMENTS = *factors*	Environment factor; must be set
MKSCORES = *pointers*	Genotype codes for each marker; must be set
CHROMOSOMES = *factors*	Linkage groups for the markers; must be set
POSITIONS = *variates*	Positions within the linkage groups of markers; must be set
MKNAMES = *texts*	Marker names
WALDSTATISTICS = *variates*	Saves the Wald test statistics
NDF = *variates*	Saves the degrees of freedom associated to the Wald test
MINLOG10P = *variates*	Saves the associated probability values of the Wald test statistics, on a −log10 scale
QSAVE = *pointers*	Saves a pointer with information and results for the significant effects
DFILENAME = *texts*	Name of the graphics file for the plots

QMATCH procedure

Matches different data structures to be used in QTL estimation (L.C.P. Keizer & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, details); default summ
GEN%MISSING = *scalar*	Percentage of missing values allowed for a genotype; default 50
MK%MISSING = *scalar*	Percentage of missing values allowed for a marker; default 50
MK%EXTREME = *scalar*	Extreme allele percentage allowed for a marker; default 5
GENSELECTION = *variate*	Logical variate containing the value one for the genotypes to retain and zero for those to remove (supersedes the options GEN%MISSING, MK%MISSING and MK%EXTREME)
MKSELECTION = *variate*	Logical variate containing the value one for the markers to retain and zero for those to remove (supersedes the options GEN%MISSING, MK%MISSING and MK%EXTREME)
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP, AMP); must be set
OUTFILEPREFIX = *text*	Prefix for the output file names; default * i.e. files not saved

Parameters

TRAITS = *pointers* or *variates*	Quantitative traits
GENOTYPES = *factors*	Genotype factors corresponding to the traits
ENVIRONMENTS = *factors*	Environment factors corresponding to the traits
MKSCORES = *pointers*	Marker scores; must be set
CHROMOSOMES = *factors*	Chromosomes corresponding to the markers
POSITIONS = *variates*	Positions on the chromosomes corresponding to the markers
MKNAMES = *texts*	Names of the markers
IDMGENOTYPES = *texts*	Labels for the genotypes corresponding to the markers
PARENTS = *pointers*	Parent information
IDPARENTS = *texts*	Labels used to identify the parents
KMATRIX = *symmetric matrices*	Kinship matrices containing coefficients of coancestries
SUBPOPULATIONS = *factors*	Groups of genotypes
STRAITS = *pointers* or *variates*	Saves the sorted quantitative traits
SGENOTYPES = *factors*	Saves the sorted genotype factors
SENVIRONMENTS = *factors*	Saves the sorted environment factors
SMKSCORES = *pointers*	Saves the sorted marker scores; must be set
SCHROMOSOMES = *factors*	Saves the sorted chromosomes corresponding to the markers
SPOSITIONS = *variates*	Saves the sorted positions on the chromosomes corresponding to the markers
SMKNAMES = *texts*	Saves the sorted names of the markers
SIDMGENOTYPES = *texts*	Saves the sorted labels for the genotypes
SPARENTS = *pointers*	Saves the sorted parent information

SIDPARENTS = *texts*	Saves the sorted labels used to identify the parents
SKMATRIX = *symmetric matrices*	Saves the sorted kinship matrices
SSUBPOPULATIONS = *factors*	Saves the sorted groups of genotypes

QMBACKSELECT procedure

Performs a QTL backward selection for loci in multi-environment trials or multiple populations (M.P. Boer, M. Malosetti, S.J. Welham & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default summ
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set
ALPHALEVEL = *scalar*	Defines a significance level; default 0.05
VCMODEL = *string token*	Defines the variance-covariance model for the set of environments (identity, diagonal, cs, hcs, outside, fa, fa2, unstructured); default cs for multi-environment trials, and diagonal for multiple populations
VCPARAMETERS = *string token*	Whether to re-estimate the variance-covariance model parameters (estimate, fix); default esti
VCSELECT = *string token*	Whether to re-select the variance-covariance model (no, yes); default no
CRITERION = *string token*	Criterion to use for model selection (aic, sic); default sic
FIXED = *formula*	Defines extra fixed effects
UNITFACTOR = *factor*	Saves the units factor required to define the random model when UNITERROR is to be used
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default expl, yvar
MAXCYCLE = *scalar*	Limit on the number of iterations; default 100
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm; default 100

Parameters

TRAIT = *variates*	Quantitative trait to be analysed; must be set
GENOTYPES = *factors*	Genotype factor; must be set
ENVIRONMENTS = *factors*	Environment factor; must be set for a multi-environment trial
POPULATIONS = factors	Population factor; must be set for a multiple-population analysis
UNITERROR = *variates*	Uncertainty on trait means (derived from individual unit or plot error) to be included in QTL analysis; default * i.e. omitted
VCINITIAL = *pointers*	Initial values for the parameters of the variance-covariance model
SELECTEDMODEL = *texts*	VCMODEL setting for the selected covariance structure
ADDITIVEPREDICTORS = *pointers*	Additive genetic predictors; must be set
ADD2PREDICTORS = *pointers*	Second (paternal) set of additive genetic predictors
DOMINANCEPREDICTORS = *pointers*	Dominance genetic predictors
CHROMOSOMES = *factors*	Chromosomes corresponding to the genetic predictors; must be set
POSITIONS = *variates*	Positions on the chromosomes corresponding to the genetic predictors; must be set
IDLOCI = *texts*	Labels for the loci
IDMGENOTYPES = *texts*	Labels for the genotypes corresponding to the genetic predictors
QTLCANDIDATES = *variates*	Specifies the locus index numbers from which to start the

	selection; must be set
QTLSELECTED = *variates*	Saves the index numbers of the selected QTLs
INTERACTIONS = *variates*	Saves a logical variate indicating whether each selected QTL showed a significant (1) or non-significant (0) QTL-by-environment or QTL-by-population interaction
DOMSELECTED = *variates*	Saves a logical variate indicating whether each selected QTL showed a significant (1) or non-significant (0) effect of the DOMINANCEPREDICTORS
DOMINTERACTIONS = *variates*	Saves a logical variate indicating whether each selected QTL showed a significant (1) or non-significant (0) dominance-by-environment or dominance-by-population interaction
WALDSTATISTICS = *variates*	Saves the Wald test statistics
PRWALD = *variates*	Saves the associated Wald probabilities

QMESTIMATE procedure

Calculates QTL effects in multi-environment trials or multiple populations (M.P Boer, M. Malosetti, S.J. Welham & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default summ
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set
NGENERATIONS = *scalar*	Number of generations of selfing for a RIL population
NBACKCROSSES = *scalar*	Number of backcrosses for a BCxSy population
NSELFINGS = *scalar*	Number of selfings for a BCxSy population
VCMODEL = *string token*	Specifies the variance-covariance model for the set of environments or populations (identity, diagonal, cs, hcs, outside, fa, fa2, unstructured); default cs for multi-environment trials, and diagonal for multiple populations
VCPARAMETERS = *string token*	Whether to re-estimate the variance-covariance model parameters (estimate, fix); default esti
VCSELECT = *string token*	Whether to re-select the variance-covariance model (no, yes); default no
CRITERION = *string token*	Criterion to use for model selection (aic, sic); default sic
FIXED = *formula*	Defines extra fixed effects
UNITFACTOR = *factor*	Saves the units factor required to define the random model when UNITERROR is to be used
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default expl, yvar
MAXCYCLE = *scalar*	Limit on the number of iterations; default 100
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm; default 100

Parameters

TRAIT = *variates*	Quantitative trait to be analysed; must be set
GENOTYPES = *factors*	Genotype factor; must be set
ENVIRONMENTS = *factors*	Environment factor; must be set for a multi-environment trial
POPULATIONS = *factors*	Population factor; must be set for a multiple-population analysis
UNITERROR = *variates*	Uncertainty on trait means (derived from individual unit or plot error) to be included in QTL analysis; default * i.e. omitted
VCINITIAL = *pointers*	Initial values for the parameters of the variance-covariance model

SELECTEDMODEL = *texts*	VCMODEL setting for the selected covariance structure
ADDITIVEPREDICTORS = *pointers*	Additive genetic predictors; must be set
ADD2PREDICTORS = *pointers*	Second (paternal) set of additive genetic predictors
DOMINANCEPREDICTORS = *pointers*	Dominance genetic predictors
CHROMOSOMES = *factors*	Chromosomes corresponding to the genetic predictors; must be set
POSITIONS = *variates*	Positions on the chromosomes corresponding to the genetic predictors; must be set
IDLOCI = *texts*	Labels for the loci; must be set
MKLOCI = *variates*	Logical variate containing the value 1 if the locus is a marker, otherwise 0; must be set
IDMGENOTYPES = *texts*	Labels for the genotypes corresponding to the genetic predictors
IDPARENTS = *texts*	Labels to identify the parents
QTLSELECTED = *variates*	Index numbers of the selected QTLs; must be set
INTERACTIONS = *variates*	Logical variate indicating whether each selected QTL has a significant (1) or non-significant (0) QTL-by-environment or QTL-by-population interaction
DOMSELECTED = *variates*	Logical variate indicating whether the dominance predictor of each selected QTL must be present (1) or absent (0) in the model
DOMINTERACTIONS = *variates*	Logical variate indicating whether the dominance-by-environment or dominance-by-population interaction of each selected QTL must be present (1) or absent (0) in the model
RESIDUALS = *variates*	Residuals from the analysis
FITTEDVALUES = *variates*	Fitted values from the analysis
WALDSTATISTICS = *variates*	Saves the Wald test statistics
PRWALD = *variates*	Saves the associated Wald probabilities
DFWALD = *variates*	Saves the degrees of freedom for the Wald test
QEFFECTS = *pointers*	Saves the estimated QTL effects
QSE = *pointers*	Saves the standard errors of the QTL effects
OUTFILENAME = *texts*	Name of the Genstat workbook file (*.gwb) to be created
QSAVE = *pointers*	Saves a pointer with information and results for the significant effects
SAVE = *REML save structures*	Save the details of each REML analysis for use in subsequent VDISPLAY and VKEEP directives

QMKDIAGNOSTICS procedure

Generates descriptive statistics and diagnostic plots of molecular marker data (D.A. Murray, S.J. Welham, M. Malosetti, M.P. Boer, L.C.P. Keizer & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, missingvalues, frequencies); default summ, miss, freq
PLOT = *string tokens*	What to plot (missingvalues, frequencies, probabilities, genotypes, map); default miss, geno, map
GEN%MISSING = *scalar*	Threshold for printing genotypes with many missing values (i.e. genotypes with a higher percentage of missing values than the specified value); default 10
MK%MISSING = *scalar*	Threshold for printing markers with many missing values (i.e. markers with a higher percentage of missing values than the specified value); default 10
MK%EXTREME = *scalar*	Threshold for printing markers with rare alleles (i.e. alleles present with a lower percentage than the specified threshold); default 10
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP, AMP); must be set

NGENERATIONS = *scalar* Number of generations for a RIL population; default 6
NBACKCROSSES = *scalar* Number of backcrosses; must be set for a BCxSy population
NSELFINGS = *scalar* Number of selfings; must be set for a BCxSy population
DCHROMOSOMES = *variate*, *text* or *scalar*
Specifies a subset of the linkage groups to be displayed
PDIRECTION = *string token* How to sort the probabilities when PRINT=frequencies with BC1, DH1, F2, RIL and BCxSy populations (ascending, descending); default * i.e. no sorting

Parameters
MKSCORES = *pointers* Genotype codes for each marker; must be set
CHROMOSOMES= *factors* Linkage groups for the markers; must be set
POSITIONS = *variates* Positions within the linkage groups of markers; must be set
MKNAMES = *texts* Marker name; must be sets
IDMGENOTYPES = *texts* Labels for genotypes corresponding to the marker scores
PARENTS = *pointers* Parent information
IDPARENTS = *texts* Labels to identify the parents
GENCHECK = *variates* Logical variates containing the value one for genotypes with missing value problems, according to the setting of the GEN%MISSING option, and zero otherwise
MKCHECK = *variates* Logical variates containing the value one for markers with missing or extreme value problems, as defined by the MK%MISSING and MK%EXTREME options, and zero otherwise
SUMMARY = *pointers* Saves a summary of counts and probabilities for the chi-square tests for BC1, DH1, F2, RIL and BCxSy populations

QMKRECODE procedure

Recodes marker and/or parental scores into separate alleles (L.C.P. Keizer & J.T.N.M. Thissen).

Options
PRINT = *string tokens* What to print (alleles, summary); default alle
POPULATIONTYPE = *string token* Type of population (BC1, DH1, F2, RIL, BCxSy, CP, AMP); must be set
MISSING = *text* Character representing a missing genotype; default '-'
USEFIRSTGENOTYPE = *string token* Makes all the first (and second) labels of the LABALLELES pointer from the first genotype of the population (yes, no); default no
ASEPARATOR = *text* Character separating allele values; default '/'

Parameters
MKSCORES = *pointers* Marker scores; must be set
MKALLELES = *pointers* Saves the marker scores per allele
LABALLELES = *pointers* Saves the allele labels
MKLABALLELES = *pointers* Saves the allele labels per marker
NALLELES = *variates* Saves the number of alleles per marker
MKNAMES = *texts* Names of the markers

QMKSELECT procedure

Obtains a representative selection of markers by means of genetic distance sampling or genetic distance optimization (J. Jansen & J.T.N.M. Thissen).

Options
PRINT = *string tokens* What to print (summary, monitoring); default summ
NCLUSTERS = *scalar* The number of markers to be selected; must be set
METHOD = *string token* Method to be used (sampling, optimization); default samp

Parameters
MKNAMES = *texts* Names of the markers; must be set
RECFREQUENCY = symmetric matices Input recombination frequencies matrix for each selection; must be set

PRIORGROUPS = *factors*	Defines prior groupings of the markers
SELECTED = *variates*	Logical variate indicating whether a marker is selected (1) as cluster centre or not (0)
NEIGHBOURS = *variates*	Saves the nearest cluster centres of the markers
DISTANCES = *variates*	Saves the distances of the markers to the nearest cluster centre
SEED = *scalars*	Seed for randomization at the start; default 0

QMQTLSCAN procedure

Performs a genome-wide scan for QTL effects (Simple and Composite Interval Mapping) in multi-environment trials or multiple populations (M.P. Boer, M. Malosetti, S.J. Welham & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, progress, model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default summ
PLOT = *string token*	Whether to plot the profile along the genome (profile); default prof
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set
ALPHALEVEL = *scalar*	Defines a genome-wide significance level to calculate the threshold; default 0.05
VCMODEL = *string token*	Specifies the variance-covariance model for the set of environments or populations (identity, diagonal, cs, hcs, outside, fa, fa2, unstructured); default cs for multi-environment trials, and diagonal for multiple populations
VCPARAMETERS = *string token*	Whether to re-estimate the variance-covariance model parameters (estimate, fix); default esti
QTLMODEL = *string token*	Type of QTL model (q, qqe); default qqe
COFACTORS = *variate*	Index numbers of loci to be used as cofactors for the genetic background
COFWINDOW = *scalar*	Specifies a window for cofactor exclusion from the model; default 10^6 which means that all cofactors on the same chromosomes are excluded
THRMETHOD = *string token*	Which method to use to calculate the threshold for QTL detection (bonferroni, liji, given); default liji
THRESHOLD = *scalar*	Threshold value for test statistic when THRMETHOD=given
DISTANCE = *scalar*	Distance between loci when THRMETHOD=bonferroni; default 4
FIXED = *formula*	Formula with extra fixed terms
UNITFACTOR = *factor*	Saves the units factor required to define the random model when UNITERROR is to be used
STATISTICTYPE = *string token*	Which test statistic to plot and save using the STATISTICS parameter (wald, minlog10p); default minl
COLOURS = *scalar*, *variate* or *text*	Colours to use for the chromosomes; default * uses the colours of pens 1, 2 up to the number of chromosomes
TITLE = *text*	General title for the plot
YLOWERTITLE = *text*	Title for the y-axis of the lower graph; default 'Environments' for multi-environment trials, and 'Populations' for multiple populations
YUPPERTITLE = *text*	Title for the y-axis of the upper graph; default uses the identifier of the STATISTICS variate or pointer
XTITLE = *string*	Title for the x-axis; default 'Chromosomes'
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default expl, yvar

MAXCYCLE = *scalar*	Limit on the number of iterations; default 100
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm; default 100

Parameters

TRAIT = *variates*	Quantitative trait to be analysed; must be set
GENOTYPES = *factors*	Genotype factor; must be set
ENVIRONMENTS = *factors*	Environment factor; must be set for a multi-environment trial
POPULATIONS = factors	Population factor; must be set for a multiple-population analysis
UNITERROR = *variate*	Uncertainty on trait means (derived from individual unit or plot error) to be included in QTL analysis; default * i.e. omitted
VCINITIAL = *pointers*	Initial values for the parameters ofthe variance-covariance model
ADDITIVEPREDICTORS = *pointers*	Additive genetic predictors; must be set
ADD2PREDICTORS = *pointers*	Second (paternal) set of additive genetic predictors
DOMINANCEPREDICTORS = *pointers*	Dominance genetic predictors
CHROMOSOMES = *factors*	Chromosomes corresponding to the genetic predictors; must be set
POSITIONS = *variates*	Positions on the chromosomes corresponding to the genetic predictors; must be set
IDLOCI = *texts*	Labels for the loci
IDMGENOTYPES = *texts*	Labels for the genotypes corresponding to the genetic predictors
IDEFFECTS = *texts*	Labels for the effects along the y-axis, in the frame below the profile plot
IDPARENTS = *texts*	Labels to use to identify the parents
QSTATISTICS = *variates*	Saves test statistics for QTL effects along the genome
QEFFECTS = *pointers*	Saves QTL effects along the genome
QSE = *pointers*	Saves standard errors of the QTL effects
OUTFILENAME = *texts*	Name of the Genstat workbook file (*.gwb) to be created
DFILENAME = *texts*	Name of the graphics file for the plots

QMTBACKSELECT procedure

Performs a QTL backward selection for loci in multi-trait trials (M.P. Boer, M. Malosetti, S.J. Welham & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default summ
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set
ALPHALEVEL = *scalar*	Defines a significance level; default 0.05
VCMODEL = *string token*	Defines the variance-covariance model for the set of traits (identity, diagonal, cs, hcs, outside, fa, fa2, unstructured); default cs
VCPARAMETERS = *string token*	Whether to re-estimate the variance-covariance model parameters (estimate, fix); default esti
VCSELECT = *string token*	Whether to re-select the variance-covariance model (no, yes); default no
STANDARDIZE = *string token*	How to standardize the traits (none, normalize) ; default norm
CRITERION = *string token*	Criterion to use for model selection (aic, sic); default sic
FIXED = *formula*	Defines extra fixed effects
UNITFACTOR = *factor*	Saves the units factor required to define the random model

	when UNITERROR is to be used
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default expl, yvar
MAXCYCLE = *scalar*	Limit on the number of iterations; default 100
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm; default 100

Parameters

Y = *variates*	Quantitative traits to be analysed; must be set
GENOTYPES = *factors*	Genotype factor; must be set
FTRAITS = *factors*	Factor indicating the trait of each y-value; must be set
UNITERROR = *variates*	Uncertainty on trait means (derived from individual unit or plot error) to be included in QTL analysis; default * i.e. omitted
VCINITIAL = *pointers*	Initial values for the parameters of the variance-covariance model
SELECTEDMODEL = *texts*	VCMODEL setting for the selected covariance structure
ADDITIVEPREDICTORS = *pointers*	Additive genetic predictors; must be set
ADD2PREDICTORS = *pointers*	Second (paternal) set of additive genetic predictors
DOMINANCEPREDICTORS = *pointers*	Dominance genetic predictors
CHROMOSOMES = *factors*	Chromosomes corresponding to the genetic predictors; must be set
POSITIONS = *variates*	Positions on the chromosomes corresponding to the genetic predictors; must be set
IDLOCI = *texts*	Labels for the loci
IDMGENOTYPES = *texts*	Labels for the genotypes corresponding to the genetic predictors
QTLCANDIDATES = *variates*	Specifies the locus index numbers from which to start the selection; must be set
QTLSELECTED = *variates*	Saves the index numbers of the selected QTLs
INTERACTIONS = *variates*	Saves a logical variate indicating whether each selected QTL showed a significant (1) or non-significant (0) QTL-by-trait interaction
DOMSELECTED = *variates*	Saves a logical variate indicating whether each selected QTL showed a significant (1) or non-significant (0) effect of the DOMINANCEPREDICTORS
DOMINTERACTIONS = *variates*	Saves a logical variate indicating whether each selected QTL showed a significant (1) or non-significant (0) dominance-by-trait interaction
WALDSTATISTICS = *variates*	Saves the Wald test statistics
PRWALD = *variates*	Saves the associated Wald probabilities

QMTESTIMATE procedure

Calculates QTL effects in multi-trait trials (M.P Boer, M. Malosetti, S.J. Welham & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default summ
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set
NGENERATIONS = *scalar*	Number of generations of selfing for a RIL population
NBACKCROSSES = *scalar*	Number of backcrosses for a BCxSy population
NSELFINGS = *scalar*	Number of selfings for a BCxSy population
VCMODEL = *string token*	Specifies the variance-covariance model for the set of traits

	(identity, diagonal, cs, hcs, outside, fa, fa2, unstructured); default cs
VCPARAMETERS = *string token*	Whether to re-estimate the variance-covariance model parameters (estimate, fix); default esti
VCSELECT = *string token*	Whether to re-select the variance-covariance model (no, yes); default no
STANDARDIZE = *string token*	How to standardize the traits (none, normalize) ; default norm
CRITERION = *string token*	Criterion to use for model selection (aic, sic); default sic
FIXED = *formula*	Defines extra fixed effects
UNITFACTOR = *factor*	Saves the units factor required to define the random model when UNITERROR is to be used
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default expl, yvar
MAXCYCLE = *scalar*	Limit on the number of iterations; default 100
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm; default 100

Parameters

Y = *variates*	Quantitative traits to be analysed; must be set
GENOTYPES = *factors*	Genotype factor; must be set
FTRAITS = *factors*	Factor indicating the trait of each y-value; must be set
UNITERROR = *variate*	Uncertainty on trait means (derived from individual unit or plot error) to be included in QTL analysis; default * i.e. omitted
VCINITIAL = *pointers*	Initial values for the parameters of the variance-covariance model
SELECTEDMODEL = *texts*	VCMODEL setting for the selected covariance structure
ADDITIVEPREDICTORS = *pointers*	Additive genetic predictors; must be set
ADD2PREDICTORS = *pointers*	Second (paternal) set of additive genetic predictors
DOMINANCEPREDICTORS = *pointers*	Dominance genetic predictors
CHROMOSOMES = *factors*	Chromosomes corresponding to the genetic predictors; must be set
POSITIONS = *variates*	Positions on the chromosomes corresponding to the genetic predictors; must be set
IDLOCI = *texts*	Labels for the loci; must be set
MKLOCI = *variates*	Logical variate containing the value 1 if the locus is a marker, otherwise 0; must be set
IDMGENOTYPES = *texts*	Labels for the genotypes corresponding to the genetic predictors
IDPARENTS = *texts*	Labels to identify the parents
QTLSELECTED = *variates*	Index numbers of the selected QTLs; must be set
INTERACTIONS = *variates*	Logical variate indicating whether each selected QTL has a significant (1) or non-significant (0) QTL-by-trait interaction
DOMSELECTED = *variates*	Logical variate indicating whether the dominance predictor of each selected QTL must be present (1) or absent (0) in the model
DOMINTERACTIONS = *variates*	Logical variate indicating whether the dominance-by-trait interaction of each selected QTL must be present (1) or absent (0) in the model
RESIDUALS = *variates*	Residuals from the analysis
FITTEDVALUES = *variates*	Fitted values from the analysis
WALDSTATISTICS = *variates*	Saves the Wald test statistics
PRWALD = *variates*	Saves the associated Wald probabilities
DFWALD = *variates*	Saves the degrees of freedom for the Wald test
QEFFECTS = *pointers*	Saves the estimated QTL effects
QSE = *pointers*	Saves the standard errors of the QTL effects

OUTFILENAME = *texts*	Name of the Genstat workbook file (*.gwb) to be created
QSAVE = *pointers*	Saves a pointer with information and results for the significant effects
SAVE = *REML save structures*	Save the details of each REML analysis for use in subsequent VDISPLAY and VKEEP directives

QMTQTLSCAN procedure

Performs a genome-wide scan for QTL effects (Simple and Composite Interval Mapping) in multi-trait trials (M.P. Boer, M. Malosetti, S.J. Welham & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, progress, model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default summ
PLOT = *string token*	Whether to plot the profile along the genome (profile); default prof
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set
ALPHALEVEL = *scalar*	Defines a genome-wide significance level to calculate the threshold; default 0.05
VCMODEL = *string token*	Specifies the variance-covariance model for the set of traits (identity, diagonal, cs, hcs, outside, fa, fa2, unstructured); default cs
VCPARAMETERS = *string token*	Whether to re-estimate the variance-covariance model parameters (estimate, fix); default esti
STANDARDIZE = *string token*	How to standardize the traits (none, normalize) ; default norm
COFACTORS = *variate*	Index numbers of loci to be used as cofactors for the genetic background
COFWINDOW = *scalar*	Specifies a window for cofactor exclusion from the model; default 10^6 which means that all cofactors on the same chromosomes are excluded
THRMETHOD = *string token*	Which method to use to calculate the threshold for QTL detection (bonferroni, liji, given); default liji
THRESHOLD = *scalar*	Threshold value for test statistic when THRMETHOD=given
DISTANCE = *scalar*	Distance between loci when THRMETHOD=bonferroni; default 4
FIXED = *formula*	Formula with extra fixed terms
UNITFACTOR = *factor*	Saves the units factor required to define the random model when UNITERROR is to be used
STATISTICTYPE = *string token*	Which test statistic to plot and save using the STATISTICS parameter (wald, minlog10p); default minl
COLOURS = *scalar*, *variate* or *text*	Colours to use for the chromosomes; default * uses the colours of pens 1, 2 up to the number of chromosomes
TITLE = *text*	General title for the plot
YLOWERTITLE = *text*	Title for the y-axis of the lower graph(s); default 'Traits'
YUPPERTITLE = *text*	Title for the y-axis of the upper graph; default uses the identifier of the STATISTICS variate or pointer
XTITLE = *string*	Title for the x-axis; default 'Chromosomes'
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default expl, yvar
MAXCYCLE = *scalar*	Limit on the number of iterations; default 100
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm; default 100

Parameters

`Y` = *variates*	Quantitative traits to be analysed; must be set
`GENOTYPES` = *factors*	Genotype factor; must be set
`FTRAITS` = *factors*	Factor indicating the trait of each y-value; must be set
`UNITERROR` = *variate*	Uncertainty on trait means (derived from individual unit or plot error) to be included in QTL analysis; default `*` i.e. omitted
`VCINITIAL` = *pointers*	Initial values for the parameters ofthe variance-covariance model
`ADDITIVEPREDICTORS` = *pointers*	Additive genetic predictors; must be set
`ADD2PREDICTORS` = *pointers*	Second (paternal) set of additive genetic predictors
`DOMINANCEPREDICTORS` = *pointers*	Dominance genetic predictors
`CHROMOSOMES` = *factors*	Chromosomes corresponding to the genetic predictors; must be set
`POSITIONS` = *variates*	Positions on the chromosomes corresponding to the genetic predictors; must be set
`IDLOCI` = *texts*	Labels for the loci
`IDMGENOTYPES` = *texts*	Labels for the genotypes corresponding to the genetic predictors
`IDEFFECTS` = *texts*	Labels for the effects along the y-axis, in the frame below the profile plot
`IDPARENTS` = *texts*	Labels to use to identify the parents
`QSTATISTICS` = *variates*	Saves test statistics for QTL effects along the genome
`QEFFECTS` = *pointers*	Saves QTL effects along the genome
`QSE` = *pointers*	Saves standard errors of the QTL effects
`OUTFILENAME` = *texts*	Name of the Genstat workbook file (`*.gwb`) to be created
`DFILENAME` = *texts*	Name of the graphics file for the plots

`QMVAF` procedure

Calculates percentage variance accounted for by QTL effects in a multi-environment analysis (S.J. Welham, M.P. Boer, M.Malosetti & J.T.N.M. Thissen).

Options

`PRINT` = *string token*	What to print (`summary`); default `summ`
`SELECTION` = *string tokens*	What types of statistics to calculate (`add`, `drop`, `cumulative`); default `add`, `drop`, `cumu`
`METHOD` = *string tokens*	What methods to use to calculate the percentage variance accounted for (`trace`, `determinant`); default `trac`, `dete`
`VCMODEL` = *string token*	Specifies the variance-covariance model for the set of environments (`identity`, `diagonal`, `cs`, `hcs`, `outside`, `fa`, `fa2`, `unstructured`); default `cs`
`FIXED` = *formula*	Defines extra fixed effects
`UNITFACTOR` = *factor*	Saves the units factor required to define the random model when `UNITERROR` is to be used
`MVINCLUDE` = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (`explanatory`, `yvariate`); default `expl`, `yvar`
`MAXCYCLE` = *scalar*	Limit on the number of iterations; default 100
`WORKSPACE` = *scalar*	Number of blocks of internal memory to be set up for use by the `REML` algorithm; default 100

Parameters

`TRAIT` = *variates*	Quantitative trait to be analysed; must be set
`GENOTYPES` = *factors*	Genotype factor; must be set
`ENVIRONMENTS` = *factors*	Environment factor; must be set
`UNITERROR` = *variate*	Uncertainty on trait means (derived from individual unit or plot error) to be included in QTL analysis; default `*` i.e. omitted

VCINITIAL = *pointers*	Initial values for the parameters of the variance-covariance model
ADDITIVEPREDICTORS = *pointers*	Additive genetic predictors; must be set
CHROMOSOMES = *factors*	Chromosomes corresponding to the genetic predictors; must be set
POSITIONS = *variates*	Positions on the chromosomes corresponding to the genetic predictors; must be set
IDLOCI = *texts*	Labels for the loci
QTLSELECTED = *variates*	Index numbers of the selected QTLs; must be set
INTERACTIONS = *variates*	Logical variate indicating whether each selected QTL has a significant (1) or non-significant (0) QTL-by-environment interaction
OUTFILENAME = *texts*	Name of the Genstat workbook file (*.gwb) to be created

QMVESTIMATE procedure

Replaces missing molecular marker scores using conditional genotypic probabilities (D.A. Murray & M. Malosetti).

Options

POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy); must be set
NGENERATIONS = *scalar*	Number of generations of selfing for a RIL population
NBACKCROSSES = *scalar*	Number of backcrosses for a BCxSy population
NSELFINGS = *scalar*	Number of selfings for a BCxSy population

Parameters

MKSCORES = *pointers*	Genotype codes for each marker; must be set
CHROMOSOMES = *factors*	The chromosome where each marker is located; must be set
POSITIONS = *variates*	The position on the chromosome of each marker; must be set
MKNAMES = *texts*	Marker names; must be set
IDMGENOTYPES = *texts*	Labels for the genotypes
PARENTS = *pointers*	Parent information; must be set
IDPARENTS = *texts*	Labels used to identify the parents; must be set
NEWMKSCORES = *pointers*	Saves the imputed genotype codes for each marker; if this is not set, the imputed values overwrite those in MKSCORES

QMVREPLACE procedure

Replaces missing marker scores with the mode scores of the most similar genotypes (L.C.P. Keizer, J.T.N.M. Thissen & F.A. van Eeuwijk).

Options

PRINT = *string tokens*	What to print (summary, similarity, neighbours, details); default summ
NNEIGHBOURS = *scalar*	Number of nearest neighbours; default 5
MAXDISTANCE = *scalar*	Maximum similarity difference; default 0.1

Parameters

MKSCORES = *pointers*	Pointer with the original marker scores; must be set
MKNAMES = *texts*	Marker names
IDMGENOTYPES = *texts*	Labels for genotypes
NEWMKSCORES = *pointers*	Pointer to store the new marker scores; must be set

QNORMALIZE procedure

Performs quantile normalization (D.B. Baird).

Options

PRINT = *string token*	What to print (summary); default summ
PLOT = *string tokens*	What to plot (cdf, histogram, ncdf, nhistogram); default hist, nhis
METHOD = *string token*	Whether to use means, medians or geometric means for the averaged normalized distribution (means, medians,

	geometricmeans); default mean
ARRANGEMENT = *string token*	Whether to use trellis or single plots for PLOT=cdf or ncdf (single, trellis); default trel
DEVICE = *scalar*	Device number on which to plot the graphs
GRAPHICSFILE = *text*	What graphics filename template to use to save the graphs; default *

Parameters

DATA = *variates* or *pointers*	Data values
GROUPS = *factors* or *texts*	Groupings of the data values
NEWDATA = *variates* or *pointers*	Saves the normalized values; if this is unset, they replace the original values in DATA

QRD directive

Calculates QR decompositions of matrices.

Option

PRINT = *string tokens*	Printed output required (orthogonalmatrix, uppertriangularmatrix); default * i.e. no printing

Parameters

INMATRIX = *matrices* or *symmetric matrices*	Matrices to be decomposed
ORTHOGONALMATRIX = *matrices*	Orthogonal matrix of each decomposition
UPPERTRIANGULARMATRIX = *matrices*	Upper-triangular matrix of each decomposition

QRECOMBINATIONS procedure

Calculates the expected numbers of recombinations and the recombination frequencies between markers (J. Jansen, J.T.N.M. Thissen & M.P. Boer).

Options

PRINT = *string tokens*	What to print (summary, positions); default summ
PLOT = *string token*	What to plot (frequencies); default freq
POPULATIONTYPE = *string token*	Type of population (F2, BC1, RIL, DH1, CP); must be set
METHOD = *string token*	Which method to use (twopoint, multipoint); default twop
USEPENALTY = *string token*	Whether to increase the number of recombinations when METHOD=twopoint by 0.5 recombination per informative meiosis for each missing marker score (yes, no); default no
TITLE = *text*	General title for the plot

Parameters

MKSCORES = *pointers*	Marker scores for each marker; must be set
CHROMOSOMES = *factors*	Factor defining the linkage groups
POSITIONS = *variates*	Saves the positions of the markers when METHOD=multipoint
MKNAMES = *texts*	Names of the markers; must be set
PARENTS = *pointers*	Marker scores of the parents; must be set
ORDER = *variates*	Order of the markers for METHOD=multipoint
NRECOMBINATIONS = *symmetric matrices* or *pointers*	Saves the number of recombinations
RECFREQUENCIES = *symmetric matrices* or *pointers*	Saves the recombination frequencies
PHASESWITCHES = *pointers*	Saves the phase switches for pairs of markers when POPULATIONTYPE=CP
INHERITANCEVECTORS = *pointers*	Saves the inheritance vectors when METHOD=multipoint
GENNRECOMBINATIONS = *variates*	Saves the numbers of recombinations of the genotypes when METHOD=multipoint

QREPORT procedure

Creates an HTML report from QTL linkage or association analysis results (D.A. Murray).

Options

OUTFILEPREFIX = *text*	Prefix to use for the files that are generated
WORKDIRECTORY = *text*	Working directory to use for files; default current Genstat working directory
CHROMOSOMES = *factor*	Factor defining linkage groups for the genetic map
POSITIONS = *variate*	Positions of markers within the linkage groups for the genetic map
HTMLHEAD = *text*	Text structure containing custom content for the header of the HTML report file

Parameter

QSAVE = *pointers*	Information and results saved from an earlier QTL analysis

QSASSOCIATION procedure

Performs marker-trait association analysis in a genetically diverse population using bi-allelic and multi-allelic markers (M. Malosetti & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, progress); default summ
PLOT = *string tokens*	What to plot (profile, qq, map); default prof, qq
RELATIONSHIPMODEL = *string token*	What model to use to account for genetic relatedness (eigenanalysis, kinship, subpopulations, null); default kins
SCORES = *pointer*	Provides the scores of significant principal components, obtained from an eigenvalue analysis
METHOD = *string token*	What model to use for GWAS (exact, fast); default fast
ALPHALEVEL = *scalar*	Defines a genome-wide significance level to calculate the threshold; default 0.05
THRMETHOD = *string token*	Method to define the threshold for significance (neffective, bonferroni, given); default neff
THRESHOLD = *scalar*	Threshold value for significant LD, on the −log10 scale; default 2
DISTANCE = *scalar*	Minimum distance gap between independent tests (i.e. distance beyond which loci are expected to be in linkeage equilibrium) when THRMETHOD=bonferroni; default *
MINORALLELE = *scalar*	Frequency of minor alleles; default 0.05
KMATRIX = *symmetric matrix*	Kinship matrix containing coefficients of coancestries
KMETHOD = *string token*	Method to use to estimate kinship matrix if not supplied by KMATRIX (correlation, dice); default dice
SUBPOPULATIONS = *factor*	Defines groupings of genotypes into subpopulations
MODELPART = *string token*	Defines which part of the model should include SUBPOPULATIONS if RELATIONSHIPMODEL is set to subpopulations, or the principal components scores if RELATIONSHIPMODEL is set to eigenanalysis (fixed, random); default rand
SCALING = *string token*	Whether to scale the scores by the square roots of their singular values (singularvalues, none); default none
STANDARDIZE = *string token*	Whether to standardize the marker scores according to their frequencies (frequency, none); default freq
COLOURS = *scalar*, *variate* or *text*	Colours to use for the chromosomes; default * uses the colours of pens 1, 2 up to the number of chromosomes
TITLE = *text*	General title for the plots
YTITLE = *text*	Title for the y-axis
XTITLE = *text*	Title for the x-axis

Parameters

TRAIT = *variates*	Phenotypic trait to analyse; must be set

GENOTYPES = *factors*	Genotype factor
MKSCORES = *pointers*	Genotype codes for each marker; must be set
CHROMOSOMES= *factors*	Linkage groups for the markers; must be set
POSITIONS = *variates*	Positions within the linkage groups of markers; must be set
MKNAMES = *texts*	Marker names
IDMGENOTYPES = *texts*	Labels for the genotypes corresponding to the markers
GENFILENAME = *texts*	Name of a comma-delimited file (`*.csv`) containing marker scores (with markers in the rows and genotypes in the columns)
MAPFILENAME = *texts*	Name of a comma-delimited file (`*.csv`) with map information
WALDSTATISTICS = *variates*	Saves the Wald test statistics
NDF = *variates*	Saves the degrees of freedom associated with the Wald test
MINLOG10P = *variates*	Saves the associated probability values of the Wald test statistics, on a −log10 scale
LAMBDA = *scalars*	Saves the inflation factor i.e. slope of the QQ plot of −log10(P) values
QSAVE = *pointers*	Saves a pointer with information and results for the significant effects
DFILENAME = *texts*	Name of the graphics file for the plots

QSBACKSELECT procedure

Performs a QTL backward selection for loci in single-environment trials (M.P. Boer, M. Malosetti, S.J. Welham & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (`summary`, `model`, `components`, `effects`, `means`, `stratumvariances`, `monitoring`, `vcovariance`, `deviance`, `Waldtests`, `missingvalues`, `covariancemodels`); default `summ`
POPULATIONTYPE = *string token*	Type of population (`BC1`, `DH1`, `F2`, `RIL`, `BCxSy`, `CP`); must be set
ALPHALEVEL = *scalar*	Defines a significance level; default 0.05
FIXED = *formula*	Formula with extra fixed effects
UNITFACTOR = *factor*	Saves the units factor required to define the random model when `UNITERROR` is to be used
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (`explanatory`, `yvariate`); default `expl`, `yvar`
MAXCYCLE = *scalar*	Limit on the number of iterations; default 100
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the `REML` algorithm; default 100

Parameters

TRAIT = *variates*	Quantitative trait to be analysed; must be set
GENOTYPES = *factors*	Genotype factor; must be set
UNITERROR = *variates*	Uncertainty on trait means (derived from individual unit or plot error) to be included in QTL analysis; default `*` i.e. omitted
ADDITIVEPREDICTORS = *pointers*	Additive genetic predictors; must be set
ADD2PREDICTORS = *pointers*	Second (paternal) set of additive genetic predictors
DOMINANCEPREDICTORS = *pointers*	Dominance genetic predictors
CHROMOSOMES = *factors*	Chromosomes corresponding to the genetic predictors; must be set
POSITIONS = *variates*	Positions on the chromosomes corresponding to the genetic predictors; must be set
IDLOCI = texts	Labels for the loci
IDMGENOTYPES = *texts*	Labels for the genotypes corresponding to the genetic

predictors

QTLCANDIDATES = *variates*	Specifies the locus index numbers from which to start the selection; must be set
QTLSELECTED = *variates*	Saves the index numbers of the selected QTLs; must be set
DOMSELECTED = *variates*	Logical indicator variable storing one where the selected QTLs show a significant effect of the dominance predictor, zero otherwise
WALDSTATISTICS = *variates*	Saves the Wald test statistics
PRWALD = *variates*	Saves the associated Wald probabilities

QSELECTIONINDEX procedure

Calculates (molecular) selection indexes by using phenotypic information and/or molecular scores of multiple traits (M. Malosetti & F.A. van Eeuwijk).

Options

PRINT = *string tokens*	What to print (summary); default summ
METHOD = *string token*	Defines which index to calculate (simple, smithhazel, landethompson); default smit
INTENSITY = *scalar*	Specifies the selection intensity expressed as the percentage of individuals of the population to select; default 10

Parameters

TRAITS = *pointers*	Pointer with a variate for each trait, supplying the phenotypic values for the genotypes; must be set
MOLECULARSCORES = *pointers*	Pointer with a variate for each trait, supplying QTL-based predictions or genomic predictions
GENOTYPES = *factors*	Genotype factor; must be set
IDMGENOTYPES = *texts*	Labels of the genotypes
WEIGHTS = *variates*	Specifies economic weights for the traits; if unset, all traits have weight one
VCPHENOTYPIC = *symmetric matrices*	Specifies the phenotypic variance-covariance matrix of the traits
VCGENETIC = *symmetric matrices*	Specifies the genotypic variance-covariance matrix of the traits
HERITABILITY = *symmetric matrices*	Specifies the heritabilities and coheritabilities of the traits
SELECTIONINDEX = *variates*	Saves the selection index

QSESTIMATE procedure

Calculates QTL effects in single-environment trials (M.P. Boer, M. Malosetti, S.J. Welham & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default summ
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set
NGENERATIONS = *scalar*	Number of generations of selfing for a RIL population
NBACKCROSSES = *scalar*	Number of backcrosses for a BCxSy population
NSELFINGS = *scalar*	Number of selfings for a BCxSy population
FIXED = *formula*	Defines extra fixed effects
UNITFACTOR = *factor*	Saves the units factor required to define the random model when UNITERROR is to be used
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default expl, yvar
MAXCYCLE = *scalar*	Limit on the number of iterations; default 100
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm; default 100

Parameters

TRAIT = *variates*	Quantitative trait to be analysed; must be set
GENOTYPES = *factors*	Genotype factor; must be set
UNITERROR = *variates*	Uncertainty on trait means (derived from individual unit or plot error) to be included in QTL analysis; default * i.e. omitted
ADDITIVEPREDICTORS = *pointers*	Additive genetic predictors; must be set
ADD2PREDICTORS = *pointers*	Second (paternal) set of additive genetic predictors
DOMINANCEPREDICTORS = *pointers*	Dominance genetic predictors
CHROMOSOMES = *factors*	Chromosomes corresponding to the additive genetic predictors; must be set
POSITIONS = *variates*	Positions on the chromosomes corresponding to the additive genetic predictors; must be set
IDLOCI = *texts*	Labels for the loci
MKLOCI = *variates*	Logical variate containing the value 1 if the locus is a marker, otherwise 0; must be set
IDMGENOTYPES = *texts*	Labels for the genotypes corresponding to the the additive genetic predictors
IDPARENTS = *texts*	Labels to identify the parents
QTLSELECTED = *variates*	Index numbers of the selected QTLs; must be set
DOMSELECTED = *variates*	Logical variate indicating whether the dominance predictor of each selected QTL must be present (1) or absent (0) in the model
RESIDUALS = *variates*	Residuals from the analysis
FITTEDVALUES = *variates*	Fitted values from the analysis
WALDSTATISTICS = *variates*	Saves the Wald test statistics
PRWALD = *variates*	Saves the associated Wald probabilities
QEFFECTS = *pointers*	Saves the estimated QTL effects
QSE = *pointers*	Saves the standard errors of the QTL effects
OUTFILENAME = *texts*	Name of the Genstat workbook file (*.gwb) to be created
QSAVE = *pointers*	Saves a pointer with information and results for the significant effects
SAVE = *REML save structures*	Save the details of each REML analysis for use in subsequent VDISPLAY and VKEEP directives

QSIMULATE procedure

Simulates marker data and QTL effects for single and multiple environment trials (M.P. Boer & J.T.N.M. Thissen).

Options

PRINT = *string token*	What to print (summary); default summ
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set
NGENERATIONS = *scalar*	Number of generations for a RIL population; default 3
NBACKCROSSES = *scalar*	Number of backcrosses for a BCxSy population; default 2
NSELFINGS = *scalar*	Number of selfings for a BCxSy population; default 3
GENOMELENGTH = *variate*	Length in cM for each chromosome
DISTANCE = *scalar*	Distance between the markers in cM; default 1 cM
COMPLETE = *string token*	Complete marker information, i.e. all parents have a different allele (yes, no); default no
FRACTIONMISSING = *scalar*	Fraction of the markers with missing values; default 0
NGENOTYPES = *scalar*	Number of genotypes; must be set
NCHROMOSOMES = *scalar*	Number of chromosomes
NPOSITIONS = *scalar*	Number of positions per chromosome
IDPARENTS = *texts*	Labels used to identify the parents
MEAN = *scalar* or *variate*	Mean of the trait for each environment; must be set if TRAIT is set

VARIANCE = *scalar* or *variate* — Variance of the trait for each environment; must be set if TRAIT is set

ADDITIVEEFFECTS = *variate* or *pointer* — Additive effects of each QTL for each environment; must be set if TRAIT is set

ADD2PREDICTORS = *pointers* — Second (paternal) set of additive genetic predictors of each QTL for each environment if POPULATIONTYPE is CP; must be set if TRAIT is set

DOMINANCEPREDICTORS = *pointers* — Dominance genetic predictors of each QTL for each environment if POPULATIONTYPE is F2 or CP; must be set if TRAIT is set

QTLCHROMOSOMES = *variate* — Chromosome number for each QTL; must be set if TRAIT is set

QTLPOSITIONS = *variate* — Position on the QTLCHROMOSOMES for each QTL; must be set if TRAIT is set

Parameters

TRAIT = *variates* — Saves the quantitative trait values

GENOTYPES = *factors* — Saves the genotype factor

ENVIRONMENTS = *factors* — Saves the environment factor

MKSCORES = *pointers* — Saves the marker scores for each marker

CHROMOSOMES = *factors* — Saves the linkage groups of the markers

POSITIONS = *variates* — Saves the position on the chromosome for each marker

MKNAMES = *texts* — Names of the markers

IDMGENOTYPES = *texts* — Labels of the genotypes

PARENTS = *pointers* — Saves the parent information

SEED = *scalars* — Specifies a seed to use for the random number generator; default 0 continues from the previous generation or (if none) initializes the seed automatically

QSQTLSCAN procedure

Performs a genome-wide scan for QTL effects (Simple and Composite Interval Mapping) in single-environment trials (M.P. Boer, M. Malosetti, S.J. Welham & J.T.N.M. Thissen).

Options

PRINT = *string tokens* — What to print (summary, progress, model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default summ

PLOT = *string token* — Whether to plot the profile along the genome (profile); default prof

POPULATIONTYPE = *string token* — Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set

ALPHALEVEL = *scalar* — Defines a genome-wide significance level to calculate the threshold; default 0.05

COFACTORS = *variate* — Index numbers of loci to be used as cofactors for the genetic background

COFWINDOW = *scalar* — Specifies a window for cofactor exclusion from the model; default 10^6 which means that all cofactors on the same chromosomes are excluded

THRMETHOD = *string token* — Which method to use to calculate the threshold for QTL detection (bonferroni, liji, given); default liji

THRESHOLD = *scalar* — Threshold value for test statistic when THRMETHOD=given

DISTANCE = *scalar* — Distance between loci when THRMETHOD=bonferroni; default 4

FIXED = *formula* — Formula with extra fixed terms

UNITFACTOR = *factor* — Saves the units factor required to define the random model when UNITERROR is to be used

STATISTICTYPE = *string token*	Which test statistic to plot and save using the STATISTICS parameter (wald, minlog10p); default minl
COLOURS = *scalar*, *variate* or *text*	Colours to use for the chromosomes; default * uses the colours of pens 1, 2 up to the number of chromosomes
TITLE = *text*	General title for plot
YTITLE = *text*	Title for the y-axis; default uses the identifier of the STATISTICS variate or pointer
XTITLE = *text*	Title for the x-axis; default 'Chromosomes'
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default expl, yvar
MAXCYCLE = *scalar*	Limit on the number of iterations; default 100
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm; default 100

Parameters

TRAIT = *variates*	Quantitative trait to be analysed; must be set
GENOTYPES = *factors*	Genotype factor; must be set
UNITERROR = *variates*	Uncertainty on trait means (derived from individual unit or plot error) to be included in QTL analysis; default * i.e. omitted
ADDITIVEPREDICTORS = *pointers*	Additive genetic predictors; must be set
ADD2PREDICTORS = *pointers*	Second (paternal) set of additive genetic predictors
DOMINANCEPREDICTORS = *pointers*	Dominance genetic predictors
CHROMOSOMES = *factors*	Chromosomes corresponding to the genetic predictors; must be set
POSITIONS = *variates*	Positions on the chromosomes corresponding to the genetic predictors; must be set
IDLOCI = *texts*	Labels for the loci
IDMGENOTYPES = *texts*	Labels for the genotypes corresponding to the genetic predictors
IDEFFECTS = *texts*	Labels for the effects along the y-axis, in the frame below the profile plot
IDPARENTS = *texts*	Labels to use to identify the parents
QSTATISTICS = *variates*	Saves test statistics for QTL effects along the genome
QEFFECTS = *pointers*	Saves QTL effects along the genome (additive effects,and, if specified, also second additive and dominance effects)
QSE = *pointers*	Saves standard errors of the QTL effects
OUTFILENAME = *texts*	Name of the Genstat workbook file (*.gwb) to be created
DFILENAME = *texts*	Name of the graphics file for the plots

QTHRESHOLD procedure

Calculates a threshold to identify a significant QTL (M.P. Boer & J.T.N.M. Thissen).

Options

PRINT = *string token*	What to print (summary); default summ
POPULATIONTYPE = *string token*	Type of population (BC1, DH1, F2, RIL, BCxSy, CP); must be set
THRMETHOD = *string token*	Which method to use (bonferroni, liji); default liji
STATISTICTYPE = *string token*	Which type of test statistic to use (wald, minlog10p); default minl
ALPHALEVEL = *scalar*	Defines the genome-wide significance level; default 0.05
DISTANCE = *scalar*	Distance between evaluation points for THRMETHOD=bonferroni; default 4
DF = *scalar*	Degrees of freedom for the Wald test; default 1

Parameters

CHROMOSOMES = *factors*	Chromosome for each locus; must be set
POSITIONS = *variates*	Position on the chromosome for each locus; must be set

ADDITIVEPREDICTORS = *pointers*	Additive genetic predictors
ADD2PREDICTORS = *pointers*	The second (paternal) additive genetic predictors if POPULATIONTYPE is CP
DOMINANCEPREDICTORS = *pointers*	The dominance genetic predictors if POPULATIONTYPE is F2 or CP
THRESHOLD = *scalars*	Saves the calculated threshold

QUANTILE procedure

Calculates quantiles of the values in a variate (P.W. Lane).

Options

PRINT = *string token*	What to print (quantiles); default quan
METHOD = *string token*	Type of quantile to form (population, sample); default samp
PROPORTION = *variate* or *scalar*	Proportions at which to calculate quantiles; default !(0,0.25,0.5,0.75,1)

Parameters

DATA = *variates*	Values whose quantiles are required; this parameter must be specified
QUANTILES = *variates* or *scalars*	Identifiers of structures to store results, if required

QUESTION procedure

Obtains a response using a Genstat menu (S.A. Harding & R.W. Payne).

Options

PREAMBLE = *text*	Text posing a question; (no default)
PROMPT = *text*	Text to be used as final prompt; the default prompt specifies the mode of response and lists the default values (if any), in brackets, followed by ">"
RESPONSE = *identifier*	Structure to store response; default * allows a menu to be saved without being executed
MODE = *string token*	Mode of response (p, t, v); default p
DEFAULT = *identifier*	Response to be assumed if just <RETURN> is given; default is to repeat the prompt until a response is obtained
LIST = *string token*	Whether a list of responses, rather than a single response, is valid (yes, no); default no
DECLARED = *string token*	Whether identifiers must already be declared (yes, no); default no
TYPE = *string tokens*	Allowed types for identifiers (ASAVE, datamatrix i.e. pointer to variates of equal lengths as required in multivariate analysis, diagonalmatrix, dummy, expression, factor, formula, LRV, matrix, pointer, RSAVE, scalar, SSPM, symmetricmatrix, table, text, tree, TSAVE, TSM, variate, VSAVE); default *, meaning no limitation
PRESENT = *string token*	Whether the identifier must have values (yes, no); default no
LOWER = *scalar*	Lower limit for numbers; default *, meaning no check
UPPER = *scalar*	Upper limit for numbers; default *, meaning no check
HELP = *text*	Text to be used in response to a general query for the question; default *
SAVE = *pointer*	Previously allowed you to save or reinput the specification of the menu, but is now no longer supported

Parameters

VALUES = *texts*	Possible codes for MODE t; (no default for MODE t; not relevant for others)
CHOICE = *texts*	Text giving explanation of each letter code; (no default for MODE t; not relevant for others)
HELP = *texts*	Text to be used in response to a specific query for a code; default *

RADIALSPLINE procedure

Calculates design matrices to fit a radial-spline surface as a linear mixed model (S.J. Welham & D.B. Baird).

Options

ORTHOGONALIZATION = *string token*	How to orthogonalize the random basis (fixed, none); default fixe
SCALING = *scalar*	Scaling of the XRANDOM terms (automatic, none); default auto

Parameters

X1 = *variates* or *factors*	Coordinates in the first dimension for which spline values are required
X2 = *variates* or *factors*	Coordinates in the second dimension for which spline values are required
XFIXED = *matrices*	Saves the design matrix to define the fixed terms (excluding the constant) for fitting the radial spline
XRANDOM = *matrices*	Saves the design matrix to define the random terms for fitting the radial spline
X1KNOTS = *variates*	Specifies the coordinates in the first dimension of the internal knots used to form the basis for the spline
X2KNOTS = *variates*	Specifies the coordinates in the second dimension of the internal knots used to form the basis for the spline
PX1 = *variates*	Specifies the coordinates in the first dimension at which to predict
PX2 = *variates*	Specifies the coordinates in the second dimension at which to predict
PFIXED = *matrices*	Saves the design matrix for the fixed terms (excluding the constant) for the radial spline at the prediction points
PRANDOM = *matrices*	Saves the design matrix for the random terms for the radial spline at the prediction points

RANDOMIZE directive

Randomizes the units of a designed experiment or the elements of a factor or variate.

Options

BLOCKSTRUCTURE = *formula*	Block model according to which the randomization is to be carried out; default * i.e. as a completely-randomized design
EXCLUDE = *factors*	(Block) factors whose levels are not to be randomized
SEED = *scalar*	Seed for the random-number generator; default 0

Parameters

factors or *variates*	Structures whose units are to be randomized according to the defined block model

RANK procedure

Produces ranks, from the values in a variate, allowing for ties (J.B. van Biezen & C.J.F. ter Braak).

Option

OMIT = *string token*	Whether units excluded by a restriction on the DATA variate should be omitted from the RANKS variate (restricted); default *, i.e. the units are not omitted, and their values are left unchanged

Parameters

DATA = *variates*	Variate containing values to be ranked
RANKS = *variates*	Variate to save vector of ranks
TIESIZE = *variates*	Variate to save the sizes of ties

RAR1 procedure

Fits regressions with an AR1 or a power-distance correlation model (R.W. Payne).

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, cparameter, cmonitoring, cplot); default mode, summ, esti, cpar
CALCULATION = *expression structures*	Calculation of explanatory variates involving nonlinear parameters
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalars*	Limit for expansion of model terms; default 3
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
SELINEAR = *string token*	Whether to calculate s.e.s for linear parameters when nonlinear parameters are also estimated (yes, no); default no
WEIGHTS = *variate*	Prior weights for the units
CMETHOD = *string token*	Estimation method (maximumlikelihood, reml); default maxi
CPARAMETER = *scalars*	Correlation parameter
CPOSITIONS = *variate*	Correlation positions
CGROUPS = *factor*	Groupings of correlation positions
MAXCYCLE = *scalars*	Maximum number of iterations; default 100
TOLERANCE = *scalars*	Convergence criterion; default 10^{-5}

Parameter

TERMS = *formula*	Terms to be fitted

RBDISPLAY directive

Displays output from a radial basis function model fitted by RBFIT.

Option

PRINT = *strings*	Controls fitted output (description, estimates, fittedvalues, summary); default desc, esti, summ

Parameter

pointers	Save structure with details of the fitted model

RBFIT directive

Fits a radial basis function model.

Options

PRINT = *string tokens*	Controls fitted output (description, estimates, fittedvalues, summary); default desc, esti, summ

RBTYPE = *string token* — Type of radial basis function (linear, cubic, thinplate, gaussian, multiquadric, inversemultiquadric, cauchy); default line

METRIC = *string token* — How to calculate distances for the radial basis functions (euclidean, cityblock, manhattan, pythagorean); default eucl

SCALING = *string token* — Type of scaling used to compute distances (sd, mahalanobis, supplied); default sd

ALPHA = *scalar* — Specifies the value for the constant α, used to calculate radial distances for RBTPYE settings multiquadric, inversemultiquadric and cauchy; default 1

LAMBDA = *scalar* — Specifies the value of the penalty constant λ

TOLERANCE = *scalar* — Tolerance for setting eigenvalues equal to zero in the singular value decomposition; default 0.000001

Parameters

Y = *variates* — Response variates

X = *pointers* — Independent variates

CENTRES = *pointers* — Centres of the radial basis functions for the dependent variates

RBSCALING = *scalars* or *variates* — Scaling parameters for the radial distance calculations when SCALING=supplied; default 1

FITTEDVALUES = *variates* — Fitted values generated for each y-variate by the model

ESTIMATES = *variates* — Saves the estimated model parameters

EXIT = *scalars* — Saves the exit code

SAVE = *pointers* — Saves details of the model and the estimated parameters for RBDISPLAY or RBPREDICT

RBPREDICT directive

Forms predictions from a radial basis function model fitted by RBFIT.

Option

PRINT = *strings* — Controls fitted output (description, predictions); default desc, pred

Parameters

X = *pointers* — X-values at which to predict

PREDICTIONS = *variates* — Predictions

SAVE = *pointers* — Details of the fitted model

RBRADLEYTERRY procedure

Fits the Bradley-Terry model for paired-comparison preference tests (R.W. Payne).

Options

PRINT = *string tokens* — What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence, preferenceprobabilities); default mode, summ, esti

GROUPS = *factor* — Factor representing different test circumstances

COVARIATE = *variates* — Other covariates to include in the model

NOMESSAGE = *string tokens* — Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *

FPROBABILITY = *string token* — Printing of probabilities for variance and deviance ratios (yes, no); default no

TPROBABILITY = *string token* — Printing of probabilities for t-statistics (yes, no); default no

SELECTION = *string tokens* — Statistics to be displayed in the summary of analysis produced by PRINT=summary (%variance, %ss, adjustedr2, r2, dispersion, %meandeviance, %deviance, aic, bic, sic); default disp

DISPERSION = *scalar* — Dispersion parameter to be used as estimate for variability in

s.e.s etc; default 1

PROBABILITY = *scalar* — Probability level for confidence intervals for parameter estimates; default 0.95

Parameters

WINNERS = *factors* — Specifies the winners in the tests

LOSERS = *factors* — Specifies the loser in the tests

NWINS = *variates* or *scalars* — Number of wins; default 1

NBINOMIAL = *variates* or *scalars* — Number of trials; default 1

PREFERENCEPROBABILITIES = *matrices* or *pointers* — Saves the estimated probability that each object is preferred to other objects

LOWERPREFERENCEPROBABILITIES = *matrices* or *pointers* — Saves the lower values of the confidence intervals for the preference probabilities

UPPERPREFERENCEPROBABILITIES = *matrices* or *pointers* — Saves the upper values of the confidence intervals for the preference probabilities

SAVE = *identifiers* — To save the regression save structure

RCATENELSON procedure

Performs a Cate-Nelson graphical analysis of bivariate data (V.M. Cave).

Options

PRINT = *string tokens* — Controls printed output (summary, quadrants, errorquadrants); default summ, quad

PLOT = *string tokens* — What graphs to plot (catenelson, criticalvalues); default cate

DIRECTION = *string token* — Direction of the association between the y and x values (ascending, descending); default asce i.e. a positive trend

YCRITICAL = *scalar* — Pre-specified critical value of y; default * i.e. the critical value of y is estimated)

XCRITICAL = *scalar* — Pre-specified critical value of x; default * i.e. the critical value of x is estimated

TITLE = *text* — Title for the Cate-Nelson plot; if unset, the title is generated automatically

YTITLE = *text* — Y-axis title for the Cate-Nelson plot; if unset, the title is generated automatically

XTITLE = *text* — X-axis title for the Cate-Nelson plot; if unset, the title is generated automatically

WINDOW = *scalar* — Window to use for the graphs; default 3

SAVE = *identifier* — Specifies the save structure of regression model holding the y-values, distribution, link function and weights; default * i.e. that from last regression fitted

Parameters

X = *variates* — Supplies the x-values for each analysis

RESULTS = *pointers* — Saves the critical value of x, the critical value of y and the quadrant allocations for each X variate

RCHECK procedure

Checks the fit of a linear, generalized linear or nonlinear regression (P.W. Lane, R. Cunningham & C. Donnelly).

Options

PRINT = *string tokens* — What to print (index, y, residuals, leverages, Cook); default *

RMETHOD = *string token* — Type of residual to use (deviance, Pearson, simple, deletion); default * i.e. as set in MODEL

INDEX = *variate* or *factor* — Which variable to use as index; default !(1...n)

ENVELOPE = *string token*	Type of envelope with Normal and half-Normal plots (none, rough, smooth, asymptotic); default none
PROBABILITY = *scalar*	Approximate probability level for envelope; default 0.95
NSIMULATIONS = *scalar*	How many simulations to generate for rough or smooth envelopes; default (1+PROB)/(1−PROB)
SHADE = *string token*	Whether to show shaded envelope rather than boundaries (no, yes); default no
RESIDUALS = *variate*	To store chosen type of residuals; default *
LEVERAGES = *variate*	To store leverages; default *
COOK = *variate*	To store modified Cook's statistics; default *
GRAPHICS = *string token*	Type of graphics to use (lineprinter, highresolution); default high
TITLE = *text*	Title for graph; default identifier of response
WINDOW = *numbers*	Window or series of windows in which to display graphs; default 4, or 5...8 for composite
SCREEN = *string token*	Treatment of previous graphics screen (clear, keep); default clea
SAVE = *regression save structure*	Specifies which model to check; default *
Parameters	
YSTATISTIC = *string tokens*	What to display in the graph (residuals, Cook, leverages, absresiduals); default resi
XMETHOD = *string tokens*	What type of graph (fittedvalues, index, normal, halfnormal, histogram, composite); default comp

RCIRCULAR procedure

Does circular regression of mean direction for an angular response (P.W. Goedhart).

Options

PRINT = *string tokens*	What to print (model, summary, estimates, fittedvalues, monitoring); default mode, summ, esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default 3
RESIDUALS = *variate*	To save the residuals
FITTEDVALUES = *variate*	To save the fittedvalues, i.e. the fitted mean directions
LEVERAGES = *variate*	To save the leverages
ESTIMATES = *variate*	To save estimates of linear parameters
SE = *variate*	To save standard errors of the estimates
VCOVARIANCE = *symmetric matrix*	To save the variance-covariance matrix of the estimates
MU0 = *scalar*	To save the estimate of the mean parameter μ_0
SEMU0 = *scalar*	To save the standard error of the estimated mea parameter μ_0
KAPPA = *scalar*	To save the estimate of the concentration parameter κ of the von Mises distribution
SEKAPPA = *scalar*	To save the standard error of the estimated concentration parameter κ
_2LOGLIKELIHOOD = *scalar*	To save the value of minus twice the maximized log likelihood
DF = *scalar*	To save the residual degrees of freedom
ITERATIVEWEIGHTS = *variate*	To save the iterative weights
LINEARPREDICTOR = *variate*	To save the linear predictor
YADJUSTED = *variate*	To save the adjusted dependent variate
I_2LOGLIKELIHOOD = *variate*	To save the contribution of each unit to the value of minus twice the maximized log likelihood
MAXCYCLE = *scalar*	Maximum number of iterations for see-saw algorithm; default 30
TOLERANCE = *scalar*	Convergence criterion; default 10^{-5}
Parameter	
TERMS = *formula*	List of explanatory variates and factors, or model formula

RCOMPARISONS procedure

Calculates comparison contrasts amongst regression means (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (aov, contrasts); default aov, cont
COMBINATIONS = *string token*	Factor combinations for which to form the predicted means (present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment to be made when forming the predicted means (marginal, equal, observed); default marg
PSE = *string tokens*	Types of standard errors to be printed with the contrasts (contrasts, differences, lsd); default cont
WEIGHTS = *table*	Weights classified by some or all of the factors in the model; default *
OFFSET = *scalar*	Value of offset on which to base predictions; default mean of offset variate
METHOD = *string token*	Method of forming margin (mean, total); default mean
ALIASING = *string token*	How to deal with aliased parameters (fault, ignore); default faul
BACKTRANSFORM = *string token*	What back-transformation to apply to the values on the linear scale, before calculating the predicted means (link, none); default link
SCOPE = *string token*	Controls whether the variance of predictions is calculated on the basis of forecasting new observations rather than summarizing the data to which the model has been fitted (data, new); default data
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, nonlinear); default *
DISPERSION = *scalar*	Value of dispersion parameter in calculation of s.e.s; default is as set in the MODEL statement
DMETHOD = *string token*	Basis of estimate of dispersion, if not fixed by DISPERSION option (deviance, Pearson); default is as set in the MODEL statement
NBINOMIAL = *scalar*	Supplies the total number of trials to be used for prediction with a binomial distribution (providing a value *n* greater than one allows predictions to be made of the number of "successes" out of *n*, whereas the value one predicts the proportion of successes); default 1
LSDLEVEL = *scalar*	Significance level (%) for least significant differences; default 5
SAVE = *identifier*	Regression save structure for the analysis from which the comparison contrasts are to be calculated

Parameters

FACTOR = *factors*	Factor whose levels are compared
CONTRASTS = *matrices*	Defines the comparisons to be estimated
ORDER = *scalars*	Number of comparisons to estimate; default is the number of rows of the CONTRASTS matrix
GROUPS = *factors* or *pointers*	Set if comparisons are to be made at different combinations of another factor or factors
ESTIMATES = *variates* or *pointers*	Saves the estimated contrasts in a variate if GROUPS is unset, or in a pointer to a set of tables
SE = *variates* or *pointers*	Saves standard errors of the contrasts in a variate if GROUPS is unset, or in a pointer to a set of tables
SED = *pointers*	Pointer to a set of symmetric matrices to save standard errors for differences between the contrasts estimated for different levels of the GROUPS factor(s)
LSD = *pointers*	Pointer to a set of symmetric matrices to save least significant differences for the contrasts estimated for different levels of

	the GROUPS factor(s)
DF = *variates*	Saves degrees of freedom for the contrasts
SS = *variates*	Saves sums of squares of the contrasts

†RCURVECOMMONNONLINEAR procedure

Refits a standard curve with common nonlinear parameters across groups to provide s.e.'s for linear parameters (R.W. Payne).

Options

PRINT = *string tokens*	Printed output from the analysis (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring); default mode, summ, esti
MAXCYCLE = *variate*	Maximum number of iterations; default 30
METHOD = *string token*	Algorithm for fitting nonlinear model (gaussnewton, newtonraphson, fletcherpowell); default newt
STEPLENGTHS = *scalar* or *variate*	Initial step lengths for the parameters
SAVE = *regression save structure*	Save structure from this analysis
INSAVE = *regression save structure*	Save structure for the curve fitted by FITCURVE, default takes the most recent regression analysis

No parameters

RCYCLE directive

Controls iterative fitting of generalized linear, generalized additive, and nonlinear models, and specifies parameters, bounds etc for nonlinear models.

Options

MAXCYCLE = *scalars*	Maximum number of iterations for Fisher-scoring algorithm (used in generalized linear models), back-fitting algorithm (used in additive models) and nonlinear algorithms; single setting implies the same limit for all; default 15, 15, 30
TOLERANCE = *scalar* or *variate*	Scalar or first unit of a variate defines the convergence criterion for the relative change in deviance and, if required, the second element of a variate defines the criterion for convergence to a zero deviance; default !(0.0001,1.0E-11)
FITTEDVALUES = *variate*	Initial fitted values for generalized linear model; default *
METHOD = *string token*	Algorithm for fitting nonlinear model (GaussNewton, NewtonRaphson, FletcherPowell); default Gaus, but Newt for scalar minimization
LINEARPARAMETERS = *scalars*	Scalars to hold current values of linear parameters used in nonlinear model, for reference within model calculations

Parameters

PARAMETER = *scalars*	Nonlinear parameters in the model
LOWER = *scalars*	Lower bound for each parameter
UPPER = *scalars*	Upper bound for each parameter
STEPLENGTH = *scalars*	Initial step length for each parameter
INITIAL = *scalars*	Initial value for each parameter

RDA procedure

Performs redundancy analysis (A.I. Glaser).

Options

PRINT = *string tokens*	What to print (variance, loadings, roots, evalues, evectors, speciesscores, sitescores, fitsitescores, correlations, fitcorrelations, weights); default vari, root
NROOTS = *scalar*	Number of eigenvalues and eigenvectors to include in output; default * takes all the non-zero eigenvalues
NORMALIZE = *string tokens*	Whether to normalize the Y, X and/or Z variates to have unit

	sums-of-squares before the analysis (`x`, `y`, `z`); default `x`, `z`
`SCALING` = *string token*	Scaling for species and site scores (`none`, `both`); default `none`
`TOLERANCE` = *scalar*	Tolerance for detecting non-zero eigenvalues; default 10^{-5}

Parameters

`Y` = *pointers*	Each pointer defines a set of response variates to be modelled
`X` = *pointers*	Explanatory variates or factors to use for for each pointer of y-variates
`Z` = *pointers*	Conditioning variates or factors to remove ("partial out") before the analysis
`LRV` = *LRVs*	LRV structure from each analysis, storing the eigenvectors, eigenvalues and total variance
`SPECIESSCORES` = *matrices*	Saves the "species scores" from each analysis
`SITESCORES` = *matrices*	Save the "site scores" from each analysis
`FITSITESCORES` = *matrices*	Save the fitted "site scores" from each analysis
`CORRELATIONS` = *matrices*	Saves the correlations between the site scores and the x-variates
`FITCORRELATIONS` = *matrices*	Saves the correlations between the fitted site scores and the x-variates
`WEIGHTS` = *matrices*	Save the weights of the x-variates in the formation of the site scores
`SAVE` = *pointers*	Save structure which provides information for use in `CRBIPLOT` and `CRTRIPLOT`

`RDESTIMATES` procedure

Plots one- or two-way tables of regression estimates (R.W. Payne).

Options

`GRAPHICS` = *string token*	Type of graph (`highresolution`, `lineprinter`); default `high`
`METHOD` = *string token*	What to plot (`estimates`, `lines`); default `esti`
`XFREPRESENTATION` = *string token*	How to label the x-axis (`levels`, `labels`); default `labels` uses the `XFACTOR` labels, if available
`PSE` = *string token*	What s.e. to plot to represent variation (`average`, `individual`); default `aver`
`SAVE` = *regression save structure*	Save structure of the analysis to display; default `*` shows the most recently fitted regression

Parameters

`XFACTOR` = *factors*	Factor providing the x-values for each plot
`GROUPS` = *factors*	Factor identifying the different sets of points from a two-way table of estimates
`XVARIATES` = *variates*	X-variates for regression coefficients or pointer
`NEWXLEVELS` = *variates*	Values to be used for `XFACTOR` instead of its existing levels
`TITLE` = *texts*	Title for the graph; default defines a title automatically
`YTITLE` = *texts*	Title for the y-axis; default `''`
`XTITLE` = *texts*	Title for the x-axis; default is to use the identifier of the `XFACTOR`

`RDISPLAY` directive

Displays the fit of a linear, generalized linear, generalized additive or nonlinear model.

Options

`PRINT` = *string tokens*	What to print (`model, deviance, summary, estimates, correlations, fittedvalues, accumulated, confidence`); default `mode,summ,esti`
`CHANNEL` = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
`DENOMINATOR` = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (`ss, ms`); default `ss`

NOMESSAGE = *string tokens* — Which warning messages to suppress (dispersion, leverage, residual, vertical, df, inflation); default *

FPROBABILITY = *string token* — Printing of probabilities for variance and deviance ratios (yes, no); default no

TPROBABILITY = *string token* — Printing of probabilities for t-statistics (yes, no); default no

SELECTION = *string tokens* — Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions

DISPERSION = *scalar* — Dispersion parameter to be used as estimate for variability in s.e.s; default is as set in the MODEL statement

RMETHOD = *string token* — Type of residuals to display (deviance, Pearson, simple); default is as set in the MODEL statement

DMETHOD = *string token* — Basis of estimate of dispersion, if not fixed by DISPERSION option (deviance, Pearson); default is as set in the MODEL statement

PROBABILITY = *scalar* — Probability level for confidence intervals for parameter estimates; default 0.95

DFDISPERSION = *scalar* — Allows you to specify the number of degrees of freedom for a dispersion parameter specified by the DISPERSION option; default is as set in the MODEL statement

SAVE = *identifier* — Specifies save structure of model to display; default * i.e. that from latest model fitted

No parameters

READ directive

Reads data from an input file, an unformatted file or a text.

Options

PRINT = *string tokens* — What to print (data, errors, summary); default erro, summ

CHANNEL = *identifier* — Channel number of file, or text structure from which to read data; default current file

SERIAL = *string token* — Whether structures are in serial order, i.e. all values of the first structure, then all of the second, and so on (yes, no); default no, i.e. values in parallel

SETNVALUES = *string token* — Whether to set number of values of vectors from the number of values read (yes, no); default no causes the number of values to be set only for structures whose lengths are not defined already (e.g. by declaration or by UNITS)

LAYOUT = *string token* — How values are presented (separated, fixedfield); default sepa

END = *text* — What string terminates data (* means there is no terminator); default ':'

SEQUENTIAL = *scalar* — To store the number of units read (negative if terminator is met); default *

ADD = *string token* — Whether to add values to existing values (yes, no); default no (available only in serial read)

MISSING = *text* — What character represents missing values; default '*'

SKIP = *scalar* — Number of characters (LAYOUT=fixe) or values (LAYOUT=sepa) to be skipped between units (* means skip to next record); default 0 (available only in parallel read)

BLANK = *string token* — Interpretation of blank fields with LAYOUT=fixe (missing,

zero, error); default miss

JUSTIFIED = *string tokens* — How values are to be assumed justified with LAYOUT=fixe (left, right); default righ

ERRORS = *scalar* — How many errors to allow in the data before reporting a fault rather than a warning, a negative setting, -*n*, causes reading of data to stop after the *n*th error; default 0

FORMAT = *variate* — Allows a format to be specified for situations where the layout varies for different units, option SKIP and parameters FIELDWIDTH and SKIP are then ignored (in the variate: 0 switches to fixed format; 0.1, 0.2, 0.3 or 0.4 to free format with space, comma, colon or semi-colon respectively as separators; * skips to the beginning of the next line; in fixed format, a positive integer *n* indicates an item in a field width of *n*, −*n* skips *n* characters; in free format, *n* indicates *n* items, −*n* skips *n* items); default *

QUIT = *scalar* — Channel number of file to return to after a fatal error; default * i.e. current input file

UNFORMATTED = *string token* — Whether file is unformatted (yes, no); default no

REWIND = *string token* — Whether to rewind the file before reading (yes, no); default no

SEPARATOR = *text* — Text containing the (single) character to be used in free format; default ' '

SETLEVELS = *string token* — Whether to define factor levels or labels (according to the setting of FREPRESENTATION) automatically from those that occur in the data (yes, no); default no causes them to be set only when they are not defined already

TRUNCATE = *string tokens* — Truncation of leading or trailing spaces of strings read in fixed format (leading, trailing); default * i.e. none

CASE = *string token* — Whether the case of letters (small and capital) should be regarded as significant or ignored when forming factor labels automatically (significant, ignored); default sign

LDIRECTION = *string token* — How to define the ordering of levels or labels when these are formed automatically (ascending, given); default asce

Parameters

STRUCTURE = *identifiers* — Structures into which to read the data

FIELDWIDTH = *scalars* — Field width from which to read values of each structure (LAYOUT=fixe only)

DECIMALS = *scalars* — Number of decimal places for numerical data containing no decimal points

SKIP = *scalars* — Number of values (LAYOUT=sepa) or characters (LAYOUT=fixe) to skip before reading a value

FREPRESENTATION = *string tokens* — How factor values are represented (labels, levels, ordinals); default leve

RECORD directive

Dumps a job so that it can later be restarted by a RESUME statement.

Option

CHANNEL = *scalar* — Channel number of the backing-store file where information is to be dumped; default 1

No parameters

REDUCE directive

Forms a reduced similarity matrix (referring to the GROUPS instead of the original units).

This directive was replaced in Release 14 by the directive HREDUCE (with exactly the same options and parameters). It is currently retained as a synonym of HREDUCE, but may be removed in a future release.

REFORMULATE directive

Modifies a formula or an expression to operate on a different set of data structures.

Options

OLDFORMULA = *formula* or *expression structure*
: Original formula or expression

NEWFORMULA = *formula* or *expression structure*
: New formula or expression, modified to operate on the new structures

Parameters

OLDSTRUCTURE = *identifiers*
: Data structures in the OLDFORMULA to be replaced in the NEWFORMULA

NEWSTRUCTURE = *identifiers*
: Identifier of the new data structure to replace each OLDSTRUCTURE

RELATE directive

Relates the observed values on a set of variates or factors to the results of a principal coordinates analysis.

This directive was replaced in Release 14 by the directive PCORELATE (with exactly the same options and parameters). It is currently retained as a synonym of PCORELATE, but may be removed in a future release.

REML directive

Fits a variance-components model by residual (or restricted) maximum likelihood.

Options

PRINT = *string tokens*
: What output to present (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default mode, comp, Wald, cova

PTERMS = *formula*
: Terms (fixed or random) for which effects or means are to be printed; default * implies all the fixed terms

PSE = *string token*
: Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default diff

WEIGHTS = *variate*
: Weights for the analysis; default * implies all weights 1

MVINCLUDE = *string tokens*
: Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default * i.e. omit units with missing values in either explanatory factors or variates or y-variates

SUBMODEL = *formula*
: Defines a submodel of the fixed model to be assessed against the full model (for METHOD=Fisher only)

RECYCLE = *string token*
: Whether to reuse the results from the estimation when printing or assessing a submodel (yes, no); default no

RMETHOD = *string token*
: Which random terms to use when calculating RESIDUALS (final, all, notspline); default fina

METHOD = *string token*
: Indicates whether to use the standard Fisher-scoring algorithm or the new AI algorithm with sparse matrix methods (Fisher, AI); default AI

MAXCYCLE = *scalar*
: Limit on the number of iterations; default 30

TOLERANCES = *variate*
: Tolerances for matrix inversion; default * i.e. appropriate default values

PARAMETERIZATION = *string token* — Parameterization to use for the variance component estimation (gammas, sigmas); default * i.e. use whichever is most appropriate for the model

CFORMAT = *string token* — Whether printed output for covariance models gives the variance matrices or the parameters (variancematrices, parameters); default vari

FMETHOD = *string token* — Controls whether and how to calculate F-statistics for fixed terms (automatic, none, algebraic, numerical); default auto

WORKSPACE = *scalar* — Number of blocks of internal memory to be allocated for use by the estimation algorithm when METHOD=AI; default 1

Parameters

Y = *variates* — Variates to be analysed

RESIDUALS = *variates* — Residuals from each analysis

FITTEDVALUES = *variates* — Fitted values from each analysis

EXIT = *scalar* — Exit status of the fit (0 if successful)

SAVE = *REML save structures* — Saves the details of each analysis for use in subsequent VDISPLAY and VKEEP directives

RENAME directive

Assigns new identifiers to data structures.

No options

Parameters

OLDIDENTIFIER = *identifiers* — Specifies the data structures to rename

NEWIDENTIFIER =*identifiers* — Specifies a new identifier for each data structure

REPPERIODOGRAM procedure

Gives periodogram-based analyses for replicated time series (R.P. Littlejohn).

Options

PRINT = *string token* — What to print (pair, randomization, glm); default * i.e. none

PLOT = *string token* — What graphs to plot (group, mean, logmean, cumulative, cv, pair); default mean, logm

TITLE = *text* — Title for each page of graphs

REPRESENTATION = *string token* — Form of data in SERIES (timeseries, meanperiodogram); default time

LENGTH = *scalar* or *variate* — Scalar specifying that the first N units of the series are to be used, or a variate specifying the first and last units of the series to be used

SEED = *scalar* — Seed for randomization; default 0

NRANDOMIZATIONS = *scalar* — Number of randomizations; default 99

TREATMENTS = *factor* — Contains ordered classification of SERIES

PAIR = *variates* — Treatment pair levels for pairwise comparisons

COLOUR = *text* or *variate* — Colours for each level of TREATMENTS; default * sets suitable colours automatically

MEANPERIODOGRAM = *pointer* — Saves mean periodograms according if REPRESENTATION=timeseries

REPLICATION = *scalar* or *variate* — Inputs or saves number of replicate series if REPRESENTATION=timeseries; scalar can be used for equal replication

Parameter

SERIES = *variates* — Specify the time series to be analysed

†RESHAPE procedure

Reshapes a data set with classifying factors for rows and columns, into a reorganized data set with new identifying factors (D.B. Baird).

Options

PRINT = *string token*	What to print (results); default *, i.e. none
ROWCLASSIFICATION = *factors*, *texts*, *variates* or *pointer*	Factors classifying the rows in the data; default a factor called Rows with a level for each row
COLCLASSIFICATION = *factors*, *texts*, *variates* or *pointer*	Factors or texts classifying the columns in the data; default a factor called Columns with labels formed from the column identifiers in DATA
MEANFACTORS = *factors*, *texts*, *variates* or *pointer*	Row or column factors whose groups are averaged in the output data set
TOTALFACTORS = *factors*, *texts*, *variates* or *pointer*	Row or column factors whose groups are totalled in the output data set
FIRSTSUMMARY = *string token*	Which summaries to form first (means, totals) default means
NEWROWFACTORS = factors	Factors to index the new rows
NEWCOLUMNFACTORS = *factors*, *texts* or *variates*	Factors to indexing the columns in the new data set
REDEFINE = *string token*	Whether to redefine the NEWROWFACTORS factors and DATA columns, if NEWROWFACTORS or NEWDATA are not set or use names used in the input data (yes, no); default no
MVINCLUDE = *string token*	Whether to include factor combinations with no observations in the output data set (*,rows, columns); default *; i.e. remove missing rows and columns

Parameters

DATA = *pointers*	Pointer containing data to be reshaped
NEWDATA = *pointers*	Pointer containing the reshaped data columns

RESTRICT directive

Defines a restricted set of units of vectors for subsequent statements.

No options

Parameters

VECTOR = *vectors*	Vectors to be restricted
CONDITION = *expression*	Logical expression defining the restriction for each vector; a zero (false) value indicates that the unit concerned is not in the set
SAVESET = *variates*	List of the units in each restricted set
NULL = *scalars*	Indicator for each restricted set, set to 1 or 0 according to whether or not it contains no units

RESUME directive

Restarts a recorded job.

Options

CHANNEL = *scalar*	Channel number of the backing-store file where the information was dumped; default 1
CLOSE = *string token*	Whether to close the file afterwards (yes, no); default no

No parameters

RETRIEVE directive

Retrieves structures from a subfile.

Options

CHANNEL = *scalar*	Specifies the channel number of the backing-store or procedure-library file containing the subfile (FILETYPE settings 'back' or 'proc'); default 0 (i.e. the workfile) for FILETYPE=back, no default for FILETYPE=proc, not relevant with other FILETYPE settings
SUBFILE = *identifier*	Identifier of the subfile; default SUBFILE
LIST = *string token*	How to interpret the list of structures (inclusive, exclusive, all); default incl
MERGE = *string token*	Whether to merge structures with those already in the job (yes, no); default no, i.e. a structure whose identifier is already in the job overwrites the existing one, unless it has a different type
FILETYPE = *string token*	Indicates the type of file from which the information is to be retrieved (backingstore, procedurelibrary, siteprocedurelibrary, Genstatprocedurelibrary); default back

Parameters

IDENTIFIER = *identifiers*	Identifiers to be used for the structures after they have been retrieved
STOREDIDENTIFIER = *identifiers*	Identifier under which each structure was stored

RETURN directive

Returns to a previous input stream (text vector or input channel).

Options

NTIMES = *scalar*	Number of streams to ascend; default 1
CLOSE = *string token*	Whether to close the channel (or text) after the return (yes, no); default no
DELETE = *string token*	Whether to delete the text or the file to which the channel was attached (only relevant if CLOSE=yes) after the return (yes, no); default no

Parameter

expression	Logical expression controlling whether or not to return to the previous input stream; default 1 (i.e. true)

†RFFAMOUNT procedure

Fits harmonic models to mean rainfall amounts for a Markov model (J.O. Ong'ala & D.B. Baird).

Options

PRINT = *string tokens*	Controls printed output for each fitted model (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence); default mode, summ, esti, accu
PLOT = *string token*	What plots to display (results); default resu
NHARMONICS = *scalar*	Defines the number of harmonics to fit (1...4); default 2
SPREADSHEET = *string tokens*	What to save in a spreadsheet (results); default *

Parameters

COUNTS = *table*	Supplies the table of counts by Markov class and day number within the year (1...366)
AMOUNTS = *tables*	Supplies the table of mean rainfall by wet Markov class and day
WINDOW = *scalars*	Window for the graph; default 3 for a single class and 1 otherwise
TITLE = *texts*	Title for the graph; default forms an automatic description
RESULTS = *pointers*	Saves a pointer to the variates of fitted rainfall means by day

	for each wet class
OUTFILE = *texts*	File(with extension .gwb, or .xlsx) to save the spreadsheet of results

†RFFPROBABILITY procedure

Fits harmonic models to rainfall probabilities for a Markov model (J.O. Ong'ala & D.B. Baird).

Options

PRINT = *string tokens*	Controls printed output for each fitted model (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence); default mode, summ, esti, accu
PLOT = *string token*	What plots to display (results); default resu
NHARMONICS = *scalar*	Defines the number of harmonics to fit (1...4); default 2
SPREADSHEET = *string tokens*	What to save in a spreadsheet (results); default *

Parameters

COUNTS = *table*	Supplies the table of counts by Markov class and day within the year (1...366)
WINDOW = *scalars*	Window to plot the graph; default 3 for a single class and 1 otherwise
TITLE = *texts*	The title for the plot; default forms an automatic description
RESULTS = *pointers*	Saves a pointer to variates of fitted rainfall probabilities by day for each wet state
OUTFILE = *texts*	File (with extension .gwb, or .xlsx) to save the selected spreadsheet components

RFINLAYWILKINSON procedure

Performs Finlay and Wilkinson's joint regression analysis of genotype-by-environment data (P.W. Lane & K. Ryder).

Options

PRINT = *string tokens*	What to print (model, summary, estimates, sortedsensitivities, monitoring); default mode, summ, esti, sort
PLOT = *string tokens*	What graphs to plot (lines, trellislines, sensitivities); default *
NBEST = *scalar*	Number of best genotypes to print in table of sorted sensitivities; default * i.e. print all of them
DIRECTION = *string token*	Direction to sort table of sorted sensitivities (ascending, descending); default asce
TOLERANCE = *scalar*	Convergence criterion; default 0.001
MAXCYCLE = *scalar*	Maximum number of cycles; default 15
SAVE = *regression save structure*	Save structure from MODEL statement defining the model; default is to use the structure from the latest MODEL statement

Parameters

GENOTYPES = *factors*	The genotype factor; no default
ENVIRONMENTS = *factors*	The environment factor; no default
SENSITIVITIES = *tables*	Saves the estimates of sensitivities; default *
GENMEANS = *tables*	Saves the estimates of genotype means; default *
ENVMEANS = *tables*	Saves the estimates of environment means; default *
ENVEFFECTS = *tables*	Saves the estimates of environment effects; default *
SESENSITIVITIES = *tables*	Saves the s.e.s of sensitivities; default *
SEGENMEANS = *tables*	Saves the s.e.s of genotype means; default *
SEENVEFFECTS = *tables*	Saves the s.e.s of environment effects; default *
MSDEVIATIONS = *tables*	Saves the mean square deviations about the line fitted to each genotype; default *
DEVIANCE = *scalar*	Saves the residual deviance
DF = *scalar*	Saves the residual d.f

TITLE = *text*	Overall title for the graphs
YTITLE = *text*	Y-axis title for the graph of the lines
XTITLE = *text*	X-axis title for the graph of the lines
EXIT = *scalar*	Exit status: set to 0 if the analysis converged, 1 otherwise

†RFSUMMARY procedure

Forms summaries for a Markov model from rainfall data (J.O. Ong'ala & D.B. Baird).

Options

PRINT = *string tokens*	Controls printed output (counts, amounts, probabilities); default *
PLOT = *string token*	What plots to display (probabilities); default prob
DAY = *variate* or *factor*	Day as a date or a day number within the year
LIMITS = *scalar* or *variate*	Values to define the daily rainfall states; default 0.85
ORDER = *scalar*	Defines the order of the Markov chain (0...5); default 1
HIGHORDER = *scalar*	Whether to use a high-order Markov chain; (no, yes); default no
INITIAL = *scalar* or *variate*	The amounts of rainfall prior to the first day; default *
SPREADSHEET = *string tokens*	What to save in a spreadsheet (counts, amounts, probabilities); default *

Parameters

DATA = *variates*	The daily rainfall amounts
WINDOW = *scalars*	Window to plot the graph; default 3 for ORDER=0 and 1 otherwise
TITLE = *texts*	The title for the plot; default uses an automatic description
COUNTS = *tables*	Saves the counts by Markov state and day
AMOUNTS = *tables*	Saves the mean rainfall by Markov wet states and day
PROBABILITIES = *pointers*	Saves a pointer to variates of probabilities of a wet day by class
CATEGORIES = *factors*	Saves the Markov class for each day
STATECOUNTS = *pointers*	Saves a pointer to tables of counts for each state
OUTFILE = *texts*	File (with extension .gwb, or .xlsx) to save selected spreadsheet components

RFUNCTION directive

Estimates functions of parameters of a linear, generalized linear, generalized additive or nonlinear model.

Options

PRINT = *string tokens*	What to print (estimates, se, correlations); default esti,se
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
CALCULATION = *expression structures*	Calculation of functions involving nonlinear and/or linear parameters; no default
SE = *variate*	To save approximate standard errors; default *
VCOVARIANCE = *symmetric matrix*	To save approximate variance-covariance matrix; default *
SAVE = *identifier*	Specifies save structure of regression model; default * i.e. that from last model fitted

Parameter

scalars	Identifiers of scalars assigned values of the functions by the calculations

RGRAPH procedure

Draws a graph to display the fit of a regression model (P.W. Lane).

Options

GRAPHICS = *string token*	Type of graphics to produce (lineprinter, highresolution); default high

TITLE = *text*	Title for the graph; default 'Fitted and observed relationship'
WINDOW = *number*	Which high-resolution graphics window to use; default 4 (redefined if necessary to fill the frame)
SCREEN = *string token*	Whether to clear the graphics screen before plotting (clear, keep); default clea
CIPLOT = *string token*	Whether to plot confidence intervals (no, yes); default no
CIPROBABILITY = *scalar*	Probability for confidence interval; default 0.95
BACKTRANSFORM = *string token*	What back-transformation to make (link, none, axis); default link
SAVE = *regression save structure*	Save structure of the model to display; default * uses the most recently fitted regression model

Parameters

INDEX = *variate*	Which explanatory variate to display; default * if GROUPS is set, otherwise INDEX is set to the first variate in the fitted model (must be set for nonlinear models other than standard curves)
GROUPS = *factor*	Which explanatory factor to display; default * if INDEX is set, otherwise GROUPS is set to the first factor in the fitted model (ignored for nonlinear models)

RIDGE procedure

Produces ridge regression and principal component regression analyses (A.J. Rook & M.S. Dhanoa).

Options

PRINT = *string token*	What to print (correlation, pcp, ridge); default corr
PLOT = *string token*	Graphical output required (ridgetrace); default *

Parameters

Y = *variates*	Response variate in regression model
X = *pointers*	Containing explanatory variates in regression model

RJOINT procedure

Does modified joint regression analysis for variety-by-environment data (P.W. Lane & K. Ryder).

Options

PRINT = *string tokens*	What to print (model, summary, estimates, monitoring, graph); default mode, summ, esti
TITLE = *text*	Overall title for graph
YTITLE = *text*	Y-axis title for graph
XTITLE = *text*	X-axis title for graph
TOLERANCE = *scalar*	Convergence criterion; default 0.001
MAXCYCLE = *scalar*	Maximum number of cycles; default 15
SAVE = *regression save structure*	Save structure from MODEL statement defining the model; default is to use the structure from the latest MODEL statement

Parameters

ENVIRONMENT = *factors*	The environment factor; no default
VARIETY = *factors*	The variety factor; no default
SENSITIVITIES = *variates*	To store estimates of sensitivities; default *
VARMEANS = *variates*	To store estimates of variety means; default *
ENVEFFECTS = *variates*	To store estimates of environment effects; default *
ENVMEANS = *variates*	To store estimates of environment means; default *
SESENSITIVITIES = *variates*	To store s.e.s of sensitivities; default *
SEVARMEANS = *variates*	To store s.e.s of variety means; default *
SEENVEFFECTS = *variates*	To store s.e.s of environment effects; default *
DEVIANCE = *scalar*	To store the residual deviance
DF = *scalar*	To store the residual d.f
EXIT = *scalar*	Exit status – set to 0 if the analysis converged, 1 otherwise

RKEEP directive

Stores results from a linear, generalized linear, generalized additive or nonlinear model.

Options

EXPAND = *string token*	Whether to put estimates in the order defined by the maximal model for linear or generalized linear models (yes, no); default no
DISPERSION = *scalar*	Dispersion parameter to be used as estimate for variability in s.e.s; default as set in the MODEL directive
RMETHOD = *string token*	Type of residuals to form if parameter RESIDUALS is set (deviance, Pearson, simple); default as set in MODEL
DMETHOD = *string token*	Basis of estimate of dispersion, if not fixed by DISPERSION option (deviance, Pearson); default as set in MODEL
PROBABILITY = *scalar*	Probability level for confidence limits; default 0.95
OMODEL = *pointer*	Pointer to settings of options of the current MODEL statement, given unit labels corresponding to the option names of MODEL (starting with 'distribution')
PMODEL = *pointer*	Pointer to settings of parameters of the current MODEL statement, given unit labels corresponding to the parameter names of MODEL (starting with 'y'), only refers to the first setting of Y, FITTEDVALUES and RESIDUAL
STATISTICS = *variates*	Saves all the statistics that could be displayed for the first Y variate by the 'summary' setting of the PRINT option of the fitting directives FIT, ADD etc
CIMETHOD = *string token*	Method to use to calculate confidence intervals for nonlinear models (exact, quadratic); default quad
IGNOREFAILURE = *string*	Whether to ignore failure to fit a generalized linear model (yes, no); default no
MAXIMALMODEL = *formula structure*	Saves the maximal model (as defined by TERMS)
FITMODEL = *formula structure*	Saves the currently-fitted model (including any contrast functions)
FITCONSTANT = *scalar*	Saves a scalar containing the value one if the constant is included in the fitted model, or zero otherwise
FITTYPE = *scalar*	Saves a scalar to indicate the type of model that has been fitted
SAVE = *identifier*	Specifies save structure of model; default * i.e. that from latest model fitted

Parameters

Y = *variates*	Response variates for which results are to be saved; default is the list of response variates in the most recent MODEL statement
RESIDUALS = *variates*	Residuals for each Y variate, as specified by the RMETHOD option
FITTEDVALUES = *variates*	Fitted values for each Y variate
LEVERAGES = *variate*	Leverages of the units for each Y variate
ESTIMATES = *variates*	Estimates of parameters for each Y variate
SE = *variates*	Standard errors of the estimates
INVERSE = *symmetric matrix*	Inverse matrix from a linear or generalized linear model, inverse of second derivative matrix from a nonlinear model
VCOVARIANCE = *symmetric matrix*	Variance-covariance matrix of the estimates
DEVIANCE = *scalars*	Residual ss or deviance
DF = *scalar*	Residual degrees of freedom
TERMS = *pointer* or *formula structure*	Fitted terms (excluding constant)
ITERATIVEWEIGHTS = *variate*	Iterative weights from a generalized linear model
LINEARPREDICTOR = *variate*	Linear predictor from a generalized linear model
YADJUSTED = *variate*	Adjusted response of a generalized linear model
EXIT = *scalar*	Exit status from a generalized linear or nonlinear model
GRADIENTS = *pointer*	Derivatives of fitted values with respect to parameters in a nonlinear model

GRID = *variate*	Grid of function or deviance values from a nonlinear model
DESIGNMATRIX = *matrix*	Design matrix whose columns are explanatory variates and dummy variates
PEARSONCHISQUARE = *scalar*	Pearson chi-square statistic from a generalized linear model
STERMS = *pointer*	Saves the identifiers of the variates that have been smoothed in the current model
SCOMPONENTS = *pointer*	Saves a pointer to variates holding the nonlinear components of the variates that have been smoothed
NOBSERVATIONS = *scalar*	Number of units used in regression, excluding missing data and zero weights and taking account of restrictions
SEFITTEDVALUES = *variate*	Saves standard errors of the fitted values
SELINEARPREDICTOR = *variate*	Saves standard errors of the linear predictor
INFLATION = *variate*	Saves the variance inflation factors of the parameter estimates
UPPER = *variates*	Saves upper confidence limits for the parameter estimates
LOWER = *variates*	Saves lower confidence limits for the parameter estimates
MEANDEVIANCE = *scalars*	Saves the residual mean deviance (or mean square)
TDEVIANCE = *scalars*	Saves the total deviance (or sum of squares)
TDF = *scalars*	Saves the total degrees of freedom (corrected for the mean or uncorrected as displayed by the fitting directives)
TMEANDEVIANCE = *scalars*	Saves the total mean deviance (or mean square)
SUMMARY = *pointer*	Saves the summary analysis-of-variance (or deviance) table as a pointer with a variate or text for each column (source, d.f. etc)
ACCUMULATED = *pointer*	Saves the accumulated analysis-of-variance (or deviance) table as a pointer with a variate or text for each column (source, d.f. etc)
STATISTICS = *variates*	Saves all the statistics that could be displayed for the Y variate by the 'summary' setting of the PRINT option of the fitting directives FIT, ADD etc

RKESTIMATES directive

Saves estimates and other information about terms in a regression analysis.

Options

FACTORIAL = *scalar*	Limit on number of factors and variates in a model term; default 3
Y = *variate*	Response variate for which results are to be saved; default is the last response variate in the save structure
SAVE = *identifier*	Provides the regression save structure for the analysis from which the estimates are to be saved; default * takes the save structure from the most recent regression

Parameters

TERMS = *formula*	Model terms for which information is required
ESTIMATES = *tables* or *scalars*	Table or scalar to store the estimated regression coefficients for each term
SE = *tables* or *scalars*	Table or scalar to store the standard errors of the estimated regression coefficients
VCOVARIANCE = *symmetric matrices*	Symmetric matrix or scalar to store the variances and covariances between the estimates of each term
DF = *scalars*	Number of degrees of freedom for each term
POSITIONS = *tables* or *scalars*	Positions of the estimates in the variate of estimates as saved from RKEEP when option EXPAND=yes

RLASSO procedure

Performs lasso using iteratively reweighted least-squares (D.A. Murray & P.H.C. Eilers).

Options

PRINT = *string token*	What output to print (correlation, crossvalidation,

	estimates, best); default best
PLOT = *string tokens*	What graphs to plot (correlation, coefficients); default * i.e. none
TERMS = *formula*	Explanatory model
FACTORIAL = *scalar*	Limit on number of factors/covariates in a model term; default 3
LAMBDA = *variate* or *scalar*	Values for the parameter lambda; must be set
VALIDATIONMETHOD = *string token*	Which cross-validation method to use (crossvalidation, gcv); default gcv
NCROSSVALIDATIONGROUPS = *scalar*	Number of groups for k-fold cross-validation; default 10
NBOOT = *scalar*	Number of times to bootstrap data to estimate standard errors and confidence limits for fitted values; default 100
SEED = *scalar*	Seed for random numbers to use in cross-validation and then in bootstrapping; default 0
CIPROBABILITY = *scalar*	Probability level for confidence interval for fitted values; default 0.95
MAXCYCLE = *scalar*	Maximum number of iterations for the iterative process
TOLERANCE = *variate*	Contains two values to define the convergence criterion for iterative least-squares and the adjustment to avoid division by zero in the penalty term; default !(0.0001,1e-08)

Parameters

Y = *variates*	Response variate
BESTLAMBDA = *scalars*	Saves the optimal lambda value from cross-validation
CVSTATISTICS = *matrices*	Saves the cross-validation statistics
RESIDUALS = *variates*	Saves residuals for the optimal LAMBDA
FITTEDVALUES = *variates*	Saves fitted values for the optimal LAMBDA
ESTIMATES = *variates*	Saves parameter estimates for the optimal LAMBDA
SE = *variates*	Saves standard errors of the parameter estimates for the optimal LAMBDA
SEFITTED = *variates*	Saves standard errors of the fitted values, from bootstrapping, for the optimal LAMBDA
LOWER = *variates*	Saves lower confidence limits for the fitted values, from bootstrapping, for the optimal LAMBDA
UPPER = *variates*	Saves upper confidence limits for the fitted values, from bootstrapping, for the optimal LAMBDA

RLFUNCTIONAL procedure

Fits a linear functional relationship model (M.S. Dhanoa & D.B. Baird).

Options

PRINT = *string token*	Controls printed output (summary, estimates, fittedvalues, confidencelimits, grouptests); default summ, esti, conf, grou
METHOD = *string tokens*	Specifies what methods to use to fit the regression (bartlett, majoraxis, errorsinvariables, yonx, xony, reducedmajoraxis, standardmajoraxis, rangedmajoraxis, geometricmean, bisector, medyonx, medxony, qgeometricmean, qbisector); default bart
PLOT = *string tokens*	Controls what to plot (fitted, residuals, bootestimates, confidencelimits); default fitt
TITLE = *text*	The title for the analysis; default title uses the Y and X identifiers
NBOOT = *scalar*	The number of samples to take for the bootstrap confidence limits; default 200
SEED = scalar	Seed for bootstrap randomization; default 0
CIPROBABILITY = *scalar*	Defines the size of the confidence interval; default 0.95 i.e. 95%

CIMETHOD = *string token* Method for confidence limits (parametric, bootstrap); default boot
GMETHOD = *string token* Method for comparing slopes, elevations and locations between groups (majoraxis, standardmajoraxis); default uses standardmajoraxis for METHOD settings standardmajoraxis, reducedmajoraxis, rangedmajoraxis, geometricmean or bisector, and majoraxis otherwise
VRATIO = *scalar* Ratio between variance of Y and X variables for METHOD=errorsinvariables; default 1
YRANGEMETHOD = *string token* Type of range used for Y when METHOD=rangedmajoraxis (relative, interval); default rela
XRANGEMETHOD = *string token* Type of range used for X when METHOD=rangedmajoraxis (relative, interval); default rela
WINDOW = *scalar* Graphics window to use for fitted-value plots; default 1
KEYWINDOW = *scalar* Graphics window to use for key; default 2

Parameters

Y = *variates* Y-variate for each model
X = *variates* X-variate for each model
SLOPE = *scalars*, *variates* or *matrices* Saves the estimated slopes
INTERCEPT = *scalars*, *variates* or *matrices* Saves the estimated intercepts
GROUPS = *factors* Defines groups of units
RESIDUALS = *variates*, *matrices* or *pointers* Saves the residuals from the fitted models
FITTEDVALUES = *variates*, *matrices* or *pointers* Saves the fitted values
ESTIMATES = *variates*, *matrices* or *pointers* Saves the estimates
SE = *variates*, *matrices* or *pointers* Saves the standard errors of the estimates
LOWER = *variates*, *matrices* or *pointers* Saves lower values of confidence intervals for the estimates
UPPER = *variates*, *matrices* or *pointers* Saves upper values of confidence intervals for the estimates
LOWFITTEDVALUES = *variates*, *matrices* or *pointers* Saves the lower confidence limits from a bootstrap analysis of fitted values
UPPFITTEDVALUES = *variates*, *matrices* or *pointers* Saves the upper confidence limits from a bootstrap analysis of fitted values
TESTPROBABILITIES = *pointers* Saves the between-group test probabilities (in a symmetric matrix) for differences in slopes, elevations and locations

RLIFETABLE procedure

Calculates the life-table estimate of the survivor function (D.A.Murray).

Options

PRINT = *string tokens* Controls printed output (lifetable); default life
PLOT = *string tokens* Type of graph to be plotted (survivor, hazard, pdf); default surv, haza, pdf
INTERVAL = *scalar* or *variate* A scalar defining the width of the intervals or a variate containing the boundaries of the intervals

Parameters

TIMES = *variates* Observed timepoints
CENSORED = *variates* Variate specifying whether the corresponding element of each TIMES variate is censored (1) or represents failures (0)
FREQUENCY = *variates* Variate containing frequencies for the elements of TIMES; by default these are all assumed to be 1

GROUPS = *factors*	Factor specifying the different groups for which to estimate life tables
LIFETABLE = *pointers*	Pointer to variates to save the information from each life table

RMGLM procedure

Fits a model where different units follow different generalized linear models (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring); default mode, summ, esti
Y = *variate*	Response variate
TERMS = *formula*	Terms in the model
NBINOMIAL = *variate*	Binomial totals
DISPERSION = *scalar*	Dispersion parameter; default * for DIST=norm, gamm, inve or calc, and 1 for DIST=pois, bino, mult, nega, geom, expo or bern
WEIGHTS = *variate*	Prior weights; default 1
OFFSET = *variate*	Offset variate to be included in model; default * i.e. none
CONSTANT = *string token*	How to treat the constant (estimate, omit, ignore); default esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default 3
FULL = *string token*	Whether to assign all possible parameters to factors and interactions (no, yes); default no
DATASET = *factor*	Indicates which generalized linear model to apply to each unit; default defined from NVALUES
LINEARPREDICTOR = *variate*	Initial values for linear predictor
MAXCYCLE = *scalar*	Maximum number of iterations; default 30
MVINCLUDE = *string token*	Whether to include units with missing values in the explanatory factors and variates (explanatory); default * i.e. omit these
SAVE = *identifier*	To name the regression save structure; default *

Parameters

NVALUES = *scalars*	Number of units for each generalized linear model
DISTRIBUTION = *string tokens*	Error distributions (normal, poisson, binomial, gamma, inversenormal, multinomial, calculated, negativebinomial, geometric, exponential, bernoulli); default norm
LINK = *string tokens*	Link functions (canonical, identity, logarithm, logit, reciprocal, power, squareroot, probit, complementaryloglog, calculated, logratio); default cano (i.e. iden for DIST=norm or calc; loga for DIST=pois; logi for DIST=bino, bern or mult; reci for DIST=gamm or expo; powe for DIST=inve; logr for DIST=nega or geom)
EXPONENT = *scalars*	Exponent for power links

RMULTIVARIATE procedure

Performs multivariate linear regression with accumulated tests; synonym FITMULTIVARIATE (H. van der Voet).

Options

PRINT = *string tokens*	Controls printed output (model, summary, accumulated); default mode, summ, accu
RPRINT = *string tokens*	Controls printed output from the univariate regression analyses (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring); default *
FACTORIAL = *scalar*	Limit for expansion of model terms; default 3

NOMESSAGE = *string tokens* — Which warning messages to suppress when fitting the complete model – messages are always suppressed when fitting models for individual tests (aliasing, marginality); default *

RESULTS = *pointer* — To save results from accumulated and summary tests in a pointer containing terms, degrees of freedom of terms, Wilks' Lambda, Rao's F-statistic, degrees of freedom for numerator and denominator of Rao's F and P-value of Rao's F

Parameter

TERMS = *formula* — List of explanatory variates and factors, or model formula

RNEGBINOMIAL procedure

Fits a negative binomial generalized linear model estimating the aggregation parameter (R.M. Harbord & R.W. Payne).

Options

†PRINT = *string tokens* — Printed output from the analysis (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence, aggregation, loglikelihood); default mode, summ, esti, aggr

AGGREGATION = *scalar* — Saves the estimate of the aggregation parameter

_2LOGLIKELIHOOD = *scalar* — Saves the value of −2 × log-likelihood

CONSTANT = *string token* — How to treat the constant (estimate, omit); default esti

FACTORIAL = *scalar* — Limit on number of factors in a treatment term; default 3

†POOL = *string token* — Whether to pool the deviance for the terms in the accumulated summary (yes, no); default no

NOMESSAGE = *string tokens* — Warnings to suppress from FIT (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *

FPROBABILITY = *string token* — Printing of probabilities for variance ratios (yes, no); default no

TPROBABILITY = *string token* — Printing of probabilities for t-statistics (yes, no); default no

SELECTION = *string tokens* — Statistics to be displayed in the summary of analysis produced by PRINT=summary (%variance, %ss, adjustedr2, r2, dispersion, %meandeviance, %deviance, aic, bic, sic); default disp

†PROBABILITY = *scalar* — Probability level for confidence intervals for parameter estimates; default 0.95

SEAGGREGATION = *scalar* — Saves the standard error of the estimated aggregation parameter

MAXCYCLE = *variate* — Maximum number of iteration for main and Newton-Raphson estimations; default !(15,15)

TOLERANCE = *variate* — Convergence criteria for deviance and *k*; default !(1E-4,1E-4)

Parameter

TERMS = *formula* — List of explanatory variates and factors, or model formula (as for FIT)

RNONNEGATIVE procedure

Fits a generalized linear model with nonnegativity constraints; synonym FITNONNEGATIVE (P.W. Goedhart & C.J.F. ter Braak).

Options

PRINT = *string tokens* — Printed output required (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring); default mode, summ, esti

CONSTANT = *string token* — How to treat the constant (estimate, omit); default esti

POOL = *string token* — Whether to pool ss in accumulated summary between all terms

fitted in a linear model (`yes`, `no`); default `no`

`DENOMINATOR` = *string token* — Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (`ss`, `ms`); default `ss`

`NOMESSAGE` = *string tokens* — Which warning messages to suppress (`dispersion`, `leverage`, `residual`, `aliasing`, `marginality`); default `*`

`FPROBABILITY` = *string token* — Printing of probabilities for variance ratios (`yes`, `no`); default `no`

`TPROBABILITY` = *string token* — Printing of probabilities for t-statistics (`yes`, `no`); default `no`

`MAXCYCLE` = *scalar* — Maximum number of iterations; default 100

`TOLERANCE` = *scalar* — Value against which the Kuhn-Tucker values are tested; default 10^{-8}

`INITIALMODEL` = *string token* — Initial model from which to start the iterative procedure (`null`, `full`, `positive`, `own`); default `null`

`OWNINITIAL` = *variates* — Specifies the variates that compose your own initial model; this option must be set when `INITIALMODEL=own`; default `*`

`FORCED` = *formula* — Model formula which is fitted irrespective of nonnegativity constraints; default `*`

Parameter

`X` = *variates* — List of predictors which are subject to nonnegativity constraints

`ROBSSPM` procedure

Forms robust estimates of sum-of-squares-and-products matrices (P.G.N. Digby).

Options

`PRINT` = *string tokens* — Controls printed output (`sspm`, `distances`, `weights`, `vcovariance`, `means`, `correlations`, `outliers`); default `*` i.e. no output

`B1` = *scalar* — The value from which the threshold distance is derived (see the Method Section); default 2

`B2` = *scalar* — The value indicating the decline in weight as the distance of a unit above the threshold increases, (see the Method Section); default 1.25

`MAXCYCLE` = *scalar* — Maximum number of iterations; default 100

`TOLERANCE` = *scalar* — The minimum change in the average squared-weight that has to be achieved for the iterative process to converge; default 1.0^{-8}

Parameters

`DATA` = *pointers* — Supplies the set of variates in each datamatrix

`SSPM` = *SSPMs* — SSPM structure to contain the robust estimates of the sums of squares and products, the robust estimates of the means, and the sum of the weights for each datamatrix

`DISTANCES` = *variates* — To contain the Mahalanobis distances of the units from the mean

`WEIGHTS` = *variates* — To contain the weights used for each unit when forming the robust estimates

`VCOVARIANCE` = *symmetric matrices* — To contain the robust estimates of the matrices of variances and covariances

`CORRELATIONS` = *symmetric matrices* — This contains on output the correlations from the robust estimates of the variances and covariances

`ROTATE` directive

Does a Procrustes rotation of one configuration of points to fit another.

Options

`PRINT` = *string tokens* — Printed output required (`rotations`, `coordinates`, `residuals`, `sums`); default `*` i.e. no printing

SCALING = *string token*	Whether or not isotropic scaling is allowed (yes, no); default no
STANDARDIZE = *string tokens*	Whether to centre the configurations (at the origin), and/or to normalize them (to unit sum of squares) prior to rotation (centre, normalize); default cent,norm
SUPPRESSREFLECTION = *string token*	Whether to suppress reflection (yes, no); default no

Parameters

XINPUT = *matrices*	Inputs the fixed configuration
YINPUT = *matrices*	Inputs the configuration to be fitted
XOUTPUT = *matrices*	To store the (standardized) fixed configuration
YOUTPUT = *matrices*	To store the fitted configuration
ROTATION = *matrices*	To store the rotation matrix
RESIDUALS = *matrices* or *variates*	To store distances between the (standardized) fixed and fitted configurations
RSS = *scalars*	To store the residual sum of squares

RPAIR procedure

Gives t-tests for all pairwise differences of means from a regression or generalized linear model (J.T.N.M. Thissen & P.W. Goedhart).

Options

PRINT = *string tokens*	What to print (differences, sed, tvalues, tprobabilities); default diff, sed, tval
SORT = *string token*	Whether to sort the means into ascending order (no, yes); default no
COMBINATIONS = *string token*	Which combinations of factors in the current model to include (full, present, estimable); default esti (similar to the PREDICT directive)
ADJUSTMENT = *string token*	Type of adjustment with linear regression models (marginal, equal); default marg (similar to the PREDICT directive)
WEIGHTS = *table*	Weights classified by some or all standardizing factors; default * (similar to the PREDICT directive)
METHOD = *string token*	Method of forming margin (mean, total); default mean (similar to the PREDICT directive)
ALIASING = *string token*	How to deal with aliased parameters (fault, ignore); default faul (similar to the PREDICT directive)
SAVE = *identifier*	Specifies save structure of model to display; default * (i.e. that of the latest model fitted)

Parameters

TREATFACTORS = *pointers*	Each pointer contains a list of treatment factors classifying the table of means to be compared (the right-most factor changes fastest, then the second from the right, etc.); this parameter must be set
LABELS = *texts*	Structures containing strings to label rows (and columns) of the symmetric matrices of pairwise differences etc; the length of the text must equal the product of the numbers of factor levels as implied by the factor list in the TREATFACTORS pointer
NEWLABELS = *texts*	To save the row labels of the DIFFERENCES, SED, TVALUES and TPROBABILITIES matrices
DIFFERENCES = *symmetric matrices*	To save pairwise differences (treatment means on the diagonal)
SED = *symmetric matrices*	To save standard errors of the pairwise differences (missing values on the diagonal)
TVALUES = *symmetric matrices*	To save t-values (missing values on the diagonal)
TPROBABILITIES = *symmetric matrices*	To save t-probabilities (missing values on the diagonal)

RPARALLEL procedure

Carries out analysis of parallelism for nonlinear functions; synonym FITPARALLEL (R.C. Butler).

Options

PRINT = *string tokens*	What to print (model, summary, accumulated, estimates, correlations, fittedvalues, monitoring); default mode, summ, accu, esti
CALCULATION = *expression structures*	Calculation(s) involving explanatory variate; no default (must be set)
METHOD = *string token*	Which models to fit (singleline, constantsseparate, linearseparate, nonlinearseparate); default nonl
CONSTANT = *string token*	How to treat constant (estimate, omit); default esti

Parameters

X = *variates*	Explanatory variate; must be set
GROUPS = *factors*	Grouping factor for data; must be set
RESULTS = *pointers*	To save results from model nonlinearseparate, if fitted; should be set only if METHOD=nonl

RPERMTEST procedure

Does random permutation tests for regression or generalized linear model analyses (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (probability, accumulated, summary, critical); default prob
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalar*	Limit on the number of variates and/or factors in the terms to be fitted; default 3
NTIMES = *scalar*	Number of permutations to make; default 999
BLOCKSTRUCTURE = *formula*	Model formula defining any blocking to consider during the randomization; default none
EXCLUDE = *factors*	Factors in the block formula whose levels are not to be randomized
SEED = *scalar*	Seed for the random number generator used to make the permutations; default 0 continues from the previous generation or (if none) initializes the seed automatically
†SUMMARY = *pointer*	Saves the summary analysis-of-variance (or deviance) table with permutation probabilities and critical values
†ACCUMULATED = *pointer*	Saves the accumulated analysis-of-variance (or deviance) table with permutation probabilities and critical values
†BINMETHOD = *string token*	How to permute binomial data (individuals, units; default indi

Parameter

TERMS = *formula*	List of explanatory variates and factors, or model formula, defining the model to fit

RPHCHANGE procedure

Modifies a proportional hazards model fitted by RPHFIT (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, loglikelihood); default mode, summ, esti
METHOD = *string token*	How to change the model (add, drop, switch); default add
POOL = *string token*	Whether to pool terms in the accumulated summary generated by the fit

Parameter

TERMS = *formula*	Model specifying the change

RPHDISPLAY procedure

Prints output for a proportional hazards model fitted by RPHFIT (R.W. Payne).

Option

PRINT = *string tokens*	Controls printed output (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, loglikelihood); default mode, summ, esti

No parameters

RPHFIT procedure

Fits a proportional hazards model to survival data as a generalized linear model (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, loglikelihood); default mode, summ, esti
MAXIMALMODEL = *formula*	Defines the full model to explore (using RPHCHANGE); default uses the model defined by the TERMS parameter
SUBJECTS = *factor*	Subject corresponding to each observation
TIMES = *factor* or *variate*	Time of each observation
CENSORED = *variate*	Contains the value 1 for censored observations, otherwise 0; if unset it is assumed that there is no censoring
OFFSET = *variate*	Offset to include in the model
POOL = *string token*	Whether to pool terms in the accumulated summary generated by the fit

Parameter

TERMS = *formula*	Model to fit

RPHKEEP procedure

Saves information from a proportional hazards model fitted by RPHFIT (R.W. Payne).

Options

RESIDUALS = *variate*	Saves the standardized residuals
FITTEDVALUES = *variate*	Saves the fitted values
ESTIMATES = *variate*	Saves estimates of the parameters
SE = *variate*	Saves standard errors of the estimates
RESPONSE = *variate*	Saves the response variate defined for the generalized linear model
OFFSET = *variate*	Saves the offset variate defined for the generalized linear model
INDEX = *variate*	Index variate used to produce the expanded covariates and factors
RISKSET = *factor*	Saves the expanded time factor
_2LOGLIKELIHOOD = *scalar*	Saves −2 × log-likelihood for the fitted model

No parameters

RPHVECTORS procedure

Forms vectors for fitting a proportional hazards model as a generalized linear model (R.W. Payne).

Options

SUBJECTS = *factor*	Subject corresponding to each observation
TIMES = *factor* or *variate*	Time of each observation
CENSORED = *variate*	Contains the value 1 for censored observations, otherwise 0; if unset it is assumed that there is no censoring
RESPONSE = *variate*	Response variate for the generalized linear model
OFFSET = *variate*	Offset variate
INDEX = *variate*	Mapping variate used to produce the expanded variables
NEWSUBJECTS = *factor*	Expanded subjects factor

NEWTIMES = *factor* or *variate*	Expanded times factor
NEWOFFSET = *variate*	Offset variate for fitting the proportional hazards model

Parameters

X = *variates* or *factors*	Lists the x-variables that are to be expanded
NEWX = *variates* or *factors*	Identifiers to store the expanded x-variables; if no NEWX is specified, the expanded values overwrite the original values of X

RPOWER procedure

Calculates the power (probability of detection) for regression models (R.W. Payne).

Options

PRINT = *string token*	Prints the power (power); default powe
TERMS = *formula*	Specifies the terms (x-variates, factors or model terms) to be fitted in the analysis when the responses to be detected are specified by the RESPONSE parameter
FACTORIAL = *scalar*	Limit on the number of factors or variates in a model term generated from TERMS; default 3
PROBABILITY = *scalar*	Significance level at which the response is required to be detected (assuming a one-sided test); default 0.05
TMETHOD = *string token*	Type of test to be made (onesided, twosided, equivalence, noninferiority, fratio, chisquare); default ones
SAVE = *rsave*	Regression save structure to provide the information about the regression model

Parameters

RESPONSE = *variates*	Variate of fitted values calculated using regression parameters of the size to be detected; default * implies that the information is to be taken from a regression save structure
RDF = *scalars*	Number of residual degrees of freedom; if unset, this is obtained from the analysis of RESPONSE or from the regression save structure
RSS = *scalars*	Anticipated residual sum of squares; if unset, this is obtained from the analysis of RESPONSE or from the regression save structure
POWER = *scalars* or *variates*	Saves the power

RPROPORTIONAL procedure

Fits the Cox proportional hazards model to survival data (A.I. Glaser & R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (estimates, vcovariance, residuals, survivor, _2loglikelihood); default esti, _2lo
FACTORIAL = *scalar*	Sets a limit on the number of factors in the terms formed from the TERMS formula
TIMES = *factor* or *variate*	Time of each observation
CENSORED = *variate*	Contains the value 1 for censored observations, otherwise 0; if unset it is assumed that there is no censoring
OFFSET = *variate*	Offset to include in the model
BLOCKS = *factor*	Blocking factor defining groups of observations with different baseline hazard functions
INITIAL = *scalar* or *variate*	Initial values for the parameters in the model
RESIDUALS = *variate*	Saves the Cox-Snell residuals
ESTIMATES = *variate*	Saves the parameter estimates
SE = *variate*	Saves standard errors of the estimates
VCOVARIANCE = *symmetric matrix*	Saves the variance-covariance matrix of the estimates
_2LOGLIKELIHOOD = *scalar*	Saves −2 × log-likelihood for the fitted model

DFTERMS = *scalar* — Saves the number of d.f. in the model specified by TERMS
SURVIVOR = *variate* or *matrix* — Saves estimates of the survivor function, in a variate if BLOCKS is unset, otherwise in a matrix with a column for each block
EXIT = *scalar* — Exit code, set to zero if the fit was successful
MAXCYCLE = *scalar* — Maximum number of iterations to use; default 50
TOLERANCE = *scalar* — Defines the convergence criterion; default 0.000001

Parameter

TERMS = *formula* — Defines the model to fit

RQLINEAR procedure

Fits and plots quantile regressions for linear models (D.B. Baird).

Options

PRINT = *string tokens* — What to print (model, estimates, summary, fittedvalues, correlations, wald, jointqtest, separateqtest); default mode, esti, summ, wald
PLOT = *string tokens* — What to plot (rhistogram, phistograms, fittedvalues, estimates, bootestimates); default rhis, phis, fitt
TERMS = *formula* — Terms to be fitted
WEIGHTS = *variate* — Weights for data values; default equally weighted
CONSTANT = *string token* — Whether to include a constant in the model (omit, estimate); default esti
FACTORIAL = *scalar* — Limit on number of factors or variates in a term; default 3.
FITINDIVIDUALLY = *string token* — Whether to fit the regression model one term at a time (yes, no); default no
FULL = *string token* — Whether to assign all possible parameters to factors and interactions (yes, no); default no
BMETHOD = *string token* — Bootstrap method (xy, weightedxy); default xy
NBOOT = *scalar* — Number of times to bootstrap data to estimate confidence limits; default 200
SEED = *scalar* — Seed for bootstrap randomization; default 0
CIPROBABILITY = *scalar* — Probability level for confidence interval; default 0.95
XPLOT = *variate* — Variate to plot fitted values against; default 1st variate in model

Parameters

Y = *variates* — Response variate
PRQUANTILES = *scalars* or *variates* — Proportions at which to calculate quantiles; default 0.5
RESIDUALS = *variates* or *pointers* — Residuals from regression for each quantile
FITTEDVALUES = *variates* or *pointers* — Fitted values from regression for each quantile
ESTIMATES = *variates* or *pointers* — Estimated coefficients of model terms for each quantile
SE = *variates* or *pointers* — Standard errors of the estimated coefficients for each quantile
VCOVARIANCE = *symmetric matrices* or *pointers* — Variance-covariance matrix of estimates for each quantile
DF = *scalars* or *variates* — Numbers of degrees of freedom fitted by the model
LOWER = *variates* or *pointers* — Lower confidence limit of coefficients for each quantile
UPPER = *variates* or *pointers* — Upper confidence limit of coefficients for each quantile
LOWFITTEDVALUES = *variates* or *pointers* — Lower confidence limit of fitted values for each quantile
UPPFITTEDVALUES = *variates* or *pointers* — Upper confidence limit of fitted values for each quantile
OBJECTIVE = *scalars* or *variates* — Optimal values of the objective function
EXIT = *scalars* or *variates* — Exit codes indicating whether the estimation was successful

RQNONLINEAR procedure

Fits and plots quantile regressions for nonlinear models (D.B. Baird).

Options

PRINT = *string tokens* — What to print (model, estimates, summary,

	fittedvalues, correlations, monitoring); default mode, esti, summ
PLOT = *string tokens*	What to plot (rhistogram, phistograms, fittedvalues, confidencelimits); default phis, fitt, conf
X = *variates*	Variates to fit in the model
DATA = *variates or factors*	Data to bootstrap in parallel with Y; default takes the variates and factors of the same length as Y involved in the CALCULATION expressions
CONSTANT = *string token*	Whether to include a constant in the model (omit, estimate); default esti
CALCULATION = *expression structures*	Calculation of explanatory variates involving nonlinear parameters
PARAMETERS = *pointer*	Pointer to scalars representing the nonlinear parameters to be optimized in the expressions
INITIAL = *variate*	Initial values for parameters
LOWPARAMETERS = *variate*	Lower bound for parameters
UPPPARAMETERS = *variate*	Upper bound for parameters
STEPLENGTHS = *variate*	Step sizes for parameters
LINEARPARAMETERS = *pointer*	Pointer to scalars representing the linear parameters in the model (including the constant)
METHOD = *string token*	Which optimization method to use (gaussnewton, newtonraphson, fletcherpowell, simplex); default gaus
NBOOT = *scalar*	Number of times to bootstrap data to estimate confidence limits; default 100
SEED = *scalar*	Seed for bootstrap randomization; default 0
CIPROBABILITY = *scalar*	Probability level for confidence interval; default 0.95
MAXCYCLE = *scalar*	Maximum number of iterations for optimization; default 200
XPLOT = *variate*	Variate to plot fitted values against; default is the first variate on the right-hand side of the CALCULATION expressions

Parameters

Y = *variates*	Response variates
PRQUANTILE = *scalars*	Proportion at which to calculate the quantile for each response variate; default 0.5
RESIDUALS = *variates*	Residuals from the nonlinear model
FITTEDVALUES = *variates*	Fitted values from the nonlinear model
ESTIMATES = *variates*	Estimates of the parameters in the model (nonlinear, linear and constant)
SE = *variates*	Standard errors of the parameters
VCOVARIANCE = *symmetric matrices*	Variance-covariance matrix for the parameters
LOWER = *variates*	Lower confidence limits for the parameters
UPPER = *variates*	Upper confidence limits for the parameters
LOWFITTEDVALUES = *variates*	Lower confidence limits for the fitted values
UPPFITTEDVALUES = *variates*	Upper confidence limits for the fitted values
OBJECTIVE = *scalars*	Optimal values of the objective function
TITLE = *texts*	Titles for fitted value graphs

RQSMOOTH procedure

Fits and plots quantile regressions for loess or spline models (D.B. Baird).

Options

PRINT = *string tokens*	What to print (model, summary, fittedvalues); default mode, summ
PLOT = *string tokens*	What to plot (rhistogram, fittedvalues); default fitt
METHOD = *string token*	Smoothing method (loess, spline); default spli
DF = *scalar*	Spline Degrees of Freedom (3-40); default 4
KNOTS = *variate*	Knot points for smoothing splines; default * uses equally

	spaced percentiles of the X variate
KERNEL = *string token*	What Kernel to use for Loess (normal, epanechnikov, quadratic, triweight, tukeybiweight, quartic, linear, uniform); default norm
LMETHOD = *string token*	Span method for Loess (constant, adaptive); default adap
BANDWIDTH = *scalar*	Bandwidth for smoothing between 0 and 1; default 0.4
ORDER = *scalar*	Order of local polynomial; default 1
NGRIDPOINTS = *scalar*	Number of points on smooth curve; default 100
NBOOT = *scalar*	Number of times to bootstrap data to estimate confidence limits; default 0 i.e. no bootstrapping
SEED = *scalar*	Seed for bootstrap randomization; default 0
CIPROBABILITY = *scalar*	Probability level for confidence interval; default 0.95
TITLE = *text*	Title for plots; default * generates titles from the structure names
ARRANGEMENT = *string token*	Whether to plot fitted regressions by the GROUPS parameter in a trellis plot (single, trellis); default sing

Parameters

Y = *variates*	Response variate
X = *variates*	Explanatory variate
PRQUANTILES = *scalars* or *variates*	Proportions at which to calculate quantiles; default 0.5
GROUPS = *factors*	Groups for which independent curves are fitted
GRID = *variates*	Grid of equidistant points at which the smooth is calculated
OUTGROUPS = *factors*	Groups for the fitted smoothed values saved by the SMOOTH parameter
SMOOTH = *variates* or *pointers*	Fitted smooth estimated at the NGRIDPOINTS points given in GRID
SLOPE = *variates* or *pointers*	Fitted slope from model for the same points as SMOOTH
RESIDUALS = *variates* or *pointers*	Residuals from regression for each quantile
FITTEDVALUES = *variates* or *pointers*	Fitted values from regression for each quantile
LOWSMOOTH = *variates* or *pointers*	Lower confidence limit of smooth for each quantile
UPPSMOOTH = *variates* or *pointers*	Upper confidence limit of smooth for each quantile
SESMOOTH = *variates* or *pointers*	Standard error of coefficients for each quantile

RQUADRATIC procedure

Fits a quadratic surface and estimates its stationary point (R.W. Payne).

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence, stationary); default mode, summ, esti
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalars*	Limit for expansion of model terms; default 3
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-

	distributed response (`%variance`, `%ss`, `adjustedr2`, `r2`, `seobservations`, `dispersion`, `%cv`, `%meandeviance`, `%deviance`, `aic`, `bic`, `sic`); default `%var`, `seob` if `DIST=normal`, `%cv` if `DIST=gamma`, and `disp` for other distributions
`PROBABILITY` = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
`STATIONARY` = *scalars*	Saves the estimated value of y at the stationary point
`SESTATIONARY` = *scalars*	Saves the standard error of the estimated value of y at the stationary point
`TYPESTATIONARY` = *scalars*	Identifies the type of stationary point (2 for maximum, 1 for maximum on a ridge, −2 for minimum, −1 for minimum on a ridge, or 0 for saddle point)
`PREDICTIONS` = *matrix*	Saves predictions
`PLOT` = *string tokens*	What to plot (contour, surface); default * i.e. nothing
`COLOURS` = *text* or *variate*	Colours for the plots

Parameters

`X` = *variates*	X-variates whose linear, quadratic and product terms define the quadratic surface
`ESTIMATE` = *scalars*	Estimated value of each x-variate at the stationary point
`SE` = *scalars*	Standard error of the estimated value of each x-variate at the stationary point
`LEVELS` = *variates*	Values at which to evaluate each `X` for plots and predictions

`RRETRIEVE` procedure

Retrieves a regression save structure from an external file (R.W. Payne).

No options

Parameters

`FILENAME` = *texts*	Name of the file storing the save structure
`EXIT` = *scalars*	Scalar that contains the value one if the save structure could not be retrieved successfully, otherwise zero
`SAVE` = *regression save structures*	Save structure that has been retrieved

`RSCHNUTE` procedure

Fits a general 4 parameter growth model to a non-decreasing Y-variate; synonym `FITSCHNUTE` (A. Keen).

Options

`PRINT` = *string tokens*	What to print (`model`, `summary`, `estimates`, `correlations`, `fittedvalues`, `accumulated`, `monitoring`); default `mode`, `summ`, `esti`
`T1` = *scalar*	Timepoint defining y_1; default the first timepoint with $\mu > 0.4 \times y_2$ (μ and y_2 are obtained by an approximating model)
`T2` = *scalar*	Timepoint defining y_2; default * takes the last observed timepoint
`NGRID` = *scalar*	The number of points for a grid search with parameters *a* and/or *b*; default 7
`PLUS` = *scalar*	The constant added to the observed and fitted values, in order to obtain a suitable variance function in case of other than normal error distribution; default * takes the smallest possible value for the response given the rounding off
`A` = *scalar*	Fixed value for parameter *a* of the growth model, defining a submodel; only 0 is appropriate; default *
`B` = *scalar*	Fixed value for parameter *b* of the growth model; default *
`ALOWER` = *scalar*	Lower bound for parameter *a* of the growth model; default $-40/(t_2-t_1)$
`AUPPER` = *scalar*	Upper bound for parameter a of the growth model; default

	$40/(t_2-t_1)$
BLOWER = *scalar*	Lower bound for parameter *b* of the growth model; default −20
BUPPER = *scalar*	Upper bound for parameter *b* of the growth model; default 20
MAXCYCLE = *scalar*	Maximum number of iterations; default 20
TOLERANCE = *scalar*	Convergence criterion; default 0.0004
Parameters	
T = *variates*	Observed timepoints for each fit
MGRID = *matrices*	Deviances from the gridsearch in *a* and/or *b*
RT = *pointers*	Pointer of two variates: the fitted growth rates and relative growth rates at the observed timepoints
OWNT = *variates*	A variate of arbitrary timepoints to be specified by the user e.g. for obtaining a smooth plot of fitted values
ROWNT = *pointers*	Pointer of three variates: the fitted values, growth rates and relative growth rates at the timepoints specified in OWNT
EXTRA = *pointers*	Pointer of eight scalars, with: 1) the starting point of the curve below which the response equals 0, 2) the endpoint of the curve where the reponse is infinite, 3) the lower asymptote of the curve, 4) the upper asymptote of the curve, 5) the inflexion point, 6) the fitted value at the point of inflexion, 7) the growth rate at the point of inflexion, 8) the relative growth rate at the point of inflexion; if no finite value for a scalar exists, the value is set to be missing

RSCREEN procedure

Performs screening tests for generalized or multivariate linear models (H. van der Voet).

Options

PRINT = *string tokens*	Printed output required (model, pool, starscheme, tests, pvalues); default mode, pool, star
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default 3
NOMESSAGE = *string tokens*	Which warning messages to suppress when fitting the complete model (aliasing, marginality): warning messages are always suppressed when fitting models for individual tests; default *
EXCLUDEHIGHER = *string token*	Whether to exclude higher-order interactions in the conditional regression model for each tested term (yes, no); default no
FORCED = *formula*	Terms always included in the model (no tests on these terms); default *
TESTED = *text*	To save the names of individual terms which are tested
NELEMENTS = *variate*	To save the number of identifiers composing each individual term
MARGINAL = *pointer*	To save results from marginal tests for each tested term in a pointer containing the test statistic, corresponding degrees of freedom and the calculated probability
CONDITIONAL = *pointer*	To save results from conditional tests for each tested term in a pointer containing the test statistic, corresponding degrees of freedom and the calculated probability
MVINCLUDE = *string token*	Whether to include units with missing values in non-relevant explanatory variates or factors when calculating conditional and marginal tests (yes, no); default no

Parameter

FREE = *formula*	List of explanatory variates and factors, or model formula; each term from the expanded FREE formula is tested in a marginal and in a conditional test, unless the term is also part of the FORCED formula

RSEARCH procedure

Helps search through models for a regression or generalized linear model (P.W. Goedhart).

Options

PRINT = *string token*	Printed output required (model, results); default mode, resu
METHOD = *string tokens*	Model selection method to employ (allpossible, forward, backward, fstepwise, bstepwise, accumulated, pooled); default allp
FORCED = *formula*	Model formula to include in every model; default *
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalar*	Limit for expansion of all model terms; default 3
DENOMINATOR = *string token*	Whether to base ratios in accumulated summaries on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
INRATIO = *scalar*	Criterion for inclusion of terms for forward selection, backward elimination and stepwise regression; default 1.0
OUTRATIO = *scalar*	Criterion for exclusion of terms for forward selection, backward elimination and stepwise regression; default 1.0
MAXCYCLE = *scalar*	Limit on number of times to repeat stepwise selection methods, unless no change is made; default 50
CRITERION = *string token*	Criterion for selecting best models among all possible models (r2, adjusted, cp, ep, aic, bic, sic, meandeviance, deviance); default adju
EXTRA = *string token*	Criterion which is also printed for the selected best models (r2, adjusted, cp, ep, aic, bic, sic, meandeviance, deviance); default cp when DISPERSION=*, and mean otherwise
AFACTORIAL = *scalar*	Limit for expansion of FREE model terms for the fitting of all possible models; default 3
PENALTY = *scalar*	Penalty for Mallows Cp and Akaike's information criterion AIC; default 2
NTERMS = *scalar*	Limit on the number of terms to be fitted when fitting all possible models; default 16
NBESTMODELS = *scalar*	Number of best models printed for each subset size; default 8
PPROBABILITY = *scalar*	When METHOD=allpossible, only models with all probabilities less than PPROBABILITY are printed; default 1 i.e. all models are printed
FINALMODELS = *pointer*	Pointer to save the final models for forward, backward, fstepwise and bstepwise regression methods
ALLMODELS = *pointer*	Pointer to save formulae for all possible regression models containing the fitted terms of all the models; every formula includes the FORCED formula if set
ESTIMATES = *pointer*	Pointer to save variates for all possible regression models containing the parameter estimates
SE = *pointer*	Pointer to save variates for all possible regression models containing standard errors of the parameter estimates
RESULTS = *pointer*	Pointer to save variates for all possible regression models containing the criteria (r2, adjusted, cp, ep, aic, sic or bic, deviance, meandeviance), degrees of freedom for residual and the total number of fitted parameters *p*
STATISTICS = *pointer*	Pointer to save variates for all possible regression models containing the test statistics. These are F-to-delete statistics (i.e. deviance ratios) when the DISPERSION option of the MODEL directive is set to *, and Chi-square-to-delete statistics (i.e. deviance differences scaled by the dispersion parameter)

for a fixed dispersion parameter

DF = *pointer*	Pointer to save variates for all possible regression models containing the degrees of freedom for the numerator of the test statistics
PROBABILITIES = *pointer*	Pointer to save variates for all possible regression models containing the probabilities of the test statistics
MARGINALTERMS = *string token*	How to treat terms that are marginal to other terms in the FREE formula (forced, free); default forc

Parameter

FREE = *formula*	Model formula specifying the candidate model terms

RSPREADSHEET procedure

Puts results from a regression, generalized linear or nonlinear model into a spreadsheet (R.W. Payne).

Options

DISPERSION = *scalar*	Dispersion parameter to be used as estimate for variability in s.e.s; default as set in MODEL
RMETHOD = *string token*	Type of residual to use (deviance, Pearson, simple, deletion); default * i.e. as set in MODEL
DMETHOD = *string token*	basis of estimate of dispersion, if not fixed by DISPERSION option (deviance, Pearson); default * i.e. as set in MODEL
SPREADSHEET = *string tokens*	Which spreadsheets to form (summary, estimates, fittedvalues, accumulated); default summary, estimates, fittedvalues
SPESTIMATES = *string tokens*	What to include in the estimates spreadsheet (estimates, se, testimates, prestimates); default esti, se, test, pres
SPFITTEDVALUES = *string tokens*	What to include in the fitted-values spreadsheet (y, fittedvalues, residuals, leverages, sefittedvalues); default y, fitt, resi, leve
SAVE = *regression save structure*	Specifies which analysis to save; default * i.e. most recent regression

Parameters

Y = *variates*	Y-variate of the analysis to be saved
RESIDUALS = *variates*	Identifier of variate to save the residuals from each analysis; default residuals
FITTEDVALUES = *variates*	Identifier of variate to save the fitted values from each analysis; default fittedvalues
LEVERAGES = *variates*	Identifier of variate to save the leverages from each analysis; default leverages
ESTIMATES = *variates*	Identifier of variate to save the estimates from each analysis; default estimates
SE = *variates*	Identifier of variate to save s.e.'s of the estimates from each analysis; default se
TESTIMATES = *variates*	Identifier of variate to save the t-statistics of the estimates from each analysis; default t_statistics
PRESTIMATES = *variates*	Identifier of variate to save the t-probabilities of the estimates from each analysis; default t_probabilities
SEFITTEDVALUES = *variates*	Identifier of variate to save s.e.'s of the fitted values from each analysis; default sefittedvalues
SUMMARY = *pointers*	Identifier of pointer to save the summary analysis-of-variance (or deviance) from each analysis; default summary
ACCUMULATED = *pointers*	Identifier of pointer to save the accumulated analysis-of-variance (or deviance) from each analysis; default accumulated
OUTFILENAME = *texts*	Name of Genstat workbook file (.gwb) or Excel (.xls or .xlsx) file to create

RSTEST procedure

Compares groups of right-censored survival data by nonparametric tests (D.A. Murray).

Options

PRINT = *string token*	Controls printed output (test); default test
METHOD = *string tokens*	Types of test required (logrank, breslow, petoprentice, taroneware); default logr, bres, peto, taro
BLOCKS = *factor*	Factor specifying groupings for a stratified test; default * i.e. none

Parameters

TIMES = *variates*	Observed timepoints
CENSORED = *variates*	Variate specifying whether the corresponding element of TIMES is censored (1) or not (0)
GROUPS = *factors*	Factor specifying the different groups
TESTS = *pointers*	Pointer to variates (length 3) to save test statistic, d.f. and probability value for each chosen method

RSTORE procedure

Stores a regression save structure in an external file (R.W. Payne).

No options

Parameters

FILENAME = *texts*	Name of the file to store the save structure
EXIT = *scalars*	Scalar that contains the value one if the save structure could not be stored successfully, otherwise zero
SAVE = *regression save structures*	Save structure to be stored; default stores the save structure from the most recent regression analysis

RSURVIVAL procedure

Models survival times of exponential, Weibull, extreme-value, log-logistic or lognormal distributions (R.W. Payne & D.A. Murray).

Options

PRINT = *string tokens*	Controls printed output (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, loglikelihood); default mode, summ, esti
TIMES = *variate*	Time of each observation
DISTRIBUTION = *string token*	Distribution of the survival times (exponential, weibull, extremevalue, loglogistic, lognormal); default expo
CENSORED = *variate*	Indicator for censored observations: 0 if uncensored, 1 if right censored (subject survived the whole trial), −1 if left censored (log-logistic distribution only); default assumes no censored observations
GRAPHICS = *string token*	Controls the plotting of diagnostic graphs of the empirical survivor function against the estimate produced by the model (lineprinter, highresolution) default * i.e. none
ALPHA = *scalar*	Saves the estimated value of the parameter α of the Weibull and extreme-value distributions, if the scalar is input with a non-missing value this provides the initial estimate for α (which will also be the final estimate if MAXCYCLE=1)
_2LOGLIKELIHOOD = *scalar*	Saves −2 multiplied by the log-likelihood
SIGMA = *scalar*	Saves the estimated value of the shape parameter sigma of the log-logistic and lognormal distributions
SURVIVOR = *variate*	Saves estimates of the survivor function
PARAMETERIZATION = *string token*	Controls the parameterization used when saving the survivor function for the Weibull distribution (ph, aft); default ph
MAXCYCLE = *scalar*	Maximum number of iterations to use to estimate α; default 20
TOLERANCE = *scalar*	Convergence limit for α; default 10^{-5}

Parameter

TERMS = *formula*	Defines the model to fit

RTCOMPARISONS procedure

Calculates comparison contrasts within a multi-way table of means (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (contrasts); default cont
COMBINATIONS = *string token*	Factor combinations for which to form the predicted means (full, present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment to be made when forming the predicted means (marginal, equal, observed); default marg
WEIGHTS = *table*	Weights classified by some or all of the factors in the model; default *
OFFSET = *scalar*	Value of offset on which to base predictions; default mean of offset variate
METHOD = *string token*	Method of forming margin (mean, total); default mean
ALIASING = *string token*	How to deal with aliased parameters (fault, ignore); default faul
BACKTRANSFORM = *string token*	What back-transformation to apply to the values on the linear scale, before calculating the predicted means (link, none); default link
SCOPE = *string token*	Controls whether the variance of predictions is calculated on the basis of forecasting new observations rather than summarizing the data to which the model has been fitted (data, new); default data
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, nonlinear); default *
DISPERSION = *scalar*	Value of dispersion parameter in calculation of s.e.s; default is as set in the MODEL statement
DMETHOD = *string token*	Basis of estimate of dispersion, if not fixed by DISPERSION option (deviance, Pearson); default is as set in the MODEL statement
NBINOMIAL = *scalar*	Supplies the total number of trials to be used for prediction with a binomial distribution (providing a value *n* greater than one allows predictions to be made of the number of "successes" out of *n*, whereas the value one predicts the proportion of successes); default 1
SAVE = *identifier*	Regression or ANOVA save structure for the analysis from which the comparisons are to be calculated

Parameters

CONTRAST = *tables*	Defines the comparisons to be estimated
ESTIMATES = *scalars*	Saves the estimated contrasts
SE = *scalars*	Saves standard errors of the contrasts

RUGPLOT procedure

Draws "rugplots" to display the distribution of one or more samples (P.W. Lane).

Options

GRAPHICS = *string token*	What type of graphics to use (highresolution, lineprinter); default high
TITLE = *text*	Title for diagram; default *
AXISTITLE = *text*	Title for axis; default *
WINDOW = *scalar*	Window in which to draw high-resolution plot; default *, taken as 11 if SCREEN=clear, or 1 if SCREEN=keep
SCREEN = *string token*	Whether to clear screen before high-resolution plot (clear, keep); default clea
ORIENTATION = *string token*	Orientation of plots (down, across); default down

JITTER = *number*	Ratio of jitter width to range of data in high-resolution plot; default 0.01
SEED = *number*	Seed for generating random numbers used in jittering; default 0, i.e. continue from last generation, or initialize from system clock

Parameters

DATA = *variates*	Data to be summarized; no default
GROUPS = *factor*	Factor to divide values of a single variate into groups; default *
RUGLABELS = *texts*	Labels for individual rugs; default *, i.e. identifiers of variates or labels or levels of factor
POSITION = *scalar* or *variate*	Position on *x*-axis (or on *y*-axis if ORIENTATION=across) at which to plot each rug; if GROUPS is set, positions for each level of the factor are taken from a variate; default is to draw a single rug on the axis, and to spread multiple rugs across the window

RUNTEST procedure

Performs a test of randomness of a sequence of observations (P.W. Goedhart).

Options

PRINT = *string token*	Controls printed output (results); default resu
NULL = *scalar*	Defines the boundary between the two types; default 0

Parameters

DATA = *variates*	Sequences of observations
SAVE = *pointers*	To save the number of runs, the number of positive and negative observations and the lower and upper tail probabilities of the test

RWALD procedure

Calculates Wald and F tests for dropping terms from a regression (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (waldtests); default wald
FACTORIAL = *scalar*	Limit on number of factors in the model terms generated from the TERMS parameter; default 3
Y = *variate*	Y-variate from whose analysis to calculate the statistics; default is the last y-variate in SAVE
RDF = *scalar*	Saves the residual d.f. used to calculate F probabilities when the dispersion is not fixed
SAVE = *regression save structure*	Specifies the save structure (from MODEL) containing the analysis for which to calculate the tests; default is the save structure from the most recent regression

Parameters

TERMS = *formula*	Model terms for which tests are required
WALDSTATISTIC = *scalar* or *pointer* to *scalars*	Saves Wald statistics
DF = *scalar* or *pointer* to *scalars*	Saves d.f. of Wald statistics
PROBABILITY = *scalar* or *pointer* to *scalars*	Saves the probabilities for the Wald statistics if the dispersion is fixed, or the corresponding F statistics if it is estimated

RXGENSTAT procedure

Submits a set of commands externally to R and reads the output (M.F. D'Antuono & D.A. Murray).

Options

PRINT = *string tokens*	Controls printed output (summary, output); default outp
RPATH = *text*	Path specifying the location of the R executable; by default Genstat searches for a version of R installed within

	C:\program files (x86)\R or C:\program files\R
REXE = *text*	Name of the R executable to run; default 'Rterm.exe'
RARGS = *text*	Command line arguments to be used with the R executable; default '--no-restore --no-save'
SCRIPT = *text*	A set of R commands to run within R
SFILE = *text*	A file containing a set of R commands to run within R
RGEN = *text*	Name of a file to save the full set of commands used within R
ROUT = *text*	Name of a file to save the output from R

Parameters

WORKDIRECTORY = *texts*	Working directory to use within R; default current Genstat working directory
IDATA = *pointers*	Pointer to data structures to export to R (the data are exported into the file specified by the IRDAFILE parameter)
IRDAFILE = *texts*	Name of an R data (rda) file to import into R
ISAVE = *texts*	Pointer to data structures to import from R (the data are imported from the file specified by the ORDAFILE parameter)
ORDAFILE = *text*	Name of an R data (rda) file used to export data from R

RYPARALLEL procedure

Fits the same regression model to several response variates, and collates the output (P. Brain, R.W. Payne & D.B. Baird).

Options

PRINT = *string tokens*	Controls printed output (model, summary); default * i.e. none
TERMS = *formula*	Defines the regression model to fit on each variate
WEIGHTS = *variate* or *symmetric matrix*	Weights for the regression; default 1
OFFSET = *variate*	Offset; default * i.e. none
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default 3
FULL = *string token*	Whether to assign all possible parameters to factors and interactions (yes, no); default no
POOL = *string token*	Whether to pool the information on each term in the analysis of variance (yes, no); default no
RMETHOD = *string token*	Type of residuals to form (deviance, Pearson, simple); default devi
SPREADSHEET = *string tokens*	What results to save in a book of spreadsheets (aov, residuals, fittedvalues, estimates, se, testimates, prestimates); default * i.e. none

Parameters

Y = *variates* or *pointers*	Y-values for each set of analyses
RESIDUALS = *matrices*	Saves residuals from each set of analyses
FITTEDVALUES = *matrices*	Saves fitted values from each set of analyses
ESTIMATES = *matrices*	Saves estimates from each set of analyses
SE = *matrices*	Saves s.e.'s of estimates
TESTIMATES = *matrices*	Saves t-statistics of estimates
PRESTIMATES = *matrices*	Saves t-probabilities of estimates
DF = *pointers*	Saves degrees of freedom for the model terms or variates in each analysis of variance
SS = *pointers* or *variates*	Saves sums of squares for the model terms in each analysis of variance
MS = *pointers* or *variates*	Saves mean squares for the model terms in each analysis of variance
RDF = *variates*	Saves degrees of freedom from the "residual" lines in each analysis of variance
RSS = *variates*	Saves sums of squares from the "residual" lines
RMS = *variates*	Saves mean squares from the "residual" lines
TDF = *variates*	Saves degrees of freedom from the "total" lines in each

	analysis of variance
TSS = *variates*	Saves sums of squares from the "total" lines
TMS = *variates*	Saves mean squares from the "total" lines
VR = *pointers* or *variates*	Saves variance ratios for the model terms in each analysis of variance
PRVR = *pointers* or *variates*	Saves probabilities of the variance ratios
OUTFILENAME = *texts*	Name of Genstat workbook file (.gwb) or Excel (.xls or .xlsx) file to create

R0INFLATED procedure

Fits zero-inflated regression models to count data with excess zeros (D.A. Murray).

Options

PRINT = *string token*	Controls printed output (model, summary, estimates, fittedvalues, monitoring); default mode, summ, esti
DISTRIBUTION = *string token*	Distribution of response variable (poisson, negativebinomial); default pois
METHOD = *string token*	Method used for model fitting (em, conditional); default em
CONSTANT = *string token*	How to treat constant for count state (estimate, omit); default esti
ZCONSTANT = *string token*	How to treat constant for zero-inflation state (estimate, omit); default esti
XTERMS = *formula*	List of explanatory variates and factors, or model formula for count state of model
ZTERMS = *formula*	List of explanatory variates and factors, or model formula for zero-inflation state of model
WEIGHTS = *variate*	Variate of weights for weighted zero-inflated regression (Lambert model only)
OFFSET = *variate*	Offset variate to be used in the model (Lambert model only)
MAXCYCLE = *scalar*	Maximum number of iterations for EM algorithm; default 100
TOLERANCE = *scalar* or *variate*	Convergence criteria for EM algorithm, k and in the generalized linear models; default !(1.E-4, 1.E-4, 1.E-4)

Parameters

Y = *variates*	Response variate
RESIDUALS = *variates*	Saves the simple residuals
FITTEDVALUES = *variates*	Saves the fitted values
ESTIMATES = *variates*	Saves the estimates of the parameters
SE = *variates*	Saves the standard errors of the estimates
RSAVE = *identifiers*	Saves the regression structure for the final generalized model fitted for the count model
ZSAVE = *identifiers*	Saves the regression structure for the final binomial regression fitted for the zero-inflation model

R0KEEP procedure

Saves information from a zero-inflated regression model for count data with excess zeros fitted by R0INFLATED (D.A. Murray).

Options

RESIDUALS = *variate*	Saves the simple residuals
FITTEDVALUES = *variate*	Saves the fitted values
ESTIMATE = *variate*	Saves the parameter estimates
SE = *variate*	Saves the standard errors of the parameter estimates
VCOVARIANCE = *symmetric matrix*	Saves the variance-covariance matrix of estimates for the ZIP and ZINB models
XFITTEDVALUES = *variate*	Saves the fitted values for the count model
XSEFITTEDVALUES = *variate*	Saves the standard errors of the fitted values for the fitted values of the count model

ZFITTEDVALUES = *variate* Saves the fitted values for the zero model
ZSEFITTEDVALUES = *variate* Saves the standard errors of the fitted values for the fitted values of the zero model
_2LOGLIKELIHOOD = *scalar* Saves −2 times the log-likelihood

No parameters

R2LINES procedure

Fits two-straight-line (broken-stick) models to data (A.W.A. Murray & J.T. Wood).

Options

PRINT = *string token* What to print (model, summary, estimates, fittedvalues, intercepts); default mode, summ, esti
PLOT = *string tokens* What to plot (breakpoint, lines, residuals); default * i.e. nothing
HORIZONTAL = *string token* Forces either the left- the or right-hand line to be horizontal (left, right); default * i.e. neither
CIPROBABILITY = *scalar* Sets the probability level of the confidence interval about the X value at the intersection; default 0.95
†NGRIDLINES = *scalar* Controls the number of points used in the initial search for the intersection of the lines; default 100
TERMS = *variates* Additional x-variates to include in the model; default none
†METHOD = *string token* Optimization method (gaussnewton, newtonraphson, fletcherpowell); default newt

Parameters

Y = *variates* Response variates to be modelled
X = *variates* Explanatory variable for each response variate
TITLE = texts Title to use on the graphs for each response variate
FITTEDVALUES = *variates* Saves fitted values
RESIDUALS = *variates* Saves standardized residuals
ESTIMATES = *variates* Saves estimates from each model (i.e. intersection coordinates and slopes of the fitted lines)
SE = *variates* Saves standard errors of the estimates
INTERCEPTS = *variates* Saves the intercepts
LOWER = *scalars* Saves the lower bound of the confidence interval about the x-value at the intersection
UPPER = *scalars* Saves the upper bound of the confidence interval about the x-value at the intersection
PARTIALLIKELIHOOD = *pointers* Saves the partial likelihood and grid values for partial likelihood plots

SAGRAPES procedure

Produces statistics and graphs for checking sensory panel performance (D.I. Hedderley).

Options

PRINT = *string tokens* Controls printed output (aovtables, graphs, summarystatistics, tables); default grap, tabl
TREATMENTS = *factor* Factor defining the different treatments that are being assessed
SESSIONS = *factor* Factor defining the sessions on which the assessments were done
ASSESSORS = *factor* Factor defining the individual assessors
SCALING = *string token* Equal scaling for x and y axes on Drift-Unreliability and Discrimination-Disagreement graphs (equal, nome); default none
DESCRIPTION = *text* Extra information to print on graphs

Parameter

DATA = *variates* Variate for each attribute, containing the recorded score

SAMPLE procedure

Samples from a set of units, possibly stratified by factors (P.W. Lane).

Options

SEED = *scalar*	Seed for the random number generator; default 0 i.e. continue from previous generation
NVALUES = *scalar*	Number of units from which a simple sample is to be taken; default * i.e. as defined by UNITS statement

Parameters

NSAMPLE = *scalars* or *tables*	Number of values in simple sample, or table of numbers of values at each combination of levels of its classifying factors; no default
SAMPLE = *identifiers*	Structure to store the result; no default

SBNTEST procedure

Calculates the sample size for binomial tests (R.W. Payne & D.A. Murray).

Options

PRINT = *string token*	What to print (replication, power); default repl, powe
PRMETHOD = *string token*	Method to be used to calculate the probabilities for the binomial test (angular, normalapproximation, exact); default norm
PROBABILITY = *scalar*	Significance level for the test; default 0.05
POWER = *scalar*	The required power (i.e. probability of detection) of the test; default 0.9
TMETHOD = *string token*	Type of test to be done (onesided, twosided); default ones
NULL = *scalar*	Probability under the null hypothesis for the one-sample test; default 0.5
RATIOREPLICATION = *scalar*	Ratio of replication sample2:sample1 (i.e. the size of sample 2 should be RATIOREPLICATION times the size of sample 1); default 1
REPLICATION = *variate*	Replication values for which to calculate and print or save the power; default * takes 11 replication values centred around the required number of replicates

Parameters

P1 = *scalars*	Probability to detect in sample 1
P2 = *scalars*	Probability to detect in sample 2
NREPLICATES = *scalars*	Saves the required number of replicates
VREPLICATION = *variates*	Numbers of replicates for which powers have been calculated
VPOWER = *variates*	Power (i.e. probability of detection) for the various numbers of replicates

SCALAR directive

Declares one or more scalar data structures.

Options

VALUE = *scalar*	Value for all the scalars; default is a missing value
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used to identify the scalars in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the scalars
VALUE = *scalars*	Value for each scalar
DECIMALS = *scalars*	Number of decimal places for printing
EXTRA = *texts*	Extra text associated with each identifier
MINIMUM = *scalars*	Minimum value for the contents of each structure
MAXIMUM = *scalars*	Maximum value for the contents of each structure

DREPRESENTATION = *scalars* or *texts*	Default format to use when the contents represents a date and time

SCORRELATION procedure

Calculates the sample size to detect specified correlations (R.W. Payne).

Options

PRINT = *string token*	What to print (replication, power); default repl, powe
PROBABILITY = *scalar*	Significance level at which the correlation or difference between correlations is to be tested; default 0.05
POWER = *scalar*	The required power (i.e. probability of detection) of the test; default 0.9
TMETHOD = *string token*	Whether to a one- or two-sided test is to be made (onesided, twosided); default ones
RATIOREPLICATION = *scalar*	Ratio of replication sample2:sample1 (i.e. the size of sample for group 2 should be RATIOREPLICATION times the size of sample for group 1); default 1
REPLICATION = *variate*	Replication values for which to calculate and print or save the power; default * takes 11 replication values centred around the required number of replicates

Parameters

COR1 = *scalars*	Anticipated correlation in group 1
COR2 = *scalars*	Anticipated correlation in group 2
NREPLICATES = *scalars*	Saves the required number of replicates
VREPLICATION = *variates*	Numbers of replicates for which powers have been calculated
VPOWER = *variates*	Power (i.e. probability of detection) for the various numbers of replicates

SDISCRIMINATE procedure

Selects the best set of variates to discriminate between groups (D.B. Baird, L.H. Schmitt & J.W. McNicol).

Options

PRINT = *string tokens*	Printed output from the analysis (summary, steps, validation, specificity, discrimination, monitoring); default summ, vali, spec, disc
PLOT = *string tokens*	What plots to produce (errorrate, steps, specificity, discriminant); default erro, steps, spec, disc
DDISCRIMINANT = *string tokens*	What to display on the discriminant plot (means, mlabels, scores, polygons, confidencecircle); default means, mlabels, scores, conf
METHOD = *string token*	The variable selection method to use (forward, backward); default forw
NSELECT = *scalar*	Number of variates to select; default 4
CRITERION = *string token*	Criterion to use to select variables (wilkslambda, crossvalidation, bootstrap, jackknife); default wilk
MODELCHOICE = *string token*	Which model to save (optimal, nselect); default opti
VALIDATIONMETHOD = *string token*	Validation method to use to calculate error rates (bootstrap, crossvalidation, jackknife, prediction); default cros
NSIMULATIONS = *variate*	Number of bootstraps or cross-validation sets to use for selection and for validation; default !(10,50)
NCROSSVALIDATIONGROUPS = *scalar*	Number of groups for cross-validation, default 10
SEED = *scalar*	Seed for random number generation; default 0
YROOT = *scalar*	Specifies the root for plotting on the y-axis
XROOT = *scalar*	Specifies the root for plotting on the x-axis

Parameters

DATA = *pointers*	Each pointer contains a set of variates that are available to be

	selected
GROUPS = *factors*	Define groupings for the units in each training set
FORCED = *pointers*	Variates that must be included in the model
SELECTED = *pointers*	Saves the variates in the final model
STEPS = *pointers*	Saves the criterion values for each step in the model selection
ERRORRATE = *scalars*	Saves the validation error rate for the final model
SPECIFICITY = *matrices*	Saves the specificity table for the final model
ALLOCATION = *factors*	Saves the groups allocated by the final model
LRV = *LRVs*	Saves the LRVs from the final discriminant analysis
SCORES= *matrices* or *pointers*	Save discriminant scores for unit from the final model

SEDLSI procedure

Calculates least significant intervals (M.C. Hannah).

Options

PRINT = *string tokens*	What to print (delta, lsi, fittedsed, discrepancy, maxdiscrepancy, %discrepancy); default delta, lsi, maxd
METHOD = *string token*	Selects the method for computing the deltas (leastsquares, max, maxpse); default leas
PLOT = *string tokens*	What to plot (sed, lsi); default sed, lsi
CHECKFIT = *string token*	Which pairwise contrasts to use in printed output or plots involving the fitted SEDs (specified, all); default spec
PROBABILITY = *scalar*	Significance level for the least significent intervals; default 0.05.
DF = *scalar*	Degrees of freedom for the t-distribution use in calculation of the least significent intervals; default * assumes an infinite number of degrees of freedom (i.e. a Normal rather than a t-distribution)
WINDOW = *scalar*	Window in which to plot the graphs
TITLE = *text*	Title for the graphs; default 'Estimates with LSIs by Treatment'
YTITLE = *text*	Title for the y-axis; default 'Estimates'

Parameters

ESTIMATES = *tables* or *variates*	Parameter estimates; if these are not supplied SEDLSI can calculate the parameters $\{\delta_i\}$ but not the LSIs
SED = *symmetric matrices*	Matrix containing standard errors of (pairwise) differences between estimates
VCOVARIANCE = *symmetric matrices*	Matrix containing variances and covariances of estimates
WEIGHTS = *symmetric matrices*	Weight (or importance) to be used for each pairwise difference; default is a matrix of ones (i.e. all pairwise differences of equal interest)
LABELS = *texts*	Text vector (e.g. treatment labels) for labelling output; default takes the labels of levels of the factor classifying an ESTIMATES table or (if ESTIMATES is a variate or unset) row labels from SED or VCOVARIANCE
DELTA = *variates*	Saves the estimated parameters $\{\delta_i\}$
LSI = *pointers*	Saves details of the least significant intervals
FITTEDSED = *symmetric matrices*	Saves the fitted SED matrices

SED2ESE procedure

Calculates effective standard errors that give good approximate standard errors of differences (R.W. Payne).

Option

PRINT = *string token*	Controls printed output (ese, discrepancy, maxdiscrepancy, %discrepancy, %accounted); default * i.e. none

Parameters

SED = *symmetric matrices*	Standard errors of differences to be approximated
ESE = *variates* or *tables*	Saves the effective standard errors
DISCREPANCY = *symmetric matrices*	Saves the discrepancies between the standard errors of differences and the approximate values calculated from the effective standard errors
%ACCOUNTED = *scalars*	Percentage of variation amongst the standard errors of differences accounted for by the approximate values calculated from the effective standard errors
TEMPLATE = *tables*	Table that can be duplicated to provide a table to store the effective standard errors

SET directive

Sets details of the "environment" of a Genstat job.

Options

INPRINT = *string tokens*	Printing of input as in PRINT option of INPUT (statements, macros, procedures, unchanged); default unch
OUTPRINT = *string tokens*	Additions to output as in PRINT option of OUTPUT (dots, page, unchanged); default unch
DIAGNOSTIC = *string tokens*	Defines the least serious class of Genstat diagnostic which should still be generated (messages, warnings, faults, extra, unchanged); default unch
ERRORS = *scalar*	Number of errors that a job may contain before it is abandoned (0 implies no limit); default is to leave unchanged
FAULT = *text*	Sets the Genstat fault indicator (for example, FAULT=* clears the last fault); default is to leave the indicator unchanged
PAUSE = *scalar*	Number of lines to output before pausing (interactive use only; 0 implies no pausing); default is no change
PROMPT = *text*	Characters to be printed for the input prompt; default is to leave unchanged
NEWLINE = *string token*	How to treat a new line ((significant, ignored);); default is no change
CASE = *string token*	Whether lower- and upper-case (small and capital) letters are to be regarded as identical in identifiers (significant, ignored); default is no change
FIELDWIDTH = *scalar*	Fieldwidth to be used as a default minimum by PRINT and other output commands
SIGNIFICANTFIGURES = *scalar*	Minimum number of significant figures to be supplied in the default formats determined by PRINT and other output commands
SEEDS = *pointer* or *scalar*	Defines the current default seeds to be used for random numbers in various parts of Genstat
RUN = *string token*	Whether or not the run is interactive (interactive, batch); by default the current setting is left unchanged
UNITS = *identifier*	To (re)set the current units structure; default is to leave unchanged
BLOCKSTRUCTURE = *identifier*	To (re)set the internal record of the most recent BLOCKSTRUCTURE statement; default is to leave unchanged
TREATMENTSTRUCTURE = *identifier*	To (re)set the internal record of the most recent TREATMENTSTRUCTURE statement; default is to leave unchanged
COVARIATE = *identifier*	To (re)set the internal record of the most recent COVARIATE statement; default is to leave unchanged
ASAVE = *identifier*	To (re)set the current ANOVA save structure; default is to leave unchanged
DSAVE = *identifier*	To (re)set the current save structure for the high-resolution

graphics environment; default is to leave unchanged

MSAVE = *identifier* — To (re)set the current save structure for multivariate analysis; default is to leave unchanged

RSAVE = *identifier* — To (re)set the current regression save structure; default is to leave unchanged

TSAVE = *identifier* — To (re)set the current time-series save structure; default is to leave unchanged

VSAVE = *identifier* — To (re)set the current REML save structure; default is to leave unchanged

VCOMPONENTS = *identifier* — To (re)set the current REML model definitions, as specified by VCOMPONENTS and VSTRUCTURE; default is to leave unchanged

WORDLENGTH = *string token* — Length of word (8 or 32 characters) to check in identifiers, directives, options, parameters and procedures (long, short); default * i.e. no change

CAPTIONS = *string tokens* — Controls which captions are displayed (minor, major, meta, unchanged); default unch

TYPESET = *string tokens* — Controls when typesetting commands within textual strings are used (output, graphics); if unset, the existing setting is left unchanged

CMETHOD = *string token* — Controls whether number settings for colour options and parameters are interpreted as RGB values or as numbers of standard colours (rgb, standard); if unset, the existing setting is left unchanged

DATASPACE = *scalar* or *variate* — Updates the current data space allocations; if unset, the existing allocations are left unchanged

WORKINGDIRECTORY = *text* — Sets the working directory; default is to leave this unchanged

ALGORITHMS = *string token* — Controls the use of enhanced computing algorithms (standard, mkl); if unset, the existing setting is left unchanged

ACTIONAFTERFAULT = *string token* — Controls what happens after a fault (continue, stop); if unset, the existing setting is left unchanged

UNSETDUMMY = *string token* — Controls what happens if you specify an unset dummy as the setting of an option or parameter that expects another type of data structure (fault, ignore, warn); if unset, the existing setting is left unchanged

LANGUAGE = *text* — Text with either one or two values to specify a preferred language for output and (optionally) a second choice in case the preferred language is unavailable

YEAR2DIGITBREAK = *scalar* — Controls how 2 digits can be used to specify years

†TIMEWITHSECONDS = *string token* — Controls whether seconds are included with the time12 and time24 date representations; (absent, present, unchanged); default unch

No parameters

SETALLOCATIONS directive

Runs through all ways of allocating a set of objects to subsets.

Options

NREQUIRED = *scalar* — Number of allocations that are required; default 1

UNIQUE = *string token* — Whether only unique allocations are to be formed, allowing the reordering of the subsets (yes, no); default no

NFOUND = *scalar* — Number of allocations that has been found

NPOSSIBLE = *scalar* — Saves the total of allocations that can be formed

GROUPS = *factor* or *pointer* — Saves the allocations, in a single factor if NREQUIRED = 1, otherwise in a pointer to NFOUND factors

UNITS = *variate* — Supplies numbers for the objects; if unset, the positive integers

1, 2 ... are used

START = *factor*	Previous allocation; if unset the allocations start as a partitioning of the objects in the ordering in the UNITS variate

Parameters

SETSIZE = *scalars*	Number of objects in each subset
ELEMENTS = *variates* or *pointers*	Saves the objects allocated to each subset, in a single variate if NREQUIRED = 1, otherwise in a pointer to NFOUND variates

SETCALCULATE directive

Performs Boolean set calculations on the contents of vectors or pointers.

Options

NULL = *scalar*	Returns either 1 or 0 according to whether or not the result is a null (i.e. empty) set
FREPRESENTATION = *string token*	How to represent factors in a calculation that contains only factors (levels, labels); default leve
TOLERANCE = *scalar*	Tolerance to use when comparing numerical values; default 10^{-6}
SUBSTITUTE = *string token*	Whether to substitute dummies within pointers in the expression (yes, no); default no
NOMESSAGE = *string tokens*	Which warning messages to suppress (novalues); default * i.e. none

Parameter

expression	Expression defining the calculation to be performed

SETDEVICE procedure

Opens a graphical file and specifies the device number on basis of its extension (M.P. Boer & J.T.N.M. Thissen).

No options

Parameters

FILENAME = *texts*	Name of the graphical file including one of the possible extensions .bmp, .emf, .eps, .gmf, .jpg, .jpeg, .pdf, .png, .tif or .tiff; must be set
NUMBER = *scalars*	Saves the device number corresponding to the graphical format specified by parameter FILENAME
ACTION = *string token*	How to create graphs for file types such as .emf, .jpg, .tif or .png (asynchronous, synchronous); default asyn

SETOPTION directive

Sets or modifies defaults of options of Genstat directives or procedures.

Option

DIRECTIVE = *string token*	Directive (or procedure) to be modified

Parameters

NAME = *string tokens*	Option names
DEFAULT = *identifiers*	New default values

SETPARAMETER directive

Sets or modifies defaults of parameters of Genstat directives or procedures.

Option

DIRECTIVE = *string token*	Directive (or procedure) to be modified

Parameters

NAME = *string tokens*	Parameter names
DEFAULT = *identifiers*	New default values

SETRELATE directive

Compares the distinct values contained in two data structures.

Options

FREPRESENTATION = *string token*	How to represent factors in a comparison between two factors (levels, labels, ordinals); default leve
LFACTORIAL = *scalar*	Limit on number of factors or variates in the terms formed from a LEFT formula; default * i.e. none
RFACTORIAL = *scalar*	Limit on number of factors or variates in the terms formed from a RIGHT formula; default * i.e. none
TOLERANCE = *scalar*	Tolerance to use when comparing numerical values; default 10^{-6}
SUBSTITUTE = *string token*	Whether to substitute dummies within LEFT or RIGHT pointers and formulae (yes, no); default no

Parameters

LEFT = *identifiers*	First structures in each comparison
RIGHT = *identifiers*	Second structures in each comparison
CONTAINS = *scalars*	Returns 1 or 0 according to whether or not LEFT contains RIGHT
EQUALS = *scalars*	Returns 1 or 0 according to whether or LEFT and RIGHT contain exactly the same distinct set of items
INCLUDEDIN = *scalars*	Returns 1 or 0 according to whether or not LEFT is included in RIGHT
DISTINCT = *scalars*	Returns 1 or 0 according to whether or not LEFT and RIGHT are distinct

SET2FORMULA directive

Forms a model formula using a set of structures supplied in a pointer.

Option

METHOD = *string token*	Relationship of the structures within the formula (combined, crossed, nested); default comb

Parameters

POINTER = *pointers*	Sets of structures to be used to form the formulae
FORMULA = *formula structures*	Formulae constructed from the sets

SHELLEXECUTE directive

Launch executables or open files in another application using their file extension, PC Windows only.

No options

Parameters

FILE = text	Name of the file to execute
STATUS = *scalar*	Indicates whether the execution of the file was successful (0) or not (1)
MESSAGE = *text*	Saves the error message associated with a failure to execute the file

SIGNTEST procedure

Performs a one or two sample sign test (E. Stephens & P.W. Goedhart).

Options

PRINT = *string token*	Whether to print the test statistic with the associated probability and sample size (test); default test
METHOD = *string token*	Type of test (twosided, greaterthan, lessthan); default twos
GROUPS = *factor*	Defines the groups for a two-sample test if only the Y1 parameter is specified
NULL = *scalar*	Median value or difference in medians under the null hypothesis; default 0

Parameters

`Y1` = *variates*	Data values for a one-sample sign test (neither `Y2` nor `GROUPS` specified), or for the first sample of a two-sample test (`Y2` also specified) or the values in both samples of a two-sample test (`GROUPS` specified but not `Y2`)
`Y2` = *variates*	Data values for the second sample of a two-sample test
`STATISTIC` = *scalars*	To save the sign test statistic
`NBINOMIAL` = *scalars*	To save the effective sample size
`PROBABILITY` = *scalars*	To save the probability level of the test

`SIMPLEX` procedure

Searches for the minimum of a function using the Nelder-Mead simplex algorithm (J.A. Nelder & W. van den Berg).

Options

`PRINT` = *string tokens*	Controls printed output (`results`, `monitoring`); default `resu`
`CALCULATION` = *expression structures*	Expressions to calculate the target function
`FUNCTIONVALUE` = *scalar*	Identifier of the scalar, calculated by `CALCULATION`, whose value is to be minimized
`DATA` = any type	Data to be used with procedure `_SIMPLEXFUNCTION`
`POINTS` = *pointer*	Saves the points of the final simplex
`FVALUES` = *pointer*	Saves the function values at the points
`MAXCYCLE` = *scalar*	Maximum number of iterations; default 500
`TOLERANCE` = *scalar*	Convergence criterion; when standard deviation of function values is lower than `TOLERANCE` convergence is assumed to be reached; default 1.E-9

Parameters

`PARAMETER` = *scalars*	Parameters to be estimated
`LOWERINITIAL` = *scalars*	Lower starting values for the parameters
`UPPERINITIAL` = *scalars*	Upper starting values for the parameters

`SKEWSYMMETRY` procedure

Provides an analysis of skew-symmetry for an asymmetric matrix (P.G.N. Digby).

Option

`PRINT` = *string tokens*	Printed output from the analysis (`roots`, `scores`); default `*` i.e. no output

Parameters

`DATA` = *matrices*	Asymmetric (square) matrices to be analysed
`ROOTS` = *diagonal matrices*	Stores the squared singular values from the analysis; the structure has one value for each plane fitted in the analysis (e.g. if the `DATA` matrix has 11 rows and columns, the `ROOTS` diagonal matrix will have 5 values)
`SCORES` = *matrices*	Stores the coordinates of the points from the analysis; each matrix has the same number of rows as the corresponding `DATA` matrix, and has 2 columns for each plane fitted in the analysis (e.g. if the `DATA` matrix has 11 rows and columns, the `SCORES` matrix will have 11 rows and 10 columns)

`SKIP` directive

Skips lines in input or output files.

Options

`CHANNEL` = *scalar*	Channel number of file; default current channel of the specified type
`FILETYPE` = *string token*	Type of the file concerned (`input`, `output`); default `inpu`
`STYLE` = *string token*	Style to use when skipping output (`plaintext`, `formatted`); default `*` uses the current style of the channel

Parameter

identifiers	How many lines to skip; for input files, a text means skip until the contents of the text have been found, further input is then taken from the following line

SLCONCORDANCE procedure

Calculates the sample size for Lin's concordance correlation coefficient (R.W. Payne).

Options

PRINT = *string token*	What to print (replication, power); default repl, powe
PROBABILITY = *scalar*	Significance level at which the non-reproducibility is to be tested; default 0.05
POWER = *scalar*	The required power (i.e. probability of detection) of the test; default 0.9
REPLICATION = *variate*	Replication values for which to calculate and print or save the power; default * takes 11 replication values centred around the required number of replicates

Parameters

CORRELATION = *scalars*	Correlation for two samples with the smallest amount of non-reproducibility required to be detected
CONCORDANCE = *scalars*	Value of Lin's concordance for two samples with the smallest amount of non-reproducibility required to be detected
MEANSHIFT = *scalars*	Value of the shift in means (divided by the harmonic mean of the standard deviations) for two samples with the smallest amount of non-reproducibility required to be detected
SDRATIO = *scalars*	Value of the ratio of the standard deviations for two samples with the smallest amount of non-reproducibility required to be detected
NREPLICATES = *scalars*	Saves the required number of replicates
VREPLICATION = *variates*	Numbers of replicates for which powers have been calculated
VPOWER = *variates*	Power (i.e. probability of detection) for the various numbers of replicates

SMANNWHITNEY procedure

Calculates the sample sizes for the Mann-Whitney test (R.W. Payne).

Options

PRINT = *string token*	What to print (replication, power); default repl, powe
PROBABILITY = *scalar*	Significance level at which the test is to be made; default 0.05
POWER = *scalar*	The required power (i.e. probability of detection) of the test; default 0.9
TMETHOD = *string token*	Whether to a one- or two-sided test is to be made (onesided, twosided); default twos
RATIOREPLICATION = *scalar*	Ratio of replication sample2:sample1 (i.e. the size of sample 2 should be RATIOREPLICATION times the size of sample 1); default 1
REPLICATION = *variate*	Sample sizes for which to calculate and print or save the power; default * takes 11 replication values centred around the required number of replicates

Parameters

NULLPROBABILITIES = *variates*	Probabilities under null hypothesis
ODDSRATIO = *scalars*	Odds ratio for test group vs. control
NREPLICATES = *scalars*	Saves the required sample size
VREPLICATION = *variates*	Sample sizes for which powers have been calculated
VPOWER = *variates*	Power (i.e. probability of detection) for the various numbers of replicates

SMCNEMAR procedure

Calculates sample sizes for McNemar's test (R.W. Payne).

Options

PRINT = *string token*	What to print (replication, power); default repl, powe
PRMETHOD = *string token*	Method to be used to calculate the power of the McNemar test (normalapproximation, exact); default exac
PROBABILITY = *scalar*	Significance level at which the test is to be made; default 0.05
POWER = *scalar*	The required power (i.e. probability of detection) of the test; default 0.9
TMETHOD = *string token*	Whether a one- or two-sided test is to be made (onesided, twosided); default twos
REPLICATION = *variate*	Sample sizes for which to calculate and print or save the power; default * takes 11 replication values centred around the required number of replicates

Parameters

CHANGEPROBABILITY = *scalars*	Probability of any sort of change
RATIOPROBABILITIES = *scalars*	Ratio of the two probabilities of change
NREPLICATES = *scalars*	Saves the required sample size
VREPLICATION = *variates*	Sample sizes for which powers have been calculated
VPOWER = *variates*	Power (i.e. probability of detection) for the various numbers of replicates

SMOOTHSPECTRUM procedure

Forms smoothed spectrum estimates for univariate time series (G. Tunnicliffe Wilson & S.J. Welham).

Options

PRINT = *string token*	Controls printed output (description); default desc
METHOD = *string token*	Method to be used for smoothing (lagwindow, direct, YuleWalker, exactautoregressive); default lagw
BANDWIDTH = *scalar*	Frequency domain bandwidth for the smoothing window; must be set if METHOD=dire
MAXLAG = *scalar*	Specifies the cut-off lag (i.e. the maximum lag of autocovariance used in the spectrum calculation) for METHOD=lagw, or the order of the autoregression for METHOD=Yule or exac; if this option is not set then BANDWIDTH must be set, and will be used to determine an appropriate value of MAXLAG
DIVISIONS = *scalar*	Determines the number of frequency divisions into which the range [0.0, 0.5] is divided for calculating the spectrum; the default is chosen so that the bandwidth covers about four intervals
PROBABILITY = *scalar*	Probability value used for confidence limits; default 0.9
TAPER = *scalar*	The proportion of data to be tapered (applied for all settings of METHOD except exac); default 0.0
SHAPE = *scalar*	The shape of the trapezium window (a value of 1.0 specifies a rectangular, and 0.0 a triangular window); default 0.5
YLOG = *string token*	Whether to plot with a log-transformed Y-axis (yes, no); default no
XLOG = *string token*	Whether to plot with a log-transformed X-axis (yes, no); default no
GRAPHICS = *string token*	What sort of graphics to use (lineprinter, highresolution); default high
WINDOW = *scalar*	Window to be used for plotting; default 1
PENS = *variate*	The two pens to be used (after being defined appropriately) for drawing the plots; default !(1,2)

Parameters

SERIES = *variates*	The series for which the spectrum is to be calculated
LENGTH = *scalars* or *variates*	Scalar specifying that the first *N* units of the series are to be used, or a variate specifying the first and last units of the series to be used
SPECTRUM = *variates*	Saves the smoothed spectrum; need not be declared in advance, but will be set up as a variate of the appropriate length within the procedure
LOWER = *scalars* or *variates*	Scalar to save the multiplier of the spectrum used to calculate the lower limit, or a variate to save the values of the lower limit
UPPER = *scalars* or *variates*	Scalar to save the multiplier of the spectrum used to calculate the upper limit, or a variate to save the values of the upper limit
FREQUENCY = *variates*	Saves the frequency values at which the spectrum is calculated

SOM procedure

Declares a self-organizing map (R.W. Payne).

No options

Parameters

IDENTIFIER = *identifiers*	Identifiers of the SOMs
VARIABLENAMES = *texts*	Names of variables corresponding to the weights of each SOM
ROWS = *scalars* or *variates*	Number of rows or row coordinates for the map
COLUMNS = *scalars* or *variates*	Number of columns or column coordinates for the map
DMETHOD = *string tokens*	Method for calculating the distances of data points from the modes (euclidean, cityblock); default eucl
WMETHOD = *string tokens*	Method for calculating the contribution of a data point to each node when revising the weights (gaussian, neighbour); default gaus

SOMADJUST procedure

Performs adjustments to the weights of a self-organizing map (R.W. Payne).

Options

SOM = *pointer*	Self-organizing map
DATA = *matrix* or *pointer*	Data values for training the map
DMETHOD = *string token*	Method for calculating the distances of data points from the modes (euclidean, cityblock); default eucl
WMETHOD = *string token*	Method for calculating the contribution of a data point to each node when revising the weights (gaussian, neighbour); default gaus

Parameters

ALPHA = *scalars*	Alpha value for each iteration
SIGMA = *scalars*	Sigma value for each iteration when WMETHOD=gaussian
THRESHOLD = *scalars*	Threshold for each iteration when WMETHOD=neighbour
ERRORS = *matrices*	Saves the reconstruction errors at the nodes of the map after each iteration
TOTALERROR = *scalars*	Saves the total reconstruction error after each iteration
FITNODES = *factors*	Saves the nodes allocated to the data points after each iteration

SOMDESCRIBE procedure

Summarizes values of variables at nodes of a self-organizing map (R.W. Payne).

Options

PRINT = *string token*	Controls whether or not the summaries are printed (summaries); default summ
DATA = *matrix* or *pointer*	Data values to identify the positions of the samples on the map
SOM = *pointer*	Specifies the map

NEWSOM = *pointer* — Saves the map, augmented by the summary information

Parameters

Y = *variates* or *factors* — Data values to be summarized

METHOD = *string tokens* — How to summarize each Y (mean, mode, median, minimum, maximum, sd, variance); default mode for factors, mean for variates

SOMESTIMATE procedure

Estimates the weights for self-organizing maps (R.W. Payne).

Options

PRINT = *string tokens* — Controls output (weights, errors, monitoring, report); default weig, repo

PLOT = *string token* — Controls what to plot (fit, totalerror); default fit

DMETHOD = *string token* — Method for calculating the distances of data points from the modes (euclidean, cityblock); default eucl

WMETHOD = *string token* — Method for calculating the contribution of a data point to each node when revising the weights (gaussian, neighbour); default gaus

ALPHA = *scalar* or *variate* — Initial alpha value for each set of iterations; default !(1, 0.1)

SIGMA = *scalar* or *variate* — Initial sigma value for each set of iterations when WMETHOD=gaussian; default !(1, 0.01) multiplied by the maximum distance between nodes

THRESHOLD = *scalar* or *variate* — Initial distance threshold for each set of iterations when WMETHOD=neighbour; default !(0.5, 0.1) multiplied by the maximum distance between nodes

NCYCLE = *scalar* or *variate* — Number of cycles in each set of iterations; default 500

NSTOP = *scalar* — Number of consecutive cycles with no changes required for convergence; default 10

Parameters

SOM = *pointers* — Save the information about each map

DATA = *matrices* or *pointers* — Data values for training each map

ERRORS = *matrices* — Reconstruction errors at the nodes of each map

FITROWS = *factors* — Save the positions of the rows allocated to the data points

FITCOLUMNS = *factors* — Save the positions of the columns allocated to the data points

Y = *variates* — Save y-values used to plot the data points

X = *variates* — Save x-values used to plot the data points

PEN = *scalars*, *variates* or *factors* — Pens used to plot the maps

SEED = *scalars* — Seed for the random numbers used to initialize the weights in each map

SOMIDENTIFY procedure

Allocates samples to nodes of a self-organizing map (R.W. Payne).

No options

Parameters

DATA = *matrices* or *pointers* — Data values used to allocate the samples to the nodes of the map

SOM = *pointers* — Save the information about each map

FITNODES = *factors* — Save nodes allocated to the data points

FITROWS = *factors* — Save the positions of the rows allocated to the data points

FITCOLUMNS = *factors* — Save the positions of the columns allocated to the data points

SOMPREDICT procedure

Makes predictions using a self-organizing map (R.W. Payne).

Options

PRINT = *string token* — Controls whether or not the predictions are printed (predictions); default pred

SOM = *pointer*	Specifies the map
YNAMES = *text*	Names of variables to predict; default * gives predictions for all the variables
METHODS = *string tokens*	Types of predictions to give (mean, mode, median, minimum, maximum, sd, variance); default mean, mode, medi, mini, maxi, sd, vari
YSAVE = *text*	Saves a text with a unit for each set of predictions giving the name of the corresponding y-variable
MSAVE = *text*	Saves a text with a unit for each set of predictions giving the name of the corresponding method

Parameters

DATA = *matrices* or *pointers*	Data values to identify the positions of the new samples on the map
UNITLABELS = *variates* or *texts*	Labels for the predictions (to identify the samples); default takes the row labels if DATA is a matrix or any unit labels if DATA is a pointer to a set of variates
PREDICTIONS = *variates* or *pointers*	Save the predictions

SORT directive

Sorts units of vectors according to an index vector.

Options

INDEX = *vectors*	Variates, texts or factors whose values are to define the ordering; default is to use the first vector in the OLDVECTOR list
DIRECTION = *string token*	Order in which to sort (ascending, descending); default asce
DECIMALS = *scalar*	Number of decimal places to which to round before sorting numbers; default * i.e. no rounding

Parameters

OLDVECTOR = *vectors* or *pointers*	Factors, pointers, texts, or variates whose values are to be sorted
NEWVECTOR = *vectors* or *pointers*	Structure to receive each set of sorted values; if any are omitted, the values are placed in the corresponding OLDVECTOR

SPCAPABILITY procedure

Calculates capability statistics (R.W. Payne).

Option

PRINT = *string tokens*	Controls output (cpk, ppk, histogram); default cpk, ppk

Parameters

DATA = *variates* or *pointers*	Data measurements
SAMPLES = *factors* or *scalars*	Factor identifying samples or scalar indicating the size of each sample
LOWERLIMIT = *scalars*	Specifies the lower specification limit for each set of data
UPPERLIMIT = *scalars*	Specifies the upper specification limit for each set of data
CPK = *scalars*	Saves the index C_{pk}
PPK = *scalars*	Saves the index P_{pk}

SPCCHART procedure

Plots *c* or *u* charts representing numbers of defective items (A.F. Kane & R.W. Payne).

Options

PRINT = *string token*	What to print (warnings); default * i.e. nothing
PLOT = *string token*	Type of chart to plot (c, u); default c
METHOD = *string token*	Method to use to obtain the control limits (given, loglinear, untransformed); default untr
TOLERANCEMULTIPLIER = *scalar*	Multiplier to use to test whether to use mean sample size for

control limits; default 1

`WINDOW` = *scalar* — Which high-resolution graphics window to use; default 3

`SCREEN` = *string token* — Whether or not to clear the graphics screen before plotting (`clear`, `keep`); default `clea`

Parameters

`NDEFECTIVE` = *variates* — Number of defective items

`NTESTED` = *scalars* or *variates* — Number of items tested

`CENTRELINE` = *scalars* — Sets or saves centre line

`LOWERCONTROLLIMIT` = *scalars* or *variates* — Sets or saves lower control limit

`UPPERCONTROLLIMIT` = *scalars* or *variates* — Sets or saves upper control limit

`SPCOMBINE` procedure

Combines spreadsheet and data files, without reading them into Genstat (D.B. Baird).

Options

`OUTFILENAME` = *text* — Name of the output file

`METHOD` = *string token* — How to add the new data from the files specified by the `FILENAME` parameter (`add`, `append`, `concatenate`, `merge`); default `appe`

`COLMATCH` = *string token* — How to match columns when appending (`name`, `position`); default `posi`

`GROUPS` = *factor* — Factor to identify sections of appended files

`OLDGLABEL` = *texts* — Label to use in the `GROUPS` factor for the original data if `GROUPS` has not already been defined

`MATCH` = *text* or *pointer* — Up to four columns in the files specified by the `FILENAME` parameter to use as keys when merging files; default `*` uses the first column in the file

`WITH` = *text* or *pointer* — Columns in the `OUTFILENAME` file to use as keys when merging files; default `*` uses as many columns of the initial columns in `OUTFILENAME` as are needed to give a column for each `MATCH` column

`UPDATE` = *string token* — Whether to use columns with matching names to replace existing columns when concatenating or merging files (`yes`, `no`); default `no` changes the names of columns with the same name as existing columns so that they become unique

`UPDATE` = *string token* — Whether to use columns with matching names to replace existing columns when concatenating or merging files (`yes`, `no`); default `no` changes the names of columns with the same name as existing columns so that they become unique

Parameters

`FILENAME` = *texts* — Names of files containing new data to be combined with the data in the `OUTFILENAME` file

`SHEETNAME` = *texts* — Name of a worksheet or a named range within an Excel, Quattro, 123 or Open Office spreadsheet file; default takes the first sheet

`CELLRANGE` = *texts* — Cell range giving the top left and bottom right cells within a worksheet; default takes all the data that it contains

`ROWSELECTION` = *variates* — Row numbers of the units of data to be included into the `OUTFILENAME` file; default takes all the rows

`COLSELECTION` = *variates* — Numbers of the columns of data to be combined with the `OUTFILENAME` file; default takes all the columns

`PAGENAME` = *texts* — Page name for each new sheet when `METHOD=add`; default `'SHEET<n>'` where n is the number of the sheet in the `OUTFILENAME` file, unless the sheet is already named in the `FILENAME` file

GLABEL = *texts* — Label to use in the GROUPS factor to identify the data from each FILENAME file; if this is unset, GROUPS is defined with only levels

SPCUSUM procedure

Prints CUSUM tables for controlling a process mean (A.F. Kane & R.W. Payne).

Options

REFERENCEVALUE = *scalars* — Specifies the upper and then the lower reference values, or just one of these if they are both the same; default 0.5

THRESHOLD = *scalars* — Detection thresholds, upper and then the lower, or just one of these if they are both the same; default 5

HEADSTART = *scalars* — Headstart values, upper and then the lower, or just one of these if they are both the same; default 0

Parameters

DATA = *variates* or *pointers* — Data measurements

SAMPLES = *factors* or *scalars* — Factor identifying samples or scalar indicating the size of each sample

MEANTARGET = *scalars* — Specifies the target value for the sample means

SIGMA = *scalars* — Specifies or saves the standard deviation of the observations

SPEARMAN procedure

Calculates Spearman's Rank Correlation Coefficient (S.J. Welham, N.M. Maclaren & H.R. Simpson).

Options

PRINT = *string tokens* — Output required (test, correlations, ranks): test produces the correlation coefficient/matrix and relevant test statistics, correlations prints out just the correlation coefficients for each pair of variates; ranks produces the vectors of ranks for each sample; default test

GROUPS = *factor* — Defines the sample membership if only one variate is specified by DATA

CORRELATION = *scalar* or *symmetric matrix* — Scalar to save the rank correlation coefficient if there are two samples, or symmetric matrix to save the coefficients between all pairs of samples if there are several

T = *scalar* or *symmetric matrix* — Scalar to save the Student's t approximation to the correlation coefficient if there are two samples, or symmetric matrix to save the t approximations for all pairs of samples if there are several (calculated only if the sample size is 8 or more)

DF = *scalars* — Scalar to save the degrees of freedom for each t-statistic

Parameters

DATA = *variates* — List of variates containing the data for each sample, or a single variate containing the data from all the samples (the GROUPS option must then be set to indicate the sample to which each unit belongs)

RANKS = *variates* — Saves the ranks

SPEWMA procedure

Plots exponentially weighted moving-average control charts (A.F. Kane & R.W. Payne).

Options

PRINT = *string token* — What to print (warnings); default * i.e. nothing

TOLERANCEMULTIPLIER = *scalar* — Multiplier to use to test whether to use mean sample size for control limits; default 1

WEIGHT = *scalar* — Weight parameter used in the calculation of the exponentially weighted moving-average statistic; default 0.25

NSIGMA = *scalar* — Number of multiples of sigma to use for control limits; default

3

WINDOW = *scalar*	Which high-resolution graphics window to use; default 3
SCREEN = *string token*	Whether or not to clear the graphics screen before plotting (clear, keep); default clea

Parameters

DATA = *variates* or *pointers*	Data measurements
SAMPLES = *factors* or *scalars*	Factor identifying samples or scalar indicating the size of each sample
MEAN = *scalars*	Sets or saves the sample mean value
SIGMA = *scalars*	Sets or saves the sample standard deviation

SPLINE procedure

Calculates a set of basis functions for M-, B- or I-splines (P.W. Goedhart).

Options

KNOTS = *scalar* or *variate*	Defines the interior knot values; no default i.e. this option must be set
ORDER = *scalar*	Defines the order of the piecewise polynomial; default 3
TYPE = *string token*	Controls which spline basis is calculated (m, b, i); default m
LOWER = *scalar*	Left-hand limit *L* of the interval [*L*, *U*); default * i.e. the minimum of the X parameter is used
UPPER = *scalar*	Right-hand limit *U* of the interval [*L*, *U*); default * i.e. a value slightly larger than the maximum of the X parameter is used
NOMESSAGE = *string token*	Which warning messages to suppress (warning); default *

Parameters

X = *variates*	Values for which the basis spline functions are calculated
BASIS = *pointers*	Pointer to save variates containing the values of the basis spline functions
DBASIS = *pointers*	Pointer to save variates containing the values of the first order derivatives of the basis spline functions

SPLOAD directive

Loads Genstat spreadsheet files.

Options

†PRINT = *string token*	What to print (catalogue, directory, summary); default cata
SCOPE = *string token*	When SPLOAD is used within a procedure, this allows the data structures to be created in program that called the procedure (SCOPE=external) or in the main program itself (SCOPE=global) rather than within the procedure (local, external, global); default loca
REDEFINE = *string token*	Whether to allow existing structures to have their type redefined (no, yes); default no
SYSTEM = *string token*	Whether to include Genstat system structures in the catalogue (yes, no); default no
UNNAMED = *string token*	Whether to include unnamed structures in the catalogue (yes, no); default no
TEMPMISSING = *string token*	Whether to read temporarily missing values as missing (yes, no); default no

Parameters

FILENAME = *texts*	Names of spreadsheet files
SHEETNAME = *texts*, *variates* or *scalars*	Names or numbers of the sheets to read from each file; default * reads them all
ISAVE = *pointers*	Stores the identifiers of the structures loaded from each file

SPNTEST procedure

Calculates the sample size for a Poisson test (R.W. Payne & D.A. Murray).

Options

PRINT = *string token*	What to print (replication, power); default repl, powe
PRMETHOD = *string token*	Method to be used to calculate the probabilities for the test (normalapproximation, exact); default norm
PROBABILITY = *scalar*	Significance level for the test; default 0.05
POWER = *scalar*	The required power (i.e. probability of detection) of the test; default 0.9
TMETHOD = *string token*	Type of test to be done (onesided, twosided); default ones
NULL = *scalar*	Mean under the null hypothesis for the one-sample test; must be set when MU2 is unset
RATIOREPLICATION = *scalar*	Ratio of replication sample2:sample1 (i.e. the size of sample 2 should be RATIOREPLICATION times the size of sample 1); default 1
REPLICATION = *variate*	Replication values for which to calculate and print or save the power; default * takes 11 replication values centred around the required number of replicates

Parameters

MU1 = *scalars*	Mean to detect in sample 1
MU2 = *scalars*	Mean to detect in sample 2
NREPLICATES = *scalars*	Saves the required number of replicates
VREPLICATION = *variates*	Numbers of replicates for which powers have been calculated
VPOWER = *variates*	Power (i.e. probability of detection) for the various numbers of replicates

SPPCHART procedure

Plots *p* or *np* charts for binomial testing for defective items (A.F. Kane & R.W. Payne).

Options

PRINT = *string token*	What to print (warnings); default * i.e. nothing
PLOT = *string token*	Type of chart to plot (p, np); default p
METHOD = *string token*	Method to use to obtain the control limits (complementaryloglog, given, logit, probit, untransformed); default untr
TOLERANCEMULTIPLIER = *scalar*	Multiplier to use to test whether to use mean sample size for control limits; default 1
WINDOW = *scalar*	Which high-resolution graphics window to use; default 3
SCREEN = *string token*	Whether or not to clear the graphics screen before plotting (clear, keep); default clea

Parameters

NDEFECTIVE = *variates*	Number of defective items
NTESTED = *scalars* or *variates*	Number of items tested
CENTRELINE = *scalars*	Sets or saves centre line
LOWERCONTROLLIMIT = *scalars* or *variates*	Sets or saves lower control limit
UPPERCONTROLLIMIT = *scalars* or *variates*	Sets or saves upper control limit

SPRECISION procedure

Calculates the sample size to obtain a specified precision (R.W. Payne).

Options

PRINT = *string token*	What to print (replication, precision); default repl, prec
NSAMPLES = *scalar*	Number of samples (1 or 2); default 2
CIPROBABILITY = *scalar*	Probability level for the confidence interval to indicate the precision; default 0.95

RATIOREPLICATION = *scalar* — Ratio of replication sample2:sample1 (i.e. the size of sample 2 should have be RATIOREPLICATION times the size of sample 1); default 1

REPLICATION = *variate* — Replication values for which to calculate and print or save the precision; default * takes 11 replication values centred around the required number of replicates

Parameters

PRECISION = *scalars* — Required precision

VAR1 = *scalars* — Anticipated variance of sample 1

VAR2 = *scalars* — Anticipated variance of sample 2; default * assumes the same variance as sample 1

NREPLICATES = *scalars* — Saves the required number of replicates

VREPLICATION = *variates* — Numbers of replicates for which precisions have been calculated

VPRDETECTION = *variates* — Precision for the various numbers of replicates

SPSHEWHART procedure

Plots control charts for mean and standard deviation or range (A.F. Kane & R.W. Payne).

Options

PRINT = *string token* — What to print (warnings); default * i.e. nothing

PLOT = *string token* — Type of chart to plot to accompany the chart of sample means (range, standarddeviation); default stan

METHOD = *string token* — Type of control limits (probability, sigma); default sigm

TOLERANCEMULTIPLIER = *scalar* — Multiplier to use to test whether to use mean sample size for control limits; default 1

PROBABILITY = *scalars* — Probability value(s) to use to calculate control limits when METHOD=probability; default 0.01, 0.025

WINDOWS = *scalar* — Which high-resolution graphics windows to use; if unset SPSHEWHART automatically sets up two windows containing the upper and lower halves of the screen

SCREEN = *string token* — Whether or not to clear the graphics screen before plotting (clear, keep); default clea

Parameters

DATA = *variates* or *pointers* — Data measurements

SAMPLES = *factors* or *scalars* — Factor identifying samples or scalar indicating the size of each sample

MEAN = *scalars* — Sets or saves the sample mean value

SIGMA = *scalars* — Sets or saves the sample standard deviation

SPSYNTAX procedure

Puts details about the syntax of commands into a spreadsheet (R.W. Payne).

Option

OUTFILENAME = *texts* — Name of Genstat file (.gsh or .gwb) or Excel (.xls or .xlsx) file to create

Parameter

COMMAND = *texts* — Single-line texts specifying the commands

SSIGNTEST procedure

Calculates the sample size for a sign test (R.W. Payne).

Options

PRINT = *string token* — What to print (replication, power); default repl, powe

PROBABILITY = *scalar* — Significance level at which the response is to be tested; default 0.05

POWER = *scalar* — The required power (i.e. probability of detection) of the test; default 0.9

TMETHOD = *string token* — Whether to a one- or two-sided test is to be made (onesided,

	`twosided`); default `twos`
`REPLICATION` = *variate*	Replication values for which to calculate and print or save the power; default `*` takes 11 replication values centred around the required number of replicates

Parameters

`RESPONSE` = *scalars*	Probability of response (i.e. the probability that an observation in one sample will be greater than the equivalent observation in the other sample) that should be detectable
`NREPLICATES` = *scalars*	Saves the required number of replicates
`VREPLICATION` = *variates*	Numbers of replicates for which powers have been calculated
`VPOWER` = *variates*	Power (i.e. probability of detection) for the various numbers of replicates

`SSPM` directive

Declares one or more SSPM data structures.

Options

`TERMS` = *formula*	Terms for which sums of squares and products are to be calculated; default `*`
`FACTORIAL` = *scalar*	Maximum number of vectors in a term; default 3
`FULL` = *string token*	Full factor parameterization (`yes`, `no`); default `no`
`GROUPS` = *factor*	Groups for within-group SSPMs; default `*`
`DF` = *scalar*	Number of degrees of freedom for sums of squares; default `*`

Parameters

`IDENTIFIER` = *identifiers*	Identifiers of the SSPMs
`SSP` = *symmetric matrices*	Symmetric matrix to contain the sums of squares and products for each SSPM
`MEANS` = *variates*	Variate to contain the means for each SSPM
`NUNITS` = *scalars*	Number of units or sum of weights for each SSPM
`WMEANS` = *pointers*	Pointers to variates of group means for each SSPM

`STACK` procedure

Combines several data sets by "stacking" the corresponding vectors (R.W. Payne).

Option

`DATASET` = *factor*	Factor to indicate the data set to which each unit originally belonged

Parameters

`STACKEDVECTOR` = *variates*, *factors* or *texts*	New vectors combining the corresponding members of the data sets specified by parameter `V1`, or parameters `V1`-`V100`
`V1` = *pointers*, *variates*, *factors*, *texts* or *scalars*	Pointers defining (all) the components to be stacked into each `STACKEDVECTOR`, or contents of the first data set
`V2` - `V100` = *variates*, *factors*, *texts* or *scalars*	Data sets 2 - 100
`FREPRESENTATION` = *string token*	How to match the values of factors (`levels`, `labels`, `ordinals`, `renumbered`); default `leve`

`STANDARDIZE` procedure

Standardizes columns of a data matrix to have mean zero and variance one (S.A. Harding & D.A. Murray).

No options

Parameters

`OLD` = *variates* or *matrices*	Structures containing data to be standardized
`NEW` = *variates* or *matrices*	Structures to contain output; by default the `OLD` structures are overwritten

STEEL procedure

Performs Steel's many-one rank test (R.W. Payne).

Options

PRINT = *string token*	Controls printed output (description, sumranks, critical, permutationtest); default desc, sumr, crit
METHOD = *string token*	Form of the alternative hypothesis (twosided, greaterthan, lessthan); default twos
TREATMENTS = *factor*	Defines the treatments
CONTROL = *scalar* or *text*	Treatment level corresponding to the control; default takes the reference level of TREATMENTS
NTIMES = *scalar*	Number of permutations for the permutation test; default 999
SEED = *scalar*	Seed to use to generate the random numbers for the permutation test; default 0

Parameters

DATA = *variates*	Data values for the tests
SUMRANKS = *tables*	Saves the sum of the ranks within the treatments from each test
RANKS = *variates*	Saves the ranks of the data values for each test

STEM procedure

Produces a simple stem-and-leaf chart (J. Ollerton & S.A. Harding).

No options

Parameters

DATA = *variates*	Data values for each plot
NDIGITS = *scalars*	Number of digits in the leaves of each plot
STEMUNITS = *scalars*	Scale units for the stem values in each plot

STEP directive

Selects terms to include in or exclude from a linear, generalized linear or generalized additive model according to the ratio of residual mean squares.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, changes, confidence); default mode, summ, esti, chan
FACTORIAL = *scalar*	Limit for expansion of model terms; default * i.e. that in previous TERMS statement
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
INRATIO = *scalar*	Criterion for inclusion of terms; default 1.0

OUTRATIO = *scalar*	Criterion for exclusion of terms; default 1.0
MAXCYCLE = *scalar*	Limit on number of times to repeat stepwise selection, unless no change is made; default 1
PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95

Parameter

formula	List of explanatory variates and factors, or model formula

STOP directive

Ends a Genstat program.

No options or parameters

STORE directive

To store structures in a subfile of a backing-store file.

Options

PRINT = *string token*	What to print (catalogue); default *
CHANNEL = *scalar*	Channel number of the backing-store file where the subfile is to be stored; default 0, i.e. the workfile
SUBFILE = *identifier*	Identifier of the subfile; default SUBFILE
LIST = *string token*	How to interpret the list of structures (inclusive, exclusive, all); default incl
METHOD = *string token*	How to append the subfile to the file (add, overwrite, replace, update); default add, i.e. clashes in subfile identifiers cause a fault (note: replace overwrites the complete file)
PASSWORD = *text*	Password to be stored with the file; default *
PROCEDURE = *string token*	Whether subfile contains procedures only (yes, no); default no
UNNAMED = *string token*	Whether to list unnamed structures (yes, no); default no
MERGE = *string token*	Whether or not to merge the structures with the existing contents of the subfile (yes, no); default no

Parameters

IDENTIFIER = *identifiers*	Identifiers of the structures to be stored
STOREDIDENTIFIER = *identifiers*	Identifier to be used for each structure when it is stored

STRUCTURE directive

Defines a compound data structure.

Options

NAME = *text*	Single-valued text defining a name for the type of structure, which must not clash with the name of any existing type of structure
STRUCTURELIST = *string token*	Whether or not the structure consists of a list (of any length) of structures of the same type or types (yes, no); default no

Parameters

LABEL = *texts*	Single-valued texts defining the labels of the elements of the structure
SUFFIX = *scalars*	Suffix numbers for the elements; default assumes the numbers 1, 2 ...
TYPE = *texts*	Texts defining the allowed types for each element
COMPATIBLE = *texts*	Defines aspects to check for compatibility with the first element

STTEST procedure

Calculates the sample size for t-tests, including equivalence tests (R.W. Payne).

Options

PRINT = *string token*	What to print (replication, power); default repl, powe

NSAMPLES = *scalar* — Number of samples for the t-test (1 or 2); default 2
PROBABILITY = *scalar* — Significance level at which the response is to be tested; default 0.05
POWER = *scalar* — The required power (i.e. probability of detection) of the test; default 0.9
TMETHOD = *string token* — Type of test to be done (onesided, twosided, equivalance, noninferiority); default ones
RATIOREPLICATION = *scalar* — Ratio of replication sample2:sample1 (i.e. the size of sample 2 should be RATIOREPLICATION times the size of sample 1); default 1
REPLICATION = *variate* — Replication values for which to calculate and print or save the power; default * takes 11 replication values centred around the required number of replicates

Parameters

RESPONSE = *scalars* — Response to be detected
VAR1 = *scalars* — Anticipated variance of sample 1
VAR2 = *scalars* — Anticipated variance of sample 2; default * assumes the same variance as sample 1
NREPLICATES = *scalars* — Saves the required number of replicates
VREPLICATION = *variates* — Numbers of replicates for which powers have been calculated
VPOWER = *variates* — Power (i.e. probability of detection) for the various numbers of replicates

SUBSET procedure

Forms vectors containing subsets of the values in other vectors (R.W. Payne).

Options

CONDITION = *expression* — Logical expression to define which units are to be included; no default – this option must be set
SETLEVELS = *string token* — Whether to reform the levels (and labels) of factors to exclude those that do not occur in the subset (yes, no); default no

Parameters

OLDVECTOR = *vectors* — Vector from which the subset is to be formed
NEWVECTOR = *vectors* — Vector to store the subsets if none is specified, the OLDVECTOR is redefined to store the subset

SUSPEND directive

Suspends execution of Genstat to carry out commands in the operating system; this directive may not be available on some computers.

Options

SYSTEM = *text* — Commands for the operating system; default: prompt for commands (interactive mode only)
CONTINUE = *string token* — Whether to continue execution of Genstat without waiting for commands to complete (yes, no); default no
MINIMIZE = *string token* — Whether to minimize the console window (yes, no); default no

No parameters

SVBOOT procedure

Bootstraps data from random surveys (S.D. Langton).

Options

PRINT = *string token* — Controls printed output (summary); default * i.e. none
SEED = *scalar* — Seed for random numbers; default 0
STRATUMFACTOR = *factor* — Stratification factor
SAMPLINGUNITS = *factor* — Sampling units (default single stage design)
WEIGHTS = *variates* — Weights variates (not required for simple bootstrap)
METHOD = *string token* — Method (simple, sarndal); default simp

POPULATION = *pointers*	Units in the population
SAVEUNITS = *variate*	Units in the bootstrapped sample
BSTRATUMFACTOR = *factor*	Bootstrapped stratification factor
BSAMPLINGUNITS = *factor*	Bootstrapped sampling units

Parameters

DATA = *variates* or *factors*	Data to bootstrap
BOOT = *variates* or *factors*	Saves bootstrap sampling units

SVCALIBRATE procedure

Performs generalized calibration of survey data (S.D. Langton).

Options

PRINT = *string token*	Controls printed output (summary, totals, monitoring); default summ, tota
PLOT = *string token*	Controls which high-resolution graphs are plotted (weights); default * i.e. none
STRATUMFACTOR = *factor*	Stratification factor; default * i.e. unstratified
SAMPLINGUNITS = *factor*	Factors indicating the sampling units in a two-stage design; default *, i.e. single-stage design
TCONSTRAINTS = *scalars*	Constraint totals or tables
X = *variates*	Variates corresponding to TCONSTRAINTS; * implies the equivalent constraint relates to a count
WEIGHTS = *variate*	Initial weights
OUTWEIGHTS = *variate*	Final (calibration) weights
METHOD = *string token*	Method to use (linear, truncatedlinear, logistic, fittedvalues); default line
LOWER = *scalar*	Lower bound for g-weights; default 0.1
UPPER = *scalar*	Upper bound for g-weights; default 10
MAXCYCLE = *scalar*	Maximum number of iterations; default 50
TOLERENCE = *scalar*	Tolerence for convergence; default 0.0001

Parameters

Y = *variates*	Response data for analysis
TOTALS = *scalars*	Saves estimated totals
SETOTALS = *scalars*	Saves standard errors of totals
FITTEDVALUES = *variates*	Saves fitted values from the regression

SVD directive

Calculates singular value decompositions of matrices.

Option

PRINT = *string tokens*	Printed output required (left, singular, right); default * i.e. no printing

Parameters

INMATRIX = *matrices*	Matrices to be decomposed
LEFT = *matrices*	Left-hand matrix of each decomposition
SINGULAR = *diagonal matrices*	Singular values (middle) matrix
RIGHT = *matrices*	Right-hand matrix of each decomposition

SVGLM procedure

Fits generalized linear models to survey data (S.D. Langton).

Options

PRINT = *string token*	What output to display (model, summary, estimates, wald, predictions, monitor); default mode, esti, wald, pred
DISTRIBUTION = *string token*	Error distribution (binomial, poisson, normal, gamma); default norm
LINK = *string token*	Link function (identity, logarithm, logit, reciprocal, probit, complementaryloglog, canonical); default cano

DISPERSION = *scalar*	Value at which to fix the residual variance, if missing the variance is estimated; default 1 for binomial or Poisson, otherwise *
TERMS = *formula*	Explanatory model
CONSTANT = *string token*	Whether to estimate or omit constant term in fixed model (omit, estimate); default esti
FACTORIAL = *scalar*	Limit on number of factors/covariates in a model term; default 3
PFACTORS = *factors* or *variates*	Variables for which predictions are to be formed; default *, or as specified in PTERMS
PLEVELS = *variates* or *scalars*	Levels or values at which predictions are to be made corresponding to PFACTORS; default (weighted) mean for variates, all levels for factors
PTERMS = *formula*	Formula specifying fixed terms for which predicted means are to be printed; default *, unless PFACTORS is set, in which case it is all main effects of and interactions between PFACTORS
STRATUMFACTOR = *factor*	Stratification factor; default *, i.e. unstratified
NUNITS = *variate* or *table*	Number of primary sampling units in each stratum
SAMPLINGUNITS = *factor*	Factor indicating the primary sampling units; default *, i.e. single stage design
WEIGHTS = *variates*	Survey weights
METHOD = *string token*	Bootstrapping method (simple, csimple, sarndal); default simp
NBOOT = *scalar*	Number of bootstrap samples to use; default 0 uses a Taylor series approximation
SEED = *scalar*	Seed for random number generator for bootstrap; default 0
CIPROBABILITY = *scalars*	The probability level for the confidence intervals; default 0.95
CIMETHOD = *string token*	Method for forming confidence intervals (automatic, tdistribution, percentile); default auto

Parameters

Y = *variates*	Dependent variates
NBINOMIAL = *scalars* or *variates*	Number of binomial trials for each unit (must be set if DISTRIBUTION=binomial)
RESIDUALS = *variates*	Variates to save residuals
FITTEDVALUES = *variates*	Variates to save fitted values
ESTIMATES = *variates*	Estimates of parameters for each Y variate
SE = *variates*	Standard errors of the estimates
VCOVARIANCE = *symmetric matrices*	Variance-covariance matrix for the estimates
LOWER = *variates*	Lower confidence limits for estimates
UPPER = *variates*	Upper confidence limits for estimates
WALD = *pointers*	Pointers to save Wald statistics for each term (pointer contains name of term, Wald statistic, F statistic, degrees of freedom, and P-value)
PREDICTIONS = *pointers*	Pointers to tables of predictions
SEPREDICTIONS = *pointers*	Pointers to tables of standard errors of predictions
LOWPREDICTIONS = *variates*	Lower confidence limits for predictions
UPPREDICTIONS = *variates*	Upper confidence limits for predictions
VCPREDICTIONS = *symmetric matrices*	Variance-covariance matrix for the predictions

SVHOTDECK procedure

Performs hot-deck and model-based imputation for survey data (S.D. Langton).

Options

PRINT = *string token*	Controls printed output (summary, monitoring, check, list, regression); default summ
METHOD = *string token*	Imputation method (hotdeck, modelbased); default hotd
DMETHOD = *string token*	Method for calculating distances (mean, minimax,

	regression); defaule mini
%THRESHOLD = *scalar*	Percentage threshold for matches
THRESHOLD = *scalar*	Absolute threshold for matches
DVARIABLES = *variates* or *factors*	Variables to use for distance calculation or factors
DRANGES = *scalars*	Ranges to use for distance calculations with each of the DVARIABLES; default * uses the observed range
LABELS = *variate*, *factor* or *text*	Provides labels for the cases
SEED = *scalar*	Seed for random numbers; default 0
IMPUTE = *variate* or *scalar*	The variate provides logical (0 or 1) values to indicate whether each unit is to be imputed, alternatively the scalar specifies a number of rows to be selected at random to be imputed to allow the effectiveness of the imputation process to be studied; default * imputes values for any units where an OLDSTRUCTURE contains a missing value
DONORS = *variate*	Logical variate indicating whether each unit can be used as a donor; default * implies that all units are used with complete data for each OLDSTRUCTURE
RSAVE = *rsave*	Regression analysis to use for METHOD=model or DMETHOD=regression
URECEPTORS = *variate*	Saves unit numbers of receptor (imputed) cases
UDONORS = *variate*	Saves unit numbers of donor cases
DISTANCES = *variate*	Saves the distances for the chosen receptor-donor pairs
Parameters	
OLDSTRUCTURES = *variates* or *factors*	Structure containing missing values
NEWSTRUCTURES = *variates* or *factors*	New structures with imputed values
OVERWRITE = *string tokens*	Whether to overwrite any existing data for imputed cases (yes, no); default no

SVMERGE procedure

Merges strata prior to survey analysis (S.D. Langton).

Options	
PRINT = *string token*	Controls printed output (summary, intable, outtable, twowaytable); default summ
OLDFACTOR = *factor*	Factor defining the original strata
NEWFACTOR = *factor*	Factor to save the merged strata
Parameters	
MERGELABELS = *texts*	Labels of strata to merge
NEWLABEL = *text*s	Label for merged stratum

SVMFIT procedure

Fits a support vector machine (D. B. Baird).

Options	
PRINT = *string tokens*	Printed output from the analysis (summary, predictions, allocations, debug); default summ, alloc
SVMTYPE = *string token*	Type of support vector machine to fit (svc, svr, nusvc, nusvr, lsvc, lsvr, lcs, svm1); default svc
KERNEL = *string token*	Type of kernel to use (linear, polynomial, radialbasis, sigmoid); default radi
PENALTY = *scalar* or *variate*	Penalty or cost for points on the wrong side of the boundary; default 1
GAMMA = *scalar* or *variate*	Gamma parameter for types with non-linear kernels; default 1
NU = *scalar* or *variate*	Nu parameter for types nusvc, nusvr, and svm1; default 0.5
EPSILON = *scalar* or *variate*	Epsilon parameter for types svr and lsvr; default 0.1
BIAS = *scalar*	Bias for allocations to groups for types lsvc and lsvr; default −1 i.e. no bias
DEGREE = *scalar*	Degree for polynomial kernel; default 3

CONSTANTVALUE = *scalar* Constant for polynomial or sigmoid kernel; default 0
LOWER = *scalar* or *variate* Lower limit for scaling data variates when SCALING = given; default −1
UPPER = *scalar* or *variate* Upper limit for scaling data variates when SCALING = given; default 1
SCALING = *string token* Type of scaling to use (none, uniform, given); default unif
NOSHRINK =*string token* Whether to suppress the shrinkage of attributes to exclude unused ones (no, yes); default no
OPTMETHOD =*string token* Whether to optimize probabilities or allocations (allocations, probabilities); default allo
REGULARIZATIONMETHOD = *string token* Regularization method for SMVTYPE = lsvc or lsvr (l1, l2); default l2
LOSSMETHOD = *string token* Loss method for SMVTYPE = lsvc or lsvr (logistic, l1, l2); default logi
DUALMETHOD = *string token* Whether to use the dual algorithm for SMVTYPE = lsvc or lsvr (yes, no); default no
NCROSSVALIDATIONGROUPS = *scalar* Number of groups for cross-validation; default 10
SEED = *scalar* Seed for random number generation; default 0
TOLERANCE = *scalar* Tolerance for termination criterion; default 0.001
WORKSPACE = *scalar* Size of workspace needed for data; default is to calculate this from the number of observations and variates

Parameters

Y = *factors or variates* Define groupings for the units in each training set, or missing values for the units to be allocated; or y-variate to be predicted via regression
X = *pointers* Each pointer contains a set of explanatory variates or factors
WEIGHTS = *variates* Weights to multiply penalties for each group when SMVTYPE = svc, nusvc, lsvc or lcs
PREDICTIONS = *factors or variates* Saves allocations to groups or predictions from regression
ERRORRATE = *scalars, variates* or *matrices* Saves the error rate for the combinations of parameters specified for the support vector machine
OPTPENALTY = *scalars* Saves the optimal value of penalty parameter
OPTGAMMA = *scalars* Saves the optimal value of gamma parameter
OPTNU = *scalars* Saves the optimal value of nu parameter
OPTEPSILON = *scalars* Saves the optimal value of epsilon parameter
OPTERRORRATE = *scalars* Saves the minimum error rate
SCALE = *texts* or *pointers* Saves the scaling used for the X variates, in a file if a text is given, or otherwise in a pointer to a pair of variates
SAVEFILE = *texts* File in which to save the model, for use by SVMPREDICT

SVMPREDICT procedure

Forms the predictions using a support vector machine (D. B. Baird).

Options

SCALE = *texts* or *pointers* Gives scaling used for the X variates
SAVEFILE = *texts* Gives support vector machine model file; default is to use the model from the last support vector machine

Parameters

X = *pointers* Each pointer contains a set of variates defining the attributes for the predictions
PREDICTIONS = *factors* or *variates* Saves the classification groupings or predicted values for each observation in X
GROUPDEFINITIONS = *factors* Supplies levels and labels for predicted groups; default uses ordinal levels

SVREWEIGHT procedure

Modifies survey weights for particular observations, adjusting other weights in the sampling unit or stratum to ensure that the overall sum of the weights remains unchanged (S.D. Langton).

Options

PRINT = *string tokens*	Controls printed output (summary); default summ
METHOD = *string tokens*	What to reweight over (all, stratum, samplingunits, lowest); default lowe
WEIGHTS = *variate*	Initial weights
OUTWEIGHTS = *variate*	Final weights
STRATUMFACTOR = *factor*	Stratification factor; default * i.e. unstratified
OUTSTRATUMFACTOR = *factor*	Saves a modified stratification factor with the reweighted observations in their own stratum
SAMPLINGUNITS = *factor*	Factor indicating the primary sampling units; default *, i.e. single stage design
LABELS = *variate*, *text* or *factor*	Labels for each unit

Parameters

OBSERVATIONS = *scalars*, *variates* or *texts*	Observation to reweight
NEWWEIGHTS = *scalars* or *variates*	New weight (default inserts a missing value, indicating that the observation should be removed)

SVSAMPLE procedure

Constructs stratified random samples (S.D. Langton).

Options

PRINT = *string token*	Controls printed output (list, summary); default summ
SAMPLE = *variate*	Saves the sample, as unit numbers of sampled units when METHOD=sample, or as a logical (1 or 0) variable indicating sampled or unsampled units when METHOD=population
STRATUMFACTOR = *factor*	Saves the stratification factor
CLUSTERS = *factor*	Specifies a factor indicating groupings of units for a cluster sample; default * i.e. sample individual rows
NUNITS = *table*, *scalar* or *variate*	Numbers of units in the full data set for each level of the STRATUMFACTOR
NSAMPLE = *table*, *scalar* or *variate*	Numbers, or proportions, of units to sample for each level of the STRATUMFACTOR
SFLEVELS = *variate*	Levels for the stratum factor, if it has not already been declared
SFLABELS = *text*	Labels for the stratum factor, if it has not already been declared
METHOD = *string token*	Whether SAMPLE should contain the numbers of the units sampled from the population, or be a variate with a value for every unit of the full population containing 0 or 1 for unsampled and sampled units respectively (population, sample); default samp
NUMBERING = *string token*	Whether to number units within each stratum, or across the whole population (withinstratum, population); default with
SEED = *scalar*	Seed for the random number generator; default 0 i.e. continue from previous generation

Parameters

OLDVECTOR = *variates*, *factors* or *texts*	Data from the full survey
NEWVECTOR = *variates*, *factors* or *texts*	Data for the sample

SVSTRATIFIED procedure

Analyses stratified random surveys by expansion or ratio raising (S.D. Langton).

Options

PRINT = *string token*	Controls printed output (summary, totals, means, influence, ratios, extra); default summ, tota, infl
PLOT = *string token*	Controls which high-resolution graphs are plotted (single, separate); default * i.e. none
XMISSING = *string token*	Action if x-variable contains missing values (estimate, fault); default esti
RESTRICTED = *string token*	Action with restricted (or filtered) observations (omit, add); default omit
STRATUMFACTOR = *factor*	Stratification factor; default * i.e. unstratified
NINFLUENCE = *scalar*	Number of influential points to print; default 10
METHOD = *string token*	Method for ratio analysis (separate, combined, classicalcombined); default sepa
SAVESUMMARY = *string token*	Whether to save just the overall summaries instead of those for each stratum (yes, no); default no
COMBINEDSTRATUM = *scalar*	Stratum for which the ratio should be set to the combined ratio estimate; default *
ROWS = *scalars*	Number of rows of plot-matrix; default * i.e. set automatically depending on number of levels of STRATUMFACTOR
COLUMNS = *scalars*	Number of columns of plot-matrix; default * i.e. set automatically depending on number of levels of STRATUMFACTOR
NBOOT = *scalar*	Number of bootstrap samples to use; default 0
SEED = *scalar*	Seed for random number generator for bootstrap; default 0
CIPROBABILITY = *scalars*	The probability level for the confidence intervals; default 0.95
CIMETHOD = *string token*	Method for forming confidence intervals (automatic, tdistribution, percentile); default auto
COMPACT = *string token*	Whether to produce output in a compact (plaintext) format (yes, no); default no

Parameters

Y = *variates*	Response data
X = *variates*	Base data; if unset expansion raising is used
LABELS = *variates*, *factors* or *texts*	Structure for labelling influential points
NUNITS = *tables*, *scalars* or *variates*	Numbers of units in each stratum in the population
XTOTALS = *tables*, *scalars* or *variates*	Population totals of the base data in each stratum
TOTALS = *tables* or *scalars*	Saves total estimates
SETOTALS = *tables* or *scalars*	Saves standard errors of estimates
MEANS = *tables* or *scalars*	Saves mean estimates
SEMEANS = *tables* or *scalars*	Saves standard errors of mean estimates
RATIOS = *tables*	Saves estimates of ratios
FITTEDVALUES = *variates*	Saves fitted values for the observations
INFLUENCE = *variates*	Saves influence statistics
LTOTALS = *tables* or *scalars*	Saves lower confidence limit for total
UTOTALS = *tables* or *scalars*	Saves upper confidence limit for total
LMEANS = *tables* or *scalars*	Saves lower confidence limit for mean
UMEANS = *tables* or *scalars*	Saves upper confidence limit for mean
VARIANCES = *tables* or *scalars*	Saves residual variances in each stratum

SVTABULATE procedure

Tabulates data from random surveys, including multistage surveys and surveys with unequal probabilities of selection (S.D. Langton).

Options

PRINT = *string token*	Controls printed output (summary, stratumsummary, psusummary, totals, means, ratios, influence, wald,

quantiles, monitor); default summ, tota, infl

PLOT = *string token* — Controls which high-resolution graphs are plotted (single, separate, weights, influence); default * i.e. none

STRATUMFACTOR = *factor* — Stratification factor; default *, i.e. unstratified

NUNITS = *table*, *scalar* or *variate* — Numbers of units in each STRATUMFACTOR level (for a multistage design these will be the number of primary sampling units)

SAMPLINGUNITS = *factor* — Factor indicating the primary sampling units; default *, i.e. single stage design

NSECONDARYUNITS = *table*, *scalar* or *variate* — Numbers of secondary sampling units for the levels of the SAMPLINGUNITS factor

CLASSIFICATION = *factors* — Domains for which separate estimates are required

NINFLUENCE = *scalar* — Number of influential points to print; default 10

MRFACTOR = *identifiers* — Identifier of factors to index the sets of multiple responses in the tables

WEIGHTS = *variate* — Survey weights

FPCOMIT = *string token* — Whether to omit the finite population correction from calculation of variances (yes, no); default no

METHOD = *string token* — Method of bootstrapping (simple, sarndal); default simp

NBOOT = *scalar* — Number of bootstrap samples to use; default 0 uses a Taylor series approximation

SEED = *scalar* — Seed for random number generator for bootstrap; default 0

CIPROBABILITY = *scalar* — The probability level for the confidence intervals; default 0.95

CIMETHOD = *string token* — Method for forming confidence intervals (automatic, tdistribution, percentile); default auto

PERCENTQUANTILES = *scalar* or *variate* — Percentage points for which quantiles are required; default 50 (i.e. median)

Parameters

Y = *variates* — Response data

X = *variates* — Base data for ratio estimation

LABELS = *variates* or *texts* — Labels for influential points

OUTWEIGHTS = *tables* — Saves weights

TOTALS = *tables* or *scalars* — Saves total estimates

SETOTALS = *tables* or *scalars* — Saves standard errors of estimates

VCTOTALS = *symmetric matrices* — Saves variance-covariance matrix of total estimates

MEANS = *tables* or *scalars* — or scalars Saves mean estimates

SEMEANS = *table or scalars* — Saves standard errors of mean estimates

VCMEANS = *symmetric matrices* — Saves variance-covariance matrix of mean estimates

RATIOS = *tables* or *scalars* — Saves estimates of ratios

SERATIOS = *tables* or *scalars* — Saves standard errors of ratios

VCRATIOS = *symmetric matrices* — Saves variance-covariance matrix of ratio estimates

NOBSERVATIONS = *tables* or *scalars* — Saves numbers of (non-missing) observations

SUMWEIGHTS = *tables* or *scalars* — Saves sums of weights

FITTEDVALUES = *variates* — Supplies fitted values for each observation

INFLUENCE = *variates* — Saves influence statistics

WALD = *variates* — Saves Wald statistics

QUANTILES = *tables* or *pointers* — Table to contain quantiles at a single PERCENTQUANTILE or pointer of tables for several PERCENTQUANTILEs

SEQUANTILES = *tables* or *pointers* — Saves standard errors of quantiles

VCQUANTILES = *tables* or *pointers* — Saves variance-covariance matrix of quantiles

LQUANTILES = *tables* or *pointers* — Saves lower confidence limits of quantiles

UQUANTILES = *tables* or *pointers* — Saves upper confidence limits of quantiles

LTOTALS = *tables* — Saves lower confidence limits of totals

UTOTALS = *tables* — Saves upper confidence limits of totals

LMEANS = *tables*	Saves lower confidence limits of means
UMEANS = *tables*	Saves upper confidence limits of means
LRATIOS = *tables*	Saves lower confidence limits of ratios
URATIOS = *tables*	Saves upper confidence limits of ratios
CELLINFLUENCE = *variates*	Saves influence statistics for individual cells

SVWEIGHT procedure

Forms survey weights (S.D. Langton).

Options

PRINT = *string token*	Controls printed output (summary, stratumsummary, psusummary); default summ, stra, psus
PLOT = *string token*	Controls which high-resolution graphs are plotted (weights); default * i.e. none
STRATUMFACTOR = *factor*	Stratification factor; default *, i.e. unstratified
NUNITS = *tables*, *scalars* or *variates*	Numbers of units in each STRATUMFACTOR (for a multistage design these will be the number of primary sampling units)
SAMPLINGUNITS = *factor*	Factor indicating the primary sampling units; default *, i.e. single stage design.
NSECONDARYUNITS = *tables*, *scalars* or *variates*	Numbers of secondary sampling units for each level of the SAMPLINGUNITS factor

Parameters

Y = *variates* of *scalars*	Response data or a scalar indicating the number of sampled units
OUTWEIGHTS = *variates*	Saves weights

SWITCH directive

Adds terms to, or drops them from a linear, generalized linear, generalized additive or nonlinear model.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence); default mode, summ, esti
NONLINEAR = *string token*	How to treat nonlinear parameters between groups (common, separate, unchanged); default unch
CONSTANT = *string token*	How to treat the constant (estimate, omit, unchanged, ignore); default unch
FACTORIAL = *scalar*	Limit for expansion of model terms; default * i.e. that in previous TERMS statement
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance,

	`%deviance`, `aic`, `bic`, `sic`); default `%var`, `seob` if `DIST=normal`, `%cv` if `DIST=gamma`, and `disp` for other distributions
`PROBABILITY` = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
`AOVDESCRIPTION` = *text*	Description for line in accumulated analysis of variance (or deviance) table when `POOL=yes`

Parameter

formula	List of explanatory variates and factors, or model formula

`SYMMETRICMATRIX` directive

Declares one or more symmetric matrix data structures.

Options

`ROWS` = *scalar*, *vector*, *pointer* or *text*	Number of rows, or labels for rows (and columns); default `*`
`VALUES` = *numbers*	Values for all the symmetric matrices; default `*`
`MODIFY` = *string token*	Whether to modify (instead of redefining) existing structures (`yes`, `no`); default `no`
`IPRINT` = *string tokens*	Information to be used by default to identify the symmetric matrices in output (`identifier`, `extra`); if this is not set, they will be identified in the standard way for each type of output

Parameters

`IDENTIFIER` = *identifiers*	Identifiers of the symmetric matrices
`VALUES` = *identifiers*	Values for each symmetric matrix
`DECIMALS` = *scalars*	Number of decimal places for printing
`EXTRA` = *texts*	Extra text associated with each identifier
`MINIMUM` = *scalars*	Minimum value for the contents of each structure
`MAXIMUM` = *scalars*	Maximum value for the contents of each structure
`DREPRESENTATION` = *scalars* or *texts*	Default format to use when the contents represent dates and times

`SYNTAX` directive

Obtains details of the syntax of a command and the source code of a procedure.

No options

Parameters

`COMMAND` = *texts*	Single-line texts specifying the commands
`NOPTIONS` = *scalars*	Number of options for each command
`NPARAMETERS` = *scalars*	Number of parameters for each command
`NAME` = *texts*	Names of the options, and then the parameters, of each command
`MODE` = *texts*	Modes of the options and parameters
`NVALUES` = *pointers*	Number of values allowed for the options and parameters
`VALUES` = *pointers*	Allowed values for the options and parameters
`DEFAULT` = *pointers*	Default values for the options and parameters
`SET` = *texts*	Whether the options and parameters must be set
`DECLARED` = *texts*	Whether the options and parameters must have been declared
`TYPE` = *pointers*	Allowed types for the options and parameters
`COMPATIBLE` = *pointers*	Aspects of the options and parameters that must be compatible with the first parameter
`PRESENT` = *texts*	Whether the options and parameters must have values
`LIST` = *texts*	Whether the options have more than one setting (not relevant for the parameters
`INPUT` = *texts*	Whether the options and parameters only supply input information
`DEFINITION` = *texts*	Saves statements to define the syntax

SOURCE = *texts* — Saves the source code of a procedure

TABINSERT procedure

Inserts the contents of a sub-table into a table (R.W. Payne).

Options

OLDTABLE = *tables* — Table containing the original values
SUBTABLE = *tables* — Sub-table to insert into the original table
NEWTABLE = *tables* — Tables to store the new values; if this is not set, these replace those in the original table

Parameters

OLDFACTOR = *factors* — Factors classifying the dimensions of the old table that are smaller in the sub-table
SUBFACTOR = *factors* — Specifies the factors classifying the corresponding dimensions of the sub-table
FREPRESENTATION = *string token* — How to match the values of each OLDFACTOR and SUBFACTOR (levels, labels); default leve

TABLE directive

Declares one or more table data structures.

Options

CLASSIFICATION = *factors* — Factors classifying the tables; default *
MARGINS = *string token* — Whether to add margins (yes, no); default no
VALUES = *numbers* — Values for all the tables; default *
MODIFY = *string token* — Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens* — Information to be used by default to identify the tables in output (identifier, extra, associatedidentifier); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers* — Identifiers of the tables
VALUES = *identifiers* — Values for each table
DECIMALS = *scalars* — Number of decimal places for printing
EXTRA = *texts* — Extra text associated with each identifier
UNKNOWN = *identifiers* — Identifier for scalar to hold summary of unclassified data associated with each table
MINIMUM = *scalars* — Minimum value for the contents of each structure
MAXIMUM = *scalars* — Maximum value for the contents of each structure
DREPRESENTATION = *scalars* or *texts* — Default format to use when the contents represent dates and times
DATAVARIATE = *variates* — Records the identifier of the variate whose summaries are in the table
SUMMARYTYPE = *string tokens* — Records the type of summary that the table contains (counts, totals, nobservations, means, minima, maxima, variances, quantiles, sds, skewness, kurtosis, semeans, seskewness, sekurtosis); default * i.e. not recorded
PERCENTQUANTILE = *scalars* — Records the percentage points for which quantiles have been formed; default * i.e. not recorded
%MARGIN = *pointers* — Records the factors defining the margin over which the table has been converted to percentages

TABMODE procedure

Forms summary tables of modes of values (R.W. Payne).

Options

PRINT = *string token* — Controls whether or not the modes are printed (mode); default

	* i.e. no printing
CLASSIFICATION = *factors*	Factors classifying the tables; if unset, the overall mode is formed for all the values in each DATA vector

Parameters

DATA = *variates* or *factors*	Data values whose modes are to be formed
MODES = *tables* or *scalars*	Save the modes for each DATA vector

TABSORT procedure

Sorts tables so their margins are in ascending or descending order (R.W. Payne).

Options

PRINT = *string tokens*	Controls output (tables, histograms); default * i.e. none
DIRECTION = *string token*	Direction of sorting (ascending, descending); default asce
METHOD = *string token*	Method to use to construct a marginal table for the sorting of a factor when there is no one-way table classified by the factor in the TABLE list, and the first table in the TABLE list classified by the factor has no margins (totals, means, minima, maxima, variances, medians); default tota
FACTORS = *pointer*	Specifies or saves a list of classifying factors of the tables in the TABLE list
NEWFACTORS = *pointer*	Specifies or saves a list of classifying factors of the new tables, corresponding to those in the FACTORS pointer
EXCLUDE = *pointer*	Factors to exclude from sorting
NBEST = *string tokens*	Number of (best) levels to include from each sorted factor; default * i.e. all of them

Parameters

TABLE = *tables*	Tables to be sorted
NEWTABLE= *tables*	Allows the new sorted tables to be saved
TITLE = *texts*	Title to be used when displaying each table
FIELDWIDTH = *scalars*	Field width for printing each table
DECIMALS = *scalars*	Decimal places for each table

TABTABLE procedure

Opens a tabbed-table spreadsheet in the Genstat client, PC Windows only (D.B. Baird).

Options

IDENTIFIER = *identifier*	Identifier for the combined table when several tables are specified by TABLE
PAGEFACTOR = *factor*	Specifies the the classifying factor to go across the tabs in the spreadsheet when TABLE is set to a single table, or gives the identifier of the factor to be created to index the tables when TABLE supplies several tables

Parameter

TABLE = *tables*	Tables to be placed into a tabbed-table spreadsheet

TABULATE directive

Forms summary tables of variate values.

Options

PRINT = *string tokens*	Printed output required (counts, totals, nobservations, means, minima, maxima, variances, quantiles, sds, skewness, kurtosis, semeans, seskewness, sekurtosis); default * i.e. no printing
CLASSIFICATION = *factors*	Factors classifying the tables; default * i.e. these are taken from the tables in the parameter lists
COUNTS = *table*	Saves a table counting the number of units with each factor combination; default *
SEQUENTIAL = *scalar*	Used for sequential formation of tables; a positive value

	indicates that formation is not yet complete (see READ); default *
MARGINS = *string token*	Whether the tables should be given margins if not already declared (yes, no); default no
IPRINT = *string token*	Whether to print the identifier of the table or the identifier of the (associated) variate that was used to form it (identifier, extra, associatedidentifier); default iden
WEIGHTS = *variate*	Weights to be used in the tabulations; default * indicates that all units have weight 1
PERCENTQUANTILES = *scalar* or *variate*	Percentage points for which quantiles are required; default 50 (i.e. median)
OWN = *scalar* or *variate*	Specifies option settings for the OWNTAB subroutine and indicates that this is to supply the data values instead of the variates in the DATA list; default *
OWNFACTORS = *factors*	Factors whose values are to be read by OWNTAB (must include the factors of the classification set); default *
OWNVARIATES = *variates*	Variates whose values are to be read by OWNTAB (must include the DATA variates); default *
INCHANNEL = *scalar*	Channel number of the file from which the OWNTAB subroutine is to read the data (previously opened by an OPEN statement)
INFILETYPE = *string token*	Type of the OWN data file (input, unformatted); default inpu

Parameters

DATA = *variates*	Data values to be tabulated
TOTALS = *tables*	Tables to contain totals
NOBSERVATIONS = *tables*	Tables containing the numbers of non-missing values in each cell
MEANS = *tables*	Tables of means
MINIMA = *tables*	Tables of minimum values in each cell
MAXIMA = *tables*	Tables of maximum values in each cell
VARIANCES = *tables*	Tables of cell variances
QUANTILES = *tables* or *pointers*	Table to contain quantiles at a single PERCENTQUANTILE or pointer of tables for several PERCENTQUANTILEs (not available for sequential or OWN tabulation)
SDS = *tables*	Tables of standard deviations
SKEWNESS = *tables*	Tables of skewness coefficients
KURTOSIS = *tables*	Tables of kurtosis coefficients
SEMEANS = *tables*	Tables of standard errors of means
SESKEWNESS = *tables*	Tables of standard errors of skewness coefficients
SEKURTOSIS = *tables*	Tables of standard errors of kurtosis coefficients

TALLY procedure

Forms a simple tally table of the distinct values in a vector (D.B. Baird & R.D. Stern).

Options

PRINT = *string tokens*	What to print out for each vector (frequencies, percentages, cumfrequencies, cumpercentages, cumgraph, all); default freq, perc
GRAPH = *string tokens*	What to display as graphs (cumulative, %cumulative); default * i.e. no graphs
NGROUPS = *scalar*	Number of groups to form from a DATA variate or factor (ignored for texts); default * forms a group for each distinct value allowing for rounding (see DECIMALS)
DECIMALS = *scalar*	Number of decimal places to which to round the DATA before forming the groups; default * i.e. no rounding
BOUNDARIES = *string token*	Whether to interpret the LIMITS as upper or lower boundaries

(upper, lower); default lowe

DIRECTION = *string token* — Order in which to sort (ascending, descending); default asce

OMITEMPTY = *string token* — Whether empty groups are omitted (yes, no); default no

WEIGHTS = *variate* — Weights to be used in the tabulations; default * indicates that all units have weight 1

PQUANTILES = *string token* — Whether to include quantiles on the plot (yes, no); default no

WINDOW = *scalar* — Window in which to plot the graphs; default 1 if GROUPS is set, or 3 otherwise

KEYWINDOW = *scalar* — Window in which to display the key when GROUPS is set; default 2

SCREEN = *string token* — Whether to clear screen before the plot (clear, keep); default clea

Parameters

DATA = *variates*, *factors* or *texts* — Data to be tallied

GROUPS = *factors* — Defines groupings of the data, to be tallied into separate tables; default * i.e. none

LIMITS = *variates* or *texts* — Limits to define the groups within the tally tables

FREPRESENTATION = *string tokens* — Specifies the representation used to define the sort order of a DATA factor (ordinals, levels, labels); default leve

VALUES = *variates*, *texts* or *pointers* — Saves the distinct groups formed for the tally tables

FREQUENCIES = *variates* or *pointers* — Saves the frequencies of the groups in the tally tables

PERCENTAGES = *variates* or *pointers* — Saves the percentage occurrences of the groups

CUMFREQUENCIES = *variates* or *pointers* — Saves the cumulative frequencies of the groups

CUMPERCENTAGES = *variates* or *pointers* — Saves the cumulative percentages of the groups

TITLE = *texts* — Title for plot; default automatically forms a title containing the identifiers of the DATA vector and any GROUPS factor

XTITLE = *texts* — Title for the axis representing data values; default uses the identifier of the DATA vector

TDISPLAY directive

Displays further output after an analysis by ESTIMATE.

Options

PRINT = *string tokens* — What to print (model, summary, estimates, correlations); default mode,summ,esti

CHANNEL = *scalar* — Channel number for output; default * i.e. current output channel

SAVE = *identifier* — Save structure to supply fitted model; default * i.e. that from the last model fitted

No parameters

TENSORSPLINE procedure

Calculates design matrices to fit a tensor-spline surface as a linear mixed model (S.J. Welham & P.H.C. Eilers).

Options

METHOD = *string token* — Type of spline to use to construct the basis (pspline, penalizedspline); default pspl

PENALTYMETHOD = *string token* — Which tensor-spline penalty to use (isotropic, semiconstrained, unconstrained); default unco

NX1SEGMENTS = *scalar* — Specifies the number of segments between boundaries in the X1 dimension; default * obtains a value automatically

NX2SEGMENTS = *scalar* — Specifies the number of segments between boundaries in the X2 dimension; default * obtains a value automatically

DEGREE = *scalar* — Degree of polynomial used to form the underlying spline basis

	functions; default 1 for METHOD=pena and 3 for METHOD=pspl
DIFFORDER = *scalar*	Differencing order for P-spline penalty; default 2
X1LOWER = *scalar*	Specifies the lower boundary in the X1 dimension; default takes the minimum value of X1
X1UPPER = *scalar*	Specifies the upper boundary in the X1 dimension; default takes the maximum value of X1
X2LOWER = *scalar*	Specifies the lower boundary in the X2 dimension; default takes the minimum value of X2
X2UPPER = *scalar*	Specifies the upper boundary in the X2 dimension; default takes the maximum value of X2
ORTHOGONALIZATION = *string token*	How to orthogonalize the random basis (fixed, none); default fixe
SCALING = *scalar*	Scaling of the XRANDOM terms (automatic, none); default auto

Parameters

X1 = *variates* or *factors*	Coordinates in the first dimension for which spline values are required
X2 = *variates* or *factors*	Coordinates in the second dimension for which spline values are required
XFIXED = *matrices*	Saves the design matrix to define the fixed terms (excluding the constant) for fitting the tensor spline
XRANDOM = *pointers*	Saves the design matrices to define the random terms for fitting the tensor spline
X1KNOTS = *variates*	Saves the coordinates in the first dimension of the internal knots used to form the basis for the spline
X2KNOTS = *variates*	Saves the coordinates in the second dimension of the internal knots used to form the basis for the spline
PX1 = *variates*	Specifies the coordinates in the first dimension at which to predict
PX2 = *variates*	Specifies the coordinates in the second dimension at which to predict
PFIXED = *matrices*	Saves the design matrix for the fixed terms (excluding the constant) for the tensor spline at the prediction points
PRANDOM = *pointers*	Saves the design matrices for the random terms for the tensor spline at the prediction points

†TEQUIVALENCE procedure

Performs equivalence, non-inferiority and non-superiority tests (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (confidence, description, test); default desc, test
PLOT = *string token*	Controls plotting of the confidence intervals (confidence); default *
CLASSIFICATION = *pointer*	Specifies the factors classifying the table of means; must be supplied for a multi-way table
METHOD = *string token*	Type of test required (equivalence, noninferiority, nonsuperiority); default equi
CIPROBABILITY = *scalar*	The probability level for the confidence interval; default 0.95
EQLIMITS = *scalar* or *variate*	Limits for equivalence, non-inferiority or non-superiority
TITLE = *text*	Title for the graph of confidence intervals; default 'Confidence plot'
WINDOW = *scalar*	Window for the graph of confidence intervals; default uses a window defined to fill the screen
SCREEN = *string token*	Whether to clear the screen before plotting the confidence intervals (clear, keep); default clea

Parameters

MEANS = *tables* or *variates*	Means to be compared
CONTROL = *scalars*, *texts* or *pointers*	Specifies the control treatment
SED = *symmetric matrix* or *scalar*	Standard errors of differences of the means
DF = *symmetric matrix* or *scalar*	Degrees of freedom for the standard errors of differences
TSTATISTICS = *tables* or *variates*	Saves the t-statistics for the tests
PROBABILITIES = *tables* or *variates*	Saves the probabilities from the tests
DIFFERENCES = *tables* or *variates*	Saves the differences from the control
SEDCONTROL = *tables* or *variates*	Saves the standard errors for the differences from the control
DFCONTROL = *tables* or *variates*	Saves the degrees of freedom for the differences from the control
LOWER = *tables* or *variates*	Saves the lower limits of the confidence intervals
UPPER = *tables* or *variates*	Saves the upper limits of the confidence intervals

TERMS directive

Specifies a maximal model, containing all terms to be used in subsequent linear, generalized linear, generalized additive, and nonlinear models.

Options

PRINT = *string tokens*	What to print (correlations, wmeans, SSPM, monitoring); default *
FACTORIAL = *scalar*	Limit for expansion of model terms; default 3
FULL = *string token*	Whether to assign all possible parameters to factors and interactions (yes, no); default no
SSPM = *SSPM*	Gives sums of squares and products on which to base calculations; default *
TOLERANCE = *scalar*	Criterion for testing for linear dependence; default is $10^7\varepsilon$, where ε is the smallest real value such that $1+\varepsilon$ is greater than 1 on the computer
DESIGNMATRIX = *matrix*	Saves the design matrix for the maximal model
MVINCLUDE = *string token*	Whether to include units with missing values in the explanatory factors and variates (explanatory); default * i.e. omit these
RIDGE = *scalar* or *variate*	Supplies values to add to the diagonal of the sums-of-squares-and-products matrix, to enable ridge methods to be used; default 0
CLDESIGNMATRIX = *text*	Saves the column labels of the design matrix for the maximal model i.e. the names of the parameters estimated in the maximal model
CLSSP = *text*	Saves the labels of the sum-of-squares-and-products matrix

Parameter

formula	List of explanatory variates and factors, or model formula

TEXT directive

Declares one or more text data structures.

Options

NVALUES = *scalar* or *vector*	Number of strings, or vector of labels; default * takes the setting from the preceding UNITS statement, if any
VALUES = *strings*	Values for all the texts; default *
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used by default to identify the texts in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the texts
VALUES = *texts*	Values for each text

CHARACTERS = *scalars* — Numbers of characters of the lines of each text to be printed by default
EXTRA = *texts* — Extra text associated with each identifier

TFILTER directive

Filters time series by time-series models.

Option

PRINT = *string tokens* — What to print (series); default *

Parameters

OLDSERIES = *variates* — Time series to be filtered
NEWSERIES = *variates* — To save filtered series
FILTER = *TSMs* — Models to filter with respect to
ARIMA = *TSMs* — ARIMA models for time series

TFIT directive

Estimates parameters in Box-Jenkins models for time series.

Options

PRINT = *string tokens* — What to print (model, summary, estimates, correlations, monitoring); default mode,summ,esti
LIKELIHOOD = *string token* — Method of likelihood calculation (exact, leastsquares, marginal); default exac
CONSTANT = *string token* — How to treat the constant (estimate, fix); default esti
RECYCLE = *string token* — Whether to continue from previous estimation (yes, no); default no
WEIGHTS = *variate* — Weights; default *
MVREPLACE = *string token* — Whether to replace missing values by their estimates (yes, no); default no
FIX = *variate* — Defines constraints on parameters (ordered as in each model, tf models first): zeros fix parameters, parameters with equal numbers are constrained to be equal; default *
METHOD = *string token* — Whether to carry out full iterative estimation, to carry out just one iterative step, to perform no steps but still give parameter standard deviations, or only to initialize for forecasting by regenerating residuals (full, onestep, zerostep, initialize); default full
MAXCYCLE = *scalar* — Maximum number of iterations; default 15
TOLERANCE = *scalar* — Criterion for convergence; default 0.0004
SAVE = *identifier* — To name save structure, or supply save structure with transfer-functions; default * i.e. transfer-functions taken from the latest model

Parameters

SERIES = *variate* — Time series to be modelled (output series)
TSM = *TSM* — Model for output series
BOXCOXMETHOD = *string token* — How to treat transformation parameter in output series (fix, estimate); default fix
RESIDUALS = *variate* — To save residual series

TFORECAST directive

Forecasts future values of a time series.

Options

PRINT = *string tokens* — What to print (forecasts, limits, setransform, sfe); default fore,limi
CHANNEL = *scalar* — Channel number for output; default * i.e. current output channel
ORIGIN = *scalar* — Number of known values to be incorporated; default 0
UPDATE = *string token* — Whether to update the forecast origin to the end of the new

observations (yes, no); default no

NEWOBSERVATIONS = *variate* — Variate of length ≥ ORIGIN providing new values of the time series to be incorporated (must be set if ORIGIN > 0)

SFE = *variate* — Saves standardized forecast errors; default *

MAXLEAD = *scalar* — Maximum lead time i.e number of forecasts to be made; default * defines the number as the length of FORECAST variate

FORECAST = *variate* — Variate of length MAXLEAD to save forecasts of output series; default *

SETRANSFORM = *variate* — Saves standard errors of the forecasts (on transformed scale, if defined); default *

LOWER = *variate* — Saves lower confidence limits; default *

UPPER = *variate* — Saves upper confidence limits; default *

PROBABILITY = *scalar* — Probability level for confidence limits; default 0.9

COMPONENTS = *pointer* — Contains variates (of length ORIGIN + MAXLEAD) to save components of the forecast

SAVE = *identifier* — Save structure to supply fitted model; default * i.e. that from last model fitted

Parameters

FUTURE = *variates* — Variates (of length ORIGIN + MAXLEAD) containing future values of input series

METHOD = *string tokens* — How to treat future values of input series (observations, forecasts); default obse

THINPLATE procedure

Calculates the basis functions for thin-plate splines (D.B. Baird).

No options

Parameters

Y = *variates* or *factors* — Y-coordinates of the data points

X = *variates* or *factors* — X-coordinates of the data points

YKNOTS = *variates* or *factors* — Y-coordinates of the knots

XKNOTS = *variates* or *factors* — X-coordinates of the knots

TPSPLINE = *variates* or *matrices* — Thin-plate spline basis, as either a pointer of variates (default if not already declared) or a matrix

TKEEP directive

Saves results after an analysis by ESTIMATE.

Option

SAVE = *identifier* — Save structure to supply fitted model; default * i.e. that from last model fitted

Parameters

OUTPUTSERIES = *variate* — Output series to which model was fitted

RESIDUALS = *variate* — Residual series

ESTIMATES = *variate* — Estimates of parameters

SE = *variate* — Standard errors of estimates

INVERSE = *symmetric matrix* — Inverse matrix

VCOVARIANCE = *symmetric matrix* — Variance-covariance matrix of parameters

DEVIANCE = *scalar* — Residual deviance

DF = *scalar* — Residual degrees of freedom

MVESTIMATES = *variate* — Estimates of missing values in series

SEMV = *variate* — Standard errors of estimates of missing values

COMPONENTS = *pointer* — Variates to save components of output series

SCORES = *variate* — To save scores (derivatives of the log-likelihood with respect to the parameters)

†TOBIT procedure

Performs a Tobit linear mixed model analysis on data with fixed-threshold censoring (M.C. Hannah & V.M. Cave).

Options

PRINT = *string token*	Controls printed output (summary); default summ
VPRINT = *string tokens*	Controls printed output from the REML analysis of the data with censored observations replaced by their estimates (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues); default mode, comp, Wald
PSE = *string token*	Standard errors to be printed with tables of effects and means from the REML analysis (differences, estimates, alldifferences, allestimates, none); default diff
PLOT = *string token*	To display a scatter plot of the data with censored observations replaced by their estimates against the observed data(scatterplot); default *
MAXCYCLE = *scalar*	Sets a limit on the number of iterations performed by the E-M algorithm; default 30
TOLERANCE = *variate*	Sets tolerance limits for convergence of the E-M algorithm on the treatment means and the variance components; default 0.1 and 0.05 for the treatment means and variance components, respectively
RMETHOD = *string token*	Which random terms to use when calculating the residuals during the E-step of the E-M algorithm (final, all); default final
DIRECTION = *string token*	The direction of the censoring (left, right); default left (i.e., the true values for the censored observations are less than or equal to the BOUND)

Parameters

Y = *variate*	Response variate to be analysed; no default, must be set
BOUND = *scalar*	Censoring threshold; no default, must be set
CENSORED = *variate*	Indicator variable for censored observations, with values of one where the response values are censored and zero otherwise
INITIAL = *scalar* or *variate*	Scalar or a variate providing starting values for the censored observations in the E-M algorithm
NEWY = *variate*	Saves a copy of the response variate with the censored observations replaced by their estimates
YCENSORED = *variate*	Saves a logical variate indicating which Y values are censored
SAVE = *REML save structure*	REML save structure from the analysis of the data with censored observations replaced by their estimates

TRANSFERFUNCTION directive

Specifies input series and transfer-function models for subsequent estimation of a model for an output series.

Option

SAVE = *identifier*	To name time-series save structure; default *

Parameters

SERIES = *variates*	Input time series
TRANSFERFUNCTION = *TSMs*	Transfer-function models; if omitted, model with 1 moving-average parameter, lag 0
BOXCOXMETHOD = *string tokens*	How to treat transformation parameters (fix, estimate); default fix
PRIORMETHOD = *string tokens*	How to treat prior values (fix, estimate); default fix
ARIMA = *TSMs*	ARIMA models for input series

TREATMENTSTRUCTURE directive

Specifies the treatment terms to be fitted by subsequent ANOVA statements.

No options

Parameter

formula	Treatment formula, specifies the treatment model terms to be fitted by subsequent ANOVAs

TREE directive

Declares one or more tree data structures and initializes each one to have a single node known as its root.

No options

Parameter

IDENTIFIER = *identifiers*	Identifiers of the trees

TRELLIS procedure

Does a trellis plot (S.J. Welham & S.A. Harding).

Options

GROUPS = *factors* or *variate*	Factors or variate defining the classification for the plots
GMETHOD = *string token*	Determines the method used to partition the range when GROUPS is set to a variate (equalspacing, quantiles, distinct, limits); default equal
NGROUPS = *scalar*	Determines the number of plots to be formed when GROUPS is set to a variate and GMETHOD is set to quantiles or equalspacing
LIMITS = *variate*	Limits to use to form groups from a GROUPS variate when GMETHOD=limits
OVERLAP = *scalar*	Proportion by which a GROUPS variate should overlap between plots (scalar in range 0 - 0.5); default 0
OMITEMPTY = *string token*	Whether to omit all empty plots from the array (all), or omit levels of a GROUPS factor where all plots are empty (levels), or keep all plots in the array (none); default level
PENGROUP = *factors*	Defines factor combinations to be plotted in different colours, note that the number of colours available may differ between devices
NROWS = *scalar*	Specifies number of rows of plots to appear on one page; default determined automatically from GROUPS
NCOLUMNS = *scalar*	Specifies number of columns of plots to appear on one page; default determined automatically from GROUPS
TITLE = *text*	Supplies a title for the plot
FIRSTPICTURE = *string token*	Whether to put the first picture at bottom or top left of the grid (bottomleft, topleft); default topl
TMETHOD = *string token*	Whether to give plot titles as factor names with labels or just labels (names, labels); default names
YTITLE = *text*	Supplies an overall y-axis title
XTITLE = *text*	Supplies an overall x-axis title
YMARGIN = *scalar*	Relative size of margins for the y-axis labels on individual plots; default 0.04
XMARGIN = *scalar*	Relative size of margins for the x-axis labels on individual plots; default 0.04
TMARGIN = *scalar*	Relative size of margin for titles of individual plots; default 0.04
PENSIZE = *scalar*	Proportionate adjustment to the pen size for individual plot titles and axis labels; default 1
USEPENS = *string token*	Whether to use current pen definitions in the procedure (no, yes); default no
USEAXES = *string token*	Which aspects of the current axis definitions of window 1 to

	use (`none`, `limits`, `style`, `marks`, `mpositions`, `nsubticks`, `transform`); default `none`
`NRMAX` = *scalar*	Maximum number of rows on page; default 8 for a square frame, 7 for a landscape frame and 10 for a portrait frame
`NCMAX` = *scalar*	Maximum number of columns on page; default 8 for a square frame, 10 for a landscape frame and 7 for a portrait frame
`KEYHEIGHT` = *scalar*	Space in y-direction to use for key (0 to suppress key); default `*` i.e. determined automatically
`YPENMETHOD` = *string token*	Whether to use the same or different pens for each y-variate (`different`, `same`); default `diff`
`FRAMESHAPE` = *string token*	Shape of the plotting frame (`landscape`, `portrait`, `square`); default `squa`

Parameters

`Y` = *variates*	Y-values of the data to be plotted
`X` = *variates* or *factors*	X-values of the data to be plotted
`METHOD` = *string tokens*	Type of plot (`point`, `line`, `mean`, `median`, `histogram`, `boxplot`, `spline`, `schematicboxplot`); default `poin`
`DESCRIPTION` = *texts*	Annotation for key

`TRY` directive

Displays results of single-term changes to a linear, generalized linear or generalized additive model.

Options

`PRINT` = *string tokens*	What to print (`model`, `deviance`, `summary`, `estimates`, `correlations`, `fittedvalues`, `accumulated`, `monitoring`, `changes`, `confidence`); default `chan`
`FACTORIAL` = *scalar*	Limit for expansion of model terms; default `*` i.e. that in previous `TERMS` statement
`POOL` = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (`yes`, `no`); default `no`
`DENOMINATOR` = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (`ss`, `ms`); default `ss`
`NOMESSAGE` = *string tokens*	Which warning messages to suppress (`dispersion`, `leverage`, `residual`, `aliasing`, `marginality`, `vertical`, `df`, `inflation`); default `*`
`FPROBABILITY` = *string token*	Printing of probabilities for variance and deviance ratios (`yes`, `no`); default `no`
`TPROBABILITY` = *string token*	Printing of probabilities for t-statistics (`yes`, `no`); default `no`
`SELECTION` = *string tokens*	Statistics to be displayed in the summary of analysis produced by `PRINT=summary`, `seobservations` is relevant only for a Normally distributed response, and `%cv` only for a gamma-distributed response (`%variance`, `%ss`, `adjustedr2`, `r2`, `seobservations`, `dispersion`, `%cv`, `%meandeviance`, `%deviance`, `aic`, `bic`, `sic`); default `%var`, `seob` if `DIST=normal`, `%cv` if `DIST=gamma`, and `disp` for other distributions
`PROBABILITY` = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95

Parameter

formula	List of explanatory variates and factors, or model formula

`TSM` directive

Declares one or more TSM data structures.

Option

`MODELTYPE` = *string token*	Type of model (`arima`, `transfer`); default `arim`

Parameters

IDENTIFIER = *identifiers*	Identifiers of the TSMs
ORDERS = *variates*	Orders of the autoregressive, integrated, and moving-average parts of each TSM
PARAMETERS = *variates*	Parameters of each TSM
LAGS = *variates*	Lags, if not default

TSUMMARIZE directive

Displays characteristics of time series models.

Options

PRINT = *string tokens*	What to print (autocorrelations, expansion, impulse, piweight, psiweight); default *
GRAPH = *string tokens*	What to display with graphs (autocorrelations, impulse, piweight, psiweight); default *
MAXLAG = *scalar*	Maximum lag for results; default 30

Parameters

TSM = *TSMs*	Models to be displayed
AUTOCORRELATIONS = *variates*	To save theoretical autocorrelations
IMPULSERESPONSE = *variates*	To save impulse-response function
STEPFUNCTION = *variates*	To save step function from impulse
PIWEIGHTS = *variates*	To save pi-weights
PSIWEIGHTS = *variates*	To save psi-weights
EXPANSION = *TSMs*	To save expanded models
VARIANCE = *scalars*	To save variance of each TSM

TTEST procedure

Performs a one- or two-sample t-test (S.J. Welham).

Options

PRINT = *string tokens*	Controls printed output (confidence, summary, test, variance, permutationtest); default conf, summ, test, vari
†METHOD = *string token*	Type of test required (twosided, greaterthan, lessthan, equivalence, noninferiority, nonsuperiority); default twos
GROUPS = *factor*	Defines the groups for a two-sample test if only the Y1 parameter is specified
CIPROBABILITY = *scalar*	The probability level for the confidence interval; for a one-sided test this will be for the mean and for a two-sided test for the difference in means; default *, i.e. no confidence interval is produced
NULL = *scalar*	The value of the mean under the null hypothesis; default 0
VMETHOD = *string token*	Selects between the standard two-sample t-test, with a pooled estimate of the variances of the samples, and the use of separate estimates for the sample variances (automatic, pooled, separate); default auto uses a pooled estimate unless there is evidence of unequal variances
NTIMES = *scalar*	Number of random allocations to make when PRINT=perm; default 999
SEED = *scalar*	Seed for the random number generator used to make the allocations; default 0 continues from the previous generation or (if none) initializes the seed automatically
†EQLIMITS = *scalar* or *variate*	Limits for equivalence, non-inferiority or non-superiority

Parameters

Y1 = *variates*	Identifier of the variate holding the first sample
Y2 = *variates*	Identifier of the variate holding the second sample
TESTRESULTS = *variates*	Identifier of variate (length 3) to save test statistic, d.f. and

probability value

LOWER = *scalars*	Identifier of scalar to save the lower limit of each confidence interval
UPPER = *scalars*	Identifier of scalar to save the upper limit of each confidence interval
W1 = *variates*	Weights (replications) of the values in Y1; default * i.e. all 1
W2 = *variates*	Weights (replications) of the values in Y2; default * i.e. all 1

TUKEYBIWEIGHT procedure

Estimates means using the Tukey biweight algorithm (D.B. Baird).

Options

CUTPOINT = *scalar*	Cut point after which weight is set to zero; default 5
TOLERANCE = *scalar*	Tolerance to avoid division by zero; default 0.00001

Parameters

DATA = *variates* or *pointers*	Data values
GROUPS = *factors*	Groupings of the data values
MEANS = *variates*	Saves the means
SE = *variates*	Saves standard errors

TVARMA procedure

Fits a vector autoregressive moving average (VARMA) model (A.I. Glaser).

Options

PRINT = *string tokens*	What to print (model, summary, estimates, correlations); default mode, summ, esti
LIKELIHOOD = *string token*	Method of likelihood calculation (exact, conditional); default exac
CONSTANT = *string token*	How to treat the constant (estimate, fixtozero); default esti
ARMA = *variate*	Variate of length two, containing the number of AR and MA parameters respectively
ARFIXED = *pointer*	Specifies fixed values of the AR parameters
MAFIXED = *pointer*	Specifies fixed values of the MA parameters
MUFIXED = *variate*	Specifies fixed values of the constant parameters
NDIFFERENCING = *variate* or *scalar*	Specifies the order of differencing for each series; default 0
NCROSSRESIDUAL = *scalar*	Number of residual cross-correlation matrices to be computed for calculating the modified portmanteau statistic; default 20
MAXCYCLE = *scalar*	Maximum number of iterations; if this is not set, an appropriate default is determined automatically according to the number of parameters
TOLERANCE = *scalar*	Convergence criterion; default 0.0001

Parameters

SERIES = *pointers*	Time series to be modelled (output series)
RESIDUALS = *pointers*	Saves the residual series
ESTIMATES = *pointers*	Saves estimates of parameters for each SERIES variate
SEESTIMATES = *pointers*	Saves standard errors of the estimates
VCRESIDUALS = *symmetric matrices*	Variance-covariance matrix of the residuals
DEVIANCE = *scalars*	Saves the residual sum of squares or deviance
CORRELATIONS = *symmetric matrices*	Saves the correlation matrix of the estimates
GRADIENTS = *variates*	Saves the first derivative of the loglikelihood function
SAVE = *pointers*	Saves information for use with TVGRAPH or TVFORECAST

TVFORECAST procedure

Forecasts future values from a vector autoregressive moving average (VARMA) model (A.I. Glaser).

Options

PRINT = *string tokens*	What to print (forecasts, se); default fore, se
MAXLEAD = *scalar*	Maximum lead time i.e. number of forecasts to be made;

default 1

Parameters

FORECASTS = *matrices*	Saves the forecasts
SE = *matrices*	Saves standard errors of the forecasts
SAVE = *pointers*	Save structure from a previous TVARMA

TVGRAPH procedure

Plots a vector autoregressive moving average (VARMA) model (A.I. Glaser).

Options

TIMEPOINTS = *variate*	X-coordinates for the graphs; default uses the integers 1, 2...
TITLE = *texts*	Overall title for the graphs
YTITLE = *texts*	Titles for the y-axes; default * forms titles automatically from the identifiers or labels of the y-variables
XTITLE = *texts*	Title for the x-axis in each set of graphs; default * uses the identifier of TIMEPOINTS (if set)
NROWS = *scalar*	Specifies the number of rows of graphs to appear on the graphics screen; default * takes the number of y-variables
NCOLUMNS = *scalar*	Specifies the number of columns of graphs to appear on the graphics screen; default 1

Parameter

SAVE = *pointers*	Save structure from TVARMA with information about the analysis; default plots information from the most recent TVARMA analysis

TXBREAK directive

Breaks up a text structure into individual words.

Option

SEPARATOR = *text*	Defines the characters separating the words in the original text; default ' ,;:.'

Parameters

TEXT = *texts*	Text to break into words
WORDS = *texts*	Saves the words contained in each text (in the order in which they occur)
COLUMNS = *variates*	Saves the number of the column in the TEXT where each word began
LINES = *variates*	Saves the number of the line where each word was found
PLACESINLINES = *variates*	Saves the place of each word (first, second &c) within the line where it was found

TXCONSTRUCT directive

Forms a text structure by appending or concatenating values of scalars, variates, texts, factors, pointers or formulae; allows the case of letters to be changed or values to be truncated and reversed.

Options

TEXT = *text*	Stores the text that is formed
CASE = *string token*	Case to use for letters (given, lower, upper, changed, sentence, title); default give leaves the case of each letter as given in the original texts
METHOD = *string token*	Whether to append or concatenate the values of the structures (append, concatenate) default conc
SEPARATOR = *string*	Characters to separate all except last two strings in each line when concatenating; default '' (i.e. none)
LASTSEPARATOR = *string*	Characters to separate last two strings in each line when concatenating; default uses the charactors defined by SEPARATOR
PREFIX = *string*	Characters to put at the start of each line when concatenating; default '' (i.e. none)

END = *string*	Characters to put at the end of each line when concatenating; default '' (i.e. none)
SIGNIFICANTFIGURES = *scalar*	Specifies the number of significant figures to include for numerical data; default 4

Parameters

STRUCTURE = *scalars*, *variates*, *factors*, *texts*, *pointers* or *formulae*	Structures whose values are to be appended or concatenated
WIDTH = *scalars* or *variates*	Number of characters to take from the strings formed from the units of each STRUCTURE, a negative value takes all the (unskipped) characters other than trailing spaces; if omitted or set to a missing value, all the (unskipped) characters are taken
DECIMALS = *scalars* or *variates*	Number of decimal places to use for numerical structures; if omitted or set to a missing value, a default is used which aims to print the value to the precision defined by the SIGNIFICANTFIGURES option
SKIP = *scalars* or *variates*	Number of characters to skip at the left-hand side of the strings formed from the units of each STRUCTURE, a negative value skips all initial spaces; if omitted or set to a missing value, no characters are skipped
FREPRESENTATION = *string tokens*	How to represent factor values (labels, levels, ordinals); default is to use labels if available, otherwise levels
DREPRESENTATION = *scalars* or *texts*	Format to use for dates and times (stored in numerical structures)
REVERSE = *string tokens*	Whether to reverse the strings of characters formed from the units of each structure (yes, no); default no
MISSING = *texts*	String to use to represent missing values of numerical structures; default '*'

TXFIND directive

Finds a subtext within a text structure.

Options

CASE = *string token*	Whether to treat the case of letters (small or capital) as significant when searching for the SUBTEXT within the TEXT (significant, ignored); default sign
REVERSE = *string token*	Whether to reverse the search to work from the end of the TEXT (yes, no); default no
MULTISPACES = *string token*	Whether to treat differences between multiple spaces and single spaces as significant, or to treat them all like a single space (significant, ignored); default sign
DISTINCT = *string tokens*	Whether to require the SUBTEXT to have one or more separators to its left or right within the TEXT (left, right; default *
SEPARATOR = *string*	Characters to use as separators; default ' ,;:.'
SAMELINE = *string token*	Whether to ignore matches in the TEXT where the SUBTEXT is not all on the same line (yes, no); default no

Parameters

TEXT = *texts*	Texts to be searched
SUBTEXT = *texts*	Text to look for in each TEXT
COLUMN = *scalars*	Position of the column within TEXT where the first character of SUBTEXT has been found
LINE = *scalars*	Number of the line within TEXT where the first character of SUBTEXT has been found
ICOLUMN = *scalars*	Column within TEXT at which to start the search
ILINE = *scalars*	Line within TEXT at which to start the search
ENDCOLUMN = *scalars*	Position of the column within TEXT where the last character of

SUBTEXT has been found

ENDLINE = *scalars* — Number of the line within TEXT where the last character of SUBTEXT has been found

TXINTEGERCODES directive

Converts textual characters to and from their corresponding integer codes.

Options

CONVERTTO = *string token* — Whether to convert from text characters to integer codes or integer codes to text characters (codes, text) ; default code

REPRESENT = *string token* — How to treat code values 128-255 (extendedascii, utf8); default exte if CODES defines no characters that can be represented only in UTF-8, otherwise utf8

Parameters

TEXT = *texts* — Text structures (each with a single line only)

CODES = *variates* or *scalars* — Integer codes corresponding to the characters in each text

TXPAD procedure

Pads strings of a text structure with extra characters so that their lengths are equal (J.T.N.M. Thissen).

Options

PADDINGCHARACTERS = *string token* — Character(s) used for padding; default uses the dot character '.'

METHOD = *string token* — Whether the character(s) of PADDINGCHARACTERS should be placed before or after the strings of OLDTEXT (before, after); default afte

REMOVESPACES = *string tokens* — Whether to remove initial and/or trailing spaces in the strings of OLDTEXT (leading, trailing); default * i.e. none

Parameters

OLDTEXT = *texts* — Texts to be padded; must be set

NEWTEXT = *texts* — Saves the padded texts

WIDTH = *scalars* — Sets a limit on the length of the strings in the padded texts; default is the width of the largest string in OLDTEXT

TXPOSITION directive

Locates strings within the lines of a text structure.

Options

CASE = *string token* — Whether to treat the case of letters as significant when searching for lines of the SUBTEXT within the TEXT (significant, ignored); default sign

REVERSE = *string tokens* — Whether to reverse the search to work from the end of the lines of the TEXT (yes, no); default no

MULTISPACES = *string token* — Whether to treat differences between multiple spaces and single spaces as significant, or to treat them all like a single space (significant, ignored); default sign

DISTINCT = *string tokens* — Whether to require the SUBTEXT to have one or more separators to its left or right within the TEXT (left, right; default *

SEPARATOR = *text* — Characters to use as separators; default ' ,;:.'

Parameters

TEXT = *texts* — Texts whose strings are to be searched

SUBTEXT = *texts* — Specifies a string or strings to find in each TEXT

POSITION = *variates* — Position of the SUBTEXT strings within the TEXT

WIDTH = *scalars* or *variates* — Right-most character(s) to search in the lines of each TEXT; default * searches up to the end of each line

SKIP = *scalars* or *variates* — Number of characters to skip at the left-hand side of the lines of each TEXT; default 0

TXPROGRESSION procedure

Forms a text containing a progression of strings (R.W. Payne).

Options

INCLUDECHARACTERS = *string tokens*	Defines the set of characters to include in the progression (lower, upper, digits, _, %, space); default lowe
DIRECTION = *string token*	Direction of the progression (ascending, descending); default asce
FIRSTLETTERS = *string token*	Controls which letters come first (alllower, allupper, lower, upper); default uppe
OWNCHARACTERSET = *text*	Can supply an alternative set of characters

Parameters

FIRST = *texts*	Single-valued text specifying the first string in each progression
SECOND = *texts*	Single-valued text specifying the second string in each progression
LAST = *texts*	Single-valued text defining the end of each progression
PROGRESSION = *texts*	Saves the progression

TXREPLACE directive

Replaces a subtext within a text structure.

Options

NTIMES = *scalar*	Number of times to search for the OLDSUBTEXT and replace it; default 1
CASE = *string token*	Whether to treat the case of letters (small or capital) as significant when searching for the OLDSUBTEXT within the OLDTEXT (significant, ignored); default sign
MULTISPACES = *string token*	Whether to treat differences between multiple spaces and single spaces as significant when locating the OLDSUBTEXT within the OLDTEXT, or to treat them all like a single space (significant, ignored); default sign
DISTINCT = *string tokens*	Whether to require the OLDSUBTEXT to have one or more separators to its left or right within the OLDTEXT (left, right); default *
SEPARATOR = *string*	Characters to use as separators; default ' ,;:.'
SAMELINE = *string token*	Whether to ignore matches in the OLDTEXT where the OLDSUBTEXT is not all on the same line (yes, no); default no

Parameters

OLDTEXT = *texts*	Texts to be edited
NEWTEXT = *texts*	Texts with OLDSUBTEXT replaced by NEWSUBTEXT; if no NEWTEXT is supplied, the new values replace those in the corresponding OLDTEXT
OLDSUBTEXT = *texts*	Text to look for in each OLDTEXT
NEWSUBTEXT = *texts*	Text to replace OLDSUBTEXT
COLUMN = *scalars*	Position of the column within OLDTEXT where the first character of NEWSUBTEXT has been placed
LINE = *scalars*	Number of the line within OLDTEXT where the first character of NEWSUBTEXT has been placed
ICOLUMN = *scalars*	Column within OLDTEXT at which to start the search
ILINE = *scalars*	Line within OLDTEXT at which to start the search
ENDCOLUMN = *scalars*	Position of the column within OLDTEXT where the last character of NEWSUBTEXT has been placed
ENDLINE = *scalars*	Number of the line within OLDTEXT where the last character of NEWSUBTEXT has been placed
NREPLACED = *scalars*	Number of subtexts replaced

TXSPLIT procedure

Splits a text into individual texts, at positions on each line marked by separator character(s) (R.W. Payne).

Options

SEPARATOR = *text*	Defines the character(s) that indicate where to split each line; default ','
INCLUDE = *string tokens*	Whether to retain the separator at the end of a split text, or any spaces at its start and end (separators, spaces) ; default * i.e. include neither

Parameters

TEXT = *texts*	Text to split
SPLITTEXTS = *texts*	Saves the texts into which TEXT is split

TX2VARIATE directive

Converts text structures to variates.

Options

PRINT = *string token*	Controls printed output (conversions) ; default * (i.e. none)
NONNUMERIC = *string token*	How to treat non-numeric values (bestmatch, missing) default miss
YEAR = *scalar*	Year to use when calculating the day within year for the date formats that specify only months and days; default is to assume that this is any year that is not a leap year
REDEFINE = *string token*	Whether to allow a structure in the VARIATE list that has already been declared (e.g. as a text) to be redefined (yes, no); default no

Parameters

TEXT = *texts*	Text structures to convert
VARIATE = *variates*	Variate for each text, containing the numbers in each of its lines
DREPRESENTATION = *scalars*	Format to use for dates and times (stored in numerical structures)
MISSING = *texts*	Strings used to represent missing values in each text; default '*'
STATUS = *variates*	Code to indicate whether the number in each unit was read successfully (1), or with conversions (2), or unsuccessfully (0)

T%CONTROL procedure

Expresses tables as percentages of control cells (R.W. Payne).

Option

PRINT = *string token*	Controls printed output (percentages); default perc

Parameters

OLDTABLE = *tables*	Tables containing the original values
NEWTABLE = *tables*	Tables to store the percentage values
FACTOR = *factors* or *pointers*	Factor, or pointer of factors, with control levels
CONTROL = *scalars*, *vaiates*, *texts* or *pointers*	Identifies the control level or levels of each FACTOR (if more than one is specified for a factor, their mean is used); default uses the reference level

UNITS directive

Defines an auxiliary vector of labels and/or the length of any vector whose length is not defined when a statement needing it is executed.

Option

NVALUES = *scalar*	Default length for vectors

Parameter

variate or *text*	Vector of labels

UNSTACK procedure

Splits vectors into individual vectors according to levels of a factor (R.W. Payne).

Options

DATASET = *factor*	Factor identifying the unstacked data sets
IDSTACKED = *factors*	Factors identifying how the units of the unstacked data sets should be matched
IDUNSTACKED = *factors*	Factors defined to identify these units in the unstacked vectors
MVINCLUDE = *strings*	Which missing values to include (datasets, idstacked); default * i.e. none

Parameters

STACKEDVECTOR = *variates*, *factors* or *texts*	Vectors to be unstacked
DATASETINDEX = *scalars* or *texts*	Level or label of the DATASET factor indicating the group whose units are to be stored in the UNSTACKEDVECTOR; default takes the levels of DATASET one at a time (and then recycling this list to match the other parameters)
UNSTACKEDVECTOR = *variates*, *factors* or *texts*	Unstacked vectors

UTMCONVERSION procedure

Converts between geographical latitude and longitude coordinates and UTM eastings and northings (D.B. Baird).

Options

CONVERTTO = *string token*	Whether to convert to UTM eastings and northings from geographical latitude and longitude coordinates, or to geographical coordinates from UTM (geographical, utm); default utm
DATUM = *string token*	The datum to use when constructing the grid for eastings and northings (WGS84, NAD83, GRS80, OSGB36, WGS72, AUSTRALIAN1965, KRASOVSKY1940, NORTHAMERICAN1927, INTERNATIONAL1924, HAYFORD1909, CLARKE1880, CLARKE1866, AIRY1830, BESSEL1841, EVEREST1830); default WGS8
CENTRALMERIDIAN = *scalar*	Central meridian in degrees for the UTM coordinates
SINGLEZONE = *string token*	Whether to convert to easting and /northings in a single zone (yes, no); default no
EORIGIN = *string token*	False origin for easting; default 500000
NORIGIN = *string token*	False origin for northing; default 0

Parameters

LATITUDE = *scalar*s or *variates*	Latitudes
LONGITUDE = *scalars* or *variates*	Longitudes
DIRECTION = *scalars* or *variates*	Directions of the angles of latitude and longitude coordinates (NE, NW, SE, SW); default NE
EASTING = *scalars* or *variates*	UTM easting grid references
NORTHING = *scalars* or *variates*	UTM northing grid references
ZONE = *scalars* or *variates*	UTM zones

VABLOCKDESIGN procedure

Analyses an incomplete-block design by REML, allowing automatic selection of random and spatial correlation models (R.W. Payne).

Options

PRINT = *string tokens*	Controls what summary output is produced about the models (deviance, aic, bic, sic, dffixed, dfrandom, change, exit, best, description; default best, desc
PBEST = *string tokens*	Controls the output from the REML analysis with the best

	model (`model`, `components`, `effects`, `means`, `stratumvariances`, `monitoring`, `vcovariance`, `deviance`, `Waldtests`, `missingvalues`, `covariancemodels`, `aic`, `sic`, `bic`); default `*` i.e. none
`PTRY` = *string tokens*	Controls the output to present from the `REML` analysis used to try each model (`model`, `components`, `effects`, `means`, `stratumvariances`, `monitoring`, `vcovariance`, `deviance`, `Waldtests`, `missingvalues`, `covariancemodels`, `aic`, `sic`, `bic`); default `*` i.e. none
`FIXED` = *formula*	Fixed model terms; default `*` i.e. none
`RANDOM` = *formula*	Additional random model terms; default `*` i.e. none
`CONSTANT` = *string token*	How to treat the constant term (`estimate`, `omit`); default `esti`
`FACTORIAL` = *scalar*	Limit on the number of factors or covariates in each fixed term; default 3
`REPLICATES` = *factor*	Replicate factor
`BLOCKS` = *factor*	Block factor; no default (must be specified)
`ROWS` = *factor*	Row factor for spatial analysis
`COLUMNS` = *factor*	Column factor for spatial analysis
`ROWCOORDINATES` = *variate* or *factor*	Row coordinates for fitting trends and spatial models if the design is irregular; if unset, these are defined from the levels of the `ROWS` factor
`COLCOORDINATES` = *variate* or *factor*	Column coordinates for fitting trends and spatial models if the design is irregular; if unset, these are defined from the levels of the `COLUMNS` factor
`PLOTFACTOR` = *factor*	Factor numbering the plots in the design; if unset, a local factor is defined automatically
`PTERMS` = *formula*	Terms (fixed or random) for which effects or means are to be printed; default `*` implies all the fixed terms
`PSE` = *string token*	Standard errors to be printed with tables of effects and means (`differences`, `estimates`, `alldifferences`, `allestimates`, `none`); default `diff`
`MVINCLUDE` = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (`explanatory`, `yvariate`); default `*` i.e. omit units with missing values in either explanatory factors or variates or y-variates
`VCONSTRAINTS` = *string token*	Whether to constrain variance components to be positive (`none`, `positive`); default `none`
`RSTRATEGY` = *string token*	Strategy for selecting the random model (`all`, `allfeasible`, `optimal`, `automatic`, `full`); default `allf`
`METHOD` = *string token*	Criterion to choose the best random model (`aic`, `sic`, `bic`); default `sic`
`TRYSPATIAL` = *string token*	Whether to try spatial models (`always`, `ifregular`); default `*` i.e. no spatial models
`TRYTRENDS` = *string token*	Whether to see whether row and column trends are needed in the fixed model (`yes`, `no`); default `no`
`SPATIALFACTOR` = *factor*	Factor to use to define the term for a 2-dimensional power-distance model; if unset, a local factor is defined automatically

Parameters

`Y` = *variates*	Response variates
`BESTMODEL` = *pointers*	Saves a model-definition structure for the best model for each y-variate
`EXIT` = *scalars*	Exit status of the best model for each y-variate
`SAVE` = *REML save structures*	Save structure from the analysis of the best model for each y-variate

VAIC procedure

Calculates the Akaike and Schwarz (Bayesian) information coefficients for REML (R.W. Payne & V.M. Cave).

Options

PRINT = *string tokens*	Controls printed output (deviance, aic, bic, sic, dffixed, dfrandom, changes); default aic
INCLUDE = *string tokens*	When LMETHOD=residual, which constants to include that depend only on the fixed model (determinant, pi); default pi
DMETHOD = *string token*	Method to use to calculate log(determinant(X′X)) (choleski, lrv); default chol
LMETHOD = *string token*	Whether the residual or full log-likelihood is used to calculate the information coefficients (residual, full); default resi
REPEAT = *string token*	Whether to repeat output from the previous VAIC (yes, no); default no

Parameters

DEVIANCE = *scalars*	Saves the deviance
AIC = *scalars*	Saves the Akaike information coefficient
SIC = *scalars*	Saves the Schwarz (Bayesian) information coefficient
DFFIXED = *scalars*	Saves the number of parameters fitted in the fixed model
DFRANDOM = *scalars*	saves the number of parameters fitted in the random model (and any covariance models)
CHANGES = *variates*	Saves changes since the previous VAIC; the units of the variates are labelled by the names of the coefficients (deviance, aic, sic, dffixed and dfrandom)
SAVE = *REML save structures*	Save structure for which to calculate the coefficients; default uses the save structure from the most recent REML

VALLSUBSETS procedure

Fits all subsets of the fixed terms in a REML analysis (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (results); default resu
FORCED = *formula*	Terms to include in every model
FACTORIAL = *scalar*	Limit for expansion of FORCED terms; default 3
SELECTION = *string tokens*	One or two criteria to be printed with the models (r2, adjusted, cp, ep, aic, sic, bic, rss, rms); default aic, sic
NBESTMODELS = *scalar*	Number of models to print; default * i.e. all
BESTMODEL = *pointer*	Saves the best model according to the selected criteria
RESULTS = *pointer*	Pointer to save variates containing the criteria for the sets, and F and Wald statistics for the terms that they contain
MARGINALTERMS = *string token*	How to treat terms that are marginal to other terms (forced, free); default forc
SAVE = *REML save structure*	Specifies the analysis whose fixed terms are to be tested; by default this will be the most recent REML

No parameters

VALINEBYTESTER procedure

Provides combinabilities and deviances for a line-by-tester trial analysed by VABLOCKDESIGN or VAROWCOLUMNDESIGN (R.W. Payne).

Options

PRINT = *string tokens*	Controls what summary output is produced about the models (combinability, tests); default comb, test
LINES = *factor*	Specifies the line (usually female parent); no default (must be specified)
TESTERS = *factor*	Specifies the tester (usually male parent); no default (must be

specified)

CONTROLS = *factor* — Distinguishes between control and test (line × tester) genotypes; default is that there are no controls

PCOMBINABILITYTERMS = *formula* — Terms whose combinability effects are to be printed (LINES and/or LINES.TESTERS; default is to print both of them

MVINCLUDE = *string tokens* — When the SAVE parameter is unset, this specifies whether to include units with missing values in the explanatory factors and variates and/or the y-variates in the analyses (explanatory, yvariate); default * i.e. omit units with missing values in either explanatory factors or variates or y-variates

Parameters

Y = *variates* — Response variates

MODELSTRUCTURE = *pointers* — Model-definition structure used for the analysis of each y-variate

COMBINABILITY = *pointers* — Pointer to tables of combinability effects for each y-variate

SECOMBINABILITY = *pointers* — Pointer to tables of standard errors of combinability effects for each y-variate

DEVIANCES = *variates* — Saves deviances for LINES and LINES.TESTERS

SAVE = *REML save structures* — Save structure from the analysis of each y-variate

VAMETA procedure

Performs a REML meta analysis of a series of trials, , previously analysed by VASERIES (R.W. Payne).

Options

PRINT = *string tokens* — Controls printed output (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic); default mode, comp, Wald

PTRY = *string tokens* — Controls the output to present from the REML analysis used to try each model (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic); default * i.e. none

PRECOVERY = *string tokens* — Controls what summary output is produced about the models that are tried during recovery (deviance, aic, bic, sic, dffixed, dfrandom, change, exit, best); default devi, aic, sic, dfra, best

FIXED = *formula* — Fixed model terms; if unset, these are taken from the MODELSTRUCTURES

RANDOM = *formula* — Additional random model terms; default * i.e. none

CONSTANT = *string token* — How to treat the constant term (estimate, omit); default esti

FACTORIAL = *scalar* — Limit on the number of factors or covariates in each fixed term; default 3

PTERMS = *formula* — Terms (fixed or random) for which effects or means are to be printed; default * implies all the fixed terms

PSE = *string token* — Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default diff

RECOVER = *string token* — Whether to try to recover with a simpler random model if REML cannot fit the model (yes, no); default no

METHOD = *string token* — How to choose the best model during recovery (aic, sic, bic); default sic

Parameters

Y = *variates* — Response variates

MODELDEFINITIONS = *pointers*	Descriptions of the models for each y-variate, saved from VASERIES
EXIT = *scalars*	Exit status for the fit (zero if successful)
SAVE = *vsaves*	REML save structure from the analysis of each y-variate

VAOPTIONS procedure

Defines options for the fitting of models by VARANDOM and associated procedures (R.W. Payne).

Options

MAXCYCLE = *scalar*	Limit on the number of iterations in REML analyses; default 100
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm
MINSPATIALCOORDINATES = *scalar*	Minimum number of different coordinates in a direction for a spatial model to be fitted by VAROWCOLUMNDESIGN; default 4
LIMPRTREND = *scalar*	Critical value for the probability of a row or column trend in the initial basic REML analysis (with replicates but no other random terms) for this to be included in the later analyses) by VAROWCOLUMNDESIGN; default 0.01
REPORTFAILURES = *string token*	Whether the accumulated summary should include models that fail to fit or that have bound variance parameters (yes, no) ; default no

No parameters

VARANDOM procedure

Finds the best REML random model from a set of models defined by VFMODEL (R.W. Payne).

Options

PRINT = *string tokens*	Controls what summary output is produced about the models (best, deviance, aic, bic, sic, dffixed, dfrandom, change, exit); default devi, aic, sic, dfra, best
PBEST = *string tokens*	Controls the output from the REML analysis with the best model (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic); default * i.e. none
PTRY = *string tokens*	Controls the output to present to present from the REML analysis used to try each model (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic); default * i.e. none
MODELSTRUCTURES = *pointer*	Model-definition structures specifying the models to try
PTERMS = *formula*	Terms (fixed or random) for which effects or means are to be printed; default * implies all the fixed terms
PSE = *string token*	Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default diff
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default * i.e. omit units with missing values in either explanatory factors or variates or y-variates
METHOD = *string token*	How to choose the best model (aic, sic, bic); default sic

Parameters

Y = *variates*	Response variates
NBESTMODEL = *scalars*	Saves the number of the best model for each y-variate, returning a missing value if no models could be fitted successfully

SAVE = *REML save structures* — Save structure from the analysis of the best model for each y-variate

VARECOVER procedure

Recovers when REML, is unable to fit a model, by simplifying the random model (R.W. Payne).

Options

PRINT = *string tokens* — Controls what summary output is produced about the simpler random models that are tried (deviance, aic, bic, sic, dffixed, dfrandom, change, exit, best); default devi, aic, sic, dfra, best

PBEST = *string tokens* — Controls the output from the REML analysis with the best simpler model (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic); default * i.e. none

PTRY = *string tokens* — Controls the output to present to present from the REML analysis used to try each model (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic); default * i.e. none

PLOTFACTOR = *factor* — Factor numbering the plots in the design, required if VARECOVER needs to try a null random model; if unset, a local factor is defined automatically

FORCED = *formula* — Specifies terms that must not be removed from the random model; by default any of the random terms can be removed

PTERMS = *formula* — Terms (fixed or random) for which effects or means are to be printed; default * implies all the fixed terms

PSE = *string token* — Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default diff

MVINCLUDE = *string tokens* — Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default * i.e. omit units with missing values in either explanatory factors or variates or y-variates

METHOD = *string token* — Criterion to choose the best model (aic, sic, bic); default sic

PROHIBIT = *string token* — Whether to exclude models where any estimated variance parameters are held at a bound (bound); default *

Parameters

Y = *variates* — Response variates

MODELSTRUCTURE = *pointers* — Model-definition structure for the unsuccessful analysis of each y-variate

BESTMODEL = *pointers* — Saves a model-definition structure for the best model for each y-variate

EXIT = *scalars* — Exit status of the best model for each y-variate

SAVE = *REML save structures* — Save structure from the analysis of the best model for each y-variate

VARIATE directive

Declares one or more variate data structures.

Options

NVALUES = *scalar* or *vector* — Number of units, or vector of labels; default * takes the setting from the preceding UNITS statement, if any

VALUES = *numbers* — Values for all the variates; default *

MODIFY = *string token* — Whether to modify (instead of redefining) existing structures

(yes, no); default no

IPRINT = *string tokens* — Information to be used by default to identify the variates in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers* — Identifiers of the variates

VALUES = *identifiers* — Values for each variate

DECIMALS = *scalars* — Number of decimal places for output

EXTRA = *texts* — Extra text associated with each identifier

MINIMUM = *scalars* — Minimum value for the contents of each structure

MAXIMUM = *scalars* — Maximum value for the contents of each structure

DREPRESENTATION = *scalars* or *texts* — Default format to use when the contents represent dates and times

VAROWCOLUMNDESIGN procedure

Analyses a row-and-column design by REML, with automatic selection of the best random and spatial covariance model (R.W. Payne).

Options

PRINT = *string tokens* — Controls what summary output is produced about the models (best, description, deviance, aic, bic, sic, dffixed, dfrandom, change, exit); default best, desc

PBEST = *string tokens* — Controls the output from the REML analysis with the best model (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic); default * i.e. none

PTRY = *string tokens* — Controls the output to present from the REML analysis used to try each model (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic); default * i.e. none

FIXED = *formula* — Fixed model terms; default * i.e. none

RANDOM = *formula* — Additional random model terms; default * i.e. none

CONSTANT = *string token* — How to treat the constant term (estimate, omit); default esti

FACTORIAL = *scalar* — Limit on the number of factors or covariates in each fixed term; default 3

REPLICATES = *factor* — Replicate factor, if relevant

ROWS = *factor* — Row factor; default * i.e. must be specified

COLUMNS = *factor* — Column factor; default * i.e. must be specified

ROWCOORDINATES = *variate* or *factor* — Row coordinates for fitting trends and spatial models if the design is irregular; if unset, these are defined from the levels of the ROWS factor

COLCOORDINATES = *variate* or *factor* — Column coordinates for fitting trends and spatial models if the design is irregular; if unset, these are defined from the levels of the COLUMNS factor

PLOTFACTOR = *factor* — Factor numbering the plots in the design; if unset, a local factor is defined automatically

PTERMS = *formula* — Terms (fixed or random) for which effects or means are to be printed; default * implies all the fixed terms

PSE = *string token* — Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default diff

MVINCLUDE = *string tokens* — Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default * i.e. omit units with

	missing values in either explanatory factors or variates or y-variates
VCONSTRAINTS = *string token*	Whether to constrain variance components to be positive (none, positive); default none
RSTRATEGY = *string token*	Strategy for selecting the random model (all, allfeasible, set, setfeasible, fastoptimal, optimal, automatic, comprehensive, full, given); default allf
METHOD = *string token*	Criterion to choose the best random model (aic, sic, bic); default sic
TRYSPATIAL = *string token*	Whether to try spatial models (always, ifregular); default * i.e. no spatial models
TRYTRENDS = *string token*	Whether to see whether row and column trends are needed in the fixed model (yes, no); default no
SPATIALFACTOR = *factor*	Factor to use to define the term for a 2-dimensional power-distance model; if unset, a local factor is defined automatically

Parameters

Y = *variates*	Response variates
BESTMODEL = *pointers*	Saves a model-definition structure for the best model for each y-variate
EXIT = *scalars*	Exit status of the best model for each y-variate
SAVE = *REML save structures*	Save structure from the analysis of the best model for each y-variate

VASDISPLAY procedure

Displays further output from an analysis by VASERIES (R.W. Payne).

Options

PRINT = *string tokens*	What output to present (model, components, effects, means, stratumvariances, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic); default mode, comp, Wald, cova
PTERMS = *formula*	Terms (fixed or random) for which effects or means are to be printed; default * implies all the fixed terms
PSE = *string token*	Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default diff
CFORMAT = *string token*	Whether printed output for covariance models gives the variance matrices or the parameters (variancematrices, parameters); default vari
FMETHOD = *string token*	Controls whether and how to calculate F-statistics for fixed terms (automatic, none, algebraic, numerical); default auto
MODELDEFINITIONS = *pointer*	Definitions of the models used by VASERIES
SAVE = *pointer*	REML save structures from the VASERIES analysis

Parameter

EXPERIMENT = *scalars* or *texts*	Specifies the experiment, from the series, whose output is to be displayed; no default, must be set

VASERIES procedure

Analyses a series of trials with incomplete-block or row-and-column designs by REML, automatically selecting the best random models (R.W. Payne).

Options

PRINT = *string tokens*	Controls what summary output is produced about the models (deviance, aic, bic, sic, dffixed, dfrandom, change, exit, best); default devi, aic, sic, dfra, best
PBEST = *string tokens*	Controls the output from the REML analysis with the best model (model, components, effects, means,

	stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic); default * i.e. none
PTRY = *string tokens*	Controls the output to present to present from the REML analysis used to try each model (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic); default * i.e. none
FIXED = *formula*	Fixed model terms; default * i.e. none
RANDOM = *formula*	Additional random model terms; default * i.e. none
CONSTANT = *string token*	How to treat the constant term (estimate, omit); default esti
FACTORIAL = *scalar*	Limit on the number of factors or covariates in each fixed term; default 3
EXPERIMENTS = *factor*	Experiment factor
REPLICATES = *factor*	Replicate factor, if required
BLOCKS = *factor*	Block factor, if required
ROWS = *factor*	Row factor, if required
COLUMNS = *factor*	Column factor, if required
ROWCOORDINATES = *variate* or *factor*	Row coordinates for fitting trends and spatial models if the design is irregular; if unset, these are defined from the levels of the ROWS factor
COLCOORDINATES = *variate* or *factor*	Column coordinates for fitting trends and spatial models if the design is irregular; if unset, these are defined from the levels of the COLUMNS factor
PLOTFACTOR = *factor*	Factor numbering the plots in the design; if unset, a local factor is defined automatically
PTERMS = *formula*	Terms (fixed or random) for which effects or means are to be printed; default * implies all the fixed terms
PSE = *string token*	Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default diff
MVINCLUDE = string tokens	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default * i.e. omit units with missing values in either explanatory factors or variates or y-variates
VCONSTRAINTS = *string token*	Whether to constrain variance components to be positive (none, positive); default none
RSTRATEGY = *string token*	Strategy for selecting the random model (all, allfeasible, fastoptimal, optimal); default allf
METHOD = *string token*	How to choose the best random model (aic, sic, bic); default sic
TRYSPATIAL = *string token*	Whether to try spatial models (always, ifregular); default * i.e. no spatial models
TRYTRENDS = *string token*	Whether to see whether row and column trends are needed in the fixed model (yes, no); default no
SPATIALFACTOR = *factor*	Factor to use to define the term for a 2-dimensional power-distance model; if unset, a local factor is defined automatically

Parameters

Y = *variates*	Response variates
MODELDEFINITIONS = *pointers*	Saves definitions of the best models for use by VAMETA
EXIT = *variates*	Exit status of the best models (zero if successful)
SAVE = *pointers*	REML save structures for the best analysis of each experiment

VASKEEP procedure

Copies information from an analysis by VASERIES into Genstat data structures (R.W. Payne).

Options

EXPERIMENT = *scalar* or *text*	Specifies the experiment, from the series, whose output is to be saved; no default, must be set
FACTORIAL = *scalar*	Limit on the number of factors or covariates in the terms generated from the TERMS parameter; default 3
RESIDUALS = *variate*	Residuals from the analysis
FITTEDVALUES = *variate*	Fitted values from the analysis
DEVIANCE = *scalar*	Residual deviance from fitting the full fixed model
DF = *scalar*	Residual degrees of freedom after fitting the full fixed model
AIC = *scalar*	Saves the Akaike information coefficient
SIC = *scalar*	Saves the Schwarz (Bayesian) information coefficient
RMETHOD = *string token*	Which random terms to use when calculating RESIDUALS (final, all); default fina
FMETHOD = *string token*	Controls how to calculate F-statistics for fixed terms (automatic, none, algebraic, numerical); default auto
WMETHOD = *string token*	Controls which Wald statistics are saved (add, drop); default drop
MODELDEFINITIONS = *pointer*	Definitions of the models used by VASERIES
SAVE = *pointer*	REML save structures from the VASERIES analysis

Parameters

TERMS = *formula*	Terms for which information is to be saved
COMPONENTS = *scalars*	Estimated variance components
MEANS = *tables*	Table of predicted means for each term
SEDMEANS = *symmetric matrices*	Standard errors of differences between the predicted means
VARMEANS = *symmetric matrices*	Variance-covariance matrix of the means
EFFECTS = *tables*	Table of estimated regression coefficients for each term
SEDEFFECTS = *symmetric matrices*	Standard errors of differences between the estimated parameters of each term
VAREFFECTS = *symmetric matrices*	Variance-covariance matrix of the effects of a term
WALD = *scalars*	Wald statistic (fixed terms only)
FSTATISTIC = *scalars*	F statistics (fixed terms only)
NDF = *scalars*	Numerator d.f. (fixed terms only)
DDF = *scalars*	Denominator d.f. (fixed terms only)

VASMEANS procedure

Saves experiment × treatment means from analysis of a series of trials by VASERIES (R.W. Payne).

Options

FACTORIAL = *scalar*	Limit on the number of factors in the terms generated from the TERMS parameter; default 3
RESIDUALVARIANCES = *table*	Saves residual variances from the experiments
MODELDEFINITIONS = *pointer*	Definitions of the models used by VASERIES
SAVE = *pointer*	REML save structures from the VASERIES analysis

Parameters

TERMS = *formula*	Terms for which means are to be saved
MEANS = *tables* or *pointers*	Experiment × term tables of means
SEMEANS = *tables* or *pointers*	Experiment × term tables of standard errors of means
AVESEDMEANS = *tables* or *pointers*	Average standard errors of differences for the experiments

VAYPARALLEL procedure

Does the same REML analysis for several y-variates, and collates the output (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (summary, monitoring); default * i.e. none
MODELDEFINITION = *pointer*	Defines the model for the analysis

FSAVETERMS = *formula*	Fixed terms for which to save information; if this is not set, information is saved for all the fixed terms
RSAVETERMS = *formula*	Random terms for which to save information; if this is not set, no information is saved for the random terms
RECOVER = *string token*	Whether to try to recover with a simpler random model if REML cannot fit the model for a particular y-variate (yes, no); default no
METHOD = *string token*	How to choose the best model during recovery (aic, sic, bic); default sic
SPREADSHEET = *string tokens*	What results to save in spreadsheets (components, fixedtests, means, vcmeans, effects, vceffects, residuals, fittedvalues); default * i.e. none
SHEETLAYOUT = *string token*	How to store the results in spreadsheets (yrows, ycolumns, onesheet); default ycol

Parameters

Y = *pointers*	Y-variates for the analyses
RESIDUALS = *matrices*	Saves the residuals
FITTEDVALUES = *matrices*	Saves the fitted values
COMPONENTS = *matrices*	Saves the variance components
MEANS = *pointers*	Pointer to a matrix for each of the terms in FSAVETERMS, saving the predicted means
VCMEANS = *pointers*	Pointer to matrices saving variances and covariances for the means
EFFECTS = *pointers*	Pointer to matrices saving effects for the terms in FSAVETERMS and RSAVETERMS
VCEFFECTS = *pointers*	Pointer to matrices saving variances and covariances for the effects
WALD = *matrices*	Saves the Wald statistics for the terms in FSAVETERMS
FSTATISTIC = *matrices*	Saves the F statistics for the terms in FSAVETERMS
NDF = *matrices*	Saves the numerator degrees of freedom for the terms in FSAVETERMS
DDF = *matrices*	Saves the denominator degrees of freedom for the terms in FSAVETERMS
PRFIXED = *matrices*	Saves the probabilities for the F statistics if available, or otherwise the Wald statistics, for the terms in FSAVETERMS
EXIT = *pointers*	Pointer to scalars saving the exit codes from the initial REML analyses
OUTFILENAME = *texts*	Name of Genstat workbook file (.gwb) or Excel (.xls or .xlsx) file to create

VBOOTSTRAP procedure

Performs a parametric bootstrap of the fixed effects in a REML analysis (C.J. Brien & R.W. Payne).

Options

†PRINT = *string tokens*	Controls printed output (observedteststatistics, pvalues, vdiagnostics, nnotconverged, monitoring, all, ownstatistics); default obse, pval
VPRINT = *string tokens*	Controls the output from the REML analysis of each sample (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default * i.e. none
PLOT = *string*	What to plot (histogram); default *
NBOOT = *scalar*	Number of bootstrap samples to take; default 99
NRETRIES = *scalar*	Maximum number of extra samples to take when some REML analyses fail to converge; default NBOOT
SEED = *scalar*	Seed for random number generation; default 0 continues an existing sequence or, if none, selects a seed automatically

METHOD = *string token* — Indicates whether to use the standard Fisher-scoring algorithm or the new AI algorithm with sparse matrix methods (Fisher, AI); default AI

MAXCYCLE = scalar — Sets a limit on the number of iterations in the REML analyses; default 30

FMETHOD = *string token* — Controls whether and how to calculate F statistics for fixed terms (automatic, none, algebraic, numerical); default none

WMETHOD = *string token* — Controls which Wald statistics are saved (add, drop); default add

WORKSPACE = *scalar* — Number of blocks of internal memory to be set up for use by the REML algorithm

†OWNMETHOD = *string token* — Type of test required for own statistics (twosided, greaterthan, lessthan); default twos

†CIPROBABILITY = *scalar* — Probability level for the confidence interval for own statistics; default 0.95

Parameters

SAVE = *REML save structures* — Specifies the (REML) save structure of the original analysis; default * uses the SAVE structure from the most recent REML analysis

UMEANS = *variates* — Specifies the expected values for the units under the null hypothesis of no effects from the FIXEDTERMS

UVCOVARIANCE = *symmetric matrices* — Specifies the variances and covariances of the units under the null hypothesis of no effects from the FIXEDTERMS

FIXEDTERMS = *formula* — Specifies the fixed terms to test; default * tests all the fixed terms in the original analysis

FSTATISTICS = *pointers* — Saves a pointer with a variate for each of the FIXEDTERMS, containing the F statistics from the bootstrap samples

PVALUES = *pointers* — Saves a pointer with a scalar for each of the FIXEDTERMS, containing the test probability obtained from the position of its F statistic within those from the bootstrap samples

NNOTCONVERGED = *scalars* — Saves the number of bootstrap samples whose REML analysis failed to converge

†OWNDATA = *pointers* — Data required to calculate own statistics

†OWNOBSERVEDVALUES = *variates* — Saves observed values of the own statistics

†OWNPROBABILITIES = *variates* — Saves bootstrap probabilities for the own statistics

†OWNESTIMATES = *variates* — Saves boostrap estimates for the own statistics

†OWNSES = *variates* — Saves boostrap standard errors for the own statistics

†OWNLOWERCIS = *variates* — Saves boostrap lower values of the confidence intervals for the own statistics

†OWNUPPERCIS = *variates* — Saves boostrap upper values of the confidence intervals for the own statistics

†OWNSTATISTICS = *pointers* — Saves the own statistics obtained from the bootstrap samples, in a pointer with a variate for each statistic

VCHECK procedure

Checks standardized residuals from a REML analysis (R.W. Payne).

Options

PRINT = *string tokens* — Controls printed output (largeresiduals, similarunits, stability); default larg

RMETHOD = *string token* — Which random terms to use when calculating the standardized residuals (final, all); default fina

RLIMIT = *scalar* — Limit for detection of large standardized residuals; if this is not set, the limit is set automatically according to the number of residual degrees of freedom

COMMONFACTORS = *factors* — Factors to define similar units; if this is not set, the factors in the fixed model are used

REPORTFACTORS = *factors*	Additional factors to include in the table of similar units
PROBABILITY = *scalar*	Critical value for the test probabilities to decide whether to generate warning messages from the Levine test for variance stability; default=0.025
NLARGERESIDUALS = *scalar*	Saves the number of large standardized residuals that have been detected
LARGERESIDUALUNITS = *variate*	Saves the unit numbers of the large standardized residuals
SIMILARINFORMATION = *pointer*	Saves details of large standardized residuals and residuals in similar units
STABILITYTEST = *pointer*	Saves the results of the Levene test for stability of the variance of the standardized residuals
SAVE = *REML save structure*	Specifies the analysis to be checked; by default this will be the most recent REML

No parameters

VCOMPONENTS directive

Defines the variance-components model for REML.

Options

FIXED = *formula*	Fixed model terms; default *
ABSORB = *factor*	Defines the absorbing factor (appropriate only when REML option METHOD=Fisher); default * i.e. none
CONSTANT = *string token*	How to treat the constant term (estimate, omit); default esti
FACTORIAL = *scalar*	Limit on the number of factors or covariates in each fixed term; default 3
CADJUST = *string token*	What adjustment to make to covariates before analysis (mean, none); default mean
RELATIONSHIP = *matrix*	Defines relationships constraining the values of the components; default *
SPLINE = *formula*	Defines random cubic spline terms to be generated: each term must contain only one variate, if there is more than one factor in a term, separate splines are calculated for each combination of levels of the factors
EXPERIMENTS = *factor*	Factor defining the different experiments in a multi-experiment (meta-) analysis

Parameters

RANDOM = *formula*	Random model terms
INITIAL = *scalars*	Initial values for each component and the residual variance
CONSTRAINTS = *string tokens*	How to constrain each variance component and the residual variance (none, positive, fixrelative, fixabsolute); default none

VCRITICAL procedure

Uses a parametric bootstrap to estimate critical values for a fixed term in a REML analysis (R.W. Payne & C.J. Brien).

Options

PRINT = *string tokens*	Prints the critical values (critical, fcritical, tcritical, wcritical, monitoring); default crit, fcri, tcri, wcri
VPRINT = *string tokens*	Controls the output from the REML analyses (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default * i.e. none
TERM = *formula*	Fixed term to be tested
UMEANS = *variate*	Specifies the expected values for the units under the null hypothesis of no effects from the TERM; default is to use the

constant from the SAVE structure

UVCOVARIANCE = *symmetric matrix* — Specifies the variances and covariances of the units under the null hypothesis of no effects from the TERM; default is to take this from the SAVE structure

WCRITICAL = *variate* — Saves the critical values of the Wald statistic

FCRITICAL = *variate* — Saves the critical values of the F statistic

NBOOT = *scalar* — Number of bootstrap samples to take; default 99

NRETRIES = *scalar* — Maximum number of extra samples to take when some REML analyses fail to converge; default NBOOT

SEED = *scalar* — Seed for random number generation; default 0 continues an existing sequence or, if none, selects a seed automatically

PROBABILITIES = *scalar* or *variate* — Significance levels for which critical values are required; default 0.05

METHOD = *string token* — Indicates whether to use the Fisher-scoring algorithm or the AI algorithm with sparse matrix methods (Fisher, AI); default AI

MAXCYCLE = *scalar* — Sets a limit on the number of iterations in the REML analyses; default 30

FMETHOD = *string token* — Controls how to calculate estimated denominator degrees of freedom when these are to be saved (automatic, none, algebraic, numerical); default auto

WMETHOD = *string token* — Controls which Wald statistics are saved (add, drop); default add

TMETHOD = *string token* — Type of test to be made for the contrasts (twosided, greaterthan, lessthan, equivalence, noninferiority); default twos

WALD = *variate* — Saves the Wald statistics from the samples

FSTATISTIC = *variate* — Saves the F statistics from the samples

NDF = *scalar* — Saves the numerator degrees of freedom for the Wald and F statistics

DDF = *variate* — Saves the estimated denominator degrees of freedom for the F statistics

NNOTCONVERGED = *scalar* — Saves the number of bootstrap samples whose REML analysis failed to converge

WORKSPACE = *scalar* — Number of blocks of internal memory to be set up for use by the REML algorithm

SAVE = *vsave* — REML save structure to provide the information about the analysis

Parameters

XCONTRASTS = *variates* or *tables* — X-variate defining a contrast to be detected

CONTRASTTYPE = *string tokens* — Type of contrast (regression, comparison) default rege

ESTIMATE = *variates* — Saves the estimated values of the contrasts from the samples

SE = *variates* — Saves the standard errors for the estimates of the contrasts from the samples

CRITICAL = *variates* — Saves the critical values for the contrasts

TCRITICAL = *variates* — Saves the critical values for the t-statistics of the contrasts

VCYCLE directive

Controls the operation of the REML algorithm.

Options

CONVERGENCE = *string token* — Type of criterion for assessing convergence (deviance, parameter); default * uses the deviance with the average-information algorithm, and the variance parameter values for the Fisher scoring algorithm

CRITERIONVALUE = *scalar* — Sets the convergence criterion value; default * i.e. determined automatically

STEPLENGTH = *scalar* — Sets the default relative step size for the average-information

algorithm; default * i.e. determined automatically

NDENSE = *scalar* — Number of equations to use as dense in the average-information algorithm; default * uses all fixed model terms as dense

EQORDER = *string token* — Method to use to reorder the mixed model equations for fitting (none, a, b); default b

No parameters

VDFIELDRESIDUALS procedure

Display residuals from a REML analysis in field layout (R.W. Payne).

Options

PRINT = *string tokens* — Controls printed output (table); default * i.e. none

PLOT = *string tokens* — Controls the graphs that are displayed (contour, shade); default cont

RMETHOD = *string token* — Which random terms to use to calculate the residuals (final, all, notspline, stfinal, stall); default all

GRAPHICS = *string token* — Type of graph (highresolution, lineprinter); default high

MARGIN = *string token* — Whether to include margins in printed tables (yes, no); default no

YORIENTATION = *string token* — Y-axis orientation of the plot (reverse, normal); default norm

PENCONTOUR = *scalar* — Pen number to be used for the contours; default 1

PENFILL = *scalar* or *variate* — Pen number(s) defining how to fill the areas between contours; default 3

PENSHADE = *scalar* or *variate* — Pen(s) to use for the shade plot; default 3

Parameters

Y = *variates* or *factors* — Specifies the y-coordinates of the plots

X = *variates* or *factors* — Specifies the x-coordinates of the plots

SAVE = *REML save structures* — Save structure of the REML analysis from which to take the residuals; default is to take the most recent REML analysis

FIELDWIDTH = *scalars* — Field width for printing the residuals; default 12

DECIMALS = *scalars* — Number of decimal places to use when printing the residuals

TITLE = *texts* — Titles for the plots

VDISPLAY directive

Displays further output from a REML analysis.

Options

PRINT = *string tokens* — What output to present (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default mode, comp, Wald, cova

CHANNEL = *identifier* — Channel number of file, or identifier of a text to store output; default current output file

PTERMS = *formula* — Terms (fixed or random) for which effects or means are to be printed; default * implies all the fixed terms

PSE = *string token* — Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default diff

CFORMAT = *string token* — Whether printed output for covariance models gives the variance matrices or the parameters (variancematrices, parameters); default vari

FMETHOD = *string token* — Controls whether and how to calculate F-statistics for fixed terms (automatic, none, algebraic, numerical); default auto

Parameter

REML save structures — Save structure containing the details of each analysis; default is to take the save structure from the latest REML analysis

VEQUATE procedure

Equates across numerical structures (P.W. Goedhart).

No options

Parameters

OLDSTRUCTURES = *pointers*	Structures whose values are to be transferred – each pointer should contain a set of structures with the same length and type (either scalar, variate, matrix, diagonal matrix, symmetric matrix, table, text or pointer)
NEWSTRUCTURES = *pointers*	Structures to contain the transferred values – each pointer contains a set of either variates, texts or pointers, as relevant to the type of the OLDSTRUCTURES

VDEFFECTS procedure

Plots one- or two-way tables of effects estimated in a REML analysis (R.W. Payne).

Options

GRAPHICS = *string token*	Type of graph (highresolution, lineprinter); default high
METHOD = *string token*	What to plot (effects, lines); default effe
XFREPRESENTATION = *string token*	How to label the x-axis (levels, labels); default labels uses the XFACTOR labels, if available
PSE = *string*	What s.e. to plot to represent variation (differences, effects, aleffects); default diff
SAVE = *REML save structure*	Save structure of the analysis to display; the default is to take the most recent REML analysis

Parameters

XFACTOR = *factors*	Factor providing the x-values for each plot
GROUPS = *factors*	Factor identifying the different sets of points from a two-way table of effects
COVARIATES = *variates*	X-variates for regression coefficients or pointer
NEWXLEVELS = *variates*	Values to be used for XFACTOR instead of its existing levels
TITLE = *texts*	Title for the graph; default defines a title automatically
YTITLE = *texts*	Title for the y-axis; default ' '
XTITLE = *texts*	Title for the x-axis; default is to use the identifier of the XFACTOR

VFIXEDTESTS procedure

Saves fixed tests from a REML analysis (R.W. Payne).

Options

FIXEDTESTS = *pointer*	Saves the fixed tests
FMETHOD = *string token*	Controls whether and how to calculate F-statistics (automatic, none, algebraic, numerical); default auto
WMETHOD = *string token*	Controls which tests are saved (add, drop); default drop
SAVE = *REML save structure*	Specifies the save structure from the required analysis; default * i.e. most recent one

No parameters

VFLC procedure

Performs an F-test of random effects in a linear mixed model based on linear combinations of the responses, i.e. an FLC test (V.M. Cave).

PRINT = *string tokens*	Controls printed output (summary, monitoring); default summ
PLOT = *string tokens*	What graphs to plot for the bootstrap and fast double bootstrap FLC tests (kerneldensity, histogram); default * i.e. none
TEST = *string tokens*	Type(s) of test to perform; (flc, bootstrap, fastdoublebootstrap); default flc

NBOOT = *scalar*	Number of bootstrap samples to take; default 99
SEED = *scalar*	Seed for random number generation; default 0 continues an existing sequence or, if none, selects a seed automatically
WINDOW = *scalar*	Window to use for the graphs; default 3
SAVE = REML save structure	Specifies the save structure of the original analysis; default is to use the save structure from the most recent REML analysis

Parameters

TERMS = *formula*	Random terms to test
STATISTIC = *scalar*	Saves the FLC test statistic
BOOTSTATISTICS = *variate*	Saves the FLC test statistics from the original data set (i.e. the observed FLC test statistic), and then the bootstrap samples
FASTDOUBLE = *pointer*	Pointer to scalars and variates to save the first-level bootstrap probability value and FLC test statistics, and the second-level fast double bootstrap FLC test statistics and resulting critical value
PROBABILITIES = *pointer*	Pointer to scalar(s) to save the probability value(s) from the test(s)
TITLE = *text*	Title for the graphs

VFMODEL procedure

Forms a model-definition structure for a REML analysis (R.W. Payne).

Options

MODELSTRUCTURE = *pointer*	Specifies the model-definition structure; no default (must be specified)
DESCRIPTION = *text*	Description of the model (for output)
FIXED = *formula*	Fixed model terms; default *
CONSTANT = *string token*	How to treat the constant term (estimate, omit); default esti
FACTORIAL = *scalar*	Limit on the number of factors or covariates in each fixed term; default 3
CADJUST = *string token*	What adjustment to make to covariates before analysis (mean, none); default mean
CHANGEITEMS = *string tokens*	What changes to make to an existing model-definition structure (description, fixed, constant, factorial, cadjust, random, initial, constraints); if this is unset, the structure is redefined completely
IMODELSTRUCTURE = *pointer*	Specifies the initial model-definition structure, to modify when CHANGEITEMS is set; default is to modify the one specified by MODELSTRUCTURE
EXPERIMENTS = *factor*	Factor defining the different experiments in a multi-experiment (meta-) analysis

Parameters

RANDOM = *formula*	Random model terms
INITIAL = *scalars*	Initial values for each component
CONSTRAINTS = *string tokens*	How to constrain each variance component and the residual variance (none, positive, fixrelative, fixabsolute); must be set unless MODIFY=yes

VFPEDIGREE procedure

Checks and prepares pedigree information from several factors, for use by VPEDIGREE and REML (S.A. Gezan & R.W. Payne).

Options

FREPRESENTATION = *string token*	Whether to match factor values by their levels or their labels (levels, labels); default leve
†SEX = *string token*	Possible sex categories of parents (fixed, either); default fixe
UNKNOWN = *scalar* or *string*	Value to be treated as unknown in the pedigree factors

†INVMETHOD = *string token* — How to represent the INVERSE (full, sparse); default spar

Parameters

INDIVIDUALS = *factors* — Individuals on which data have been measured
MALEPARENTS = *factors* — Male parents (or sires) of the progeny
FEMALEPARENTS = *factors* — Female parents (of dams) of the progeny
NEWINDIVIDUALS = *factors* — New individuals factor, with levels standardized for use in VPEDIGREE
NEWMALEPARENTS = *factors* — New males factor, with levels standardized to match those in the NEWINDIVIDUALS factor
NEWFEMALEPARENTS = *factors* — New females factor, with levels standardized to match those in the NEWINDIVIDUALS factor
OTHERFACTORS = *pointers* — Pointer containing additional factors, that may be used in the REML models, whose levels must also be standardized to match those in the NEWINDIVIDUALS factor
NEWOTHERFACTORS = *pointers* — Pointer containing new additional factors, with standardized levels
†INVERSE = *pointer* — Inverse relationship matrix in sparse matrix form
†POPULATION = *variates* — Full list of identifiers generated from the individuals and parents

VFRESIDUALS procedure

Obtains residuals, fitted values and their standard errors from a REML analysis (S.J. Welham).

Options

RESIDUALS = *variate* — Saves the residuals
SERESIDUALS = *variate* — Saves standard errors of the residuals
FITTEDVALUES = *variate* — Saves the fitted values
SEFITTEDVALUES = *variate* — Saves prediction standard errors for the fitted values
RMETHOD = *string token* — Which random terms to use when calculating the residuals (final, all); default fina
MAXNUNITS = *scalar* — Maximum number of units for which the full variance-covariance matrix will be formed; default 1000
EXIT = *scalar* — Exit code set to zero if the saving was successful, one otherwise
SAVE = *REML save structure* — Save structure for the required analysis; default uses the save structure from the most recent REML

No parameters

VFSTRUCTURE procedure

Adds a covariance-structure definition to a REML model-definition structure (R.W. Payne).

Options

MODELSTRUCTURE = *pointer* — Supplies the model-definition structure; no default (must be specified)
EXPERIMENT = *scalar* — Level of the EXPERIMENTS factor for which a residual is to be defined (using the VRESIDUAL directive)
TERMS = *formula* — Model terms for which the covariance structure is to be defined
FORMATION = *string token* — Whether the structure is formed by direct product construction or by definition of the whole matrix (direct, whole); default dire
COORDINATES = *identifiers* — Coordinates of the data points to be used in calculating distance-based models (list of variates or matrix)

Parameters

MODELTYPE = *string tokens* — Type of covariance model associated with the term(s), or with individual factors in the term(s) if FORMATION=direct (identity, fixed, AR, MA, ARMA, power, banded, correlation, antedependence, unstructured,

	diagonal, uniform, FA, FAequal) default iden
ORDER = *scalar*	Order of model
HETEROGENEITY = *string token*	Heterogeneity for correlation matrices (none, outside); default none
METRIC = *string token*	How to calculate distances when MODELTYPE=power (cityblock, squared, euclidean); default city
FACTOR = *factors*	Factors over which to form direct products

VFUNCTION procedure

Calculates functions of variance components from a REML analysis (S.J. Welham).

Options

PRINT = *string token*	Output required (function); default func
RANDOM = *formula*	Random model (excluding residual stratum) used for the REML analysis
NCONSTANT = *scalar*	Value to be used as constant in the numerator function; default 0
DCONSTANT = *scalar*	Value to be used as constant in the denominator function; default 0
SAVE = *REML save structure*	Specifies the (REML) save structure from which the variance components are to be taken; by default they are taken from the save structure of the most recent REML analysis

Parameters

NUMERATOR = *variates*	Each variate contains a list of coefficients, one for each variance component, defining a linear combination of the components to use as the numerator of the function
DENOMINATOR = *variates*	Each variate contains coefficients defining a linear combination of the variance components to use as the denominator of the function
FUNCTIONVALUE = *scalars*	Saves the calculated value of the function
SE = *scalars*	Saves the approximate standard error of the function value

VGESELECT procedure

Selects the best variance-covariance model for a set of environments (M.P. Boer, M. Malosetti, S.J. Welham & J.T.N.M. Thissen).

Options

PRINT = *string tokens*	What to print (summary, best, model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, waldtests, missingvalues, covariancemodels); default summ, best, comp, cova
VCMODELS = *string tokens*	Specifies the variance-covariance models that are to be compared for the set of environments (identity, diagonal, cs, hcs, outside, fa, fa2, unstructured); default iden, diag, cs, hcs, outs, fa, fa2, unst
CRITERION = *string token*	Defines which criterion is used to compare the different covariance structures (aic, sic); default sic
FIXED = *formula*	Defines extra fixed effects
UNITFACTOR = *factor*	Saves the units factor required to define the random model when UNITERROR is to be used
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default expl, yvar
MAXCYCLE = *scalar*	Limit on the number of iterations; default 100
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm; default 100

Parameters

TRAIT = *variates*	Quantitative trait to be analysed; must be set

GENOTYPES = *factors*	Genotype factor; must be set
ENVIRONMENTS = *factors*	Environment factor; must be set
UNITERROR = *variate*	Uncertainty on trait means (derived from individual unit or plot error) to be included in QTL analysis; default * i.e. omitted
SELECTEDMODEL = *texts*	VCMODELS setting for the best variance-covariance model
SAVE = *REML save structures*	Save the details of each REML analysis for use in subsequent VDISPLAY and VKEEP directives

VGRAPH procedure

Plots tables of means from REML (R.W. Payne).

Options

GRAPHICS = *string token*	Type of graph (highresolution, lineprinter); default high
METHOD = *string token*	What to plot (points, means, linesandpoints, onlylines, data, barchart, splines); default poin when XFACTOR is a factor, and only when it is a variate
XFREPRESENTATION = *string token*	How to label the x-axis (levels, labels); default labe uses the XFACTOR labels, if available
PSE = *string token*	What to plot to represent variation when points are plotted at the means (differences, lsd, means, allmeans); default diff
LSDLEVEL = *scalar*	Significance level (%) to use for approximate least significant differences; default 5
DFSPLINE = *scalar*	Number of degrees of freedom to use when METHOD=splines
YTRANSFORM = *string tokens*	Transformed scale for additional axis marks and labels to be plotted on the right-hand side of the y-axis (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default iden i.e. none
PENYTRANSFORM = *scalar*	Pen to use to plot the transformed axis marks and labels; default * selects a pen, and defines its properties, automatically
†KEYMETHOD = *string token*	What to use for the key descriptions when GROUPS specifies more than one factor (labels, namesandlabels); default name
†PLOTTITLEMETHOD = *string token*	What to use for the titles of the plots when TRELLISGROUPS specifies more than one factor (labels, namesandlabels); default name
†PAGETITLEMETHOD = *string token*	What to use for the titles of the pages when PAGEGROUPS specifies more than one factor (labels, namesandlabels); default name
†USEAXES = *string token*	Which aspects of the current axis definitions of window 1 to use (none, limits, marks, mpositions, nsubticks,); default none
SAVE = *REML save structure*	Save structure to provide the table of means; default uses the save structure from the most recent REML

Parameters

XFACTOR = *factors* or *variates*	Provides the x-values for each plot; by default this is chosen automatically
GROUPS = *factors* or *pointers*	Factor or factors identifying groups in each plot; by default chosen automatically
TRELLISGROUPS = *factors* or *pointers*	Factor or factors specifying the different plots of a trellis plot of a multi-way table
PAGEGROUPS = *factors* or *pointers*	Factor or factors specifying plots to be displayed on different pages
NEWXLEVELS = *variates*	Values to be used for XFACTOR; default uses the existing levels if XFACTOR is a factor, and the minimum and maximum values

	if it is a variate
TITLE = *texts*	Title for the graph; default is to define a title automatically if GROUPS is set, or to have none if it is unset
YTITLE = *texts*	Title for the y-axis; default is to use the identifier of the y-variate, or to have no title if this is unnamed
XTITLE = *texts*	Title for the x-axis; default is to use the identifier of the XFACTOR
PENS = *variates*	Defines the pen to use to plot the points and/or line for each group defined by the GROUPS factors

VHERITABILITY procedure

Calculates generalized heritability for a random term in a REML analysis (R.W. Payne).

PRINT = *string tokens*	Controls printed output (heritability); default heri
SAVE = *REML save structure*	Save structure of the analysis from which to calculate the heritabilities; default uses the most recent REML analysis

Parameters

TERMS = *formula*	Random terms whose heritabilities are to be calculated
HERITABILITY = *scalar* or *variate*	Saves the heritabilities
EXIT = *scalar* or *variate*	Exit status for the calculations: one if unsuccessful, otherwise zero

VHOMOGENEITY procedure

Tests homogeneity of variances and variance-covariance matrices (R.W. Payne).

PRINT = *string tokens*	Controls printed output (test, variances); default test
GROUPS = *factors*	Define the groups whose variances are to be compared; these need be given only if DATA is set

Parameters

DATA = *variates* or *pointers*	Data variate from which variances are calculated, or pointer to a list of variates from which variance-covariance matrices are calculated
VARIANCES = *any numerical structures* or *pointers*	Supplies the variances (in any numerical structure) or variance-covariance matrices in a pointer to a list of symmetric matrices if the DATA parameter is not set, or saves variances (in a table) and variance-covariance matrices (in a pointer to a list of symmetric matrices) if they have been calculated from DATA and GROUPS
DF = *any numerical structure*	Supplies the degrees of freedom for variances (in any numerical structure) or for variance-covariance matrices (as a pointer to a list of scalars) if the DATA parameter is not set, or saves the degrees of freedom for variances (in a table) or variance-covariance matrices (as a pointer to a list of scalars) if they have been calculated from DATA and GROUPS
SAVE = *pointers*	Saves the results i.e. type of test, chi-square statistic, degrees of freedom and probability

VINTERPOLATE procedure

Performs linear & inverse linear interpolation between variates (R.J. Reader).

Options

METHOD = *string token*	Type of interpolation required (interval, value): for METHOD=value, y-values are interpolated for each point in the NEWINTERVALS variates and stored in the NEWVALUES variates, while for METHOD=interval, x-values are estimated for the y-values in the NEWVALUES variates and stored in the NEWINTERVALS variates; default inte

RANGEMETHOD = *string token* — Whether the smallest value, largest value or the mean of the two is returned if more than one value is valid (`first`, `middle`, `last`); default `midd`

Parameters

OLDVALUES = *pointers* — Each one contains variates specifying the y-values (data values) with which an interpolation is to be carried out

NEWVALUES = *pointers* — For `METHOD=value`, each pointer contains variates to store the results of an interpolation; for `METHOD=interval`, it contains either variates or scalars to specify y-values for which inverse interpolation is to be carried out

OLDINTERVALS = *variates* — Contains the x-values (intervals) corresponding to the variates in the `OLDVALUES` pointer

NEWINTERVALS = *pointers* — For `METHOD=interval`, each pointer contains variates to store the results of an inverse interpolation; for `METHOD=value`, it contains either variates or scalars to specify x-values at which interpolation is to be performed

`VKEEP` directive

Copies information from a `REML` analysis into Genstat data structures.

Options

RESIDUALS = *variate* — Residuals from the analysis

FITTEDVALUES = *variate* — Fitted values from the analysis

SIGMA2 = *scalar* — Variance component for the lowest stratum

VCOVARIANCE = *symmetric matrix* — Variance-covariance matrix for the estimates of the variance components

VESTIMATES = *variate* — Saves a vector of all parameters in the variance model

VARESTIMATES = *symmetric matrix* — Variance-covariance matrix for the parameters in the variance model (as saved by `VESTIMATES`)

VLABELS = *text* — Vector of text labels for the `VESTIMATES` and `VARESTIMATES` structures

MVESTIMATES = *variate* — Estimates of missing values

MVSE = *variate* — Standard errors of missing-value estimates

MVUNITS = *variate* — Unit numbers of missing values

ALLEFFECTS = *variate* — Full set of estimated fixed and random effects

ALLVCOVARIANCE = *symmetric matrix* — Variance-covariance matrix for the full set of fixed and random effects not associated with the absorbing factor

DEVIANCE = *scalar* — Residual deviance from fitting the full fixed model

DF = *scalar* — Residual degrees of freedom after fitting the full fixed model

SUBDEVIANCE = *scalar* — Residual deviance after fitting the submodel of the fixed model

SUBDF = *scalar* — Residual degrees of freedom after fitting the submodel of the fixed model

RSS = *scalar* — Residual sum of squares from fitting the `FIXED` model by general least squares with a covariance matrix derived from the estimated variance components

INDEX = *variate* — Index of units included in the analysis

MODELS = *pointer* — Pointer to formulae giving the fixed, random, spline and residual terms fitted

RMATRIX = *pointer* — Saves details of the covariance model fitted to the residual

RMETHOD = *string token* — Which random terms to use when calculating `RESIDUALS` (`final`, `all`, `notspline`); default uses the setting from the `REML` statement

CFORMAT = *string token* — Whether the covariance matrices or the parameters are saved for a `COVARIANCEMODEL` (`variancematrices`, `parameters`); default `vari`

UVCOVARIANCE = *symmetric matrix* — Unit-by-unit variance-covariance matrix

DFFIXED = *scalar* — Number of degrees of freedom in the fixed model

DFRANDOM = *scalar* Number of degrees of freedom in the random model
FMETHOD = *string token* Controls how to calculate F-statistics for fixed terms (automatic, none, algebraic, numerical); default auto
WMETHOD = *string token* Controls which Wald statistics are saved (add, drop); default drop
WORKSPACE = *scalar* Saves the workspace setting that was used by the REML command
YVARIATE = *dummy* Dummy to be set to the y-variate of the analysis
EXIT = *scalar* Exit status of the fit (0 if successful)
SAVE = *REML save structures* Save structure from the required analysis; default * takes the save structure from the latest REML statement

Parameters

TERMS = *formula* Terms for which information is to be saved
COMPONENTS = *scalars* Estimated variance components
COVARIANCEMODEL = *pointers* Saves details of the covariance model fitted to a random term
MEANS = *tables* Table of predicted means for each term
SEDMEANS = *symmetric matrices* Standard errors of differences between the predicted means
VARMEANS = *symmetric matrices* Variance-covariance matrix of the means
EFFECTS = *tables* Table of estimated regression coefficients for each term
SEDEFFECTS = *symmetric matrices* Standard errors of differences between the estimated parameters of each term
VAREFFECTS = *symmetric matrices* Variance-covariance matrix of the effects of a term
DESIGNMATRIX = *matrices* Saves the design matrix for the term
SPLBLUP = *pointers* Best linear unbiased predictors for spline terms, saved in a pointer with a variate for each combination of the levels of the factors in the term
SPLDESIGN = *pointers* Design matrices (Z) for spline terms, saved in a pointer with a matrix for each combination of the levels of the factors in the term
SPLX = *pointers* Knot points for spline terms, saved in a pointer with a variate for each combination of the levels of the factors in the term
SPLSMOOTH = *pointers* Smoothing parameters estimated for spline terms, saved in a pointer with a scalar for each combination of the levels of the factors in the term
CADJUSTMENT = *scalars* For a term involving covariates, saves the adjustment made to its values during the analysis
WALD = *scalars* Wald statistic (fixed terms only)
FSTATISTIC = *scalars* F statistics (fixed terms only)
NDF = *scalars* Numerator d.f. (fixed terms only)
DDF = *scalars* Denominator d.f. (fixed terms only)

VLINEBYTESTER procedure

Analyses a line-by-tester trial by REML (R.W. Payne).

Options

PRINT = *string tokens* Specifies the output to be produced (model, components, effects, means, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels, aic, sic, bic, combinability, tests); default mode, comp, wald, comb, test
PRECOVERY = *string tokens* Controls what summary output is produced about the models that are tried during recovery (deviance, aic, bic, sic, dffixed, dfrandom, change, exit, best); default devi, aic, sic, dfra, best
LINES = *factor* Specifies the line (usually female parent); no default (must be specified)
TESTERS = *factor* Specifies the tester (usually male parent); no default (must be

specified)

CONTROLS = *factor*	Distinguishes between control and test (line × tester) genotypes; default is that there are no controls
FIXED = *formula*	Fixed model terms, in addition to the TESTERS main effect and any control comparisons; default * i.e. none
RANDOM = *formula*	Random model terms, in addition to the terms involving LINES, TESTERS and EXPERIMENTS that are included automatically; default * i.e. none
CONSTANT = *string token*	How to treat the constant term (estimate, omit); default esti
FACTORIAL = *scalar*	Limit on the number of factors or covariates in each fixed term; default 3
EXPERIMENTS = *factor*	Specifies the different experiments for a REML meta analysis; default is that the data are all from a single experiment
PCOMBINABILITYTERMS = *formula*	Terms whose combinability effects are to be printed, selected from LINES, LINES.TESTERS and their interactions with EXPERIMENTS; default is to print all of them
PTERMS = *formula*	Terms (fixed or random) for which effects or means are to be printed; default * implies all the fixed terms
PSE = *string token*	Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default diff
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default * i.e. omit units with missing values in either explanatory factors or variates or y-variates
RECOVER = *string token*	Whether to try to recover with a simpler random model if REML cannot fit the model (yes, no); default no
METHOD = *string token*	How to choose the best model during recovery (aic, sic, bic); default sic

Parameters

Y = *variates*	Response variates
COMBINABILITY = *pointers*	Pointer to tables of combinability effects for each y-variate
SECOMBINABILITY = *pointers*	Pointer to tables of standard errors of combinability effects for each y-variate
DEVIANCES = *variates*	Saves deviances for LINES, LINES.TESTERS and their interactions with EXPERIMENTS
EXIT = *scalars*	Exit status for each y-variate (zero to indicate that the analysis was successful)
SAVE = *REML save structures*	Save structure from the analysis of each y-variate

VLSD procedure

Prints approximate least significant differences for REML means (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (means, sed, lsd, df); default lsd
FACTORIAL = *scalar*	Limit on the number of factors in each term; default 3
LSDLEVEL = *scalar*	Significance level (%) to use in the calculation of least significant differences; default 5
DFMETHOD = *string token*	Specifies which degrees of freedom to use for the t-statistics (fddf, given, tryfddf); default fddf
DFGIVEN = *scalar*	Specifies the number of degrees of freedom to use for the t-statistics when DFMETHOD=given, or if d.d.f. are unavailable when DFMETHOD=tryfddf
FMETHOD = *string token*	Controls how to calculate denominator degrees of freedom for the F-statistics, if these are not already available in the REML

	save structure (`automatic`, `algebraic`, `numerical`); default `auto`
`SAVE` = *REML save structure*	Save structure to provide the table of means; default uses the save structure from the most recent `REML`

Parameters

`TERMS` = *formula*	Treatment terms whose means are to be compared; default `*` takes the `REML` fixed model
`MEANS` = *pointer* or *table*	Saves the means for each term
`SED` = *pointer* or *symmetric matrix*	Saves standard errors of differences between means
`LSD` = *pointer* or *symmetric matrix*	Saves approximate least significant differences matrix for the means
`DF` = *pointer* or *scalar*	Saves the degrees of freedom used to calculate the t critical values for the LSDs
`DDF` = *pointer* or *scalar*	Saves the range of denominator degrees of freedom in the F tests for the term and any terms that are marginal to the term (available only when denominator degrees of freedom of F-statistics are being used)

`VMATRIX` procedure

Copies values and row/column labels from a matrix to variates or texts (D.A. Murray).

No options

Parameters

`MATRIX` = *matrices*, *symmetric matrices* or *diagonal matrices*	Matrices to copy into variates
`VARIATE` = *variates*	Saves the values from each matrix
`ROWS` = *variates*	Saves the row coordinates
`COLUMNS` = *variates*	Saves the column coordinates
`ROWLABELS` = *texts*	Saves the row labels
`COLLABELS` = *texts*	Saves the column labels

`VMCOMPARISON` procedure

Performs pairwise comparisons between REML means (D.M. Smith).

Options

`PRINT` = *string tokens*	Controls printed output (`comparisons`, `critical`, `description`, `lines`, `letters`, `plot`, `mplot`, `pplot`); default `lett`
`METHOD` = *string token*	Test to be performed (`fplsd`, `fulsd`, `bonferroni`, `sidak`); default `fuls`
`FACTORIAL` = *scalar*	Limit on the number of factors in each term; default 3
`DIRECTION` = *string token*	How to sort means (`ascending`, `descending`); default `asce`
`PROBABILITY` = *scalar*	The required significance level; default 0.05
`STUDENTIZE` = *string token*	Whether to use the alternative LSD test where the Studentized Range statistic is used instead of Student's t (`yes`, `no`); default `no`
`DFMETHOD` = *string token*	Specifies which degrees of freedom to use for the tests (`fddf`, `given`, `tryfddf`); default `fddf`
`DFGIVEN` = *scalar*	Specifies the number of degrees of freedom to use for the tests when `DFMETHOD=given`, or if d.d.f. are unavailable when `DFMETHOD=tryfddf`
`FMETHOD` = *string token*	Controls how to calculate denominator degrees of freedom for the F-statistics, if these are not already available in the `REML` save structure (`automatic`, `algebraic`, `numerical`); default `auto`
`SAVE` = *REML save structure*	Save structure to provide the tables of means and associated information; default uses the save structure from the most recent `REML`

Parameters

TERMS = *formula*	Treatment terms whose means are to be compared
MEANS = *pointer* or *variate*	Saves the (sorted) means
LABELS = *pointer* or *text*	Saves labels for the (sorted) means
LETTERS = *pointer* or *text*	Saves letters indicating groups of means that do not differ significantly
SIGNIFICANCE = *pointer*	Indicators to show significant comparisons between or symmetric matrix (sorted) means

VMETA procedure

Performs a multi-treatment meta analysis using summary results from individual experiments (V.M. Cave).

Options

PRINT = *string tokens*	Controls printed output from the REML analysis (model, components, effects, means, monitoring, vcovariance, deviance, Waldtests, covariancemodels); default mode, comp, cova, mean
PSE = *string token*	Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default alle
EMETHOD = *string token*	Specifies whether the EXPERIMENTS main effect is fitted as a fixed or random term in the REML model; default fixe
VCMODEL = *string token*	Specifies the between-experiment variance-covariance model (identity, diagonal, cs, hcs, unstructured, faequal1, faequal2, fa1); default iden for fixed EXPERIMENTS effects and cs for random effects
INITIAL = *scalars*, *variates*, *matrices*, *symmetric matrices* or *pointers*	Initial parameter values for the variance-covariance model specified by VCMODEL (supplied in the structures appropriate for the model concerned); default generates values automatically
MAXCYCLE = *scalar*	Sets a limit on the number of iterations in the REML analysis; default 30

Parameters

MEANS = *variates*	Supplies the TREATMENTS by EXPERIMENTS means
TREATMENTS = *factors*	Identifier of the treatments factor
EXPERIMENTS = *factors*	Identifier of the experiments factor
SEDMEANS = *variates*	Supplies the (average) standard error of differences in each experiment
VARIANCES = *variates*	Identifier for the variate containing the sampling variance for each experiment
MODERATOR = *factors* or *variates*	Identifier for a moderator variable
SAVE = *REML save structures*	Saves the details of each analysis for use in subsequent VDISPLAY and VKEEP directives

VMODEL procedure

Specifies the model for a REML analysis using a model-definition structure defined by VFMODEL (R.W. Payne).

Option

PRINT = *string tokens*	Controls printed output (model, structure); default * i.e. none

Parameter

MODELSTRUCTURE = *pointer*	Model-definition structure

VNEARESTNEIGHBOUR procedure

Analyses a field trial using nearest neighbour analysis (D.B. Baird).

Options

PRINT = *string tokens*	Controls printed output (model, wald, components, means, effects, sed); default mode, wald, comp, mean, effe, sed
NDIFFERENCES = *scalar*	Specifies the number of neighbours to use in differencing the plots, either 1 for first or 2 for second differences; default 1
TMETHOD = *string token*	Indicates how the treatments effects are to be included in the model (fixed, random); default fixe
UMETHOD = *string token*	Whether to include a unit-error term in the model (include, omit); default incl
SEDMETHOD = *string token*	Specifies how the estimates of standard errors of differences of treatment effects are to be calculated (REML, simulation); default REML
NTIMES = *scalar*	Specifies the number of simulations to make; default 100

Parameters

Y = *variates*	Variates to be analysed
TREATMENTS = *factors*	Treatment factor for each y-variate
BLOCKS = *factors*	Block factor for each y-variate, defining groups of plots to be detrended independently
UNITS = *factors*	Unit-within-block factor for each y-variate, defining the order of plots within each block
MEANS = *tables*	Saves the estimated treatment means from each analysis
EFFECTS = *tables*	Saves the estimated treatment effects from each analysis
SED = *matrices* or *symmetric matrices*	Saves the estimated standard errors of differences between treatments
COMPONENTS = *variates*	Saves the estimated variance components from the fitted model
SEED = *scalars*	Seed for the random number generator used in the simulations to calculate standard error of differences; default 0 continues from the previous generation or (if none) initializes the seed automatically

VORTHPOLYNOMIAL procedure

Forms orthogonal polynomials over time for repeated measures (J.T.N.M. Thissen).

Options

TIMEPOINTS = *variate*	Variate of timepoints; default uses the suffixes of the DATA pointer
MAXDEGREE = *scalar*	The number of contrasts (excluding the mean); default is the number of identifiers in the CONTRAST pointer minus 1

Parameters

DATA = *pointers*	Each pointer contains the data variates (observed at successive times); must be set
CONTRAST = *pointers*	To save the calculated contrasts: the first variate contains the means, the second the linear polynomial contrasts, the third the quadratic polynomial contrasts etc; must be set

VPEDIGREE directive

Generates an inverse relationship matrix for use when fitting animal or plant breeding models by REML.

Options

SEX = *string token*	Possible sex categories of parents (fixed, either); default fixe
UNKNOWN = *scalar*	Value to be treated as unknown

Parameters

INDIVIDUALS = *factors*	Individuals on which data has been measured

MALEPARENTS = *factors*	Male parents of the progeny
FEMALEPARENTS = *factors*	Female parents of the progeny
INVERSE = *pointer*	Inverse relationship matrix in sparse matrix form
POPULATION = *variates*	Full list of identifiers generated from the individuals and parents

VPERMTEST procedure

Does random permutation tests for the fixed effects in a REML analysis (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (prwald, criticalwald, ownstatistics, monitoring); default prwa, crit
NTIMES = *scalar*	Number of permutation samples to make; default 99
NRETRIES = *scalar*	Maximum number of extra samples to take when some REML analyses fail to converge; default NTIMES
BLOCKSTRUCTURE = *formula*	Model formula defining any blocking to consider during the randomization; default none
EXCLUDE = *factors*	Factors in the block formula whose levels are not to be randomized
SEED = *scalar*	Seed for random number generation; default 0 continues an existing sequence or, if none, selects a seed automatically
WMETHOD = *string token*	Controls which Wald statistics are used (add, drop); default add
OWNMETHOD = *string token*	Type of test required for own statistics (twosided, greaterthan, lessthan); default twos
CIPROBABILITY = *scalar*	Probability level for the confidence interval for own statistics; default 0.95

Parameters

SAVE = *REML save structures*	Specifies the (REML) save structure of the original analysis; default * uses the SAVE structure from the most recent REML analysis
WALD = *pointers*	Wald statistics saved in a pointer with a variate for each term
CRITICALWALD = *pointers*	Saves a pointer with variates for the 5%, 1% and 0.1% significance levels containing the corresponding critical values for the fixed terms, obtained from the quantiles of the Wald statistics from the permuted data sets
PRWALD = *pointers*	Critical values for Wald statistics saved in a pointer with a scalar for each term
NNOTCONVERGED = *scalars*	Saves the number of permutations whose REML analysis failed to converge
OWNDATA = *pointers*	Data required to calculate own statistics
OWNOBSERVEDVALUES = *variates*	Saves observed values of the own statistics
OWNPROBABILITIES = *variates*	Saves probabilities for the own statistics
OWNESTIMATES = *variates*	Saves estimates for the own statistics
OWNSES = *variates*	Saves standard errors for the own statistics
OWNLOWERCIS = *variates*	Saves lower values of the confidence intervals for the own statistics
OWNUPPERCIS = *variates*	Saves upper values of the confidence intervals for the own statistics
OWNSTATISTICS = *pointers*	Saves the own statistics obtained from the permutation samples, in a pointer with a variate for each statistic

VPLOT procedure

Plots residuals from a REML analysis (S.J. Welham).

Options

RMETHOD = *string token*	Which random terms to use when calculating the residuals (final, all, notspline, stfinal, stall); default uses the

	setting from the REML statement
INDEX = *variate* or *factor*	X-variable for an index plot; default !(1,2...)
GRAPHICS = *string token*	What type of graphics to use (lineprinter, highresolution); default high
TITLE = *text*	Overall title for the plots; if unset, the identifier of the y-variate is used
SAVE = *REML save structure*	Specifies the (REML) save structure from which the residuals and fitted values are to be taken; default * uses the SAVE structure from the most recent REML analysis

Parameters

METHOD = *string tokens*	Type of graph for residuals (fittedvalues, normal, halfnormal, histogram, absresidual, index); default fitt, norm, half, hist
PEN = *scalars*, *variates* or *factors*	Pens to be used for the plots

VPOWER procedure

Uses a parametric bootstrap to estimate the power (probability of detection) for terms in a REML analysis (R.W. Payne & C.J. Brien).

Options

PRINT = *string tokens*	Controls printed output (power, nnotconverged, monitoring); default powe
VPRINT = *string tokens*	Controls the output from the REML analyses (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default * i.e. none
TERM = *formula*	Fixed term to be assessed in the analysis
UVCOVARIANCE = *symmetric matrix*	Specifies the variances and covariances of the units; default is to take this from the SAVE structure
PROBABILITY = *scalar*	Significance level at which the response is to be detected; default 0.05
TMETHOD = *string token*	Type of test to be made (fratio, wald, twosided, greaterthan, lessthan, equivalence, noninferiority); default frat
XCONTRASTS = *variate*	X-variate defining a contrast to be detected
CONTRASTTYPE = *string token*	Type of contrast (regression, comparison) default rege
CRITICALVALUE = *scalar*	Supplies a critical value for the test statistic
NBOOT = *scalar*	Number of bootstrap samples to analyse; default 500
NRETRIES = *scalar*	Maximum number of extra samples to take when some REML analyses fail to converge; default NBOOT
SEED = *scalar*	Seed for random number generation; default 0 continues an existing sequence or, if none, selects a seed automatically
METHOD = *string token*	Indicates whether to use the standard Fisher-scoring algorithm or the new AI algorithm with sparse matrix methods (Fisher, AI); default AI
MAXCYCLE = *scalar*	Sets a limit on the number of iterations in the REML analyses; default 30
FMETHOD = *string token*	Controls whether and how to calculate F statistics for fixed terms (automatic, none, algebraic, numerical); default auto
WMETHOD = *string token*	Controls which Wald statistics are saved (add, drop); default add
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm
SAVE = *vsave*	REML save structure to provide the unit-by-unit variance-covariance matrix if UVCOVARIANCE is not specified

Parameters

RESPONSE = *scalars*, *variates* or *tables*	Specifies the response to be detected
POWER = *scalars*	Saves the power (i.e. probability of detection) for RESPONSE
NCONVERGED = *scalars*	Saves the number of bootstrap samples whose REML analyses converged
NNOTCONVERGED = *scalars*	Saves the number of bootstrap samples whose REML analyses failed to converge

VPREDICT directive

Forms predictions from a REML model.

Options

PRINT = *string tokens*	What to print (description, predictions, se, sed, avesed, vcovariance); default desc, pred, se, aves
CHANNEL = *scalar*	Channel number for output; default * i.e. current output channel
MODEL = *formula*	Indicates which model terms (fixed and/or random) are to be used in forming the predictions; default * includes all the fixed terms and relevant random terms
OMITTERMS = *formula*	Specifies terms to be excluded from the MODEL; default * i.e. none
FACTORIAL = *scalar*	Limit on the number of factors or variates in each term in the models specified by MODEL or OMITTERMS; default 3
PRESENTCOMBINATIONS = *identifiers*	Lists factors for which averages should be taken across combinations that are present
WEIGHTS = *tables*	Weights classified by some or all of the factors in the model; default *
PREDICTIONS = *table* or *scalar*	To save the predictions; default *
SE = *table* or *scalar*	To save standard errors of predictions; default *
SED = *symmetric matrix*	To save standard errors of differences between predictions; default *
VCOVARIANCE = *symmetric matrix*	To save variances and covariances of predictions; default *
SAVE = *REML save structure*	Specifies the save structure from which to predict; default * i.e. that from most recent REML

Parameters

CLASSIFY = *vectors*	Variates and/or factors to classify table of predictions
LEVELS = *variates*, *scalars* or *texts*	To specify values of variates and/or levels of factors for which predictions are calculated
PARALLEL = *identifiers*	For each vector in the CLASSIFY list, allows you to specify another vector in the CLASSIFY list with which the values of this vector should change in parallel (you then obtain just one dimension in the table of predictions for these vectors)
NEWFACTOR = *identifiers*	Identifiers for new factors that are defined when LEVELS are specified

VRACCUMULATE procedure

Forms a summary accumulating the results of a sequence of REML random models (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (deviance, aic, bic, sic, dffixed, dfrandom, change, exit); default devi, aic, sic, dfra
METHOD = *string token*	How to accumulate the current analysis (add, printonly, restart); default add
INCLUDE = *string tokens*	Which constants to include that depend only on the fixed model (determinant, pi); default pi
DMETHOD = *string token*	Method to use to calculate log(determinant(X′X)) (choleski, lrv); default chol
ACCUMULATED = *pointer*	Saves the summary

Parameters

DESCRIPTION = *text*	Single-line text to describe the analysis; default lists the random terms added or deleted from the previous model
SAVE = *REML save structure*	Save structure for the REML analysis to put into the summary; default uses the save structure from the most recent REML

VRADD procedure

Adds terms from a REML fixed model into a Genstat regression (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (model, deviance, summary, estimates, correlations, fittedvalues, accumulated); default mode, summ, esti, accu
FACTORIAL = *scalar*	Limit for expansion of terms; default 3
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
SELECTION = *string tokens*	One or two criteria to be printed with the models (%variance, %ss, adjustedr2, r2, dispersion, aic, sic, bic); default %var, aic, sic

Parameter

TERMS = *formula*	Fixed terms to be added

VRCHECK procedure

Checks effects of a random term in a REML analysis (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (largeblups, stability); default larg
TERM = *formula*	Random term whose BLUPs are to be assessed; must be set
RMETHOD = *string token*	Which random terms to use to form the residuals that are subtracted from the y-variate to provide the fitted values (all, term); default all
RLIMIT = *scalar*	Limit for detection of large standardized BLUPs; if this is not set, the limit is set automatically according to the number of BLUPs
NLARGEBLUPS = *scalar*	Saves the number of large standardized BLUPs that have been detected
LARGEBLUPUNITS = *pointer*	Saves the factor levels of the large standardized BLUPs
STABILITYTEST = *pointer*	Saves the results of the Levene test for stability of the variance of the standardized BLUPs
SAVE = *REML save structure*	Specifies the analysis from which the BLUPs are to be taken; by default this will be the most recent REML

No parameters

VRDISPLAY procedure

Displays output for a REML fixed model fitted in a Genstat regression (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (model, deviance, summary, estimates, correlations, fittedvalues, accumulated); default mode, summ, esti, accu
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
SELECTION = *string tokens*	One or two criteria to be printed with the models (%variance, %ss, adjustedr2, r2, dispersion, aic, sic, bic); default %var, aic, sic

No parameters

VRDROP procedure

Drops terms in a REML fixed model from a Genstat regression (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (model, deviance, summary, estimates, correlations, fittedvalues, accumulated); default mode, summ, esti, accu
FACTORIAL = *scalar*	Limit for expansion of terms; default 3
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
SELECTION = *string tokens*	One or two criteria to be printed with the models (%variance, %ss, adjustedr2, r2, dispersion, aic, sic, bic); default %var, aic, sic

Parameter

TERMS = *formula*	Fixed terms to be dropped

VREGRESS procedure

Performs regression across variates (M.W. Patefield & D. Tandy).

No options

Parameters

Y = *pointers*	Pointers each containing a set of y-variates for each of whose units a regression is to be done
X = *pointers*	Pointer containing x-variates for each set of y-variates
SLOPE = *variates*	Variate to save the estimated slopes from each set of regressions
INTERCEPT = *variates*	Variate to save the estimated intercepts from each set of regressions

VRESIDUAL directive

Defines the residual term for a REML analysis, or the residual term for an experiment within a meta-analysis (combined analysis of several experiments)

Options

EXPERIMENT = *scalar*	Level of the EXPERIMENTS factor for which the residual is being defined
TERM = *formula*	Model term to be used as the residual
FORMATION = *string token*	Whether the structure is formed by direct product construction or by definition of the whole matrix (direct, whole); default dire
VARIANCE = *scalar*	Allows an initial estimate to be provided for the residual variance of the experiment
CONSTRAINT = *string token*	Allows the residual variance to be fixed at its initial value (fix, positive) default posi
COORDINATES = *matrix* or *variates*	Coordinates of the data points to be used in calculating distance-based models

Parameters

MODELTYPE = *string tokens*	Type of covariance model associated with the term(s), or with individual factors in the term(s) if FORMATION=direct (identity, fixed, AR, MA, ARMA, power, boundedlinear, circular, spherical, linearvariance, banded, correlation, antedependence, unstructured, diagonal, uniform, FA, FAequal) default iden
ORDER = *scalar*	Order of model
HETEROGENEITY = *string token*	Heterogeneity for correlation matrices (none, outside); default none
METRIC = *string token*	How to calculate distances when MODELTYPE=power

(cityblock, squared, euclidean); default city

FACTOR = *factors* — Factors over which to form direct products

MATRIX = *identifiers* — To define matrix values for the term or the factors when MODELTYPE=fixed

INVERSE = *identifiers* — To define values for matrix inverses (instead of the fixed matrices themselves) when MODELTYPE=fixed

INITIAL = *identifiers* — Initial parameter values for each correlation matrix

CONSTRAINTS = *texts* — Texts containing strings none, fix or positive to define constraints for the parameters in each model

EQUALITYCONSTRAINTS = *variates* — Non-zero values in the variate indicate groups of parameters whose values are to be constrained to be equal

VRFIT procedure

Fits terms from a REML fixed model in a Genstat regression (R.W. Payne).

Options

PRINT = *string tokens* — Controls printed output (model, deviance, summary, estimates, correlations, fittedvalues, accumulated); default mode, summ, esti, accu

FACTORIAL = *scalar* — Limit for expansion of terms; default 3

DENOMINATOR = *string token* — Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss

SELECTION = *string tokens* — One or two criteria to be printed with the models (%variance, %ss, adjustedr2, r2, dispersion, aic, sic, bic); default %var, aic, sic

Parameter

TERMS = *formula* — Fixed terms to be fitted

VRKEEP procedure

Saves output for a REML fixed model fitted in a Genstat regression (R.W. Payne).

Options

FACTORIAL = *scalar* — Limit for expansion of terms; default 3

RESIDUALS = *variate* — Residuals, as specified by the RMETHOD option

FITTEDVALUES = *variate* — Fitted values

RMETHOD = *string token* — Type of residuals to form (simple, standardized); default simp

RDF = *scalar* — Residual degrees of freedom

RSS = *scalar* — Residual sum of squares

ACCUMULATED = *pointer* — Accumulated analysis-of-variance table

DENOMINATOR = *string token* — Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss

Parameters

TERMS = *formula* — Terms whose information is to be saved

ESTIMATES = *table*, *scalar* or *pointer* to *tables* or *scalars* — Estimated regression coefficients for each term

SE = *table*, *scalar* or *pointer* to *tables* or *scalars* — Standard errors of estimated regression coefficients for each term

VCOVARIANCE = *symmetric matrix* or *pointer* to *symmetric matrices* — Variances and covariances between the estimates of each term

NDF = *scalar* or *pointer* to *scalars* — Numerator degrees of freedom for each term

DDF = *scalar* or *pointer* to *scalars* — Denominator degrees of freedom for each term

VRMETAMODEL procedure

Forms the random model for a REML meta analysis (R.W. Payne).

Options

RANDOM = *formula structure*	Saves the random model
EXPERIMENTSFACTOR = *factor*	Factor defining which units are in each experiment
TERMS = *formula*	Specifies terms, if any, to be fitted over the whole data set; default * i.e. none

Parameters

EXPERIMENT = *scalars*, *variates* or *texts*	Experiments on which additional random terms are to be fitted
LOCALTERMS = *formula structures*	Random terms that are to be fitted only on the corresponding experiment
SAVEVECTORS = *pointers*	Saves the factors (and/or any variates) defined to represent the local terms on each experiment

VRPERMTEST procedure

Performs permutation tests for random terms in REML analysis (V.M. Cave).

Options

PRINT = *string tokens*	Controls printed output (summary, monitoring, vdiagnostics); default summ
VPRINT = *string tokens*	Controls the output from the REML analysis of the full and reduced models (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default * i.e. none
PLOT = *string tokens*	What graphs to plot (kerneldensity, histogram); default *
MODELDEFINITION = *pointer*	REML model definition structure, defined using the VFMODEL and VFSTRUCTURE procedures, to specify the full model; no default, must be set
RDROP = *formula*	Random term(s) to drop from the full model; no default, must be set
NTIMES = *scalar*	Number of permutations to make; default 99
NRETRIES = *scalar*	Maximum number of extra permutations to make when some REML analyses fail to converge; default NTIMES
SEED = *scalar*	Seed for random number generation; default 0 continues an existing sequence or, if none, selects a seed automatically
WINDOW = *scalar*	Window to use for the graphs; default 3

Parameters

Y = *variates*	Variates to be analysed
STATISTICS = *scalars* or *pointers*	Saves the test statistics
PROBABILITIES = *scalars* or *pointers*	Saves the p-values
TITLE = *text*	Title for the graphs
SAVE = *pointers*	Saves the test statistics and permuted values

VRSETUP procedure

Sets up Genstat regression to assess terms from a REML fixed model (R.W. Payne).

Option

SAVE = *REML save structure*	Specifies the analysis whose fixed terms are to be tested; by default this will be the most recent REML

No parameters

VRSWITCH procedure

Adds or drops terms from a REML fixed model in a Genstat regression (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (model, deviance, summary,

	estimates, correlations, fittedvalues, accumulated); default mode, summ, esti, accu
FACTORIAL = *scalar*	Limit for expansion of terms; default 3
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
SELECTION = *string tokens*	One or two criteria to be printed with the models (%variance, %ss, adjustedr2, r2, dispersion, aic, sic, bic); default %var, aic, sic

Parameter

TERMS = *formula*	Fixed terms to be added or dropped

VRTRY procedure

Tries the effect of adding and dropping individual terms from a REML fixed model in a Genstat regression (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (changes); default chan
FACTORIAL = *scalar*	Limit for expansion of terms; default 3
CHANGES = *pointer*	Saves details of the changes

Parameter

TERMS = *formula*	Fixed terms to be added or dropped

VSAMPLESIZE procedure

Estimates the replication to detect a fixed term or contrast in a REML analysis, using parametric bootstrap (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (power, replication, monitoring); default powe, repl, moni
TERM = *formula*	Fixed term to be assessed in the analysis
REPLICATES = *factor*	Factor identifying the replication in the design
TRYREPLICATION = *variate*	Replication values to try first; default !(2,4)
MAXREPLICATION = *scalar*	Maximum feasible replication; default * i.e. not defined
FIXED = *formula*	Fixed terms in the analysis; if unset, determined automatically from the most recent VCOMPONENTS
RANDOM = *formula*	Random terms in the analysis; if unset, determined automatically from the most recent VCOMPONENTS
COMPONENTS = *variate* or *scalar*	Variate of variance components of the random terms; must be set
FACTORIAL = *scalar*	Limit on the number of factors or variates in fixed terms; default 3
PROBABILITY = *scalar*	Significance level at which the term is required to be detected (assuming a one-sided test); default 0.05
POWER = *scalar*	The required power (i.e. probability of detection) of the test; default 0.9
TMETHOD = *string token*	Type of test to be made (fratio, wald, twosided, lessthan, greaterthan, equivalence, noninferiority; default frat
XCONTRASTS = *variate*	X-variate defining a contrast to be detected
CONTRASTTYPE = *string token*	Type of contrast (regression, comparison) default rege
CRITICALVALUE = *scalar*	Supplies a critical value for the test statistic
NBOOT = *scalar* or *variate*	Number of bootstrap samples to analyse, in a variate with 2 values if there is to be preliminary search, otherwise in a scalar; default 1000
NRETRIES = *scalar* or *variate*	Maximum number of extra samples to take when some REML analyses fail to converge, in a variate with 2 values if there is to be preliminary search, otherwise in a scalar; default NBOOT

SEED = *scalar*	Seed for random number generation; default 0 continues an existing sequence or, if none, selects a seed automatically
METHOD = *string token*	Indicates whether to use the standard Fisher-scoring algorithm or the new AI algorithm with sparse matrix methods (Fisher, AI); default AI
MAXCYCLE = *scalar*	Sets a limit on the number of iterations in the REML analyses; default 30
FMETHOD = *string token*	Controls whether and how to calculate F statistics for fixed terms (automatic, none, algebraic, numerical); default auto
WMETHOD = *string token*	Controls which Wald statistics are saved (add, drop); default add
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm

Parameters

RESPONSE = *scalars* or *tables*	Specifies the response to be detected
NREPLICATES = *scalars*	Number of replicates required to detect RESPONSE

VSCREEN procedure

Performs screening tests for fixed terms in a REML analysis (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (ftests, waldtests); default ftes, wald
EXCLUDEHIGHER = *string token*	Whether to exclude higher-order interactions in the conditional models (yes, no); default no
FORCED = *formula*	Terms that must always be included in the model (no tests on these terms); default *
FSAVE = *pointer*	Saves the F tests
WSAVE = *pointer*	Saves the Wald tests
SAVE = *REML save structure*	Specifies the analysis whose fixed terms are to be tested; by default this will be the most recent REML

No parameters

VSOM procedure

Analyses a simple REML variance components model for outliers using a variance shift outlier model (S.J. Welham, F.N. Gumedze & D.B. Baird).

Options

PRINT = *string tokens*	Specifies the output to be produced (fdr, outliers); default fdr, outl
VPRINT = *string tokens*	Controls the output from the REML analysis of the baseline model (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default mode, comp, Wald, cova
PLOT = *string tokens*	Controls which plots are produced (indexplots, residual); default inde, resi
INDEXPLOT = *string tokens*	Selects the index plots to produce (omega, sigma2, tsquared, lrt, method, all); default meth
TERM = *formula*	Random term to scan for outliers; default is the residual term
METHOD = *string token*	Method for calculating the statistics used to indicate an outlier (full, partial, t); default t
THRMETHOD = *string token*	Method for obtaining the threshold statistics (approximate, bootstrap); default appr for METHOD=full and boot otherwise
NBOOT = *scalar*	Number of bootstrap samples to take to form the threshold statistics; default 99 for METHOD=full and 499 otherwise
FIXED = *formula*	Fixed model terms

RANDOM = *formula*	Random model terms
CONSTANT = *string token*	How to treat the constant term (estimate, omit); default esti
FACTORIAL = *scalar*	Limit on the number of factors or covariates in each fixed term; default 3
VCONSTRAINTS = *string token*	How to constrain the variance components and the residual variance (none, positive, fixrelative, fixabsolute); default posi
INITIAL = *variate*	Initial values for the variance components; default 1
SEED = *scalar*	Seed for random number generation; default 0 continues an existing sequence or, if none, selects a seed automatically
SAVEITEMS = *string tokens*	Selects the items to save (residuals, omega, sigma2, gamma, tsquared, lrt, fdr, approxthresholds, thresholdstats, outliers, method, all); default resi, omeg, sigm, meth, fdr, outl

Parameters

Y = *variates*	Response variates
TITLE = *texts*	Specifies the title or titles to use for the plots
SAVE = *pointer*s	Saves information from the analysis of each y-variate

VSPECTRALCHECK procedure

Forms the spectral components from the canonical components, and constrains any negative spectral components to zero (C.J. Brien).

Options

PRINT = *string tokens*	Controls printed output (relationshipsmatrix, canonicalcomponentestimates, spectralcomponentestimates, nconstrainedcomponents, all); default spec
VPRINT = *string tokens*	Controls the output from the final REML refit (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default * i.e. none
INITIALMETHOD = *string token*	Whether to use the estimates from the unconstrained fit as initial values in constrained fits or the default REML initial values (remldefault, unconstrainedanalysis); default unco
MAXCYCLE = *scalar*	Sets a limit on the number of iterations in the REML analyses; default 30
TOLERANCE = *scalar*	Tolerance for zero values; default 10^{-10}
DPRINT = *string tokens*	Controls output of diagnostic information (spectralcomponents, canonicalcomponents, relationshipmatrix, all); default * i.e. none

Parameters

Y = *variates*	Response variates
CORRESPONDENCE = *matrices*	Upper-triangular matrix giving the spectral components in terms of the canonical components
SPECTRALESTIMATES = *variates*	Saves estimates of the spectral components
CANONICALESTIMATES = *variates*	Saves estimates of the canonical components
NCONSTRAINEDCOMPONENTS = *scalars*	Saves the number of spectral components constrained to zero, returns a missing value if some components could not be constrained
EXIT = *scalars*	Exit status of the final REML refit
SAVE = *REML save structures*	Supplies the save structure from the prior analysis of each Y variate; this need not be set, if that was the most recent REML analysis

VSPREADSHEET procedure

Saves results from a REML analysis in a spreadsheet (R.W. Payne).

Options

COMPONENTS = *variate*	Variate to contain the variance components; default components
MEANS = *pointer*	Pointer to tables to contain the means; default means
SEDMEANS = *pointer*	Pointer to matrices to contain the standard errors of differences of the means; default sedmeans
VARMEANS = *pointer*	Pointer to matrices to contain the variance-covariance matrices of the means; default varmeans
EFFECTS = *pointer*	Pointer to tables to contain the effects; default effects
SEDEFFECTS = *pointer*	Pointer to matrices to contain the standard errors of differences of the effects; default sedeffects
VAREFFECTS = *pointer*	Pointer to matrices to contain the variance-covariance matrices of the effects; default vareffects
REPLICATIONS = *pointer*	Pointer to tables of replications; default replication
WALDTABLE = *pointer*	Pointer to a text and variates containing the information in the table of tests for fixed effects; default waldtable
PTERMS = *formula*	Terms (fixed or random) for which effects or means are to be saved; default * implies all the fixed terms
FMETHOD = *string token*	Controls whether and how to calculate F-statistics for fixed terms (automatic, none, algebraic, numerical); default auto
SPREADSHEET = *string tokens*	What to include in the spreadsheet (components, waldtable, effects, sedeffects, vareffects, means, sedmeans, varmeans, replications); default comp, wald, mean, sedm, repl
OUTFILENAME = *texts*	Name of Genstat workbook file (.gwb) or Excel (.xls or .xlsx) file to create
SAVE = *REML save structure*	Specifies which REML analysis to save; default * i.e. most recent one

No parameters

VSTATUS directive

Prints the current model settings for REML.

Option

PRINT = *string tokens*	What to print (model); default mode

No parameters

VSTRUCTURE directive

Defines a variance structure for random effects in a REML model.

Options

TERMS = *formula*	Model terms for which the covariance structure is to be defined
FORMATION = *string token*	Whether the structure is formed by direct product construction or by definition of the whole matrix (direct, whole); default dire
CORRELATE = *string token*	Whether to impose correlation across the model terms if several are specified (none, positive, unrestricted); default none
CINITIAL = *scalars*	Initial values for covariance matrix across terms
COORDINATES = *matrix* or *variates*	Coordinates of the data points to be used in calculating distance-based models

Parameters

MODELTYPE = *string tokens*	Type of covariance model associated with the term(s), or with individual factors in the term(s) if FORMATION=direct

(identity, fixed, AR, MA, ARMA, power, boundedlinear, circular, spherical, linearvariance, banded, correlation, antedependence, unstructured, diagonal, uniform, FA, FAequal) default iden

ORDER = *scalar* Order of model

HETEROGENEITY = *string token* Heterogeneity for correlation matrices (none, outside); default none

METRIC = *string token* How to calculate distances when MODELTYPE=power (cityblock, squared, euclidean); default city

FACTOR = *factors* Factors over which to form direct products

MATRIX = *symmetric matrices*, *diagonal matrices* or *pointers*
Defines matrix values for a term or the factors when MODELTYPE=fixed

INVERSE = *symmetric matrices*, *diagonal matrices* or *pointers*
Define values for matrix inverses (instead of the fixed matrices themselves) when MODELTYPE=fixed

DISTANCES = *symmetric matrices* Symmetric matrix of pre-formed distances to be used in distance-based models of order one

COORDINATES = *matrices*, *variates* or *pointers*
Specifies coordinates of each factor level to be used in calculating distance-based models

INITIAL = *scalars*, *variates*, *matrices*, *symmetric matrices* or *pointers*
Initial parameter values for each correlation matrix (supplied in the structures appropriate for the model concerned)

CONSTRAINTS = *texts* Texts containing strings none, fix or positive to define constraints for the parameters in each model

EQUALITYCONSTRAINTS = *variates* Non-zero values in the variate indicate groups of parameters whose values are to be constrained to be equal

†VSUMMARY procedure

Summarizes a variate, with classifying factors, into a data matrix of variates and factors (D.B. Baird).

Options

PRINT = *string token* What to print (summaries); default * i.e. none

CLASSIFICATION = *factors* Factors classifying the summary groups

NEWCLASSIFICATION = *factors* Factors in the data matrix to classify the output variates

REDEFINE = *string token* Whether to redefine the CLASSIFICATION factors and DATA variates, if NEWCLASSIFICATION or NEWDATA are not set (yes, no); default no

CMETHOD = *string token* How to form levels for carried factors (median, minimum, maximum); default median

MVINCLUDE = *string token* Whether to include factor combinations with no observations in summaries (yes, no); default no

WARNING = *string token* What warnings to output (carry); default carry warns when carried factors have varying values within classification groups

Parameters

DATA = *variates*, *factors* or *pointers* Data to be summarized

STATISTIC = *texts* What statistic to calculate (carry, counts, sums, totals, nobservations, means, minima, maxima, variances, quantiles, sds, skewness, kurtosis, semeans, seskewness, sekurtosis); default mean

PERCENTILE = *scalars* or *variates* Percentile to be used for quantiles; default 50.

NEWDATA = *variates*, *factors* or *pointers* Summary statistics as variates or factors for STATISTIC=carry

VSURFACE procedure

Fits a 2-dimensional spline surface using REML, and estimates its extreme point (D.B. Baird).

Options

PRINT = *string tokens*	What to print (description, model, components, effects, vcovariance, deviance, waldtests, extreme, confidence, monitoring); default desc, mode, comp, wald, extr
PLOT = *string tokens*	What to plot (contour, surface); default * i.e. nothing
BASIS = *string token*	Spline basis to use (thinplate, pspline, penalizedspline); default thin
KNOTS = *scalar*, *variate* or *pointer*	Knots to be fitted in spline model, if a scalar, this is the total number of knots to be fitted; if a variate of length 2, this is the number of knots in the X1 and X2 directions; and if a pointer to 2 variates, these are the values for knots in the X1 and X2 directions; default 16
PENALTYMETHOD = *string token*	Which tensor spline penalty to use (isotropic, semiconstrained, unconstrained); default unco
DEGREE = *scalars*	Degree of polynomial used to form the underlying spline; default 1 for METHOD=penalizedspline and 3 for METHOD=pspline
DIFFORDER = *scalars*	Differencing order for p-spline penalty; default 2
EXTREME = *scalars*	Saves the estimated value of y at the extreme point
SEEXTREME = *scalars*	Saves the standard error of the estimated value of y at the extreme point
TYPEEXTREME = *string token*	Type of extreme to be identified (minimum, maximum); default maxi
PREDICTIONS = *matrix* or *pointer*	Saves predictions
PMETHOD = *string tokens*	Method of returning predictions (grid, list); default grid
NBOOT = *scalars*	The number of bootstrap samples to estimate standard errors and confidence limits; default 100
NRETRIES = *scalars*	Number of times to retry bootstrap sampling when the REML fit fails; default is the same value as NBOOT
SEED = *scalars*	The seed used to initialize the randomization in the bootstrap sampling; default 0 continues an existing sequence or, if none, selects a seed automatically
CIPROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
COLOURS = *text* or *variate*	Colours for the plots

Parameters

Y = *variates*	Y-variate to which the spline surface will be fitted
X1 = *variates*	The first X-variate which defines the spline surface
X2 = *variates*	The second X-variate which defines the spline surface
ESTIMATE = *variates*	Estimated value of each x-variate at the extreme point
SE = *variates*	Standard error of the estimated value of each x-variate at the extreme point
LEVELS = *scalars*, *variates* or *pointers*	Number of values or values at which to evaluate each X for plots and predictions
TITLE = *texts*	Title to use for graphs; default automatically made from the variate identifiers used for Y, X1 and X2.
WINDOW = *scalars*	Window number for the graphs; default 3
SCREEN = *string tokens*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
EXIT = *scalars*	Exit code from the REML fit and location of extreme point

VTABLE procedure

Forms a variate and set of classifying factors from a table (P.W. Goedhart).

No options

Parameters

TABLE = *tables*	Tables to be copied
VARIATE = *variates*	New variate to contain the body of each table
CLASSIFICATION = *pointers*	Pointer containing the factors by which each new variate is classified
LABELS = *texts*	Labels for the new variate, indicating the values of the classifying factors corresponding to each of its units

VTCOMPARISONS procedure

Calculates comparison contrasts within a multi-way table of predicted means from a REML analysis (R.W. Payne).

Options

PRINT = *string tokens*	Controls printed output (contrasts, Waldtests); default cont
MODEL = *formula*	Indicates which model terms (fixed and/or random) are to be used in forming the predictions; default * includes all the fixed terms and relevant random terms
OMITTERMS = *formula*	Specifies terms to be excluded from the MODEL; default * i.e. none
FACTORIAL = *scalar*	Limit on the number of factors or variates in each term in the models specified by MODEL or OMITTERMS; default 3
PRESENTCOMBINATIONS = *identifiers*	Lists factors for which averages should be taken across combinations that are present
WEIGHTS = *tables*	One-way tables of weights classified by factors in the model; default *
GROUPS = *factors*	Groups for which to estimate each contrast
DFMETHOD = *string token*	Specifies which degrees of freedom to use for the comparisons (fddf, given, tryfddf, none); default fddf
DFGIVEN = *scalar*	Specifies the number of degrees of freedom to use for the comparisons when DFMETHOD=given, or if d.d.f. are unavailable when DFMETHOD=tryfddf
FMETHOD = *string token*	Controls how to calculate denominator degrees of freedom for the F-statistics, if these are not already available in the REML save structure (automatic, algebraic, numerical); default auto
SAVE = *identifier*	REML save structure for the analysis from which the comparisons are to be calculated

Parameters

CONTRAST = *tables*	Defines the comparisons to be estimated
ESTIMATE = *scalars* or *variates*	Saves the estimated contrasts
SE = *scalars* or *variates*	Saves standard errors of the contrasts
VCOVARIANCE = *symmetric matrices*	Save the variance-covariance matrices of contrasts estimated for GROUPS
STATISTIC = *scalars* or *variates*	Saves saves the test statistic (t or Wald)
DF = *scalars* or *variates*	Saves estimated numbers of residual degrees of freedom of the contrasts
PROBABILITY = *scalars* or *variates*	Saves the probabilities of the contrasts
WALD = *scalars*	Wald statistic for each comparison, combining the tests within groups
FSTATISTIC = *scalars*	F statistics for each comparison, if available, combining the tests within groups
NDF = *scalars*	Numerator d.f. for FSTATISTIC
DDF = *scalars*	Denominator d.f. for FSTATISTIC

VUVCOVARIANCE procedure

Forms the unit-by-unit variance-covariance matrix for specified variance components in a REML model (R.W. Payne).

Options

FIXED = *formula*	Fixed model terms; default *
CONSTANT = *string token*	How to treat the constant term (estimate, omit); default esti
FACTORIAL = *scalar*	Limit on the number of factors or covariates in each fixed term; default 3
SEED = *scalar*	Seed for the random numbers used to generate a dummy y-variate; default 12345

Parameters

RANDOM = *formula structures*	Random model terms
COMPONENTS = *variates*	Values for the variance components and residual variance
UVCOVARIANCE = *symmetric matrices*	Saves the unit-by-unit variance-covariance matrices

WADLEY procedure

Fits models for Wadley's problem, allowing alternative links and errors (D.M. Smith).

Options

PRINT = *string tokens*	Controls printed output (deviance, estimates, correlations, monitoring); default devi, esti
DISTRIBUTION = *string token*	Distribution of the response variate (poisson, negativebinomial, qlnegativebinomial, qlscaledpoisson); default pois
LINK = *string token*	Link transformation (logit, probit, complementaryloglog, cauchit); default logi
TERMS = *formula*	Model to be fitted
CONTROL = *factor*	Factor to distinguish the control, or zero, dose (level 1) from the other treatments (level 2)
MAXIMAL = *factor*	Factor to define the maximal model i.e. with a level for every combination of values of the variates and factors in TERMS
RMETHOD = *string token*	Type of residuals to be formed (deviance, Pearson); default devi

Parameters

Y = *variates*	Response variate for each fit
RESIDUALS = *variates*	Variate to save the residuals from each fit
FITTEDVALUES = *variates*	Variate to save the fitted values from each fit

WILCOXON procedure

Performs a Wilcoxon Matched-Pairs (Signed-Rank) test (S.J. Welham, N.M. Maclaren & H.R. Simpson).

Option

PRINT = *string tokens*	Output required (test, ranks): test gives the relevant test statistics, ranks prints out the signed ranks for the vector of differences; default test

Parameters

DATA = *variates*	Variates holding the differences between each pair of samples
RANKS = *variates*	Variate to save the signed ranks
STATISTIC = *scalars*	Scalar to save the value of the test statistic
PROBABILITY = *scalars*	Saves the probability for each test statistic
SIGN = *scalars*	Scalar to indicate the sign of the total sum of the signed ranks: 1 if the sum is positive, 0 otherwise

WINDROSE procedure

Plots rose diagrams of circular data like wind speeds (P.W. Goedhart & R.W. Payne).

Options

PRINT = *string token*	What to print (table); default * i.e. nothing
SEGMENT = *scalar*	Width of sectors (in degrees) into which to group an ANGLES variates before plotting; default 20
MSEGMENT = *scalar*	Defines the centre (in degrees) of the sectors; default 0
INTERVALS = *scalar* or *variate*	Scalar to define the intervals at which to summarize the data values, or a variate defining the boundaries between the intervals; default * i.e. determined automatically
%INTERVAL = *scalar*	Interval (on the percent scale) between the circles drawn to provide a scale on the diagram; default * i.e. determined automatically
COLOURS = *text* or *variate*	Colours to shade the triangles segment for each interval; default * sets suitable colours automatically
SCREEN = *string token*	Whether to clear screen before displaying the graphs (keep, clear); default clea

Parameters

DATA = *variates*	Data values
ANGLES = *factors* or *variates*	Directions of the data values
TITLE = *text*	Title for the graph; default * i.e. identifier of the DATA variate
WINDOW = *scalar*	Window for the graph; default 3

WORKSPACE directive

Accesses private data structures for use in procedures.

No options

Parameters

NAME = *texts*	Texts, each containing a single line, to give the names used to identify the private data structures
DUMMY = *identifiers*	Dummy structure to be used to refer to each private data structure

WSTATISTIC procedure

Calculates the Shapiro-Wilk test for Normality (R.W. Payne).

Option

PRINT = *string tokens*	What to print (test); default test

Parameters

DATA = *variates*	Samples of data to be tested for Normality
W = *scalars*	Saves the Shapiro-Wilk W statistic for each sample
PROBABILITY = *scalars*	Saves the probability for W under the assumption that the data are Normal

XAXIS directive

Defines the x-axis in each window for high-resolution graphics.

Option

RESET = *string token*	Whether to reset the axis definition to the default values (no, yes); default no

Parameters

WINDOW = *scalars*	Numbers of the windows
TITLE = *texts*	Title for the axis
TPOSITION = *string tokens*	Position of title (middle, end)
TDIRECTION = *string tokens*	Direction of title (parallel, perpendicular)
LOWER = *scalars*	Lower bound for axis
UPPER = *scalars*	Upper bound for axis
MARKS = *scalars* or *variates*	Distance between each tick mark (scalar) or positions of the marks along the axis (variate)

MPOSITION = *string tokens*	Positioning of the tick marks on the axis (inside, outside, across)
LABELS = *texts* or *variates*	Labels at each major tick mark
LPOSITION = *string tokens*	Position of the axis labels (inside, outside)
LDIRECTION = *string tokens*	Direction of the axis labels (parallel, perpendicular)
LROTATION = *scalars* or *variates*	Rotation of the axis labels
NSUBTICKS = *scalars*	Number of subticks per interval (ignored if MARKS is a variate)
YORIGIN = *scalars*	Position on y-axis at which the axis is drawn
ZORIGIN = *scalars*	Position on z-axis at which the axis is drawn
PENTITLE = *scalar*	Pen to use to write the axis title
PENAXIS = *scalar*	Pen to use to draw the axis
PENLABELS = *scalars*	Pen to use to write the axis labels
ARROWHEAD = *string tokens*	Whether the axis should have an arrowhead (include, omit)
ACTION = *string tokens*	Whether to display or hide the axis (display, hide)
TRANSFORM = *string tokens*	Transformed scale for the axis (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default iden
LINKED = *scalars*	Linked axis whose definitions should be used for this axis in 2-dimension graphs; default * i.e. none
MLOWER% = *scalars*	How large a margin to set between the lowest x-value and the lower value of the axis, if not set explicitly by LOWER (expressed as a percentage of the range of the x-values)
MUPPER% = *scalars*	How large a margin to set between the largest x-value and the upper value of the axis, if not set explicitly by UPPER (expressed as a percentage of the range of the x-values)
DECIMALS = *scalars* or *variates*	Number of decimal places to use for numbers printed at the marks
DREPRESENTATION = *scalars*, *variates* or *texts*	Format to use for dates and times printed at the marks
VREPRESENTATION = *string tokens*	Format to use for numbers printed at the marks (decimal, engineering, scientific); default deci
YOMETHOD = *string tokens*	Method to use to set the position of the origin on the y-axis if not set explicitly by YORIGIN (upper, lower, center, centre)
ZOMETHOD = *string tokens*	Method to use to set the position of the origin on the z-axis if not set explicitly by ZORIGIN (upper, lower, center, centre)
REVERSE = *string tokens*	Whether to reverse the axis direction to run from upper to lower instead of the default lower to upper (yes, no); default no
SAVE = *pointers*	Saves details of the current settings for the axis concerned

XOCATEGORIES procedure

Performs analyses of categorical data from cross-over trials (D.M. Smith & M.G.Kenward).

Options

PRINT = *string token*	What to print at each fit (model, summary, accumulated, estimates, correlations, fittedvalues, monitoring); default *
PDATA = *string token*	Whether or not a display of category combination by sequence is required (yes, no); default no
METHOD = *string token*	Type of analysis for which factors are required (subject, loglinear, ownsubject, ownloglinear); default subj
CARRYOVER = *string token*	Whether or not models with carryover effects in are to be produced (yes, no); default no

Parameters

SEQUENCE = *factors*	The identifier of the sequence of treatments

RESULTS = *pointers*	Pointer containing factors (one for each period) giving the category scores observed
NUMBER = *variates*	Numbers recorded in the sequence/category combinations
SAVE = *pointers*	Saves the factors constructed to do the analysis
REUSE = *pointers*	To reuse factors saved earlier using SAVE
MODEL = *formula*	Additional terms to be fitted to model if OWNSUBJECT or OWNLOGLINEAR options used; default *

XOEFFICIENCY procedure

Calculates efficiency of estimating effects in cross-over designs (B. Jones & P.W. Lane).

Options

PRINT = *string tokens*	What reports to produce (summary, efficiency, variance, carryover, contrasts, dummyanalysis, incidence); default summ, effi, cont
NPERIODS = *scalar*	Number of periods in the design; no default
CARRYOVER = *string token*	Whether to included effects of carryover (yes, no); default no
CONTRASTTYPE = *string token*	Type of treatment contrasts if POLYNOMIAL and OWN parameters are unset (pairwise, control); default pair
INCIDENCE = *pointer*	Saves incidence matrices; default *

Parameters

SEQUENCES = *formula*	Text, variate or factor with sequence of levels of a single treatment; no default
POLYNOMIAL = *scalars*	Order of polynomials to represent each term in the SEQUENCES parameter; default *, i.e. represent effects according to OWN parameter or CONTRASTTYPE option
OWN = *matrices*	Specific contrasts for each term in the sequences parameter; default *, i.e. represent effects according to POLYNOMIAL parameter or CONTRASTTYPE option
EFFICIENCY = *symmetric matrices*, *variates* or *diagonal matrices*	Saves efficiencies; default *
VARIANCE = *symmetric matrices*, *variates* or *diagonal matrices*	Saves variances; default *

XOPOWER procedure

Estimates the power of contrasts in cross-over designs (P.W. Lane & B. Jones).

Options

PRINT = *string tokens*	What reports to produce (summary, contrasts, nonequality, equivalence, noninferiority, superiority); default summ, none
NPERIODS = *scalar*	Number of periods in the design; default 2
NREPEATS = *scalar*	Number of repeats of supplied sequences, or variate or a series of numbers to get power for multiples of a design; default 1
CARRYOVER = *string token*	Whether to include the carry-over term (yes, no); default no
CONTRASTTYPE = *string token*	Type of treatment contrasts if POLYNOMIAL and OWN parameters are unset (pairwise, control); default pair
ALPHALEVEL = *scalar*	Significance level at which to test each contrast, adjusted if necessary for multiplicity; default 0.05
DELTA = *scalar*	Tolerance for equivalence & non-inferiority tests; default 0.2231 i.e. log(1.25)
VARWITHIN = *scalar*	Variance of response within subjects; default 1
VARBETWEEN = *scalar*	Variance of response between subjects; default 1
NSIMULATIONS = *scalar*	Number of simulations; default 1000
SEED = *scalar*	Seed for random-number generator; default 0 i.e. continue from previous or use system clock
MONITOR = *string token*	What summary of power values to report every 50 simulations for each report chosen in PRINT option (minimum, mean,

median, maximum); default * i.e. no monitoring

Parameters

SEQUENCES = *texts*, *variates* or *factors*	Sequence of levels of a single treatment factor; no default
POLYNOMIAL = *scalars*	Order of polynomials to represent the treatment factor; default * i.e. represent effects according to OWN parameter or CONTRASTTYPE option
OWN = *matrices*	Specific contrasts for the treatment factor; default * i.e. represent effects according to POLYNOMIAL parameter or CONTRASTTYPE option
MEANS = *variates*	Pattern of means for each treatment level for which to establish power; default * i.e. all zero
NONEQUALITY = *symmetric matrices* or *matrices*	Structure to save calculated power values for nonequality; default *
EQUIVALENCE = *symmetric matrices* or *matrices*	Structure to save calculated power values for equivalence; default *
NONINFERIORITY = *symmetric matrices* or *matrices*	Structure to save calculated power values for noninferiority; default *
SUPERIORITY = *symmetric matrices* or *matrices*	Structure to save calculated power values for superiority; default *

YAXIS directive

Defines the y-axis in each window for high-resolution graphics.

Option

RESET = *string token*	Whether to reset the axis definition to the default values (no, yes); default no

Parameters

WINDOW = *scalars*	Numbers of the windows
TITLE = *texts*	Title for the axis
TPOSITION = *string tokens*	Position of title (middle, end)
TDIRECTION = *string tokens*	Direction of title (parallel, perpendicular)
LOWER = *scalars*	Lower bound for axis
UPPER = *scalars*	Upper bound for axis
MARKS = *scalars* or *variates*	Distance between each tick mark (scalar) or positions of the marks along the axis (variate)
MPOSITION = *string tokens*	Positioning of the tick marks on the axis (inside, outside, across)
LABELS = *texts* or *variates*	Labels at each major tick mark
LPOSITION = *string tokens*	Position of the axis labels (inside, outside)
LDIRECTION = *string tokens*	Direction of the axis labels (parallel, perpendicular)
LROTATION = *scalars* or *variates*	Rotation of the axis labels
NSUBTICKS = *scalars*	Number of subticks per interval (ignored if MARKS is a variate)
XORIGIN = *scalars*	Position on x-axis at which the axis is drawn
ZORIGIN = *scalars*	Position on z-axis at which the axis is drawn
PENTITLE = *scalars*	Pen to use to write the axis title
PENAXIS = *scalars*	Pen to use to draw the axis
PENLABELS = *scalar*	Pen to use to write the axis labels
ARROWHEAD = *string tokens*	Whether the axis should have an arrowhead (include, omit)
ACTION = *string tokens*	Whether to display or hide the axis (display, hide)
TRANSFORM = *string tokens*	Transformed scale for the axis (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default iden
LINKED = *scalars*	Linked axis whose definitions should be used for this axis in 2-

dimensional graphs; default * i.e. none

MLOWER% = *scalars* — How large a margin to set between the lowest y-value and the lower value of the axis, if not set explicitly by LOWER (expressed as a percentage of the range of the y-values)

MUPPER% = *scalars* — How large a margin to set between the largest y-value and the upper value of the axis, if not set explicitly by UPPER (expressed as a percentage of the range of the y-values)

DECIMALS = *scalars* or *variates* — Number of decimal places to use for numbers printed at the marks

DREPRESENTATION = *scalars*, *variates* or *texts* — Format to use for dates and times printed at the marks

VREPRESENTATION = *string tokens* — Format to use for numbers printed at the marks (decimal, engineering, scientific); default deci

XOMETHOD = *string tokens* — Method to use to set the position of the origin on the x-axis if not set explicitly by XORIGIN (upper, lower, center, centre)

ZOMETHOD = *string tokens* — Method to use to set the position of the origin on the z-axis if not set explicitly by ZORIGIN (upper, lower, center, centre)

REVERSE = *string tokens* — Whether to reverse the axis direction to run from upper to lower instead of the default lower to upper (yes, no); default no

SAVE = *pointers* — Saves details of the current settings for the axis concerned

YTRANSFORM procedure

Estimates the parameter lambda of a single parameter transformation (D.M. Smith).

Options

TRANSFORM = *string token* — Type of transformation (power, modulus, foldedpower, GuerreroJohnson, Aranda1, Aranda2, powerlogit); default powe

METHOD = *string tokens* — Method of evaluating transformation parameter lambda (Atkinson, Andrews, BoxCox, Robust); default boxc

K = *scalar* — Cut-off value for robust method; default *

LOWER = *scalar* — Lower limit of range of lambda; default *

UPPER = *scalar* — Upper limit of range of lambda; default *

STEPLENGTH = *scalar* — Increment of lambda; default (UPPER - LOWER)/20

LAMBDA = *scalar* — Single value of lambda; default *

FVBOUND = *string token* — Replace illegal fitted values by the corresponding boundary values (no, yes); default no

GRAPHICS = *string token* — What sort of graphics to use (lineprinter, highresolution); default high

TERMS = *formula* — Terms of model

Parameters

Y = *variates* — Response variate

NBINOMIAL = *variates* — Denominator for a binomial variate

SAVE = *pointers* — Structures to save the output

ZAXIS directive

Defines the z-axis in each window for high-resolution graphics.

Option

RESET = *string token* — Whether to reset the axis definition to the default values (no, yes); default no

Parameters

WINDOW = *scalars* — Numbers of the windows

TITLE = *texts* — Title for the axis

TPOSITION = *string tokens* — Position of title (middle, end)

TDIRECTION = *string tokens*	Direction of title (parallel, perpendicular)
LOWER = *scalars*	Lower bound for axis
UPPER = *scalars*	Upper bound for axis
MARKS = *scalars* or *variates*	Distance between each tick mark (scalar) or positions of the marks along the axis (variate)
MPOSITION = *string tokens*	Positioning of the tick marks on the axis (inside, outside, across)
LABELS = *texts*	Labels at each major tick mark
LPOSITION = *string tokens*	Position of the axis labels (inside, outside)
LDIRECTION = *string tokens*	Direction of the axis labels (parallel, perpendicular)
LROTATION = *scalars* or *variates*	Rotation of the axis labels
NSUBTICKS = *scalars*	Number of subticks per interval (ignored if MARKS is a variate)
XORIGIN = *scalars*	Position on x-axis at which the axis is drawn
YORIGIN = *scalars*	Position on y-axis at which the axis is drawn
PENTITLE = *scalars*	Pen to use to write the axis title
PENAXIS = *scalars*	Pen to use to draw the axis
PENLABELS = *scalar*	Pen to use to write the axis labels
ARROWHEAD = *string tokens*	Whether the axis should have an arrowhead (include, omit)
ACTION = *string tokens*	Whether to display or hide the axis (display, hide)
MLOWER% = *scalars*	How large a margin to set between the lowest z-value and the lower value of the axis, if not set explicitly by LOWER (expressed as a percentage of the range of the z-values)
MUPPER% = *scalars*	How large a margin to set between the largest z-value and the upper value of the axis, if not set explicitly by UPPER (expressed as a percentage of the range of the z-values)
DECIMALS = *scalars* or *variates*	Number of decimal places to use for numbers printed at the marks
DREPRESENTATION = *scalars variates* or *texts*	Format to use for dates and times printed at the marks
VREPRESENTATION = *string tokens*	Format to use for numbers printed at the marks (decimal, engineering, scientific); default deci
XOMETHOD = *string tokens*	Method to use to set the position of the origin on the x-axis if not set explicitly by XORIGIN (upper, lower, center, centre)
YOMETHOD = *string tokens*	Method to use to set the position of the origin on the y-axis if not set explicitly by YORIGIN (upper, lower, center, centre)
REVERSE = *string tokens*	Whether to reverse the axis direction to run from upper to lower instead of the default lower to upper (yes, no); default no
SAVE = *pointers*	Saves details of the current settings for the axis concerned

%CD directive

Changes current directory, PC Windows only.

No options

Parameters

DIRECTORY = text	Directory to change to
CURRENT = text	Saves new directory

%CLOSE directive

Closes the binary file opened by %OPEN.

No options or parameters

%FPOSITION directive

Returns the current position in the binary file opened by %OPEN.

No options

Parameter

scalar	Number of bytes of the current position from the start of the file

%LOG directive

Adds text into the Input Log window in the Genstat client.

No options

Parameter

text	Text to display in the Input Log window

%MESSAGEBOX directive

Display text in a dialog in the Genstat client.

Options

TITLE = *text*	Title for the dialog; default 'Genstat'
ICON = *string token*	Icon to display in the dialog (information, warning, error, question); default info

Parameter

text	text to display in the dialog

%OPEN directive

Open a binary file for use with %WRITE.

No options

Parameter

NAME = *text*	Name of file to be opened for binary output using %WRITE

%SLEEP directive

Pauses execution of the server for a time specified in seconds.

No options

Parameter

scalar	specifies the time in seconds to pause

%TEMPFILE directive

Creates a unique temporary file in the Genstat temporary folder.

No options

Parameters

PREFIX = *string*	Prefix for the filename
FILENAME = *text*	Saves the filename
INDEX = *scalar*	Saves the index number that follows the prefix in the filename

%WRITE directive

Writes values of data structures to a binary file opened by %OPEN.

Options

SEPARATOR = *scalar or text*	Separator character as a literal character or a scalar giving an ASCII code (0-255); default * i.e. none
TERMINATOR = *string token*	Terminator to use at the end of a text (null, newline) default null
POSITION = *scalar*	File position at which to write the data; default 0 writes at the current position

Parameters

DATA = *texts, scalars, variates or matrices*	Data structures to write to the file
FORMATTED = *string tokens*	Output format to use when writing the structures (bit, byte, shortint, longint, real, double, string, text, rawtext, factor); default depends on the type of data structure

NBYTES = *scalars*	Saves the number of bytes written to the file

4.2 Functions for calculations

Function name	Description
ABS(x)	the absolute value of x: |x|.
ACOS	synonym for ARCCOS.
ANG	synonym of ANGULAR.
ANGLE(y;x)	inverse tangent of y/x, result in radians in range (−π,π].
ANGULAR(p)	the angular transformation: for a percentage p (0 <p < 100), forms x = (180/pi) × arcsin(sqrt(p/100)).
ARCCOS(x)	inverse cosine of x, where −1 <= x <= 1.
ARCSIN(x)	inverse sine of x, where −1 <= x <= 1.
ARCTAN(x)	arctangent (inverse tangent) of x, result in radians.
AREA(y;x)	numerically integrates the curve running through the points specified by variates y and x using the trapezoidal method.
ASIN	synonym for ARCSIN.
ATAN	synonym for ARCTAN.
BASE(i;n)	column matrix with n rows, value one in row i and zero elsewhere.
BBELOW(t;n;m)	provides a variate containing numbers of all the nodes below node n of tree t; if m=1 this gives only the terminal nodes below n, otherwise it includes internal nodes as well.
BBRANCHES(t;n)	provides a variate containing the numbers of the branches taken on the path to node n in tree t (the result is of the same length as the results of BPATH, and includes a * as the final element, corresponding to n itself).
BDEPTH(t;x)	calculates the depths of nodes x in tree t.
BETA(a;b;x)	Beta function B(a,b) or, if x is set, regularized incomplete Beta function I(a,b,x).
BI0(x)	modified Bessel function of the first kind I_0(x).
BI1(x)	modified Bessel function of the first kind I_1(x).
BIN(x;n)	modified Bessel function of the first kind I_n(x; n).
BJ0(x)	Bessel function of the first kind J_0(x).
BJ1(x)	Bessel function of the first kind J_1(x).
BJN(x;n)	Bessel function of the first kind J^n(x; n).
BK0(x)	modified Bessel function of the second kind K_0(x).
BK1(x)	modified Bessel function of the second kind K_1(x).
BKN(x;n)	modified Bessel function of the second kind K_n(x; n).
BLUE(x)	calculates the blue components of the RGB colour values in x.
BMAXNODE(t)	provides the maximum node number in tree t.
BNBRANCHES(t;x)	provides the number of branches below nodes x in tree t (0 if n is a terminal node).
BNEXT(t;x;y)	finds the numbers of the nodes on branches y from nodes x in tree t (or * for any terminal node).
BNNODES(t)	provides the number of nodes in tree t.
BOUND(x;l;u)	sets values of x less than l to l, and values greater than u to u; missing values can be set in l or u to imply no boundary.
BPATH(t;n)	provides a variate containing the numbers of the nodes on the branch to node n in tree t (includes n itself as the final element).
BPREVIOUS(t;x)	finds the numbers of the nodes immediately above nodes x in tree t (or * for the root of the tree).
BSCAN(t;x)	finds the numbers of the nodes immediately after nodes x in

	tree t in an standard branch-by-branch order that visits each node once (or * for the root of the tree).
BTERMINAL(t;x)	finds the next terminal nodes after nodes x in tree t (or * for the node after the last terminal node).
BY0(x)	Bessel function of the second kind Y_0(x).
BY1(x)	Bessel function of the second kind Y_1(x).
BYN(x;n)	Bessel function of the second kind Y_n(x; n).
C	synonym of CONSTANTS.
CED	synonym of EDCHI.
CEILING(x)	ceiling of x: returns for each value x_j of x the least integer i such that $i \geq x_j$.
CHARACTERS(g;c)	returns a variate giving the length of each line of text t: if c is omitted or set to 0 the length is the "raw" length (with no checking for any typesetting commands); if c = 1 it is the formatted length (taking account of typesetting commands, see 1.4.2 for their syntax); finally, if c = −1 it is the number of storage units ("bytes") required to store the text (standard characters like letters and digits require only one, more complicated characters like Chinese or Thai characters may require as many as four).
CHISQ	synonym of CLCHI.
CHOLESKI(x)	the Choleski decomposition of a symmetric matrix x: such that x = LL' where L is square with upper off-diagonal elements zero.
CIRCULATE(x;s)	shifts the values of x, treating x as a circular stack. If s is omitted, values are shifted one to the right, as for s=1.
CLBETA(x;a;b)	cumulative lower probability for a beta distribution with parameters a and b.
CLBINOMIAL(j;n;p)	probability of j or fewer successes out of n binomial trials with probability of success p.
CLBVARIATENORMAL(x;y;r)	cumulative lower probability for a bivariate normal distribution with means 0, variances 1, and correlation r.
CLCHISQUARE(x;df;c)	cumulative lower probability for a non-central chi-square distribution with noncentrality parameter c; if the third parameter c is omitted, it is assumed to be zero, giving the ordinary (central) chi-square distribution.
CLF(x;df1;df2;c)	cumulative lower probability for a non-central F distribution with degrees of freedom df1 and df2, and noncentrality parameter c; if the fourth parameter c is omitted, it is assumed to be zero, giving the ordinary (central) F distribution.
CLGAMMA(x;k;t)	cumulative lower probability for a gamma distribution with shape parameter k (kappa) and scale parameter t (theta).
CLHYPERGEOMETRIC(j;l;m;n)	probability of j or fewer positive samples out of a total sample of size m from a population of size n of which l are positive (hypergeometric distribution).
CLINVNORMAL(x;m;l)	cumulative lower probability for an inverse Normal (or inverse Gaussian) distribution with mean m and reciprocal dispersion parameter l (variance is m^3/l).
CLLOGNORMAL(x)	cumulative lower probability for a lognormal distribution corresponding to a normal distribution with mean 0 and variance 1.
CLNORMAL(x;m;v)	cumulative lower probability for a Normal distribution with mean m (default 0) and variance v (default 1).
CLOGLOG(p)	takes the complementary log-log transformation of the percentages p (0 < p < 100%).
CLPOISSON(j;m)	probability of value of j or less for a Poisson distribution with

	mean `m`.
`CLSMMODULUS(x;df;n)`	cumulative lower probability for a Studentized maximum modulus distribution with degrees of freedom `df` and number of means `n`.
`CLSRANGE(x;df;n)`	cumulative lower probability for a Studentized range distribution with degrees of freedom `df` and number of means `n`.
`CLT(x;df;c)`	cumulative lower probability for a non-central Student's t distribution with degrees of freedom `df` and noncentrality parameter `c`; if the third parameter `c` is omitted, it is assumed to be zero, giving the ordinary (central) t distribution.
`CLUNIFORM(x;a;b)`	cumulative lower probability for a uniform distribution on [`a`,`b`].
`COLBIND(x;y)`	joins matrices `x` and `y` side by side.
`COLCENTRE(x)`	centres the columns of matrix `x` by subtracting their means.
`COLMEANS(x)`	mean of the non-missing elements of each row of matrix `x`.
`COLNOBSERVATIONS(x)`	number of non-missing elements in each column of matrix `x`.
`COLSUMS(x)`	sum of the non-missing elements of each column of matrix `x`.
`COL1(n)`	column matrix of 1's with `n` rows.
`CONSTANTS(g)`	provides the value of various constants, according to the contents of `g`: e (for a string of `'e'` or `'E'`), π (`'pi'` or `'PI'`), missing value (`'*'`), the conversion factor by which to multiply radians to get degrees (`'degrees'`), the conversion factor by which to multiply degrees to get radians (`'radians'`) and the number ε defined as the smallest number such that the calculation 1+ε is detectable on the computer as greater than one (`'epsilon'`).
`CORRELATION(x;y)`	if both `x` and `y` are specified, returns a scalar giving the correlation between the values of `x` and `y`; if `y` is omitted, forms a correlation matrix from a symmetric matrix `x` of sums of squares and products.
`CORRMAT`	synonym of `CORRELATION`.
`COS(x)`	cosine of `x`, for `x` in radians.
`COSH(x)`	hyperbolic cosine of `x`.
`COV`	synonym of `COVARIANCE`.
`COVARIANCE(x;y)`	returns a scalar giving the covariance between the values of `x` and `y`.
`CPUTIME(x)`	returns a scalar containing the currently used cpu time in seconds (argument `x` is ignored).
`CUBETA(x;a;b)`	cumulative upper probability for a beta distribution with parameters `a` and `b`.
`CUBINOMIAL(j;n;p)`	probability of more than `j` successes out of `n` binomial trials with probability of success `p`.
`CUBVARIATENORMAL(x;y;r)`	cumulative upper probability for a bivariate normal distribution with means 0, variances 1, and correlation `r`.
`CUCHISQUARE(x;df;c)`	cumulative upper probability for a non-central chi-square distribution with noncentrality parameter `c`; if the third parameter `c` is omitted, it is assumed to be zero, giving the ordinary (central) chi-square distribution.
`CUF(x;df1;df2;c)`	cumulative upper probability for a non-central F distribution with degrees of freedom `df1` and `df2`, and noncentrality parameter `c`; if the fourth parameter `c` is omitted, it is assumed to be zero, giving the ordinary (central) F distribution.
`CUGAMMA(x;k;t)`	cumulative upper probability for a gamma distribution with shape parameter `k` (kappa) and scale parameter `t` (theta).
`CUHYPERGEOMETRIC(j;l;m;n)`	probability of more than `j` positive samples out of a total

	sample of size m from a population of size n of which l are positive (hypergeometric distribution).
CUINVNORMAL(x;m;l)	cumulative upper probability for an inverse Normal (or inverse Gaussian) distribution with mean m and reciprocal dispersion parameter l (variance is m^3/l).
CULOGNORMAL(x)	cumulative upper probability for a lognormal distribution corresponding to a normal distribution with mean 0 and variance 1.
CUM	synonym of CUMULATE.
CUMULATE(x)	forms the cumulative sum of the values of x; i.e. x_1, x_1+x_2, $x_1+x_2+x_3$, and so on.
CUNORMAL(x;m;v)	cumulative upper probability for a Normal distribution with mean m (default 0) and variance v (default 1).
CUPOISSON(j;m)	probability of a value greater than j for a Poisson distribution with mean m.
CUSMMODULUS(x;df;n)	cumulative upper probability for a Studentized maximum modulus distribution with degrees of freedom df and number of means n.
CUSRANGE(x;df;n)	cumulative upper probability for a Studentized range distribution with degrees of freedom df and number of means n.
CUT(x;df;c)	cumulative upper probability for a non-central Student's t distribution with degrees of freedom df and noncentrality parameter c; if the third parameter c is omitted, it is assumed to be zero, giving the ordinary (central) t distribution.
CUUNIFORM(x;a;b)	cumulative upper probability for a uniform distribution on [a,b].
D	synonym of DETERMINANT.
DATE(d;m;y)	constructs the date value corresponding to day d, month m and year y.
DAY(x)	the day of month corresponding to date-time value x.
DEGREES(x)	converts angles x from radians to degrees.
DET	synonym of DETERMINANT.
DETERMINANT(x)	the determinant of a square or symmetric matrix
DIAGONAL(x;b)	form a diagonal matrix from a variate x, or takes diagonal of a square, symmetric or diagonal matrix x; b may be set if x is a matrix, to request a banded diagonal matrix of order b (returned as a square matrix with the values off the bands set to zero).
DIFFERENCE(x;s)	forms the differences of x, i.e. $x_i - x_{i-s}$; if s is omitted, first differences are formed, as for s=1
DIGAMMA(x)	digamma function of x, $\Psi(x)$.
DPRODUCT(x;y)	direct or Kronecker product of matrices x and y: $x \otimes y$.
DSUM(x;y)	direct sum of matrices x and y ($x \oplus y$); alternatively, if the second argument is omitted, x can be a pointer and the function then gives x[1] ⊕ x[2] ⊕ ... x[n].
EDBETA(p;a;b)	equivalent deviate corresponding to cumulative lower probability p for a beta distribution with parameters a and b.
EDBINOMIAL(p;n;bp)	equivalent deviate corresponding to cumulative lower probability p for a binomial distribution with n trials and probability of success bp (returns the smallest integer x such that the probability of up to x successes is greater than or equal to p).
EDCHISQUARE(p;df;c)	equivalent deviate corresponding to cumulative lower probability p for a non-central chi-square distribution with noncentrality parameter c; if the third parameter c is omitted,

	it is assumed to be zero, giving the ordinary (central) chi-square distribution.
EDF(p;df1;df2;c)	equivalent deviate corresponding to cumulative lower probability p for a non-central F distribution with degrees of freedom df1 and df2, and noncentrality parameter c; if the fourth parameter c is omitted, it is assumed to be zero, giving the ordinary (central) F distribution.
EDGAMMA(p;k;t)	equivalent deviate corresponding to cumulative lower probability p for a gamma distribution with shape parameter k (kappa) and scale parameter t (theta).
EDHYPERGEOMETRIC(p;l;m;n)	equivalent deviate corresponding to cumulative lower probability p for a hypergeometric distribution with samples of size m from a population of size n of which l are positive (returns the smallest integer x such that the probability of up to x successes is greater than or equal to p).
EDINVNORMAL(p;m;l)	equivalent deviate corresponding to cumulative lower probability p for an inverse Normal (or inverse Gaussian) distribution with mean m and reciprocal dispersion parameter l (variance is m^3/l).
EDLOGNORMAL(p)	equivalent deviate corresponding to cumulative lower probability p for a lognormal distribution corresponding to a normal distribution with mean 0 and variance 1.
EDNORMAL(p;m;v)	equivalent deviate corresponding to cumulative lower probability p for a Normal distribution with mean m (default 0) and variance v (default 1).
EDPOISSON(p;m)	equivalent deviate corresponding to cumulative lower probability p for a Poisson distribution with mean m (returns the smallest integer x such that the probability of up to x successes is greater than or equal to p).
EDSMMODULUS(p;df;n)	equivalent deviate corresponding to cumulative lower probability p for a Studentized maximum modulus distribution with degrees of freedom df and number of means n.
EDSRANGE(p;df;n)	equivalent deviate corresponding to cumulative lower probability p for a Studentized range distribution with degrees of freedom df and number of means n.
EDT(p;df;c)	equivalent deviate corresponding to cumulative lower probability p for a non-central Student's t distribution with degrees of freedom df and noncentrality parameter c; if the third parameter c is omitted, it is assumed to be zero, giving the ordinary (central) t distribution.
EDUNIFORM(p;a;b)	equivalent deviate corresponding to cumulative lower probability p for a uniform distribution on [a,b].
ELEMENTS(x;e1;e2)	forms a sub-structure of x. If x is a vector or a diagonal matrix, then only e1 should be specified; this then indicates the selected elements of x. If x is a rectangular matrix, then both e1 and e2 should be given, to specify respectively the selected rows and columns of x. For a symmetric matrix x, if the same rows and columns are to be selected (giving a symmetric matrix) then only e1 should be specified; otherwise both e1 and e2 should be given (and the result is a matrix).
EVALUES(x)	eigenvalues of x (as a diagonal matrix).
EVECTORS(x)	eigenvectors of x (as a rectangular matrix).
EXP(x)	exponential: e^x.
EXPAND(x;s)	forms a variate of length s, containing zeroes and ones; if s is omitted and the length cannot be determined from the context,

	the length of the current units structure, if any, is taken. The values in x specify the numbers of the units that are to contain the value 1.
FACTORIAL(x)	factorial of x (x!): the values in x must be non-negative, missing values are given for results that are too large to be stored.
FED	synonym of EDF.
FLOOR(x)	floor of x: returns for each value x_j of x the largest integer i such that $i \leq x_j$.
FPROBABILITY	synonym of CLF.
FRACTION(x)	fractional part of x i.e. x-INTEGER(x).
FRATIO	synonym of CLF.
GAMMA(a;x)	Gamma function, Γ(a) for a>0 or, if x is set, lower incomplete Gamma function γ(a,x).
GCONSTANTS(g)	provides type numbers of Genstat data structures. The string g can therefore be either 'scalar', 'factor', 'text', 'variate', 'matrix', 'diagonalmatrix', 'symmetricmatrix', 'table', 'asave', 'tsave', 'expression', 'formula', 'dummy', 'pointer', 'lrv', 'sspm', 'tsm', 'rsave', 'tree', or 'vsave'.
GETFIRST(g)	gives a variate containing the position of the first non-space character in each string of the text g.
GETLAST(g)	gives a variate containing the position of the last non-space character in each string of the text g.
GETPOSITION(g1;g2;x)	for each unit, if the string in the text g2 occurs as a substring of the string in the text g1, this returns the position at which the substring starts; otherwise it returns the value zero. The text g2 may contain a single string (to be checked against every string of g1). The structure x (scalar or variate) supplies a logical value to indicate whether to ignore the case of any letters; if x is omitted, the logical is assumed to be false (case not ignored).
GINVERSE(x)	Moore-Penrose generalized inverse of x.
GRAY(x)	calculates RGB colour values for the values on the gray (grey) scale in x.
GRBETA(n;a;b)	generates n pseudo-random numbers from a Beta distribution with parameters a and b.
GRBINOMIAL(n;t;p)	generates n pseudo-random numbers from a Binomial distribution with t trials and probability p.
GRCHISQUARE(n;df;c)	generates n pseudo-random numbers from a chi-square distribution with degrees of freedom df and non-centrality parameter c (default c=0).
GREEN(x)	calculates the green components of the RGB colour values in x.
GREY(x)	calculates RGB colour values for the values on the gray (grey) scale in x.
GRF(n;df1;df2;c)	generates n pseudo-random numbers from an F distribution with df1 and df2 degrees of freedom, and non-centrality parameter c (by default c1=0).
GRGAMMA(n;k;t)	generates n pseudo-random numbers from a Gamma distribution with shape parameter k (kappa) and scale parameter t (theta).
GRHYPERGEOMETRIC(n;l;m;p)	generates n pseudo-random numbers from a Hypergeometric distribution representing the number of positive values or successes in samples of size m from a population of size p of which l are positive.

GRLOGNORMAL(n;m;v)	generates n pseudo-random numbers from a lognormal distribution such that log(x) has a Normal distribution with mean m and variance v.
GRNORMAL(n;m;v)	generates n pseudo-random numbers from a Normal distribution with mean m (default 0) and variance v (default 1).
GRPOISSON(n;m)	generates n pseudo-random numbers from a Poisson distribution with mean m.
GRSAMPLE(n; v; p)	forms a variate of size n by sampling with replacement from variate v with probabilities (or relative weights) p; if p is omitted, the probabilities are assumed to be equal; if v is omitted, sampling is from a variate containing the integers 1...n.
GRSELECT(n; v; r)	forms a variate of size n by sampling from a population defined as NEXPAND(r; v); if r is omitted, the population contains just one of each element of v; if v is omitted, sampling is from a variate containing the integers 1...n.
GRT(n;df;c)	generates n pseudo-random numbers from a Student's t distribution with degrees of freedom df and non-centrality parameter c (default c=0).
GRUNIFORM(n;a;b)	generates n pseudo-random numbers from a uniform distribution on [a,b].
HOURS(x)	the number of hours during the day corresponding to x (i.e. the number of hours recorded on a 24 hour clock at date-time value x).
I	synonym of INVERSE.
IANGULAR(x)	gives the inverse of the angular transformation (result in percentages).
ICLOGLOG(x)	gives the inverse of the complementary log-log transformation (result in percentages).
IDENTITY(n)	identity matrix of order n (returned as a diagonal matrix).
ILOGIT(x)	gives the inverse of the logit transformation (result in percentages).
INT	synonym of INTEGER.
INTEGER(x)	integer part of x: [x].
INV	synonym of INVERSE.
INVERSE(x)	the inverse of a non-singular square, symmetric or diagonal matrix x.
IMBEQUALIZE(r;l;u)	performs histogram equalization of brightness of RGB image in matrix r; scalar l specifies the lower threshold and scalar h specifies the upper threshold.
IMBLUR(r;b)	blurs the RGB image in matrix r by the amount specified in scalar b (0<b<100; default 2).
IMCEQUALIZE(r;l;u)	performs an independent histogram equalization of colours in RGB image in matrix r; scalars l and h specify the lower and upper threshold.
IMBRIGHTNESS(r;l;h;m)	modifies brightness of the RGB image in matrix r, setting pixels in each channel with brightness less than l (default 0) to 0 and those brighter than h (default 255) to 255; m defines mode of adjustment (default 0 stretches brightness and 1 distributes brightness evenly across the range).
IMBSTRETCH(r;l;u;m)	performs a histogram stretch of brightness in RGB image in matrix r; scalar l (default 0) specifies percentage of pixels to set to 0 (i.e. black), scalar h (default 0) specifies percentage of pixels to set to white, and scalar m (0≤m≤255; default 128) specifies the colour value in each channel to be set to the middle intensity.

`IMCONTRAST(r;c;b)`	modifies contrast and brightness of RGB image in matrix `r`; `c` (−1≤c≤1; default 0 i.e. no adjustment) defines adjustment to the contrast, and `b` (−1≤b≤1; default 0 i.e. no adjustment) defines adjustment to the brightmess.
`IMCREPLACE(r;c;d;t)`	replaces colour `c` in RGB image in matrix `r` with colour `d`, using tolerance `t` (default 0).
`IMCSTRETCH(r;l;h;m)`	performs a histogram stretch of the individual colours in RGB image in matrix `r`; scalar `l` (default 0) specifies percentage of pixels to set to 0 (i.e. black), scalar `h` (default 0) specifies percentage of pixels to set to white, and scalar `m` (0≤m≤255; default 128) specifies the colour value in each channel to be set to the middle intensity.
`IMDESPECKLE(r)`	despeckles the RGB image in matrix `r`.
`IMELLIPSE(r;cx;cy;hr;vr;c;cf;p)`	draws an ellipse with centre (`cx`, `cy`), horizontal radius `hx` (default 40), vertical radius `vr` (default 40), colour `cl`, fill colour `cf` (default 0) and opacity `p` (0≤p≤1, where 0 is transparent and default of 1 is solid) on RGB image in matrix `r`.
`IMEMBOSS(r;b;t;a;e;d)`	embosses RGB image in matrix `r`; matrix `b` specifies a "bump map" defining the peaks and valleys in the output image (typically this is a grey scale version of `r`); matrix `t` defines the texture to apply to the input matrix; scalar `a` gives the angle of the light source in radians; scalar `e` is the elevation of the light source in radians; scalar `d` defines the depth of the effect.
`IMGAMMA(r;g)`	applies gamma correction `g` (g≥0; default 1.5) to the brightness of RGB image in matrix `r`; g<1 decreases brightness, and g>1 increases brightness.
`IMGBLUR(r;s)`	applies a Gaussian blur with standard deviation `s` to RGB image in matrix `r`.
`IMGRAYSCALE(r)` or `IMGREYSCALE(r)`	convert RGB image in matrix `r` to grey scale.
`IMHFLIP(r)`	performs a horizontal flip on RGB image in matrix `r`.
`IMLINE(r;x1;y1;x2;y2;c)`	draws a line from point (`x1`, `y1`) to (`x2`, `y2`) in colour `c` on RGB image in matrix `r`.
`IMMCONVOLUTION(r;f;i;cr;cg;cb;m)`	applies the convolution filter in matrix `f` to RGB image in matrix `r` ; scalar `i` (default 1) defines intensity parameter; scalars `cr`, `cg` and `cb` contain 0 or 1 (default) according to whether red, green and blue channels, respectively, are to be modified. If mode defined by scalar `m` is 0 (default), the new value at each point is `i` multiplied by the sum of the values at the point and nearby points multiplied by the convolution matrix. Alternatively, if m=1 (default), the new value at each point is the current value at the point minus i multiplied by the sum of the values at the point and nearby points multiplied by the values in the convolution matrix.
`IMMEDIANFILTER(r)`	performs a median filter on the RGB image in a matrix `r`.
`IMOVERLAY(rt;rb;m;mp;p;x;y)`	overlays RGB image in matrix `rt` over RGB image in matrix `rb`; `m` controls how images are blended (0 = fast blend, 1 = slower, more accurate blend, 2 = pixels combined with logical `AND`, 3 = pixels combined with logical `OR`, 4 = pixels combined with logical `XOR`, 5 = output pixel is maximum of top and bottom as in Photoshop "Lighten", 6 = output pixel is minimum of top and bottom as in Photoshop "Darken", 7 = output pixel is sum of top and bottom, 8 = output pixel is difference of top and bottom, 9 = if top > `mp`, output top, 10 = if top < `mp`, output top, 11 = absolute value of the difference of top and bottom, 12 = take top × bottom / maximum component, 13 = take top × bottom × ModeParameter /

	maxComponent, 14 = screen, 15 = define bottom to be bottom + top – mp, 16 = define bottom to bottom – top – mp, 17 = pixels combined with logical NAND, 18 = pixels combined with logical NOR, 19 = pixels combined with logical NXOR/XNOR, 20 = color dodge, 21 = color burn, 22 = soft dodge, 23 = soft burn, 24 = Photoshop "overlay", 25 = soft light, 26 = hard light, 27 = XFader reflect, 28 = XFader glow, 29 = XFader freeze, 30 = XFader heat); p defines the opacity of the blended image; and (x, y) specifies the position of bottom left-hand corner of the top image on the bottom image.
IMPUSH(r;x1;y1;x2;y2)	applies a point-to-point warp on RGB image in matrix r, "pushing" point (x1, y1) to (x2, y2).
IMRECTANGLE(r;x1;y1;x2;y2;c)	colours rectangle with bottom left corner (x1, y1) and top right corner (x2, y2) in RGB image in matrix r to be colour c.
IMROTATE(r;a;b)	rotates RGB image in matrix r; a is the angle in radians (default π/2); b is the background colour to put into the (blank) corners.
IMSATURATE(r;s)	adjusts the saturation level of RGB image in matrix r according to the value of scalar s (default 1.1): when s>1 the saturation is increased, when 0<s<1 saturation is decreased, and when s<0 photo-negative is generated.
IMSHARPEN(r;s)	sharpens RGB image in matrix r by the amount specified in scalar s (0<s<100; default 2).
IMSIZE(r;w;h;m)	changes the size of RGB image in matrix r to have width w and height h; m selects the algorithm to use to assign colours in the new image: 0 = box filter, 1 = triangle filter, 2 = Hamming filter, 3 = Gaussian filter, 4 = bell filter, 5 = B-spline filter, 6 = cubic 1 filter, 7 = cubic 2 filter, 8 = Lanczos3 filter, 9 = Mitchell filter, 10 = sinc filter, 11 = Hermite filter, 12 = Hanning filter, 13 = Catrom filter, 14 = fast area-average, 15 = area-average, 16 = bi-linear interpolation, 17 (default) = bi-cubic interpolation, 18 = nearest neighbour.
IMSSTRETCH(r;l;h;m)	performs a histogram stretch of the saturation in RGB image in matrix r; scalar l (default 0) specifies percentage of pixels to set to 0 (i.e. black), scalar h (default 0) specifies percentage of pixels to set to white, and scalar m (0≤m≤255; default 128) specifies the colour value in each channel to be set to the middle intensity.
IMSTEXT(r;st;c;fh;y1;x1;y2;x2;ft;tr;sm)	draws the text in string st with height fh, font ft, colour c, transparency tr and smoothness sm (sm=1 for none, or 2 or 4) within the bounding rectangle with top left corner at (x1, y1) and bottom right corner at (x2, y2) on RGB image in matrix r.
IMTEXT(r;st;c;fh;y1;x1;y2;x2;ft)	draws the text in string st with height fh, font ft and colour c within the bounding rectangle with top left corner at (x1, y1) and bottom right corner at (x2, y2) on RGB image in matrix r.
IMUNSHARPEN(r;t;a;s)	applies an unsharp mask to RGB image in matrix r: this first applies a Gaussian blur with standard deviation s; it then finds the difference between pixels in the blurred image and in the original and, if this is greater than t in each channel, it adds the amount specified by scalar a multiplied by the difference from the original value.
IMVFLIP(r)	performs a vertical flip on RGB image in a matrix r.
IMXSHEAR(r;x;b)	shears RGB image in matrix r by moving the top of the image |x| pixels to the right (x>0) or left (x<0); the blank parts of the new image are given (background) colour b.

`IMYSHEAR(r;y;b)` shears RGB image in matrix `r` by moving the right-hand side of the image |`y`| pixels up or down; the blank parts of the new image are given (background) colour `b`.

`IM3CONVOLUTION(r;f;i;cr;cg;cb;d)` applies convolution filter in the 3×3 matrix `f` to RGB image in matrix `r`; scalar `i` (default 1) defines intensity parameter; scalars `cr`, `cg` and `cb` contain 0 or 1 (default) according to whether red, green and blue channels, respectively, are to be modified. If the "feedback" defined by scalar `d` is 0 (default), the new value at each point is `i` multiplied by the sum of the values at the point and nearby points multiplied by the convolution matrix. Alternatively, if `d`=i (default), the new value at each point is calculated by taking (1-`i`) multiplied by the current value at the point, and then subtracting `i` multiplied by the sum of the values at the point and nearby points multiplied by the values in the convolution matrix.

`IPROBIT(x)` gives the inverse of the probit transformation (result in percentages).

`KRONECKER` synonym for `DPRODUCT`.

`KURTOSIS(x)` kurtosis of the non-missing values in `x`.

`LEAPYEAR(x)` returns 1 if the year corresponding to date-time value `x` is a leap year, 0 otherwise.

`LEVELS(f)` forms a variate containing the levels of the factor `f`.

`LLB` synonym of `LLBINOMIAL`.

`LLBINOMIAL(x;n;p)` log-likelihood function for the Binomial distribution; `n` is the sample size and `p` the mean proportion (or the probability).

`LLG` synonym of `LLGAMMA`.

`LLGAMMA(x;k;t)` log-likelihood function for the Gamma distribution with shape parameter `k` (kappa) and scale parameter `t` (theta).

`LLN` synonym of `LLNORMAL`.

`LLNORMAL(x;m;v)` log-likelihood function for the Normal distribution; `m` is the mean and `v` the variance.

`LLP` synonym of `LLPOISSON`.

`LLPOISSON(x;m)` log-likelihood function for the Poisson distribution; `m` is the mean.

`LNFACTORIAL(x)` log of `x`! for non-negative integer values `x`.

`LNGAMMA(x)` log-Gamma function, $\log_e(\Gamma(x))$, for `x`>0.

`LOG(x)` natural logarithm of `x`, for `x` > 0.

`LOG10(x)` logarithm to base 10 of `x`, for `x` > 0.

`LOGIT(p)` takes the logit transformation log(`p`/(100-`p`)) of the percentages `p` (0 < `p` < 100%).

`LSVECTORS(x)` matrix of vectors from the left-hand side of a singular-value decomposition of `x`.

`LTPRODUCT(x;y)` left transposed product of `x` and `y`: a more efficient way of calculating `TRANSPOSE(x)*+y`.

`LTRIANGLE(m;d)` returns the lower triangle of square matrix `m`, as a square matrix with the upper triangular set to zero; putting `d`=1 (default) indicates that the diagonal is to be included, while putting `d`=0 excludes the diagonal.

`MAT0` synonym of `MZERO`.

`MAT1(r;c)` matrix of ones of size `r` by `c`.

`MAX` synonym of `MAXIMUM`.

`MAXIMUM(x)` finds the maximum of the values in `x`.

`MAXPOSITION(x)` finds the position of the first instance of the maximum value within `x`. For a variate this is the number of the unit containing the maximum. For a matrix the row of the maximum value can

	then be calculated as
	`row = INTEGER((MAXPOSITION(x)-1)/NROWS(x)) + 1`
	and the column as
	`col = MAXPOSITION(x) - NROWS(x)*(row-1)`
	For a symmetric matrix, the column is
	`col = INTEGER((SQRT(8*MAXPOSITION(x)+1)+1)/2)`
	and the row is
	`row = MAXPOSITION(x) - col*(col-1)/2`

`MBASE(r;c;i;j)`	matrix of size `r` by `c` which is zero, except for position(s) `i,j` which are set to one.
`MCENTRE(m)`	doubly centres the matrix `m` so that its rows and columns have mean zero.
`MEAN(x)`	forms the mean of the values of `x`.
`MED`	synonym of `MEDIAN`.
`MEDIAN(x)`	finds the median of the values in `x`.
`MEXP(m)`	calculates the matrix exponential of `m`.
`MFRACTION(x;p;m)`	returns the period within a month that date-time value `x` belongs to; `p` is the length of the period (e.g. 5 for pentade, 10 for decade), and `m` is the starting month (default 1).
`MIN`	synonym of `MINIMUM`.
`MINIMUM(x)`	finds the minimum of the values in `x`.

`MINPOSITION(x)`	finds the position of the first instance of the minimum value within `x`. For a variate this is the number of the unit containing the minimum. For a matrix the row of the minimum value can then be calculated as
	`row = INTEGER((MINPOSITION(x)-1)/NROWS(x)) + 1`
	and the column as
	`col = MINPOSITION(x) - NROWS(x)*(row-1)`
	For a symmetric matrix, the column is
	`col = INTEGER((SQRT(8*MINPOSITION(x)+1)+1)/2)`
	and the row is
	`row = MINPOSITION(x) - col*(col-1)/2`

`MINSERT(x;m;i;j)`	inserts matrix `m` into matrix `x`, putting its top-left element into row `i` and column `j` of `x`; elements of `m` that are defined to lie outside `x` are ignored.
`MINUTES(x)`	the number of minutes during the hour corresponding to `x` (i.e. the number of minutes recorded on a clock at date-time value `x`).
`MODULO(x;y)`	Form modulus of `x` to base `y`.
`MONTH(x)`	the month corresponding to date-time value `x`.
`MPOWER(m; n)`	raises matrix `m` to the `n`'th power.
`MSQRT(m)`	calculates the matrix square root of `m`.
`MVINSERT(x;y)`	replaces values in `x` by missing value wherever the second identifier stores a non-zero value (logical `.TRUE.`).
`MVREPLACE(x;y)`	replaces missing values in `x` with the values in the corresponding units of `y`.
`MZERO(r;c)`	zero matrix of size `r` by `c`.
`NCOLUMNS(x)`	gives the number of columns of `x`.
`NCOMBINATIONS(n;r)`	number of combinations ${}^{n}C_{r}$ of `r` objects taken from a set of size `n`.
`NDAYINYEAR(x;m)`	the number of the day in year corresponding to date-time value `x`, and starting the year at the beginning of month `m` (default 1).
`NED`	synonym of `EDNORMAL`.
`NEWLEVELS(f;x)`	forms a variate from the factor `f`; the variate `x` defines a value for each level and should be the same length as the number of levels of the factor; if the second argument `x` is omitted, the ordinals (1, 2...) are given.
`NEXPAND(n;v)`	expands structure `v` to repeat each value the number of times

	specified by the corresponding element of n.
NLEVELS(f)	gives the number of levels of factor f.
NMV(x)	counts the number of missing values in x.
NOBSERVATIONS(x)	counts the number of observations (that is non-missing values) in x.
NORMAL	synonym of CLNORMAL.
NOW(x)	returns a scalar containing the current date and time (argument x is ignored).
NPERMUTATIONS(n;r)	number of permutations ${}^{n}P_{r}$ of r objects taken from a set of size n.
NROWS(x)	gives the number of rows of x.
NVALUES(x)	gives the number of values of x including missing values and taking account of any restriction.
NVRESTRICTED(x)	synonym of NVALUES.
NVUNRESTRICTED(x)	number of values of x ignoring any restriction (i.e. gives the full "length" of x).
NWEEKINYEAR(x;s)	number of the week through the year for date-time value x. The default setting for s is 'iso'; this uses the definition of ISO Standard IS-8601 (1988) in which any week (starting on Monday) that lies in more than one year is assigned a week number for the year in which most of its days occur. The alternative setting, 'simple', takes the first week of the year as the one containing 1st January.
OWN(x; 'name'; p1; p2...pn)	calls an external function with data in a variate x and n scalar parameters; the function is in a DLL defined by the EXTERNAL directive.
PAREA(y;x)	area of a polygon with vertices specified by y and x.
PERCENTILES(x;p)	percentiles (defined in variate p) of the values of x.
POSITION(x;y)	finds the position, within the vector y, of each value of x.
PRBETA(x;a;b)	probability density function for a beta distribution with parameters a and b.
PRBINOMIAL(j;n;p)	probability of j successes out of n binomial trials with probability of success p.
PRCHISQUARE(x;df;c)	probability density function for a non-central chi-square distribution with noncentrality parameter c; if the third parameter c is omitted, it is assumed to be zero, giving the ordinary (central) chi-square distribution.
PRF(x;df1;df2;c)	probability density function for a non-central F distribution with degrees of freedom df1 and df2, and noncentrality parameter c; if the fourth parameter c is omitted, it is assumed to be zero, giving the ordinary (central) F distribution.
PRGAMMA(x;k;t)	probability density function for a gamma distribution with shape parameter k (kappa) and scale parameter t (theta).
PRHYPERGEOMETRIC(j;l;m;n)	probability of j successes out of a sample of m from a population of size n of which l are positive (hypergeometric distribution).
PRINVNORMAL(x;m;l)	probability density function for an inverse Normal (or inverse Gaussian) distribution with mean m and reciprocal dispersion parameter l (variance is m^3/l).
PRLOGNORMAL(x)	probability density function for a lognormal distribution corresponding to a normal distribution with mean 0 and variance 1.
PRNORMAL(x;m;v)	probability density function for a Normal distribution with mean m (default 0) and variance v (default 1).
PROBIT(p)	takes the probit transformation of the percentages p ($0 < p < 100\%$). This is equal to the Normal equivalent deviate of

	p/100.
PRODUCT(x;y)	forms the matrix product of x and y (that is x *+ y).
PRPOISSON(j;m)	probability of the value j for a Poisson distribution with mean m.
PRSMMODULUS(x;df;n)	probability density function for a Studentized maximum modulus distribution with degrees of freedom df and number of means n.
PRSRANGE(x;df;n)	probability density function for a Studentized range distribution with degrees of freedom df and number of means n.
PRT(x;df;c)	probability density function for a non-central Student's t distribution with degrees of freedom df and noncentrality parameter c; if the third parameter c is omitted, it is assumed to be zero, giving the ordinary (central) t distribution.
PRUNIFORM(p;a;b)	probability density function for a uniform distribution on [a,b].
QPRODUCT(x;y)	forms the quadratic product of x and y (that is x *+ y *+ TRANSPOSE(x)), where x is a rectangular matrix or variate and y is a symmetric or diagonal matrix or a scalar.
QTPRODUCT(x;y)	quadratic matrix product of x′ and y (that is TRANSPOSE(x) *+ y *+ x), where x is a rectangular matrix or variate and y is a symmetric or diagonal matrix or a scalar.
QUANTILES(x;q)	quantiles (defined in variate q) of the values of x.
RADIANS(x)	converts angles x from degrees to radians.
RANGE(x)	range of values in x, i.e. MAX(x) - MIN(x).
RANKS(x)	ranks of the values in x.
RED(x)	calculates the red components of the RGB colour values in x.
REPLACE(x;y;z)	searches x for all occurrences of each value in y, and replaces them with the corresponding value from z.
RESTRICTION(x)	forms a variate with the value 1 in the units to which x is currently restricted.
REVERSE(x)	reverses the values of x.
RGB(x;y;z)	calculates RGB colour values from the red, green and blue components in x, y and z, respectively; these components must all be between 0 and 255.
RGB(t)	gives the RGB colour values of the standard Genstat colours in text t. The text can contain the string 'match' in its second and subsequent units, to repeat the colour in the previous unit. It can also contain strings made up of three pairs of hexadecimal digits (00-FF) prefixed by #, 0x or 0X: i.e. '#rgb', '0xrgb' or '0Xrgb' where rgb are pairs of hexadecimal digits 00-FF that define the red, green and blue intensities of the colour respectively.
RMEANS(x;p;q)	running means of x using a window around each unit that includes p preceding and q succeeding observations; p must be set, default for q is 0.
RNOBSERVATIONS(x;p;q)	number of observations contributing to the computation of a running mean or total involving p preceding and q succeeding observations about each unit of x; p must be set, default for q is 0.
ROUND(x)	rounds the values of x to the nearest integer.
ROWBIND(x;y)	joins matrices x and y vertically (i.e. stacks y below x).
ROWCENTRE(x)	centres the rows of matrix x by subtracting their means.
ROWMEANS(x)	mean of the non-missing elements of each row of matrix x.
ROWNOBSERVATIONS(x)	number of non-missing elements in each row of matrix x.
ROWSUMS(x)	sum of the non-missing elements of each row of matrix x.
ROW1(n)	row matrix of 1's with n columns.

RQOBJECTIVE(y;d;p;t)	returns the objective function from fitting a quantile linear regression with a response variate y, a design matrix d, a probability value specified by the scalar p, and using a tolerance defined by the scalar t; if the fourth argument is omitted, a default tolerance of 10^{-12} is used.
RSVECTORS(x)	matrix of vectors from the right-hand side of a singular-value decomposition of x.
RTOTALS(x;p;q)	running totals of x using a window around each unit that includes p preceding and q succeeding observations; p must be set, default for q is 0.
RTPRODUCT(x;y)	forms the right transposed product of x and y (that is x *+ TRANSPOSE(y)).
RUNS(x)	length of run of identical values up to each unit in x.
SD(x)	standard deviation of the non-missing values in x.
SECONDS(x)	the number of seconds (including fraction of seconds) during the minute corresponding to date-time value x.
SEMEAN(x)	standard error of the mean of the non-missing values in x.
SET(x)	returns a scalar logical value containing the values 1 or 0 according to whether or not dummy x is set (i.e. the opposite of the function UNSET).
SHIFT(x;s)	shifts the values of x by s places (to the right or left according to the sign of s). This is not a circular shift, so some positions lose their values and are given missing values.
SIGN(x)	sign of x (−1, 0 or 1 for x<0, x==0 or x>0 respectively).
SIN(x)	sine of x, for x in radians.
SINH(x)	hyperbolic sine of x.
SKEWNESS(x)	skewness of the non-missing values in x.
SOLUTION(x;y)	finds the solution b of the set of simultaneous linear equations x *+ b = y.
SORT(x;y)	sorts the elements of x into the order that would put the values of y into ascending order; if y is omitted, the values of x are sorted.
SQRT(x)	gives the square root of x (x ≥ 0).
SSPLINE(y;x;df;p)	fits a smoothing-spline of y on x, with df degrees of freedom or (if df is missing) smoothing parameter p.
STANDARIZE(x)	standardizes x to (x-MEAN(x))/SD(x).
SUBMAT(x)	forms sub-triangles or sub-rectangles of a rectangular or symmetric matrix. The rows and columns to be included are determined by matching the pointers indexing the resultant matrix with the pointers indexing x. (SUBMAT does not allow for indexing by variates or texts.)
SUM(x)	forms the sum of the values in x (synonym TOTAL).
SVALUES(x)	singular values of x (as a diagonal matrix).
T	synonym of TRANSPOSE.
TAN(x)	tangent of x, for x in radians.
TANH(x)	hyperbolic tangent of x.
TCOLUMN(t)	converts one-way table t into a matrix with a single column.
TDIAGONAL(t)	converts one-way table t into a diagonal matrix.
TIME(h;m;s)	constructs the time value (days and fractions of days) corresponding to h hours, m minutes and s seconds.
TKURTOSIS(x)	forms margins containing the kurtosis of the cells in table t.
TMATRIX(t;f1;f2)	converts two-way table t into a matrix, with classifying factor f1 corresponding to the rows, and classifying factor f2 corresponding to the columns.
TMAXIMA(t)	forms margins of maxima for table t.
TMEANS(t)	forms margins of means for table t.

TMEDIANS(t)	forms margins of medians for table t.
TMINIMA(t)	forms margins of minima for table t.
TNMV(t)	forms margins counting the numbers of missing values in table t.
TNOBSERVATIONS(t)	forms margins counting the numbers of observations (non-missing values) in table t.
TNVALUES(t)	forms margins counting the numbers of values (missing or non-missing) in table t.
TOTAL(x)	forms the total of the values in x (synonym SUM).
TPROJECT(t)	converts table t into a variate, using the values of its classifying factors to determine which value of the table to put into each unit of the variate.
TRACE(x)	calculates the trace of the square, diagonal, or symmetric matrix x (that is the sum of all its diagonal elements).
TRANSPOSE(x)	forms the transpose of a rectangular matrix x.
TRIGAMMA(x)	trigamma function of x.
TROW(t)	converts one-way table t into a matrix with a single row.
TSD(t)	forms margins of between-cell standard deviations for table t.
TSEMEANS(t)	forms margins of standard errors for between-cell means of table t.
TSKEWNESS(x)	forms margins containing the skewness of the cells in table t.
TSUMS	synonym of TTOTALS.
TTOTALS(t)	forms margins of totals for table t.
TVARIANCES(t)	forms margins of between-cell variances for table t.
TVECTOR(t; s; p)	copies the values from table t into a variate. The scalar s is zero if the margins of the table are to be omitted, or a non-zero (and non-missing) value if they are included. The pointer p contains the classifying factors of the table, defining the order in which the values are to be copied; this can be omitted if t is a one-way table. If margins are not to be included from a one-way table, s can also be omitted.
TYPE(x)	gives the type number of the data structure x.
UNIQUE(x)	the unique values in x.
UNSET(d)	returns a scalar logical value according to whether or not the dummy d is set.
URAND(seed;s)	provides s uniform pseudo-random numbers in the range (0,1). If s is not supplied and URAND cannot determine the length of the result from the context of the expression, the length of the current units structure (if any) is taken. Scalar seed initializes the generator. If zero in the first use of URAND in a job, the system clock is used to provide a seed; subsequent calls may use zero to continue the sequence of random numbers.
UTRIANGLE(m;d)	returns the upper triangle of square matrix m; as a square matrix with the lower triangular set to zero; putting d=1 (default) indicates that the diagonal is to be included, while putting d=0 excludes the diagonal.
VAR	synonym of VARIANCE.
VARIANCE(x)	gives the variance of the values in x.
VCORRELATION(p1;p2)	gives the correlation, at every unit, between the values of the corresponding structures in the pointers p1 and p2.
VCOVARIANCE(p1;p2)	gives the covariance, at every unit, between the values of the corresponding structures in the pointers p1 and p2.
VEC(x)	stacks columns of a matrix x into a single variate (*VEC* operator).
VECH(x)	stacks columns of the lower triangle of a matrix x (*VECH* operator).

VKURTOSIS(x)	kurtosis of the non-missing values in each unit of the variates (or scalars) in pointer p.
VMAXIMA(p)	finds the maximum of the values in each unit of the variates (or scalars) in pointer p.
VMEANS(p)	gives the mean of the non-missing values in each unit of the variates (or scalars) in pointer p.
VMEDIANS(p)	finds the median of the values in each unit of the variates (or scalars) in pointer p.
VMINIMA(p)	finds the minimum of the values in each unit of the variates (or scalars) in pointer p.
VNMV(p)	counts the number of missing values in each unit of the variates (or scalars) in pointer p.
VNOBSERVATIONS(p)	counts the number of observations (non-missing values) in each unit of the variates (or scalars) in pointer p.
VNVALUES(p)	gives the number of values in each unit of the variates (or scalars) in pointer p: that is the number of values of p.
VPERCENTILES(p;s)	calculates percentiles for the value supplied in scalar s, across the set of variates in pointer p.
VPOSITIONS(x;p)	gives the suffix of the first vector in the pointer p containing the value in each unit of the variate or text x.
VQUANTILES(p;s)	calculates quantiles for the probability supplied in scalar s, across the set of variates in pointer p.
VRANGE(p)	range of values within the units of the variates in pointer p.
VSD(x)	standard deviation of the non-missing values in each unit of the variates (or scalars) in pointer p.
VSEMEANS(x)	standard error of the mean of non-missing values in each unit of the variates (or scalars) in pointer p.
VSKEWNESS(x)	skewness of the non-missing values in each unit of the variates (or scalars) in pointer p.
VSUMS(p)	gives the sum of the non-missing values in each unit of the variates (or scalars) in pointer p (synonym VTOTALS).
VTOTALS(p)	gives the total of the non-missing values in each unit of the variates (or scalars) in pointer p (synonym VSUMS).
VVARIANCES(p)	gives the variance of the non-missing values in each unit of the variates (or scalars) in pointer p.
WEEKDAY(x)	the day of the week (where Monday is weekday 1) corresponding to data-time value x.
WHERE(x)	produces a variate listing the units of x that are logically true (i.e. non-zero).
WHICH(x)	synonym of WHERE.
YEAR(x)	the year corresponding to date-time value x.

4.3 Functions for model formulae

Function name	Description
COMPARISON(f;s;m)	estimates the comparisons amongst the levels of factor f specified by the first s rows of the matrix m. In regression models, the first argument may be a variate instead of a factor; COMP(v;s;m) then fits a set of associated variates stored in the first s rows of the rows of the matrix m. In either case, the comparisons define explanatory variates to be included in the regression, and their parameter estimates are the resulting regression coefficients. In TREATMENTSTRUCTURE formulae (specifying a model for analysis-of-variance), the parameter estimates are the estimates of the comparisons themselves (i.e.

m*+e, where e is the vector of estimated effects of factor f). This differs from the use of COMPARISON in regression models (and the use of the REG function in either regression or analysis of variance) as there the parameter estimates are regression coefficients. Another difference is that in analysis of variance each comparison is fitted ignoring the other comparisons, but in regression they are adjusted for each other.

LO synonym of LOESS.

LOESS(x;d;s;l) fits a locally weighted regression of order l (= 1 for linear, 2 for quadratic) with approximately d degrees of freedom or using smoothing parameter s (regression models only): x is a variate for univariate smoothing, or a pointer to up to four variates for multivariate smoothing; when x is a variate l is a scalar, when x is a pointer it is either a scalar or a variate with an element for each variate in the pointer.

POL(f;s;v) indicates that the effects of factor f are to be partitioned into polynomial contrasts (linear, quadratic etc) up to order s, where s is a scalar containing an integer between 1 and 4. Variate v defines a numerical value for each level of the factor; if omitted, the factor levels themselves are used. In a TREATMENT formula, the contrasts are orthogonalized, but they are not in a regression or generalized linear model. In regression models, POL(v;s) can be used to fit simple (non-orthogonalized) polynomials of a variate v up to order s.

POLND(f;s;v) has the same effect as POL, except that no Dev components are fitted for factor f in interactions (TREATMENT formulae only).

REG(f;s;m) indicates that the effects of factor f are to be partitioned into the orthogonal regression contrasts specified by the first s rows of the matrix m. In regression models, the first argument may be a variate instead of a factor; REG(v;s;m) then orthogonalizes and fits a set of associated variates stored in the first s rows of the rows of the matrix m. The matrix m may be omitted in a regression model, in which case orthogonal polynomial contrasts are constructed for either f or v. Note, though, that the orthogonalization is with respect to the replication of the main effect of the factor or variate, so interactions of the contrasts with other vectors in a regression model may not be orthogonal.

REGND(f;s;m) has the same effect as REG, except that no Dev components are fitted for factor f in interactions (TREATMENT formulae only).

S synonym of SSPLINE.

SSPLINE(v;s;p) indicates that the effect of a variate v is to be fitted by a smoothing spline with approximately s degrees of freedom or using "smoothing parameter" p (only in regression models or expressions).

5 List of commands

This lists the directives in Release 22, together with the procedures in the Procedure Library PL30 that accompanies Release 22.

ABIVARIATE produces graphs and statistics for bivariate analysis of variance.
ABLUPS calculates BLUPs for block terms in an ANOVA analysis.
ABOXCOX estimates the power λ in a Box-Cox transformation, that maximizes the partial log-likelihood in ANOVA.
ACANONICAL determines the orthogonal decomposition of the sample space for a design, using an analysis of the canonical relationships between the projectors derived from two or more model formulae.
ACDISPLAY provides further output from an analysis by ACANONICAL.
ACHECK checks assumptions for an ANOVA analysis.
ACKEEP saves information from an analysis by ACANONICAL.
ACONFIDENCE calculates simultaneous confidence intervals for ANOVA means.
ADD adds extra terms to a linear, generalized linear, generalized additive, or nonlinear model.
ADDPOINTS adds points for new objects to a principal coordinates analysis.
ADETECTION calculates the minimum size of effect or contrast detectable in an analysis of variance.
ADISPLAY displays further output from analyses produced by ANOVA.
ADPOLYNOMIAL plots single-factor polynomial contrasts fitted by ANOVA.
ADSPREADSHEET puts the data and plan of an experimental design into Genstat spreadsheets.
AEFFICIENCY calculates efficiency factors for experimental designs.
AFALPHA generates alpha designs.
AFAUGMENTED forms an augmented design.
AFCARRYOVER forms factors to represent carry-over effects in cross-over trials.
AFCOVARIATES defines covariates from a model formula for ANOVA.
AFCYCLIC generates block and treatment factors for cyclic designs.
AFDISCREPANCY calculates the discrepancy of a design.
AFFYMETRIX estimates expression values for Affymetrix slides.
AFIELDRESIDUALS display residuals in field layout.
AFLABELS forms a variate of unit labels for a design.
AFMEANS forms tables of means classified by ANOVA treatment factors.
AFMINABERRATION forms minimum aberration factorial or fractional-factorial designs.
AFNONLINEAR forms D-optimal designs to estimate the parameters of a nonlinear or generalized linear model.
AFORMS prints data forms for an experimental design.
AFPREP searches for an efficient partially-replicated design.
AFRCRESOLVABLE forms doubly resolvable row-column designs, with output.
AFRESPONSESURFACE uses the BLKL algorithm to construct designs for estimating response surfaces.
AFUNITS forms a factor to index the units of the final stratum of a design.
AGALPHA forms alpha designs by standard generators for up to 100 treatments.
AGBIB generates balanced incomplete block designs.
AGBOXBEHNKEN generates Box-Behnken designs.
AGCENTRALCOMPOSITE generates central composite designs.
AGCROSSOVERLATIN generates Latin squares balanced for carry-over effects.
AGCYCLIC generates cyclic designs from standard generators.
AGDESIGN generates generally balanced designs.
AGFACTORIAL generates minimum aberration block or fractional factorial designs.
AGFRACTION generates fractional factorial designs.
AGHIERARCHICAL generates orthogonal hierarchical designs.
AGINDUSTRIAL helps to select and generate effective designs for use in industrial experiments.
AGLATIN generates mutually orthogonal Latin squares.
AGLOOP generates loop designs e.g. for time-course microarray experiments
AGMAINEFFECT generates designs to estimate main effects of two-level factors.
AGNATURALBLOCK forms 1- and 2-dimensional designs with blocks of natural size
AGNEIGHBOUR generates neighbour-balanced designs.
AGNONORTHOGONALDESIGN generates non-orthogonal multi-stratum designs.
AGQLATIN generates complete and quasi-complete Latin squares.

AGRAPH plots tables of means from ANOVA.
AGRCRESOLVABLE forms doubly resolvable row-column designs.
AGREFERENCE generates reference-level designs e.g. for microarray experiments
AGSEMILATIN generates semi-Latin squares.
AGSPACEFILLINGDESIGN generates space filling designs.
AGSQLATTICE generates square lattice designs.
AKAIKEHISTOGRAM prints histograms with improved definition of groups.
AKEEP copies information from an ANOVA analysis into Genstat data structures.
AKEY generates values for treatment factors using the design key method.
ALIAS finds out information about aliased model terms in analysis of variance.
ALIGNCURVE forms an optimal warping to align an observed series of observations with a standard series.
ALLDIFFERENCES shows all pairwise differences of values in a variate or table.
ALLPAIRWISE performs a range of all pairwise multiple comparison tests.
AMCOMPARISON performs pairwise multiple comparison tests for ANOVA means.
AMDUNNETT forms Dunnett's simultaneous confidence interval around a control.
AMERGE merges extra units into an experimental design.
AMMI allows exploratory analysis of genotype × environment interactions.
AMTDISPLAY displays further output for a multi-tiered design analysed by AMTIER.
AMTIER analyses a multi-tiered design with up to 3 structures.
AMTKEEP saves information from the analysis of a multitiered design by AMTIER.
ANOVA analyses y-variates by analysis of variance according to the model defined by earlier BLOCKSTRUCTURE, COVARIATE, and TREATMENTSTRUCTURE statements.
ANTMVESTIMATE estimates missing values in repeated measurements.
ANTORDER assesses order of ante-dependence for repeated measures data.
ANTTEST calculates overall tests based on a specified order of ante-dependence.
AN1ADVICE aims to give useful advice if a design that is thought to be balanced fails to be analysed by ANOVA.
AONEWAY performs one-way analysis of variance.
AOVANYHOW performs analysis of variance using ANOVA, regression or REML as appropriate.
AOVDISPLAY provides further output from an analysis by AOVANYHOW.
APAPADAKIS analysis of variance with an added Papadakis covariate, formed from neighbouring residuals.
APERMTEST does random permutation tests for analysis-of-variance tables
APLOT plots residuals from an ANOVA analysis.
APOLYNOMIAL forms equations for single-factor polynomial contrasts fitted by ANOVA.
APOWER calculates the power (probability of detection) for terms in an aov.
APPEND appends a list of vectors of compatible types.
APRODUCT forms a new experimental design from the product of two designs.
ARANDOMIZE randomizes and prints an experimental design.
ARCSPLITPLOT adds extra treatments onto the replicates of a resolvable row-column design, and generates factors giving the row and column locations of the plots within the design.
AREPMEASURES produces an analysis of variance for repeated measurements.
ARESULTSUMMARY provides a summary of results from an ANOVA analysis.
ARETRIEVE retrieves an ANOVA save structure from an external file.
ASAMPLESIZE finds the replication to detect a treatment effect or contrast.
ASPREADSHEET saves results from an analysis of variance in a spreadsheet.
ASRULES derives association rules from transaction data.
ASCREEN performs screening tests for designs with orthogonal block structure
ASSIGN sets elements of pointers and dummies.
ASTATUS provides information about the settings of ANOVA models and variates.
ASTORE stores an ANOVA save structure in an external file.
ASWEEP performs sweeps for model terms in an analysis of variance.
AUDISPLAY produces further output for an unbalanced design (after AUNBALANCED).
AUGRAPH plots tables of means from AUNBALANCED.
AUKEEP saves output from analysis of an unbalanced design (by AUNBALANCED).

AUNBALANCED performs analysis of variance for unbalanced designs.
AUMCOMPARISON performs pairwise multiple comparison tests for means from an unbalanced analysis of variance, performed previously by AUNBALANCED.
AUPREDICT forms predictions from an unbalanced design (after AUNBALANCED).
AUSPREADSHEET saves results from an analysis of an unbalanced design (by AUNBALANCED) in a spreadsheet.
AU2RDA saves results from an unbalanced analysis of variance, by AUNBALANCED, in R data frames.
AXES defines the axes in each window for high-resolution graphics.
AXIS defines an oblique axis for high-resolution graphics.
AYPARALLEL does the same analysis of variance for several y-variates, and collates the output.
A2DISPLAY provides further output following an analysis of variance by A2WAY
A2KEEP copies information from an A2WAY analysis into Genstat data structures
A2PLOT plots effects from two-level designs with robust s.e. estimates.
A2RDA saves results from an analysis of variance in R data frames.
A2RESULTSUMMARY provides a summary of results from an analysis by A2WAY.
A2WAY performs analysis of variance of a balanced or unbalanced design with up to two treatment factors.
BACKTRANSFORM calculates back-transformed means with approximate standard errors and confidence intervals.
BAFFYMETRIX estimates expression values from an Affymetrix CED and CDF file.
BANK calculates the optimum aspect ratio for a graph.
BARCHART plots bar charts in high-resolution graphics.
BASELINE estimates a baseline for a series of numbers whose minimum value is drifting.
BASSESS assesses potential splits for regression and classification trees.
BBINOMIAL estimates the parameters of the beta binomial distribution.
BCDISPLAY displays a classification tree.
BCFDISPLAY displays information about a random classification forest.
BCFIDENTIFY identifies specimens using a random classification forest.
BCFOREST constructs a random classification forest.
BCIDENTIFY identifies specimens using a classification tree.
BCKEEP saves information from a classification tree.
BCLASSIFICATION constructs a classification tree.
BCONSTRUCT constructs a tree.
BCUT cuts a tree at a defined node, discarding nodes and information below it.
BCVALUES forms values for nodes of a classification tree.
BGIMPORT imports MCMC output in CODA format produced by WinBUGS or OpenBUGS.
BGPLOT produces plots for output and diagnostics from MCMC simulations.
BGRAPH plots a tree.
BGROW adds new branches to a node of a tree.
BGXGENSTAT runs WinBUGS or OpenBUGS from Genstat in batch mode using scripts.
BIDENTIFY identifies specimens using a tree.
BIPLOT produces a biplot from a set of variates.
BJESTIMATE fits an ARIMA model, with forecast and residual checks.
BJFORECAST plots forecasts of a time series using a previously fitted ARIMA.
BJIDENTIFY displays time series statistics useful for ARIMA model selection.
BJOIN extends a tree by joining another tree to a terminal node.
BKDISPLAY displays an identification key.
BKEY constructs an identification key.
BKIDENTIFY identifies specimens using a key.
BKKEEP saves information from an identification key.
BLANDALTMAN produces Bland-Altman plots to assess the agreement between two variates.
BLOCKSTRUCTURE defines the blocking structure of the design and hence the strata and the error terms.
BNTEST calculates one- and two-sample binomial tests.
BOOTSTRAP produces bootstrapped estimates, standard errors and distributions.
BOXPLOT draws box-and-whisker diagrams or schematic plots.
BPCONVERT converts bit patterns between integers, pointers of set bits and textual descriptions.

BPRINT displays a tree.
BPRUNE prunes a tree using minimal cost complexity.
BRDISPLAY displays a regression key.
BREAK suspends execution of the statements in the current channel or control structure and takes subsequent statements from the channel specified.
BREGRESSION constructs a regression tree.
BRFDISPLAY displays information about a random regression forest.
BRFOREST constructs a random regression forest.
BRFPREDICT makes predictions using a random regression forest.
BRKEEP saves information from a regression tree.
BRPREDICT makes predictions using a regression tree.
BRVALUES forms values for nodes of a regression tree.
CABIPLOT plots results from correspondence analysis or multiple correspondence analysis.
CALCULATE calculates numerical values for data structures.
CALLS lists library procedures called by a procedure.
CANCORRELATION does canonical correlation analysis.
CAPTION prints captions in standardized formats.
CASE introduces a "multiple-selection" control structure.
CASSOCIATION calculates measures of association for circular data.
CATALOGUE displays the contents of a backing-store file.
CATRENDTEST calculates the Cochran-Armitage chi-square test for trend.
CCA performs canonical correspondence analysis.
CCOMPARE tests whether samples from circular distributions have a common mean direction or have identical distributions.
CDESCRIBE calculates summary statistics and tests of circular data.
CDNAUGMENTEDDESIGN constructs an augmented block design, using CycDesigN if the controls are in an incomplete-block design.
CDNBLOCKDESIGN constructs a block design using CycDesigN.
CDNPREP constructs a multi-location partially-replicated design using CycDesigN.
CDNROWCOLUMNDESIGN constructs a row-column design using CycDesigN.
CENSOR pre-processes censored data before analysis by ANOVA.
CHECKARGUMENT checks the arguments of a procedure.
CHIPERMTEST performs a random permutation test for a two-dimensional contingency table.
CHISQUARE calculates chi-square statistics for one- and two-way tables.
CINTERACTION clusters rows and columns of a two-way interaction table.
CLASSIFY obtains a starting classification for non-hierarchical clustering.
CLOSE closes files.
CLUSTER forms a non-hierarchical classification.
CMHTEST performs the Cochran-Mantel-Haenszel test.
COKRIGE calculates kriged estimates using a model fitted to the sample variograms and cross-variograms of a set of variates.
COLOUR defines the red, green and blue intensities to be used for the Genstat colours with certain graphics devices.
COMBINE combines or omits "slices" of a multi-way data structure (table, matrix, or variate).
COMMANDINFORMATION provides information about whether (and how) a command has been implemented.
CONCATENATE concatenates and truncates lines (units) of text structures; allows the case of letters to be changed.
CONCORD is a synonym for KCONCORDANCE.
CONFIDENCE calculates simultaneous confidence intervals.
CONTOUR is a synonym for LPCONTOUR.
CONVEXHULL finds the points of a single or a full peel of convex hulls.
COPY forms a transcript of a job.
CORANALYSIS does correspondence analysis, or reciprocal averaging.
CORRELATE forms correlations between variates, autocorrelations of variates, and lagged cross-correlations between variates.

CORRESP is a synonym for CORANALYSIS.
COVARIATE specifies covariates for use in subsequent ANOVA statements.
COVDESIGN produces experimental designs efficient under analysis of covariance.
CSPRO reads a data set from a CSPro survey data file and dictionary, and loads it into Genstat or puts it into a spreadsheet file.
CUMDISTRIBUTION fits frequency distributions to accumulated counts.
CRBIPLOT plots correlation or distance biplots after RDA, or ranking biplots after CCA.
CRTRIPLOT plots ordination biplots or triplots after CCA or RDA.
CVA performs canonical variates analysis.
CVAPLOT plots the mean and unit scores from a canonical variates analysis.
CVASCORES calculates scores for individual units in canonical variates analysis.
DARROW adds arrows to an existing plot.
DAYLENGTH calculates daylengths at a given period of the year.
DBARCHART produces bar charts for one or two-way tables.
DBCOMMAND runs an SQL command on an ODBC database.
DBEXPORT updates an ODBC database table using data from Genstat.
DBIMPORT loads data into Genstat from an ODBC database.
DBINFORMATION loads information on the tables and columns in an ODBC database.
DBIPLOT plots a biplot from an analysis by PCP, CVA or PCO.
DBITMAP plots a bit map of RGB colours.
DCIRCULAR plots circular data.
DCLEAR clears a graphics screen.
DCLUSTERLABELS labels clusters in a single-page dendrogram plotted by DDENDROGRAM.
DCOLOURS forms a band of graduated colours for graphics.
DCOMPOSITIONAL plots 3-part compositional data within a barycentric triangle.
DCONTOUR draws contour plots on a plotter or graphics monitor.
DCORRELATION plots a correlation matrix.
DCOVARIOGRAM plots 2-dimensional auto- and cross-variograms.
DDEEXPORT sends data or commands to a Dynamic Data Exchange server.
DDEIMPORT gets data from a Dynamic Data Exchange (DDE) server.
DDENDROGRAM draws dendrograms with control over structure and style.
DDESIGN plots the plan of an experimental design.
DDISPLAY redraws the current graphical display.
DEBUG puts an implicit BREAK statement after the current statement and after every NSTATEMENTS subsequent statements, until an ENDDEBUG is reached.
DECIMALS sets the number of decimals for a structure, using its round-off.
DECLARE declares one or more customized data structures.
DELETE deletes the attributes and values of structures.
DELLIPSE draws a 2-dimensional scatter plot with confidence, prediction and/or equal-frequency ellipses superimposed.
DEMC performs Bayesian computing using the Differential Evolution Markov Chain algorithm.
DERRORBAR adds error bars to a graph.
DESCRIBE saves and/or prints summary statistics for variates.
DESIGN helps to select and generate effective experimental designs.
DEVICE switches between (high-resolution) graphics devices.
DFINISH ends a sequence of related high-resolution plots.
DFONT defines the default font for high-resolution graphics.
DFOURIER performs a harmonic analysis of a univariate time series.
DFRTEXT adds text to a graphics frame.
DFUNCTION plots a function.
DGRAPH draws graphs on a plotter or graphics monitor.
DHELP provides information about Genstat graphics.
DHISTOGRAM draws histograms on a plotter or graphics monitor.
DHSCATTERGRAM plots an h-scattergram.
DIAGONALMATRIX declares one or more diagonal matrix data structures.
DIALLEL analyses full and half diallel tables with parents.

DILUTION calculates Most Probable Numbers from dilution series data.
DIRECTORY prints or saves a list of files with names matching a specified mask.
DISCRIMINATE performs discriminant analysis.
DISPLAY prints, or reprints, diagnostic messages.
DISTRIBUTION estimates the parameters of continuous and discrete distributions.
DKALMAN plots results from an analysis by KALMAN.
DKEEP saves information from the last plot on a particular device.
DKEY adds a key to a graph.
DKSTPLOT produces diagnostic plots for space-time clustering.
DLOAD loads the graphics environment settings from an external file.
DMADENSITY plots the empirical CDF or PDF (kernel smoothed) by groups.
DMASS plots discrete data like mass spectra, discrete probability functions.
DMSCATTER produces a scatter-plot matrix for one or two sets of variables.
DMST gives a high resolution plot of an ordination with minimum spanning tree.
DOTHISTOGRAM plots dot histograms.
DOTPLOT produces a dot-plot using line-printer or high-resolution graphics.
DPARALLEL displays multivariate data using parallel coordinates.
DPIE draws a pie chart on a plotter or graphics monitor.
DPOLYGON draws polygons using high-resolution graphics.
DPROBABILITY creates a probability distribution plot of the values in a variate.
DPSPECTRALPLOT calculates an estimate of the spectrum of a spatial point pattern.
DPTMAP draws maps for spatial point patterns using high-resolution graphics.
DPTREAD adds points interactively to a spatial point pattern.
DQMAP displays a genetic map.
DQMKSCORES plots a grid of marker scores for genotypes and indicates missing data.
DQMQTLSCAN plots the results of a genome-wide scan for QTL effects in multi-environment trials.
DQRECOMBINATIONS plots a matrix of recombination frequencies between markers.
DQSQTLSCAN plots the results of a genome-wide scan for QTL effects in single-environment trials.
DREAD reads the locations of points from an interactive graphical device.
DREFERENCELINE adds reference lines to a graph.
DREPMEASURES plots profiles and differences of profiles for repeated measures data.
DRESIDUALS plots residuals.
DROP drops terms from a linear, generalized linear, generalized additive, or nonlinear model.
DRPOLYGON reads a polygon interactively from the current graphics device.
DSAVE saves the current graphics environment settings to an external file.
DSCATTER produces a scatter-plot matrix using high-resolution graphics.
DSEPARATIONPLOT creates a separation plot for visualising the fit of a model with a dichotomous (i.e. binary) or polytomous (i.e. multi-categorical) outcome.
DSHADE plots a shade diagram of 3-dimensional data.
DSTART starts a sequence of related high-resolution plots.
DSTTEST plots power and significance for t-tests, including equivalence tests.
DSURFACE produces perspective views of a two-way arrays of numbers.
DTABLE plots tables.
DTEXT adds text to a graph.
DTIMEPLOT produces horizontal bars displaying a continuous time record.
DUMMY declares one or more dummy data structures.
DUMP prints information about data structures, and internal system information.
DUPLICATE forms new data structures with attributes taken from an existing structure.
DVARIOGRAM plots fitted models to an experimental variogram.
DXDENSITY produces one-dimensional density (or violin) plots.
DXYDENSITY produces density plots for large data sets.
DXYGRAPH draws two-dimensional graphs with marginal distribution plots alongside the y- and x-axes.
DYPOLAR produces polar plots.
D3GRAPH plots a 3-dimensional graph.
D3HISTOGRAM plots three-dimensional histograms.
ECABUNDANCEPLOT produces rank/abundance, ABC and *k*-dominance plots

ECACCUMULATION plots species accumulation curves for samples or individuals.
ECANOSIM perform's an analysis of similarities (ANOSIM)
ECDIVERSITY calculates measures of diversity with jackknife or bootstrap estimates
ECFIT fits models to species abundance data
ECNICHE generates relative abundance of species for niche-based models
ECNPESTIMATE calculates nonparametric estimates of species richness.
ECRAREFACTION calculates individual or sample-based rarefaction
EDDUNNETT calculates equivalent deviates for Dunnett's simultaneous confidence interval around a control.
EDFTEST performs empirical-distribution-function goodness-of-fit tests.
EDIT edits text vectors.
ELSE introduces the default set of statements in block-if or in multiple-selection control structures.
ELSIF introduces a set of alternative statements in a block-if control structure.
ENDBREAK returns to the original channel or control structure and continues execution.
ENDCASE indicates the end of a "multiple-selection" control structure.
ENDDEBUG cancels a DEBUG statement.
ENDFOR indicates the end of the contents of a loop.
ENDIF indicates the end of a block-if control structure.
ENDJOB ends a Genstat job.
ENDPROCEDURE indicates the end of the contents of a Genstat procedure.
ENQUIRE provides details about files opened by Genstat.
EQUATE transfers data between structures of different sizes or types (but the same modes i.e. numerical or text) or where transfer is not from single structure to single structure.
ESTIMATE is a synonym for TFIT.
EXAMPLE obtains and runs a Genstat example program.
EXECUTE executes the statements contained within a text.
EXIT exits from a control structure.
EXPORT outputs data structures in foreign file formats, including Excel, Quattro, dBase, SPlus, Gauss, MatLab and Instat, or as plain or comma-delimited text.
EXPRESSION declares one or more expression data structures.
EXTERNAL declares an external function in a DLL for use by the OWN function.
EXTRABINOMIAL fits the models of Williams (1982) to overdispersed proportions.
FACAMEND permutes the levels and labels of a factor.
FACCOMBINATIONS forms a factor to indicate observations with identical combinations of values of a set of variates, texts or factors.
FACDIVIDE represents a factor by factorial combinations of a set of factors.
FACEXCLUDEUNUSED redefines the levels and labels of a factor to exclude those that are unused.
FACGETLABELS obtains the labels for a factor if it has been defined with labels, or constructs labels from its levels otherwise.
FACLEVSTANDARDIZE standardizes the levels or labels of a list of factors.
FACMERGE merges levels of factors.
FACPRODUCT forms a factor with a level for every combination of other factors.
FACROTATE rotates factor loadings from a principal components, canonical variates or factor analysis.
FACSORT sorts the levels of a factor according to an index vector.
FACTOR declares one or more factor data structures.
FACUNIQUE redefines a factor so that its levels and labels are unique.
FALIASTERMS forms information about aliased model terms in analysis of variance.
FARGUMENTS forms lists of arguments involved in an expression.
FAULT checks whether to issue a diagnostic, i.e. a fault, warning or message.
FBASICCONTRASTS breaks a model term down into its basic contrasts.
FBETWEENGROUPVECTORS forms variates and classifying factors containing within-group summaries to use e.g. in a between-group analysis.
FCA performs factor analysis.
FCLASSIFICATION forms a classification set for each term in a formula, breaks a formula up into separate formulae (one for each term), and applies a limit to the number of factors and variates in the terms of a formula.

FCOMPLEMENT forms the complement of an incomplete block design.
FCONTRASTS modifies a model formula to contain contrasts of factors.
FCOPY makes copies of files.
FCORRELATION forms the correlation matrix for a list of variates.
FCOVARIOGRAM forms a covariogram structure containing auto-variograms of individual variates and cross-variograms for pairs from a list of variates.
FDELETE deletes files.
FDESIGNFILE forms a backing-store file of information for AGDESIGN.
FDIALLEL forms the components of a diallel model for REML or regression.
FDISTINCTFACTORS checks sets of factors to remove any that define duplicate classifications.
FDRBONFERRONI estimates false discovery rates by a Bonferroni-type procedure.
FDRMIXTURE estimates false discovery rates using mixture distributions.
FEXACT2X2 does Fisher's exact test for 2×2 tables.
FFRAME forms multiple windows in a plot-matrix for high-resolution graphics.
FFREERESPONSEFACTOR forms multiple-response factors from free-response data.
FHADAMARDMATRIX forms Hadamard matrices.
FHAT calculates an estimate of the F nearest-neighbour distribution function.
FIELLER calculates effective doses or relative potencies.
FILEREAD reads data from a file.
FILTER is a synonym for TFILTER.
FIT fits a linear, generalized linear, generalized additive, or generalized nonlinear model.
FITCURVE fits a standard nonlinear regression model.
FITINDIVIDUALLY fits regression models one term at a time.
FITMULTINOMIAL fits generalized linear models with multinomial distribution.
FITNONLINEAR fits a nonlinear regression model or optimizes a scalar function.
FITNONNEGATIVE is a synonym for RNONNEGATIVE.
FITPARALLEL is a synonym for RPARALLEL.
FITSCHNUTE is a synonym for RSCHNUTE.
FKEY forms design keys for multi-stratum experimental designs, allowing for confounded and aliased treatments.
FLRV forms the values of LRV structures.
FMEGAENVIRONMENTS forms mega-environments based on winning genotypes from an AMMI-2 model.
FMFACTORS forms a pointer of factors representing a multiple-response.
FNCORRELATION calculates correlations from variances and covariances, together with their variances and covariances.
FNLINEAR estimates linear functions of random variables, and calculates their variances and covariances.
FNPOWER estimates products of powers of two random variables, and calculates their variances and covariances.
FOCCURRENCES counts how often each pair of treatments occurs in the same block.
FOR introduces a loop.
FORECAST is a synonym for TFORECAST.
FORMULA declares one or more formula data structures.
FOURIER calculates cosine or Fourier transforms of real or complex series.
FPARETOSET forms the Pareto optimal set of non-dominated groups.
FPLOTNUMBER forms plot numbers for a row-by-column design.
FPROJECTIONMATRIX forms a projection matrix for a set of model terms.
FPSEUDOFACTORS determines patterns of confounding and aliasing from design keys, and extends the treatment model to incorporate the necessary pseudo-factors.
FRAME defines the positions and appearance of the plotting windows within the frame of a high-resolution graph.
FREGULAR expands vectors onto a regular two-dimensional grid.
FRENAME renames files.
FRESTRICTEDSET forms vectors with the restricted subset of a list of vectors.
FRIEDMAN performs Friedman's non-parametric analysis of variance.

FROWCANONICALMATRIX puts a matrix into row canonical, or reduced row echelon, form.
FRQUANTILES forms regression quantiles.
FRTPRODUCTDESIGNMATRIX forms summation, or relationship, matrices for model terms.
FSIMILARITY forms a similarity matrix or a between-group-elements similarity matrix or prints a similarity matrix.
FSPREADSHEET creates a Genstat Spreadsheet file (GSH) from specified data structures.
FSSPM forms the values of SSPM structures.
FSTRING forms a single string from a list of strings in a text.
FTEXT forms a text structure from a variate.
FTSM forms preliminary estimates of parameters in time-series models.
FUNIQUEVALUES redefines a variate or text so that its values are unique.
FVARIOGRAM forms experimental variograms.
FVCOVARIANCE forms the variance-covariance matrix for a list of variates.
FVSTRING forms a string listing the identifiers of a set of data structures.
FWITHINTERMS forms factors to define terms representing the effects of one factor within another factor.
FZERO gives the F function expectation under complete spatial randomness.
F2DRESIDUALVARIOGRAM calculates and plots a 2-dimensional variogram from a 2-dimensional array of residuals.
GALOIS forms addition and multiplication tables for a Galois finite field.
GBGRIDCONVERSION converts GB grid references to or from latitudes and longitudes or to or from UTM coordinates.
GEE fits models to longitudinal data by generalized estimating equations.
GENERATE generates factor values for designed experiments.
GENPROCRUSTES performs a generalized Procrustes analysis.
GESTABILITY calculates stability coefficients for genotype-by-environment data.
GET accesses details of the "environment" of a Genstat job.
GETATTRIBUTE accesses attributes of structures.
GETLOCATIONS finds locations of an identifier within a pointer, or a string within a factor or text, or a number within any numerical data structure.
GETNAME forms the name of a structure according to its IPRINT attribute.
GETRGB gets the RGB values of the standard graphics colours.
GGEBIPLOT plots displays to assess genotype+genotype-by-environment variation.
GHAT calculates an estimate of the G nearest-neighbour distribution function.
GINVERSE calculates the generalized inverse of a matrix.
GLDISPLAY displays further output from a GLMM analysis.
GLKEEP saves results from a GLMM analysis.
GLM analyses non-standard generalized linear models.
GLMM fits a generalized linear mixed model.
GLPERMTEST does random permutation tests for generalized linear mixed models.
GLPLOT plots residuals from a GLMM analysis.
GLPREDICT forms predictions from a GLMM analysis.
GLRTEST calculates likelihood tests to assess random terms in a generalized linear mixed model.
GPREDICTION produces genomic predictions (breeding values) using phenotypic and molecular marker information.
GRANDOM generates pseudo-random numbers from probability distributions.
GRAPH is a synonym for LPGRAPH.
GRCSR generates completely spatially random points in a polygon.
GREJECTIONSAMPLE generates random samples using rejection sampling.
GRIBIMPORT reads data from a GRIB2 meteorological data file, and loads it or converts it to a spreadsheet file.
GRLABEL randomly labels two or more spatial point patterns.
GRMNOMIAL generates multinomial pseudo-random numbers.
GRMULTINORMAL generates multivariate normal pseudo-random numbers.
GROUPS forms a factor (or grouping variable) from a variate or text, together with the set of distinct values that occur.

GRTHIN randomly thins a spatial point pattern.
GRTORSHIFT performs a random toroidal shift on a spatial point pattern.
GSTATISTIC calculates the gamma statistic of agreement for ordinal data.
G2AEXPORT forms a dbase file to transfer ANOVA output to Agronomix Generation II.
G2AFACTORS redefines block and treatment variables as factors.
G2VEXPORT forms a dbase file to transfer REML output to Agronomix Generation II.
HANOVA does hierarchical analysis of variance or covariance for unbalanced data.
HBOOTSTRAP performs bootstrap analyses to assess the reliability of clusters from hierarchical cluster analysis.
HCLUSTER performs hierarchical cluster analysis.
HCOMPAREGROUPINGS compares groupings generated, for example, from cluster analyses.
HDISPLAY displays results ancillary to hierarchical cluster analyses: matrix of mean similarities between and within groups, a set of nearest neighbours for each unit, a minimum spanning tree, and the most typical elements from each group.
HEATUNITS calculates accumulated heat units of a temperature dependent process.
HELP provides help information about Genstat.
HFAMALGAMATIONS forms an amalgamations matrix from a minimum spanning tree.
HFCLUSTERS forms a set of clusters from an amalgamations matrix.
HGANALYSE analyses data using a hierarchical or double hierarchical generalized linear model.
HGDISPLAY displays results from a hierarchical or double hierarchical generalized linear model.
HGDRANDOMMODEL defines the random model in a hierarchical generalized linear model for the dispersion model of a double hierarchical generalized linear model.
HGFIXEDMODEL defines the fixed model for a hierarchical or double hierarchical generalized linear model.
HGFTEST calculates likelihood tests for fixed terms in a hierarchical generalized linear model
HGGRAPH draws a graph to display the fit of an HGLM or DHGLM analysis.
HGKEEP saves information from a hierarchical or double hierarchical generalized linear model analysis.
HGNONLINEAR defines nonlinear parameters for the fixed model of a hierarchical generalized linear model.
HGPLOT produces model-checking plots for a hierarchical or double hierarchical generalized linear model.
HGPREDICT forms predictions from a hierarchical or double hierarchical generalized linear model.
HGRANDOMMODEL defines the random model for a hierarchical or double hierarchical generalized linear model.
HGRTEST calculates likelihood tests for random terms in a hierarchical generalized linear model.
HGSTATUS displays the current HGLM model definitions.
HGWALD prints or saves Wald tests for fixed terms in an HGLM.
HISTOGRAM is a synonym for LPHISTOGRAM.
HLIST lists the data matrix in abbreviated form.
HPCLUSTERS prints a set of clusters.
HREDUCE forms a reduced similarity matrix (referring to the GROUPS instead of the original units).
HSUMMARIZE forms and prints a group by levels table for each test together with appropriate summary statistics for each group.
IDENTIFY identifies an unknown specimen from a defined set of objects.
IF introduces a block-if control structure.
IFUNCTION estimates implicit and/or explicit functions of parameters.
IMPORT reads data from a foreign file format, and loads it or converts it to a spreadsheet file.
INPUT specifies the input file from which to take further statements.
INSIDE determines whether points lie within a specified polygon.
INTERPOLATE interpolates values at intermediate points.
IRREDUNDANT forms irredundant test sets for the efficient identification of a set of objects.
JACKKNIFE produces Jackknife estimates and standard errors.
JOB starts a Genstat job.
JOIN joins or merges two sets of vectors together, based on classifying keys.
KALMAN calculates estimates from the Kalman filter.
KAPLANMEIER calculates the Kaplan-Meier estimate of the survivor function.

KAPPA calculates a kappa coefficient of agreement for nominally scaled data.
KCONCORDANCE calculates Kendall's Coefficient of Concordance.
KCROSSVALIDATION computes cross validation statistics for punctual kriging.
KCSRENVELOPES simulates K function bounds under complete spatial randomness.
KERNELDENSITY uses kernel density estimation to estimate a sample density.
KHAT calculates an estimate of the K function.
KLABENVELOPES gives bounds for K function differences under random labelling.
KNEARESTNEIGHBOURS classifies items or predicts their responses by examining their *k* nearest neighbours.
KOLMOG2 performs a Kolmogorov-Smirnoff two-sample test.
KRIGE calculates kriged estimates using a model fitted to the sample variogram.
KRUSKAL carries out a Kruskal-Wallis one-way analysis of variance.
KSED calculates the standard error for K function differences under random labelling.
KSTHAT calculates an estimate of the K function in space, time and space-time.
KSTMCTEST performs a Monte-Carlo test for space-time interaction.
KSTSE calculates the standard error for the space-time K function.
KTAU calculates Kendall's rank correlation coefficient τ
KTORENVELOPES gives bounds for the bivariate K function under independence.
K12HAT calculates an estimate of the bivariate K function.
LCONCORDANCE calculates Lin's concordance correlation coefficient.
LIBEXAMPLE accesses examples and source code of library procedures.
LIBFILENAME supplies the names of information files for library procedures.
LIBHELP provides help information about library procedures.
LIBSOURCE obtains the source code of a Genstat procedure.
LIBVERSION provides the name of the current Genstat Procedure Library.
LINDEPENDENCE finds the linear relations associated with matrix singularities.
LIST lists details of the data structures currently available within Genstat.
LORENZ plots the Lorenz curve and calculates the Gini and asymmetry coefficients.
LPCONTOUR produces contour maps of two-way arrays of numbers using character (i.e. line-printer) graphics.
LPGRAPH produces point and line plots using character (i.e. line-printer) graphics.
LPHISTOGRAM produces histograms using character (i.e. line-printer) graphics.
LRIDGE does logistic ridge regression.
LRV declares one or more LRV data structures.
LRVSCREE prints a scree diagram and/or a difference table of latent roots.
LSIPLOT plots least significant intervals, saved from SEDLSI.
LSPLINE calculates design matrices to fit a natural polynomial or trignometric L-spline as a linear mixed model.
LVARMODEL analyses a field trial using the Linear Variance Neighbour model.
MAANOVA does analysis of variance for a single-channel microarray design.
MABGCORRECT performs background correction of Affymetrix slides.
MACALCULATE corrects and transforms two-colour microarray differential expressions.
MADESIGN assesses the efficiency of a two-colour microarray design.
MAEBAYES modifies t-values by an empirical Bayes method.
MAESTIMATE estimates treatment effects from a two-colour microarray design.
MAHISTOGRAM plots histograms of microarray data.
MANNWHITNEY performs a Mann-Whitney U test.
MANOVA performs multivariate analysis of variance and covariance.
MANTEL assesses the association between similarity matrices.
MAPCLUSTER clusters probes or genes with microarray data.
MAPLOT produces two-dimensional plots of microarray data.
MAREGRESSION does regressions for single-channel microarray data.
MARGIN forms and calculates marginal values for tables.
MARMA calculates Affymetrix expression values.
MAROBUSTMEANS does a robust means analysis for Affymetrix slides.
MASCLUSTER clusters microarray slides.

MASHADE produces shade plots to display spatial variation of microarray data.
MATRIX declares one or more matrix data structures.
MAVDIFFERENCE applies the average difference algorithm to Affymetrix data.
MAVOLCANO produces volcano plots of microarray data.
MA2CLUSTER performs a two-way clustering of microarray data by probes (or genes) and slides.
MCNEMAR performs McNemar's test for the significance of changes.
MCOMPARISON performs pairwise multiple comparison tests within a table of means.
MCORANALYSIS does multiple correspondence analysis.
MCOVARIOGRAM fits models to sets of variograms and cross-variograms.
MCROSSPECTRUM performs a spectral analysis of a multiple time series.
MC1PSTATIONARY gives the stationary probabilities for a 1st-order Markov chain.
MDS performs non-metric multidimensional scaling.
MEDIANTETRAD gives robust identification of multiple outliers in 2-way tables.
MERGE copies subfiles from backing-store files into a single file.
META combines estimates from individual trials.
MICHAELISMENTEN fits the Michaelis-Menten equation for substrate concentration versus time data.
MINFIELDWIDTH calculates minimum field widths for printing data structures.
MINIMIZE finds the minimum of a function calculated by a procedure.
MIN1DIMENSION finds the minimum of a function in one dimension.
MMPREDICT predicts the Michaelis-Menten curve for a particular set of parameter values.
MNORMALIZE normalizes two-colour microarray data.
MODEL defines the response variate(s) and the type of model to be fitted for linear, generalized linear, generalized additive, and nonlinear models.
MONOTONIC fits an increasing monotonic regression of y on x.
MOVINGAVERAGE calculates and plots the moving average of a time series.
MPOLISH performs a median polish of two-way data.
MPOWER forms integer powers of a square matrix.
MSEKERNEL2D estimates the mean square error for a kernel smoothing.
MTABULATE forms tables classified by multiple-response factors.
MULTMISSING estimates missing values for units in a multivariate data set.
MVAOD does an analysis of distance of multivariate data.
MVARIOGRAM fits models to an experimental variogram.
MVFILL replaces missing values in a vector with the previous non-missing value.
NAG calls an algorithm from the NAG Library.
NCONVERT converts integers between base 10 and other bases.
NCSPLINE calculates natural cubic spline basis functions (for use e.g. in REML)
NLAR1 fits curves with an AR1 or a power-distance correlation model.
NLCONTRASTS fits nonlinear contrasts to quantitative factors in ANOVA.
NNDISPLAY displays output from a multi-layer perceptron neural network fitted by NNFIT.
NNFIT fits a multi-layer perceptron neural network.
NNPREDICT forms predictions from a multi-layer perceptron neural network fitted by NNFIT.
NORMTEST performs tests of univariate and/or multivariate Normality.
NOTICE provides news and other information about Genstat.
OPEN opens files.
OPLS performs orthogonal partial least squares regression.
OPTION defines the options of a Genstat procedure with information to allow them to be checked when the procedure is executed.
OR introduces a set of alternative statements in a "multiple-selection" control structure.
ORTHPOLYNOMIAL calculates orthogonal polynomials.
OUTPUT defines where output is to be stored or displayed.
OWN does work specified in Fortran subprograms linked into Genstat by the user.
PAGE moves to the top of the next page of an output file.
PAIRTEST performs t-tests for pairwise differences.
PARAMETER defines the parameters of a Genstat procedure with information to allow them to be checked when the procedure is executed.
PARTIALCORRELATIONS calculates partial correlations for a list of variates.

PASS does work specified in subprograms supplied by the user, but not linked into Genstat. This directive may not be available on some computers.
PCO performs principal coordinates analysis, also principal components and canonical variates analysis (but with different weighting from that used in CVA) as special cases.
PCOPROCRUSTES performs a multiple Procrustes analysis.
PCORELATE relates the observed values on a set of variables to the results of a principal coordinates analysis.
PCP performs principal components analysis.
PDESIGN prints or stores treatment combinations tabulated by the block factors.
PDUPLICATE duplicates a pointer, with all its components.
PEAKFINDER finds the locations of peaks in an observed series.
PEN defines the properties of "pens" for high-resolution graphics.
PENSPLINE calculates design matrices to fit a penalized spline as a linear mixed model.
PERCENT expresses the body of a table as percentages of one of its margins.
PERIODTEST gives periodogram-based tests for white noise in time series.
PERMUTE forms all possible permutations of the integers 1...*n*.
PFACLEVELS prints levels and labels of factors.
PLINK prints a link to a graphics file into an HTML file.
PLS fits a partial least squares regression model.
PNTEST calculates one- and two-sample Poisson tests.
POINTER declares one or more pointer data structures.
POSSEMIDEFINITE calculates a positive semi-definite approximation of a non-positive semi-definite symmetric matrix.
PPAIR displays results of t-tests for pairwise differences in compact diagrams.
PRCORRELATION calculates probabilities for product moment correlations.
PRDOUBLEPOISSON calculates the probability density for the double Poisson distribution.
PREDICT forms predictions from a linear or generalized linear model.
PREWHITEN filters a time series before spectral analysis.
PRIMEPOWER decomposes a positive integer into its constituent prime powers.
PRINT prints data in tabular format in an output file, unformatted file, or text.
PRKTAU calculates probabilities for Kendall's rank correlation coefficient τ
PRMANNWHITNEYU calculates probabilities for the Mann-Whitney U statistic.
PROBITANALYSIS fits probit models allowing for natural mortality and immunity.
PROCEDURE introduces a Genstat procedure.
PRSPEARMAN calculates probabilities for Spearman's rank correlation statistic.
PRWILCOXON calculates probabilities for the Wilcoxon signed-rank statistic.
PSPLINE calculates design matrices to fit a P-spline as a linear mixed model.
PTAREAPOLYGON calculates the area of a polygon.
PTBOX generates a bounding or surrounding box for a spatial point pattern.
PTCLOSEPOLYGON closes open polygons.
PTDESCRIBE gives summary and second order statistics for a point process.
PTGRID generates a grid of points in a polygon.
PTINTENSITY calculates the overall density for a spatial point pattern.
PTKERNEL2D performs kernel smoothing of a spatial point pattern.
PTK3D performs kernel smoothing of space-time data.
PTREMOVE removes points interactively from a spatial point pattern.
PTROTATE rotates a point pattern.
PTSINPOLYGON returns points inside or outside a polygon.
QBESTGENOTYPES sorts individuals of a segregating population by their genetic similarity with a target genotype, using the identity by descent (IBD) information at QTL positions.
QCANDIDATES selects QTLs on the basis of a test statistic profile along the genome.
QCOCHRAN performs Cochran's *Q* test for differences between related-samples
QDESCRIBE calculates descriptive statistics of molecular markers.
QDIALOG produces a modal dialog box to obtain a response from the user.
QDISCRIMINATE performs quadratic discrimination between groups i.e. allowing for different variance-covariance matrices.

QEIGENANALYSIS uses principal components analysis and the Tracy-Widom statistic to find the number of significant principal components to represent a set of variables.
QEXPORT exports genotypic data for QTL analysis.
QFACTOR allows the user to decide to convert texts or variates to factors.
QFLAPJACK creates a Flapjack project file from genotypic and phenotypic data.
QGSELECT obtains a representative selection of genotypes by means of genetic distance sampling or genetic distance optimization.
QIBDPROBABILITIES reads molecular marker data and calculates IBD probabilities.
QIMPORT imports genotypic and phenotypic data for QTL analysis.
QKINSHIPMATRIX forms a kinship matrix from molecular markers.
QLDDECAY estimates linkage disequilibrium (LD) decay along a chromosome.
QLINKAGEGROUPS forms linkage groups using marker data from experimental populations.
QLIST gets the user to select a response interactively from a list.
QMAP constructs genetic linkage maps using marker data from experimental populations.
QMASSOCIATION performs multi-environment marker-trait association analysis in a genetically diverse population using bi-allelic and multi-allelic markers.
QMATCH matches different data structures to be used in QTL estimation.
QMBACKSELECT performs a QTL backward selection for loci in multi-environment trials or multiple populations.
QMESTIMATE calculates QTL effects in multi-environment trials or multiple populations.
QMKDIAGNOSTICS generates descriptive statistics and diagnostic plots of molecular marker data.
QMKRECODE recodes marker scores into separate alleles.
QMKSELECT obtains a representative selection of markers by means of genetic distance sampling or genetic distance optimization.
QMQTLSCAN performs a genome-wide scan for QTL effects (Simple and Composite Interval Mapping) in multi-environment trials or multiple populations.
QMTBACKSELECT performs a QTL backward selection for loci in multi-trait trials.
QMTESTIMATE calculates QTL effects in multi-trait trials.
QMTQTLSCAN performs a genome-wide scan for QTL effects (Simple and Composite Interval Mapping) in multi-trait trials.
QMVAF calculates percentage variance accounted for by QTL effects in a multi-environment analysis.
QMVESTIMATE replaces missing molecular marker scores using conditional genotypic probabilities.
QMVREPLACE replaces missing marker scores with the mode scores of the most similar genotypes.
QNORMALIZE performs quantile normalization.
QRD calculates QR decompositions of matrices.
QRECOMBINATIONS calculates the expected numbers of recombinations and the recombination frequencies between markers.
QREPORT creates an HTML report from QTL linkage or association analysis results.
QSASSOCIATION performs marker-trait association analysis in a genetically diverse population using bi-allelic and multi-allelic markers.
QSBACKSELECT performs a QTL backward selection for loci in single-environment trials.
QSELECTIONINDEX calculates (molecular) selection indexes by using phenotypic information and/or molecular scores of multiple traits.
QSESTIMATE calculates QTL effects in single-environment trials.
QSIMULATE simulates marker data and QTL effects for single and multiple environment trials.
QSQTLSCAN performs a genome-wide scan for QTL effects (Simple and Composite Interval Mapping) in single-environment trials.
QTHRESHOLD calculates a threshold to identify a significant QTL.
QUANTILE calculates quantiles of the values in a variate.
QUESTION obtains a response using a Genstat menu.
RADIALSPLINE calculates design matrices to fit a radial-spline surface as a linear mixed model.
RANDOMIZE randomizes the units of a designed experiment or the elements of a factor or variate.
RANK produces ranks, from the values in a variate, allowing for ties.
RAR1 fits regressions with an AR1 or a power-distance correlation model.
RBDISPLAY displays output from a radial basis function model fitted by RBFIT.
RBFIT fits a radial basis function model.

RBPREDICT forms predictions from a radial basis function model fitted by RBFIT.
RBRADLEYTERRY fits the Bradley-Terry model for paired-comparison preference tests.
RCATENELSON performs a Cate-Nelson graphical analysis of bivariate data.
RCHECK checks the fit of a linear or generalized linear regression.
RCIRCULAR does circular regression of mean direction for an angular response.
RCOMPARISONS calculates comparison contrasts amongst regression means.
RCURVECOMMONNONLINEAR refits a standard curve with common nonlinear parameters across groups to provide s.e.'s for linear parameters.
RCYCLE controls iterative fitting of generalized linear, generalized additive, and nonlinear models, and specifies parameters, bounds etc for nonlinear models.
RDA performs redundancy analysis.
RDESTIMATES plots one- or two-way tables of regression estimates.
RDISPLAY displays the fit of a linear, generalized linear, generalized additive, or nonlinear model.
READ reads data from an input file, an unformatted file, or a text.
RECORD dumps a job so that it can later be restarted by a RESUME statement.
REDUCE is a synonym for HREDUCE.
REFORMULATE modifies a formula or an expression to operate on a different set of data structures.
RELATE is a synonym for PCORELATE.
REML fits a variance-components model by residual (or restricted) maximum likelihood.
RENAME assigns new identifiers to data structures.
REPPERIODOGRAM gives periodogram-based analyses for replicated time series.
RESHAPE reshapes a data set with classifying factors for rows and columns, into a reorganized data set with new identifying factors.
RESTRICT defines a restricted set of units of vectors for subsequent statements.
RESUME restarts a recorded job.
RETRIEVE retrieves structures from a subfile.
RETURN returns to a previous input stream (text vector or input channel).
RFFAMOUNT fits harmonic models to mean rainfall amounts for a Markov model.
RFFPROBABILITY fits harmonic models to rainfall probabilities for a Markov model.
RFINLAYWILKINSON performs Finlay and Wilkinson's joint regression analysis of genotype-by-environment data.
RFSUMMARY forms summaries for a Markov model from rainfall data.
RFUNCTION estimates functions of parameters of a nonlinear model.
RGRAPH draws a graph to display the fit of a regression model.
RIDGE produces ridge regression and principal component regression analyses.
RJOINT does modified joint regression analysis for variety-by-environment data.
RKEEP stores results from a linear, generalized linear, generalized additive, or nonlinear model.
RKESTIMATES saves estimates and other information about individual terms in a regression analysis.
RLASSO performs lasso using iteratively reweighted least-squares.
RLFUNCTIONAL fits a linear functional relationship model
RLIFETABLE calculates the life-table estimate of the survivor function.
RMGLM fits a model where different units follow different generalized linear models.
RMULTIVARIATE performs multivariate linear regression with accumulated tests.
RNEGBINOMIAL fits a negative binomial generalized linear model estimating the aggregation parameter.
RNONNEGATIVE fits a generalized linear model with nonnegativity constraints.
ROBSSPM forms robust estimates of sum-of-squares-and-products matrices.
ROTATE does a Procrustes rotation of one configuration of points to fit another.
RPAIR gives t-tests for all pairwise differences of means from a regression or generalized linear model.
RPARALLEL carries out analysis of parallelism for nonlinear functions.
RPERMTEST does random permutation tests for regression or generalized-linear-model analyses
RPHCHANGE modifies a proportional hazards model fitted by RPHFIT.
RPHDISPLAY prints output for a proportional hazards model fitted by RPHFIT.
RPHFIT fits the proportional hazards model to survival data as a generalized linear model.
RPHKEEP saves information from a proportional hazards model fitted by RPHFIT.
RPHVECTORS forms vectors for fitting proportional hazards data as a generalized linear model.
RPOWER calculates the power (probability of detection) for regression models.

RPROPORTIONAL fits the proportional hazards model to survival data as a generalized linear model.
RQLINEAR fits and plots quantile regressions for linear models.
RQNONLINEAR fits and plots quantile regressions for nonlinear models.
RQSMOOTH fits and plots quantile regressions for loess or spline models.
RQUADRATIC fits a quadratic surface and estimates its stationary point.
RRETRIEVE retrieves a regression save structure from an external file.
RSCHNUTE fits a general 4 parameter growth model to a non-decreasing Y-variate.
RSCREEN performs screening tests for generalized or multivariate linear models.
RSEARCH helps search through models for a regression or generalized linear model.
RSPREADSHEET puts results from a regression, generalized linear or nonlinear model into Genstat spreadsheets.
RSTEST compares groups of right-censored survival data by nonparametric tests.
RSTORE stores a regression save structure in an external file.
RSURVIVAL models survival times of exponential, Weibull, extreme-value, log-logistic or lognormal distributions.
RTCOMPARISONS calculates comparison contrasts within a multi-way table of means.
RUGPLOT draws "rugplots" to display the distribution of one or more samples.
RUNTEST performs a test of randomness of a sequence of observations.
RWALD calculates Wald and F tests for dropping terms from a regression.
RXGENSTAT submits a set of commands externally to R and reads the output.
RYPARALLEL fits the same regression model to several response variates, and collates the output.
R0INFLATED fits zero-inflated regression models to count data with excess zeros.
R0KEEP saves information from a zero-inflated regression model for count data with excess zeros fitted by R0INFLATED.
R2LINES fits two-straight-line (broken-stick) models to data
SAGRAPES produces statistics and graphs for checking sensory panel performance.
SAMPLE samples from a set of units, possibly stratified by factors.
SBNTEST calculates the sample size for binomial tests.
SCALAR declares one or more scalar data structures.
SCORRELATION calculates the sample size to detect specified correlations.
SDISCRIMINATE selects the best set of variates to discriminate between groups.
SEDLSI calculates least significant intervals.
SED2ESE calculates effective standard errors that give good approximate sed's.
SET sets details of the "environment" of a Genstat job.
SETALLOCATIONS runs through all ways of allocating a set of objects to subsets.
SETCALCULATE performs Boolean set calculations on the contents of vectors or pointers.
SETDEVICE opens a graphical file and specifies the device number on basis of its extension.
SETNAME sets the identifier of a data structure to be one specified in a text.
SETOPTION sets or modifies defaults of options of Genstat directives or procedures.
SETPARAMETER sets or modifies defaults of parameters of Genstat directives or procedures.
SETRELATE compares two sets of values in two data structures.
SET2FORMULA forms a model formula using structures supplied in a pointer.
SHELLEXECUTE launches executables or opens files in another application using their file extension.
SIGNTEST performs a one or two sample sign test.
SIMPLEX searches for the minimum of a function using the Nelder-Mead algorithm.
SKEWSYMMETRY provides an analysis of skew-symmetry for an asymmetric matrix.
SKIP skips lines in input or output files.
SLCONCORDANCE calculates the sample size for Lin's concordance coefficient.
SMANNWHITNEY calculates sample sizes for the Mann-Whitney test.
SMCNEMAR calculates sample sizes for McNemar's test.
SMOOTHSPECTRUM forms smoothed spectrum estimates for univariate time series.
SOM declares a self-organizing map.
SOMADJUST performs adjustments to the weights of a self-organizing map.
SOMDESCRIBE summarizes values of variables at nodes of a self-organizing map.
SOMESTIMATE estimates the weights for self-organizing maps.
SOMIDENTIFY allocates samples to nodes of a self-organizing map.

SOMPREDICT makes predictions using a self-organizing map.
SORT sorts units of vectors according to an index vector.
SPCAPABILITY calculates capability statistics.
SPCCHART plots c or u charts representing numbers of defective items.
SPCOMBINE combines spreadsheet and data files, without reading them into Genstat.
SPCUSUM prints CUSUM tables for controlling a process mean.
SPEARMAN calculates Spearman's rank correlation coefficient.
SPEWMA plots exponentially weighted moving-average control charts.
SPLINE calculates a set of basis functions for M-, B- or I-splines.
SPLOAD loads Genstat spreadsheet files.
SPPCHART plots p or np charts for binomial testing for defective items.
SPNTEST calculates the sample size for a Poisson test.
SPRECISION calculates the sample size to obtain a specified precision.
SPSHEWHART plots control charts for mean and standard deviation or range.
SPSYNTAX puts details about the syntax of commands into a spreadsheet.
SSIGNTEST calculates the sample size for a sign test.
SSPM declares one or more SSPM data structures.
STACK combines several data sets by "stacking" the corresponding vectors.
STANDARDIZE standardizes columns of a data matrix to have mean zero and variance one.
STEEL performs Steel's many-one rank test.
STEM produces a simple stem-and-leaf chart.
STEP selects terms to include in or exclude from a linear, generalized linear, or generalized additive model according to the ratio of residual mean squares.
STOP ends a Genstat program.
STORE to store structures in a subfile of a backing-store file.
STRUCTURE defines a compound data structure.
STTEST calculates the sample size for t-tests (including equivalence tests).
SUBSET forms vectors containing subsets of the values in other vectors.
SUSPEND suspends execution of Genstat to carry out commands in the operating system. This directive may not be available on some computers.
SVBOOT bootstraps data from random surveys.
SVCALIBRATE performs generalized calibration of survey data.
SVD calculates singular value decompositions of matrices.
SVGLM fits generalized linear models to survey data.
SVHOTDECK performs hot-deck and model-based imputation for survey data.
SVMERGE merges strata prior to survey analysis.
SVMFIT fits a support vector machine.
SVMPREDICT forms the predictions using a support vector machine.
SVREWEIGHT modifies survey weights, adjusting other weights to ensure that their overall sum remains unchanged.
SVSAMPLE constructs stratified random samples.
SVSTRATIFIED analyses stratified random surveys by expansion or ratio raising.
SVTABULATE tabulates data from random surveys, including multistage surveys and surveys with unequal probabilities of selection.
SVWEIGHT forms survey weights.
SWITCH adds terms to, or drops them from a linear, generalized linear, generalized additive, or nonlinear model.
SYMMETRICMATRIX declares one or more symmetric matrix data structures.
SYNTAX obtains details of the syntax of a command and the source code of a procedure.
TABINSERT inserts the contents of a sub-table into a table.
TABLE declares one or more table data structures.
TABMODE forms summary tables of modes of values
TABSORT sorts tables so their margins are in ascending or descending order.
TABTABLE opens a tabbed-table spreadsheet in the Genstat client.
TABULATE forms summary tables of variate values.
TALLY forms a simple tally table of the distinct values in a vector.

TCOMBINE combines several tables into a single table.
TDISPLAY displays further output after an analysis by TFIT.
TENSORSPLINE calculates design matrices to fit a tensor-spline surface as a linear mixed model.
TERMS specifies a maximal model, containing all terms to be used in subsequent linear, generalized linear, generalized additive, and nonlinear models.
TEXT declares one or more text data structures.
TFILTER filters time series by time-series models.
TFIT estimates parameters in Box-Jenkins models for time series.
TFORECAST forecasts future values of a time series.
THINPLATE calculates the basis functions for thin-plate splines.
TKEEP saves results after an analysis by TFIT.
TOBIT performs a Tobit linear mixed model analysis on data with fixed-threshold censoring.
TRANSFERFUNCTION specifies input series and transfer-function models for subsequent estimation of a model for an output series.
TREATMENTSTRUCTURE specifies the treatment terms to be fitted by subsequent ANOVA statements.
TREE declares a tree, & initializes it to have a single node known as the root.
TRELLIS does a trellis plot.
TRY displays results of single-term changes to a linear, generalized linear, or generalized additive model.
TSM declares one or more TSM data structures.
TSUMMARIZE displays characteristics of time series models.
TTEST performs a one- or two-sample t-test.
TUKEYBIWEIGHT estimates means using the Tukey biweight algorithm.
TVARMA fits a vector autoregressive moving average (VARMA) model.
TVFORECAST forecasts future values from a vector autoregressive moving average (VARMA) model.
TVGRAPH plots a vector autoregressive moving average (VARMA) model.
TXBREAK breaks up a text structure into individual words.
TXCONSTRUCT forms a text structure by appending or concatenating values of scalars, variates, texts, factors, pointers or formulae; allows the case of letters to be changed or values to be truncated and reversed.
TXFIND finds a subtext within a text structure.
TXINTEGERCODES converts textual characters to and from their corresponding integer codes.
TXPAD pads strings of a text structure with extra characters so that their lengths are equal.
TXPOSITION locates strings within the lines of a text structure.
TXPROGRESSION forms a text containing a progression of strings.
TXREPLACE replaces a subtext within a text structure.
TXSPLIT splits a text into individual texts, at positions on each line marked by separator character(s).
TX2VARIATE converts text structures to variates.
T%CONTROL expresses tables as percentages of control cells.
UNITS defines an auxiliary vector of labels and/or the length of any vector whose length is not defined when a statement needing it is executed.
UNSTACK splits vectors into individual vectors according to levels of a factor.
UTMCONVERSION converts between geographical latitude and longitude coordinates and UTM eastings and northings.
VABLOCKDESIGN analyses an incomplete-block design by REML, allowing automatic selection of random and spatial covariance models.
VAIC calculates the Akaike and Schwarz (Bayesian) information coefficients for REML.
VALINEBYTESTER provides combinabilities and deviances for a line-by-tester trial analysed by VABLOCKDESIGN or VAROWCOLUMNDESIGN.
VALLSUBSETS fits all subsets of the fixed terms in a REML analysis.
VAMETA performs a REML meta analysis of a series of trials.
VAOPTIONS defines options for the fitting of models by VARANDOM and associated procedures.
VARANDOM finds the best REML random model from a set of models defined by VFMODEL.
VARECOVER recovers when REML, is unable to fit a model, by simplifying the random model.
VARIATE declares one or more variate data structures.
VAROWCOLUMNDESIGN analyses a row-and-column design by REML, with automatic selection of the best

random and spatial covariance model.
VASDISPLAY displays further output from an analysis by VASERIES.
VASERIES analyses a series of trials with incomplete-block or row-and-column designs by REML, automatically selecting the best random models.
VASKEEP copies information from an analysis by VASERIES into Genstat data structures.
VASMEANS saves experiment × treatment means from analysis of a series of trials by VASERIES.
VAYPARALLEL does the same REML analysis for several y-variates, and collates the output.
VBOOTSTRAP performs a parametric bootstrap of the fixed effects in a REML analysis.
VCHECK checks standardized residuals from a REML analysis.
VCOMPONENTS defines the variance-components model for REML.
VCRITICAL uses a parametric bootstrap to estimate critical values for a fixed term in a REML analysis.
VCYCLE controls details of the REML algorithm.
VDEFFECTS plots one- or two-way tables of effects estimated in a REML analysis.
VDFIELDRESIDUALS display residuals from a REML analysis in field layout.
VDISPLAY displays further output from a REML analysis.
VEQUATE equates values across a set of structures.
VFIXEDTESTS saves fixed tests from a REML analysis.
VFLC performs an F-test of random effects in a linear mixed model based on linear combinations of the responses, i.e. an FLC test.
VFMODEL forms a model-definition structure for a REML analysis.
VFPEDIGREE checks and prepares pedigree information from several factors, for use by VPEDIGREE and REML.
VFRESIDUALS obtains residuals, fitted values and their standard errors from a REML analysis.
VFSTRUCTURE adds a covariance-structure definition to a REML model-definition structure.
VFUNCTION calculates functions of variance components from a REML analysis.
VGESELECT selects the best variance-covariance model for a set of environments.
VGRAPH plots tables of means from REML.
VHERITABILITY calculates generalized heritability for a random term in a REML analysis.
VHOMOGENEITY tests homogeneity of variances and variance-covariance matrices.
VINTERPOLATE performs linear & inverse linear interpolation between variates.
VKEEP copies information from a REML analysis into Genstat data structures.
VLINEBYTESTER analyses a line-by-tester trial by REML.
VLSD prints approximate least significant differences for REML means.
VMATRIX copies values and row/column labels from a matrix to variates or texts.
VMCOMPARISON performs pairwise comparisons between REML means.
VMETA performs a multi-treatment meta analysis using summary results from individual experiments.
VMODEL specifies the model for a REML analysis using a model-definition structure defined by VFMODEL.
VNEARESTNEIGHBOUR analyses a field trial using nearest neighbour analysis.
VORTHPOLYNOMIAL forms orthogonal polynomials over time for repeated measures.
VPEDIGREE generates an inverse relationship matrix for use when fitting animal or plant breeding models by REML.
VPERMTEST does random permutation tests for the fixed effects in a REML analysis.
VPLOT plots residuals from a REML analysis.
VPOWER uses a parametric bootstrap to estimate the power (probability of detection) for terms in a REML analysis.
VPREDICT forms predictions from a REML model.
VRACCUMULATE forms a summary accumulating the results of a sequence of REML random models.
VRADD adds terms from a REML fixed model into a Genstat regression.
VRCHECK checks effects of a random term in a REML analysis.
VRDISPLAY displays output for a REML fixed model fitted in a Genstat regression.
VRDROP drops terms in a REML fixed model from a Genstat regression.
VREGRESS performs regression across variates.
VRESIDUAL defines the residual term for a REML model.
VRFIT fits terms from a REML fixed model in a Genstat regression.

VRKEEP saves output for a REML fixed model fitted in a Genstat regression.
VRMETAMODEL forms the random model for a REML meta analysis.
VRPERMTEST performs permutation tests for random terms in REML analysis.
VRSETUP sets up Genstat regression to assess terms from a REML fixed model.
VRSWITCH adds or drops terms from a REML fixed model in a Genstat regression.
VRTRY tries the effect of adding and dropping individual terms from a REML fixed model in a Genstat regression.
VSAMPLESIZE estimates the replication to detect a fixed term or contrast in a REML analysis, using parametric bootstrap.
VSCREEN performs screening tests for fixed terms in a REML analysis.
VSOM analyses a simple REML variance components model for outliers using a variance shift outlier model.
VSPECTRALCHECK forms the spectral components from the canonical components of a multitiered design, and constrains any negative spectral components to zero.
VSPREADSHEET saves results from a REML analysis in a spreadsheet.
VSTATUS prints the current model settings for REML.
VSTRUCTURE defines a variance structure for random effects in a REML model.
VSUMMARY summarizes a variate, with classifying factors, into a data matrix of variates and factors.
VSURFACE fits a 2-dimensional spline surface using REML, and estimates its extreme point.
VTABLE forms a variate and set of classifying factors from a table.
VTCOMPARISONS calculates comparison contrasts within a multi-way table of predicted means from a REML analysis.
VUVCOVARIANCE forms the unit-by-unit variance-covariance matrix for specified variance components in a REML model.
WADLEY fits models for Wadley's problem, allowing alternative links and errors.
WILCOXON performs a Wilcoxon Matched-Pairs (Signed-Rank) test.
WINDROSE plots rose diagrams of circular data like wind speeds.
WORKSPACE accesses private data structures for use in procedures.
WSTATISTIC calculates the Shapiro-Wilk test for Normality.
XAXIS defines the x-axis in each window for high-resolution graphics.
XOCATEGORIES performs analyses of categorical data from cross-over trials.
XOEFFICIENCY calculates efficiency of estimating effects in cross-over designs.
XOPOWER estimates the power of contrasts in cross-over designs.
YAXIS defines the y-axis in each window for high-resolution graphics.
YTRANSFORM estimates the parameter lambda of a single parameter transformation.
ZAXIS defines the z-axis in each window for high-resolution graphics.
%CD changes the current directory.
%CLOSE closes the binary file opened by %OPEN.
%FPOSITION returns the current position in the binary file opened by %OPEN.
%LOG adds text into the Input Log window in the Genstat client.
%MESSAGEBOX displays text in a dialog in the Genstat client.
%OPEN open a binary file for use with %WRITE.
%SLEEP pauses execution of the server for a time specified in seconds.
%TEMPFILE creates a unique temporary file in the Genstat temporary folder.
%WRITE writes values of data structures to a binary file opened by %OPEN.