GENSTAT

Newsletter

Issue No. 23

Editors

P W Lane
AFRC Institute of Arable Crops Research
Rothamsted Experimental Station
HARPENDEN
Hertfordshire
United Kingdom AL5 2JQ

K I Trinder
NAG Limited
Wilkinson House
Jordan Hill Road
OXFORD
United Kingdom OX2 8DR

Printed and produced by the Numerical Algorithms Group Limited

ISSN 0269-0764

Please note that the cover of this Newsletter has been adapted by kind permission of Oxford University Press, from the cover of the Genstat 5 Reference Manual.

Genstat Newsletter
Issue No. 23

NP1904

March 1989

Contents

Published Twice Yearly by
Rothamsted Experimental Station Statistics Department
and the Numerical Algorithms Group Ltd

Editorial

The main date for your diaries in the coming months is the Genstat Conference on 11-15 September 1989 in Edinburgh. A booking form giving further information is enclosed with this Newsletter and the Editors look forward to meeting you there!

A one-day conference about procedures was held on 10 May at Rothamsted. As with previous meetings of this type, it was again well attended. The article 'A Guide to Procedures in Genstat' is based on material presented by the authors at the meeting. The next one-day conference is scheduled for the Spring of 1990; any suggestions for topics will be welcomed.

The second introductory guide to Genstat 5 is being published by Oxford University Press. 'Genstat 5: a second course' will be available soon and will cost £15 for the paperback version and £35 for hardback. It introduces many of the statistical facilities of Genstat that are not covered in 'Genstat 5: an introduction' – generalized linear and nonlinear regression, multivariate and cluster analysis, analysis of time series as well as giving details about programming and procedure writing.

As reported in the last issue of the Newsletter, versions of Genstat 5 have now been completed for Sun 4 and Sun 386i, as well as Sun 3. All of these are Release 1.3 and have graphics interfaces to GKS. In addition, Release 1.2 has also now become available for the Sequent Symmetry running Dynix. Please contact NAG for news about the status of other Genstat 5 implementations.

Training Courses

NAG and Rothamsted have now established a series of Genstat 5 Introductory Courses. The latest of these courses was held on 6-9 June 1989 and the next course is likely to be in December 1989. The course can also be run at user sites by arrangement with NAG, and in this case the material can be tailored to meet the needs of those users exactly. The course provides a general introduction to Genstat 5 for those with a basic understanding of statistical analysis but little or no knowledge of how to use Genstat. We are also considering running more specialised or advanced courses and we would like advice from you as users on what you feel the courses should cover. For example, we could run a course on the advanced regression facilities of Genstat which assumes basic knowledge about how to use Genstat, or we could run a course that aimed to teach statistical methodology alongside putting that methodology into practice using Genstat. Please let us know what you think and do take into account your colleagues who do not know Genstat but could benefit from its facilities.

Australasian Genstat 5 Workshop, 21-23 November 1988

J Thompson
Applied Mathematics Division
DSIR
Wellington
New Zealand

This highly successful workshop was held at Melbourne University under the joint auspices of the Department of Agriculture and Rural Affairs, Victoria and the Statistical Consulting Centre of Melbourne University.

Stuart Crosbie and Richard Jarrett did a fine job of co-ordinating three days of intensive information transfer. There were 61 participants. The main speakers were Roger Payne and Peter Lane of Rothamsted and they did a quite remarkable 'Cox and Box' act (1867, Arthur Sullivan of Gilbert and Sullivan fame) – each one following the other in turn, keeping the flow of information going at a pace which was manageable, but had me reeling in amazement by the third day that I had been able to absorb so much. It was clear they had done this before and we benefitted from their experience.

The programme followed a logical sequence, starting with an overview in which we were told the philosophy of Genstat 5 and the syntax rules and their rationale. After coffee we got down to tin tacks and learned about data structures, input and output. After an excellent lunch we were into calculating and manipulating, regressing, GLM-ing, curve-fitting and nonlinear modelling. That left the last session of the afternoon for analysis of variance, time-series and multivariate analysis. After a more challenging dinner (for those who chose to eat at the college) the hardy returned to the university for hands-on experience in the Statistical Consulting Centre. The session was rudely interrupted by a bolt of lightning which killed the computer system for a while; however, despite this, those who attended found the session useful. We all got more than a little wet getting back to the college and on awakening in the morning heard it had been the highest 12-hour rainfall ever recorded in Melbourne and some areas were flooded. We were OK though, aside of the odd leak in the old college roof, and the workshop went on.

Day two covered high-resolution graphics, the Procedure Library, procedure writing, Genstat 5 on workstations and PCs and the efficient use of the program. The day closed with an application: a procedure for first-difference analysis, by David Baird.

For the evening, the organisers had arranged a joint meeting with the Statistical Society of Australia (Victoria Branch). We attended their annual Belz Lecture delivered by Dr David Scott, Head of the Statistics Department, La Trobe University. His general message was that all statisticians should be at home with a wide range of computing tools. We then moved on to dinner. The dinner was very well organised and most enjoyable. It was a good idea to combine the conference dinner with the Statistical Society meeting as it gave us a wider group of statisticians to interact with and a let up from the intensity of the workshop.

Day three concentrated initially on the internal structure of Genstat 5 and extending it using Fortran 77. We then moved on to teaching aspects – teaching Genstat to non-statisticians and experiences with introductory and conversion courses from Emlyn Williams' experience as well as Rothamsted's. A code generator was also demonstated by Nam Nguyen. After a critique of Genstat's position with respect to modern statistical practice by Tony Pettit, we were told about plans for Genstat 5 Releases 2 and 3. We had an effective discussion forum when the users were beiing asked to state their needs. For three days we had listened to the Genstat 'Gurus' telling us what they had done. Now it was our turn to tell them what we wanted. The ideas were collected during the discussion by Richard Jarrett and turned rapidly into a hand-out.

Discussion points covered:

- Distribution/installation/pricing policy
- A plea for a timetable for implementation on machines for which Genstat 5 is not yet available
- A local information network was sought. Bob Forrester, Biometrics Unit, CSIRO volunteered to get a newsgroup started on ASCnet.
- A call was made for a future meeting, suggested to be in New Zealand in 1992. I volunteered to get the ball rolling at least.
- A need was seen for rapid improvement of the PC version, problems being speed and workspace. Reliability and efficiency were considered more important than windows and other 'fancy stuff'.
- Documentation. There were requests for more (more complicated) examples.
- New features sought:
  - Graphics: the standard needs to be raised to publication standard. Roger Payne said that graphics were already publication standard, but requested information on comparisons between Genstat 5 and other packages (e.g. S). He would like to be sent examples of what users consider 'good graphics'.
  - smoothing
  - getting out design matrices
  - fractional factorials
  - new procedures
  - menu-driven interface – seen as needed for low-level use but undesirable for regular users.

The forum was wound up with a range of comments on the current version, all of which I am sure were noted by Roger and Peter, who I must add appeared to weather the onslaught very well.

And that was it. We staggered away replete and exhausted.

For me and for my New Zealand colleagues it was a very valuable three days and I am pretty confident the same would be true for the overwhelming majority of the Australian contingent. This sort of total immersion therapy works provided it is not done too frequently. I think it worked so well because it was well organised and there was something real and new to say. We will probably all be ready for another dose in four year's time.

A Guide to Procedures in Genstat

R W Payne
P G N Digby
Statistics Department
AFRC Institute of Arable Crops Research
Rothamsted Experimental Station
Harpenden
Hertfordshire
United Kingdom AL5 2JQ

G M Arnold
Biometrics Section
Department of Agricultural Sciences
University of Bristol
AFRC Institute of Arable Crops Research
Long Ashton Research Station
Long Ashton
Bristol
United Kingdom BS18 9AF

1. Introduction

The Genstat 5 procedure provides a convenient way of storing a Genstat program for future use. The program is self-contained, with the information that it requires being transferred via the options and parameters defined for the procedure. So, once a program has been formed into a procedure, it can be made available to be used by anyone, without their having to worry about clashes between identifier names within and outside the procedure. They also do not need to know (or should not need to know) anything about how the procedure works.

In this article we explain how to define a procedure, with its options and parameters, and how to form libraries of procedures for convenient accessing in future. We cover some of the techniques that have been devised for such things as argument-checking and for efficient coding. We also give information about the Genstat Procedure Library.

2. Writing and Storing Genstat Procedures

The rules of syntax for procedures are given in Section 6.3 of the Genstat Manual. Below, these are recapitulated, emphasising the most important aspects – and those that have caused confusion. The example below shows a procedure called `TRANSFORM`, for making various transformations of percentages. Notice that we can use a name that differs from one of the Genstat directives only in characters 5-8 – in this case the directive `TRANSFERFUNCTION`. For the Genstat Procedure Library, though, we have been more stringent, requiring the procedure names to be distinct in at least one of the first four characters, but this is in case any of them were to become a directive in future (as may happen to `LIBHELP`). The procedure `TRANSFORM` will have an option called `METHOD` to control which transformation is done, and two parameters: `%` for the percentages, and `RESULT` to contain the transformed values (remember that `%` is considered to be a letter in Genstat programs).

```
PROCEDURE 'TRANSFORM'     "notice that the first six letters of the name
                           are the same as the directive TRANSFERFUNCTION"
" Define the arguments of the procedure."
OPTION    NAME='METHOD';  \
          MODE=T;         "T => mode text i.e. strings" \
          DEFAULT='LOGIT' "default to be taken if METHOD is not set"
PARAMETER NAME='%','RESULT'; \
          MODE=P          "P => mode pointer i.e. identifiers"
```

```

IF METHOD .EQS. 'LOGIT'
  CALCULATE RESULT = LOG( % / (100-%) )
ELSIF METHOD .EQS. 'COMPLOGLOG'
  CALCULATE RESULT = LOG( -LOG((100-%)/100) )
ELSIF METHOD .EQS. 'ANGULAR'
  CALCULATE RESULT = ANGULAR(%)
ELSE
  PRINT '***** Invalid setting of METHOD option.*****'
  EXIT [CONTROL=procedure]
ENDIF

ENDPROCEDURE
```

The definition of a procedure begins with a PROCEDURE statement, and terminates with an ENDPROCEDURE statement. The PROCEDURE statement also defines the name of the procedure. This can be of any length, with the same rules as for Genstat identifiers: the first character must be a letter, subsequent characters can be either letters or digits; any characters after the first eight will be ignored.

Next the options and parameters of the procedure are defined. The OPTION directive has parameters NAME, MODE and DEFAULT; while PARAMETER just has NAME and MODE. NAME defines the keyword used for that option or parameter when the procedure is used. (It can then be given in upper or lower or mixed case.) It also defines a Genstat dummy used within the procedure to refer to whatever setting may have been supplied from the program outside. However (and this is one potential snare even though it is mentioned clearly in the Manual at the top of page 246), the identifier of the dummy must be in capital letters when used in the procedure. MODE specifies a single-letter code like those used to specify unnamed structures:

P	(pointer)	→	identifier
V	(variate)	→	number
T	(text)	→	string
E	(expression)		
F	(formula)		

Finally, for the OPTION directive, there is the DEFAULT parameter to define default settings for use if any option is not specified when the procedure is used.

One thing that the Manual does not tell you is that you should not mix mode P with the other modes in the parameters of the procedure (although this is mentioned in the instructions for authors of Genstat library procedures, [1]). The difficulty arises because the settings of parameters of mode other than P are combined together by Genstat to form a single unnamed structure. For example, if a procedure is defined as:

```
PROCEDURE 'FACREAD'
PARAMETER NAME='STRUCTURE','FREPRESENTATION'; MODE=P,T
```

and then called with

```
FACREAD Sex,Dose; labels,levels
```

the second parameter will be combined into the structure !T(labels,levels) and supplied in both calls to the procedure, firstly with the first parameter set as Sex and then with it set as Dose. Within the procedure there is no way of knowing whether it is being called for the first or second time, and so the rules of parallelism would break down.

Having defined the arguments of the procedure, there then follow the statements to be executed when the procedure is invoked (that is, the body of the procedure), and the whole construct is terminated by an ENDPROCEDURE statement.

Next, in the example, the procedure is executed with the `%` parameter set to the variate `Percent`, and the results are stored in a variate called `Logit`. (The default for the `METHOD` option is to take a logit transformation.) Then it is used to take a complementary log-log transformation. Finally, there is an attempt to take an angular transformation, but this fails as the tests in the `IF` block within the procedure assume that `METHOD` is given in full and in capitals.

```
" Try out the procedure.."
VARIATE   [VALUES=1,5,10,25,50,75,90,95,99] Percent
TRANSFORM Percent; RESULT=Logit          "default setting for METHOD"
PRINT     Percent,Logit; DECIMALS=0,3

   Percent       Logit
         1      -4.595
         5      -2.944
        10      -2.197
        25      -1.099
        50       0.000
        75       1.099
        90       2.197
        95       2.944
        99       4.595

TRANSFORM [METHOD=COMPLOGLOG] Percent; Cloglog
PRINT     Percent,Cloglog; DECIMALS=0,3

   Percent     Cloglog
         1      -4.600
         5      -2.970
        10      -2.250
        25      -1.246
        50      -0.367
        75       0.327
        90       0.834
        95       1.097
        99       1.527

TRANSFORM [METHOD=ang] Percent; RESULT=Angle


***** Invalid setting of METHOD option.*****
```

This is clearly an irritation, so the procedure is redefined below to illustrate how to check for abbreviated and alternative forms of textual settings. The expression

```
METHOD .IN. !T(ANGULAR,ANGULA,ANGUL,ANGU,ANG,AN,A,\
               angular,angula,angul,angu,ang,an,a )
```

will generate the result 1 (corresponding to ***true***) if any abbreviation of `ANGULAR` or `angular` is specified. Note that the options and parameters of a procedure cannot be changed within a job and, in Release 1, Genstat ensures this by giving a fault message if a redefined procedure contains any `OPTION` or `PARAMETER` statements.

```
" You can redefine the procedure but not its arguments."
PROCEDURE 'TRANSFORM'

IF METHOD .IN. !T(LOGIT,LOGI,LOG,LO,L,logit,logi,log,lo,l)
  CALCULATE RESULT = LOG( % / (100 - %) )
ELSIF METHOD .IN. !T( \
    COMPLOGLOG,COMPLOGLO,COMPLOGL,COMPLOG,COMPLO,COMPL,COMP,COM,CO,C,\
    comploglog,comploglo,complogl,complog,complo,compl,comp,com,co,c )
  CALCULATE RESULT = LOG( -LOG((100-%)/100) )
ELSIF METHOD .IN. !T(ANGULAR,ANGULA,ANGUL,ANGU,ANG,AN,A,\
                    angular,angula,angul,angu,ang,an,a )
  CALCULATE RESULT = ANGULAR(%)
```

```
 41  ELSE
 42    PRINT '***** Invalid setting of METHOD option.*****'
 43    EXIT [CONTROL=procedure]
 44  ENDIF
 45
 46  ENDPROCEDURE
 47  " Try out the new procedure.."
 48  TRANSFORM [METHOD=ang] Percent; RESULT=Angle
 49  PRINT     Percent,Angle; DECIMALS=0,2

    Percent       Angle
          1        5.74
          5       12.92
         10       18.43
         25       30.00
         50       45.00
         75       60.00
         90       71.57
         95       77.08
         99       84.26
```

Procedures can be stored in backing-store files like any other data structure. They need to be stored separately from other structures; the option setting `PROCEDURE=yes` must be given in the `STORE` statement to indicate that a subfile specifically for procedures is to be used. For efficiency it is generally best to store a single procedure in each subfile, and give the subfile the same name as the procedure. However, if a main procedure calls utility procedures that are specific to the main procedure, then the complete suite of procedures might be stored together.

```
  50  " Open a backing-store file to store the procedure."
  51  OPEN 'MYLIB'; CHANNEL=1; FILETYPE=backingstore
  52  " Store TRANSFORM in a procedure subfile. Note: retrieval is
 -53    faster if the procedure is stored in a subfile of the same name."
  54  STORE [CHANNEL=1; SUBFILE=TRANSFORM; PROCEDURE=yes] TRANSFORM
  55  " Close the file."
  56  CLOSE CHANNEL=1; FILETYPE=backingstore
  57  ENDJOB

******** End of job.  Maximum of 3720 data units used at line 54 (38762 left)
Genstat 5  Release 1.3  (Vax/VMS4)
Copyright 1988, Lawes Agricultural Trust (Rothamsted Experimental Station)

  58  " Open MYLIB as a procedure library."
  59  OPEN 'MYLIB'; CHANNEL=1; FILETYPE=procedurelibrary
  60  " TRANSFORM (which will have been deleted at ENDJOB) will
 -61    now be accessed automatically from MYLIB when required.."
  62  VARIATE     [VALUES=10,20...90] Percentage
  63  TRANSFORM   [METHOD=COMPL] Percentage; CompLogLog
  64  PRINT       Percentage,CompLogLog; DECIMALS=0,3

   Percenta    CompLogL
         10      -2.250
         20      -1.500
         30      -1.031
         40      -0.672
         50      -0.367
         60      -0.087
         70       0.186
         80       0.476
         90       0.834
```

A procedure library is a backing-store file, containing procedures, which is attached to Genstat using one of the channels specifically allocated to procedure libraries (see line 59 of the example). Then (as is explained on page 243 of the Genstat 5 Reference Manual), if a statement name is not recognised as one of the Genstat 5 directives, Genstat will check for a procedure of that name, firstly already stored within the program, then within any procedure libraries explicitly attached to the program (like MYLIB above), then the Site Procedure Library, and finally the (official) Genstat Procedure Library.

3. Techniques Useful for Procedure Writing

There are many useful techniques that have been devised, discovered or were designed into Genstat to help with the writing of procedures.

First of all, if you intend to make your procedure available to other users, you need to think about argument checking, and the production of clear error diagnostics. Otherwise the procedure may fail for reasons that are only indirectly due to the actual mistake, and it can be hard for the user to see where to make the correction.

In the Genstat 5 Library there is a procedure CHECKARGUMENT (Payne, 1987) which provides a stringent check of many aspects. In Release 2 of Genstat, there will be similar facilities in the OPTION and PARAMETER directives, to enable more checking to be done automatically within Genstat. The help information for CHECKARGUMENT is obtained as follows:

```
  65  LIBHELP     [PRINT=index,description,options,parameters] 'CHECKARGUMENT'

   Procedure       CHECKARG

Help['index']
CHECKARGUMENT checks the arguments of a procedure

Help['description']
This procedure can be used to check that each argument of a procedure is set,
that it is set to a structure of a valid type, that the structure has values,
and (for structures of type text or variate) that the values belong to a
defined set. The requirements for each argument are specified by the
parameters of CHECKARGUMENT; if there is anything that is not to be checked,
the corresponding parameter should be left unset. The scalar specified by the
ERROR option is set to 1 if an error is discovered, and an explanatory
message is printed; this scalar should be initialized to zero before calling
CHECKARGUMENT.

Option: ERROR.  Parameters: STRUCTURE,SET,TYPE,DECLARED,VALUES,PRESENT.

Help['options']
ERROR      = scalar        This scalar is given the value 1 if any errors are
                           detected; it should have the value 0 on entry

Help['parameters']
STRUCTURE = identifiers    Lists the structures (arguments) to be checked
SET        = string        Indicates whether or not each structure must be set
                           (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); default *
DECLARED   = string        Indicates whether or not each structure must have
                           been declared (no, yes); default no
VALUES     = variate       Defines the allowed values for a structure of type
             or text       variate or text
PRESENT    = string        Indicates whether or not each structure must have
                           values (no, yes); default no
```

Here is an example showing how the arguments of procedure AONEWAY (Payne, 1987) are checked.

```
" check arguments "
CALCULATE E=0
CHECKARGUMENT [ERROR=E] \
  PRINT,GROUPS,HOMOGENEITY,EXPLAIN,Y; \
  SET='yes'; \
  TYPE='text',!T(factor,text,variate),2('text'),'variate'; \
  DECLARED='yes'; \
  VALUES=2(*,!T(no,n,yes,ye,y,NO,N,YES,YE,Y)),*; \
  PRESENT='yes'
EXIT [CONTROL=procedure] E
```

CHECKARGUMENT operates by accessing the attributes of each argument, either by functions within Genstat expressions or by GETATTRIBUTE, and then checking them in IF statements, with EXIT to leave the procedure (after printing a suitable message) if there are errors.

Even if you do not want to do stringent checks, you may still want to use the UNSET function to check whether an argument has been set (and ASSIGN a default if not); this example is taken from GENPROC (Arnold and Payne, 1988):

```
" declare (and initialize) output structures or set
  to locals if the corresponding parameters are unset "
IF UNSET(XOUTPUT)
  ASSIGN LXOUTPUT; POINTER=XOUTPUT
ENDIF
MATRIX [ROWS=Nrows; COLUMNS=Ncols] XOUTPUT[1...Nconfig]; VALUES=XINPUT[]
```

The procedure TRANSFORM showed one way of interpreting an argument with a string setting. Where the argument may have several settings, the expression becomes more complicated. In this example from BIPLOT (Harding, 1988) scalars centre, normal and tprint[1,2] are set to logical values to indicate whether there is to be centering (STANDARDIZE setting centre), or normalization, and whether singular values and scores are to be printed. Since the STANDARDIZE and PRINT options can have more than one setting the result of the expression

```
PRINT .IN. !T(SCORES,SCORE,SCOR,SCO,SC,scores,score,scor,sco,sc)
```

will have as many values as the structure to which PRINT is set. By using the SUM function the resulting logical scalar, tprint[2], will indicate whether any of them match.

```
CALC centre,normal,tprint[1,2] = SUM( \
     2(STANDARDIZE,PRINT) .IN. \
     !T(CENTRE,CENTR,CENT,CEN,CE,C,centre,centr,cent,cen,ce,c), \
     !T(NORMALIZE,NORMALIZ,NORMALI,NORMAL,NORMA,NORM,NOR,NO,N,  \
        normalize,normaliz,normali,normal,norma,norm,nor,no,n), \
     !T(SINGULAR,SINGULA,SINGUL,SINGU,SING,SIN,SI,S,            \
        singular,singula,singul,singu,sing,sin,si,s),           \
     !T(SCORES,SCORE,SCOR,SCO,SC,scores,score,scor,sco,sc))
```

The CONCATENATE directive can be very useful for checking strings up to a specified number of characters. For example, in LIBHELP, the name of the procedure (specified by the PROCEDURE parameter) needs to be exactly eight characters to match the way in which the names have been stored in the backing-store file that holds the help information for the Library; the statement below abbreviates a setting of PROCEDURE that is longer than eight characters, or pads out with spaces one that is shorter.

```
CONCATENATE [NEWTEXT=Procname] PROCEDURE; WIDTH=8
```

In some procedures, the strings are sufficiently distinct that only the first few characters are checked, as illustrated by this example from APLOT (Todd, 1988); this further shows how an integer-valued scalar can be calculated for use in a subsequent CASE statement.

```
IF unset(METHOD)
  VARIATE vmeth; VALUE=1
ELSE
  CONCATENATE [t1] METHOD; WIDTH=2
  CALC vmeth = t1 .in. !t('F ','f ',Fi,fI,FI,fi) + \
               t1 .in. !t('N ','n ',No,nO,NO,no) * 2 + \
               t1 .in. !t('h ','H ',ha,HA,Ha,hA) * 3 + \
               t1 .in. !t(hi,HI,hI,Hi) * 4
ENDIF
...
...
FOR dmeth=#vmeth
  VARIATE quant,yy; VALUES=!(#nmv(*),(1...nobs)),residuals
  CASE dmeth
    "fitted values versus residuals"
    GRAPH [YTITLE='residuals'; XTITLE='fitted values'] residuals; fitted
  OR
    "Normal plot"
    SORT yy
    CALC quant = ned( (quant-0.375) / (nobs+0.25) )
    GRAPH [TITLE='Normal plot'; YTITLE='residuals'; \
      XTITLE='expected Normal quantiles'] yy; quant
  OR
    "half-Normal plot"
    CALC yy = sort(abs(yy))
    & quant = ned( 0.5 + (quant-0.375) / (nobs+0.25) / 2 )
    GRAPH [TITLE='half-Normal plot'; YTITLE='absolute residuals'; \
      XTITLE='expected Normal quantiles'] yy; quant
  OR
    "histogram"
    CALC  hse = 0.5 * sqrt( var(residuals) )
    & min,max = int(min,max/hse) * hse
    &    min1 = min + hse
    HISTOGRAM [LIMITS=!(min,min1...max)] residuals
  ELSE
    PRINT [IPRINT=*] '***** Invalid setting for METHOD parameter:', \
                     METHOD,'*****.'
  ENDCASE
ENDFOR
```

A procedure may modify the environment of the Genstat job. Sometimes this may be a specified intention of the procedure, but if it is a side effect the original environment should be saved (by GET) on entry to the procedure and restored (by SET) before exit. For example, three aspects of the environment are the current settings for the block model (as set by the BLOCKSTRUCTURE directive), for the treatment model (from TREATMENTSTRUCTURE), and of the analysis-of-variance save structure (as set by ANOVA and used by ADISPLAY and AKEEP). This example shows how these are saved, and then restored, within ALIAS (Payne, 1987).

```
" save special structures "
GET [SPECIAL=Savesp]
...
...
" restore special structures "
SET [BLOCKSTRUCTURE=Savesp['blockstructure']; \
  TREATMENTSTRUCTURE=Savesp['treatmentstructure']; \
  ASAVE=Savesp['asave']]
```

Similarly, the Help procedures restore the PAUSE setting on exit, having set it to 20 while they are printing the information.

```
GET [ENVIRONMENT=Esave]
...
...
" reset PAUSE "
SET [PAUSE=Esave[5]]
```

Efficiency of coding is important in any Genstat program, but particularly for procedures that are intended for repeated use. The general principle is to use the power of the Genstat language to cut down on the number of statements. For example, most directives operate on lists, so declarations and calculations should be combined into a single statement whenever possible. Declarations of vectors can be combined even when they do not share a common length, as shown in the following statement from ANTTEST (Payne and Ridout, 1989).

```
VARIATE Dtemp, Resdf,Resss, \
  Treatdf[1...NTREATTERMS],Treatss[1...NTREATTERMS];\
  VALUES=!(#NUnits(0)), 2(!(#Q(*))), 2(#NTREATTERMS(!(#Q(*))))
```

Calculations can be made more efficient not only by using lists in the expression, but also by the use of embedded assignments, as illustrated by this example from MANOVA (Payne and Arnold, 1988).

```
CALC BartChi = - (a = ErrDF-(NYvars-TreatDF[Term]+1)/2 ) * \
                 (WLambda = SUM( LOG( 1 / (1+TermLRV['Roots']) ) ))
&    WLambda = EXP(WLambda)
&    b       = MVREP( SQRT( ((NYvars**2) * (TreatDF[Term]**2) - 4) / \
                      ((NYvars**2) + (TreatDF[Term]**2) - 5) ); 1)
&    DF2     = ( a*b ) - ( (DF1 = NYvars * TreatDF[Term]) - 2 ) / 2
&    DF1,DF2 = DF1,DF2 * (DF1,DF2 >= 0) / (DF1,DF2 >= 0)
```

When a Genstat pointer, e.g. Vlist, consists of a set of variates the VSUM function can be used to sum them; for example:

```
CALCULATE Varsum = VSUM(Vlist)
```

For other types of structure there is no corresponding function; however, a parallel calculation can be used to avoid using a FOR loop, as illustrated in the example below for forming the matrix Z in GENPROC (Arnold and Payne, 1988).

```
MATRIX [ROWS=Nrows; COLUMNS=Ncols] Y,Z
...
...
CALCULATE WSS,Z = 0
&   #Nconfig(Z) = Z + XOUTPUT[1...Nconfig]
```

By using an assignment in the statements of an IF block, the results of a logical test can be saved for future use, as shown in another example from GENPROC.

```
EXIT Success = LastRSS-RSS < TOLERANCE
...
...
IF .NOT. Success
  PRINT '****** Generalized Procrustes has failed to converge:
      decrease TOLERANCE or increase MAXCYCLE ******'; JUSTIFICATION=left
  EXIT [CONTROL=procedure]
ENDIF
```

It is often also possible to print a complete sequence of results with a single PRINT statement by printing in series: this is a further example from GENPROC.

```
PRINT [SERIAL=yes; IPRINT=*; ORIENTATION=across] \
  '*** Percentage variance ***',DiagNcol, \
  '*** Coordinates ***',X, \
  '*** Rotation matrix ***',ROTATIONS[Config], \
  '*******************************************************************'; \
  FIELDWIDTH=8; DECIMALS=2(2,4,3)
```

As a final point concerning efficiency, it is important not to forget well established techniques such as the ability to keep the details of an experimental design using the DESIGN option of ANOVA (this example is taken from MANOVA).

```
IF UNSET(APRINT)
  ANOVA [PRINT=*; DESIGN=Design] Y[]; SAVE=Save
ELSE
  ANOVA [PRINT=#APRINT; DESIGN=Design] Y[]; SAVE=Save
ENDIF
COVARIATE  Y[]
ANOVA [PRINT=*; DESIGN=Design] Dum
```

When it is necessary to refer specifically to elements of pointers, it is important to remember that the user may not be using suffixes 1 to *n*; for example,

```
VARIATE Yield[1980...1988]
```

will create the pointer Yield, with suffixes 1980, 1981, ... 1988. Inside a procedure this can cause difficulty and the easiest way to avoid it is to make a pointer local to the procedure with the same values but with suffixes running from 1 to *n*. In the example below from VINTERPOLATE (Reader, 1989), the pointer Oval is used to reference the input structure OLDVALUES, and P1 and P2 are used to reference its elements. It is important that the original pointer is reset to itself at the end, as shown below; otherwise Genstat might identify the structures as Oval[1...*n*] if they were printed later in the job.

```
POINTER Oval; OLDVALUES
GETATTRIBUTES [ATTRIBUTE=nvalues] Oval[1]; Attributes
CALCULATE Noldvalues=NVALUES(Oval)
&         Nvalues=Attributes['nvalues']
&         Noldi1=( Noldintervals=NVALUES(OLDINTERVALS) )-1
...
...
"Set up variates to reference elements of pointers easily"
VARIATE P1,P2,P3; VALUES=!(1...Noldi1),!(2...Noldintervals),\
  !(1...Noldintervals)
...
...
CALCULATE #Noldi1(Adiff)=#Noldi1(Adiff)+((Oval[#P2]-Oval[#P1]).LT.0)
...
...
"Reset OLDVALUES pointer so that it refers to its original suffixes."
POINTER OLDVALUES; Oval
```

Similarly, you should not assume that the levels of a factor are the numbers 1 to *n*. You can get a variate containing the levels by repeated use of GETATTRIBUTE, although you will need to handle the situation where a levels vector has not been defined explicitly in the factor declaration: the example below from DISCRIMINATE (Digby, 1988) shows how this can be done to form the variate Levvals and the dummy Levs from the input factor GROUPS.

```
GETATTRIBUTE [type,nvalues,nlevels,nmv,levels,labels] \
             DATA,GROUPS; Pd,Pg
...
...
    " get sizes, and levels and labels "
    CALC Nv,Nu,Ng = (Pd,Pg)['nvalues'],Pg['nlevels']
      &  Nr = Nv*(Nv < Ng) + (Ng-1)*(Nv >= Ng)
    GETATTRIBUTE [type] Pg[4]; Plevs
    DUMMY Levs
    IF nmv(Plevs['type']) == 0
      ASSIGN Pg['levels']; Levs
      CALC Levvals = Pg['levels']
    ELSE
      ASSIGN Ng; Levs
      VARIATE Levvals; !(1...Ng)
    ...
    ...
    FACTOR [NVALUES=Nu; LEVELS=Levs] NEWGROUPS
```

A potential clash of identifiers will arise if a structure local to the procedure is used as an attribute of an output structure (defined by one of the options or parameters), since the local structure will also have to be returned to the user's program. In the previous example, the factor NEWGROUPS will be returned to the user with Levs as one of its attributes. The dummy Levs either holds a scalar value or else points to the levels vector of the GROUPS parameter, which therefore must already exist in the user's program outside the procedure; so there will not be a problem. In the next example, from SKEWSYMMETRY (Digby, 1988), it is required to create the parameter SCORES as a matrix with columns indexed by the numbers 1.1, 1.2, 2.1, 2.2, ... *n*.1, *n*.2 (where *n* is determined from the input parameter and is stored in the scalar Nbimens below). The variate Plane is formed to hold the column indexes, but when the SCORES parameter is declared, Plane is used indirectly by inserting its values into an unnamed structure which can then be returned quite safely to the user.

```
VARIATE Plane; !((0.9,0.1)#Nbimens); DECIMAL=1
CALCULATE Plane = cum(Plane) + 0.2
...
...
MATRIX [ROWS=Nrc; COLUMNS=!(#Plane)] SCORES
```

Among the other techniques that have been discovered or devised, is the use of the SORT directive to transfer all the relevant attributes of one vector to another. This example, taken from SUBSET (Payne, 1988), then modifies the attributes that need to be changed in the new vectors.

```
" define the new vectors (SORT does default declarations
                           & carries all attributes across) "
SORT [INDEX=Format] Oldvec[]; Newvec[]
...
...
FOR Old=Oldvec[]; New=Newvec[]
  GETATTRIBUTES [ATTRIBUTE=type] Old; SAVE=OldvAtt
  " modify the numbers of values "
  CASE OldvAtt['type']
    PRINT ' ***** OLDVECTOR must be a factor, text or variate' "scalar"
    EXIT [CONTROL=procedure]
  OR "factor"
    FACTOR  [NVALUES=NSubset; MODIFY=yes] New
  OR "text"
    TEXT    [VALUES=#NSubset(' '); MODIFY=yes] New
  OR "variate"
    VARIATE [NVALUES=NSubset; MODIFY=yes] New
  ELSE
    PRINT ' ***** OLDVECTOR must be a factor, text or variate'
    EXIT [CONTROL=procedure]
  ENDCASE
ENDFOR
```

The identifier of a structure, `X` say, can be printed using an unnamed pointer:

```
PRINT !p(X)
```

The final example shows how `CHECKARGUMENT` arranges to print `ARGUMENT['?']` if the argument being tested is an unnamed structure.

```
POINTER [NVALUES=!T('?')] Argument; VALUES=!P(STRUCTURE)
...
...
PRINT [IPRINT=*; SQUASH=yes] 'Type of structure',Argument, \
  'incorrect: should be '; FIELDWIDTH=1
PRINT [IPRINT=*; SQUASH=yes] TYPE
EXIT [CONTROL=procedure]
```

4. The Genstat Procedure Library

Genstat 5 is distributed with a library of procedures which is attached automatically whenever Genstat is run. The procedures in the library are thus immediately available to every Genstat user. The library is regularly updated and is labelled with a release number to differentiate each update; for example, Release 1.3/2 indicates that it is the second version of the library to be attached to Release 1.3 of Genstat 5. New releases of the library will be distributed to sites on floppy disks, thus being independent of the central distribution of Genstat releases and more easily kept up-to-date.

The library is structured in modules based on statistical subject area; in Release 1.3/2 the modules are *AOV* (analysis of variance), *BASIC* (basic statistics and exploratory data analysis), *DISTRIBUTIONS* (distributions and simulation), *GLM* (generalized linear models), *HELP* (help information), *MANIPULATION* (manipulation), *MVA* (multivariate analysis including graphical models), *NONPARAMETRIC* (non-parametric methods), *REGRESSION* (regression analysis), *TIMESERIES* (time series analysis) and *UTILITY* (utilities). Other subjects planned for future modules include econometrics, survival analysis, teaching facilities, text handling and overhead-projector slide production.

Information about the library is accessible from within Genstat using the procedures in the help module which provide on-line help information. To get initial information the procedure `LIBHELP` (Payne, 1987) may be used; with option and parameter settings left to their defaults, a description of the `LIBHELP` procedure is given which includes information on how to find out further information about the library or about specific procedures. Other help procedures are `LIBINFORM` (Payne, 1987), which gives information about the modules and contents of the library, `LIBMANUAL` (Payne, 1988) which enables a manual for the procedure library to be printed, and `LIBEXAMPLE` (Payne, 1987) which allows examples for each procedure to be accessed as well as the full source code for any procedure. A general description and details of the options and parameters of these procedures is reproduced in Appendix 2.

Users are encouraged to submit procedures for inclusion in the library. Instructions for authors were published in Genstat Newsletter 20, [1], or can be obtained from the Secretary of the Editorial Committee (currently P G N Digby), who can also give information on procedures in progress in order to avoid duplication of effort. The Instructions for Authors contain rather precise advice on the style of code. These are not imposed rigorously, but the intention is that the code should be easily readable and thus maintainable, and that it should be laid out in a way that can be easily followed by someone other than the author; this may be a referee, but it may also be another user who has accessed the code via `LIBEXAMPLE`, as described above. The general rule throughout is to avoid the cryptic, either in names of identifiers within the procedure, or in abbreviations of statements. The name of the procedure, or of its options or parameters may be changed by the editor or referee so that consistency can be maintained across the library and with Genstat 5 itself (see Section 3 of the Instructions for Authors).

We also expect a submitted procedure to be accompanied by a reasonable amount of evidence that it works, preferably test programs that can be checked against published results.

Once a procedure has been submitted it will be looked at initially by one of the editors (listed in Appendix 1) and then refereed by an expert in that area. A report will then be sent to the editor, or the referee may contact the author directly to sort out any problems. We reserve the right to rewrite the help information submitted so that a reasonably consistent style is achieved and maintained for the library. If necessary a redraft of this information will be sent to the author to check that the sense has still been retained.

5. Conclusion

The procedure is becoming recognised as one of the major extensions added in Genstat 5. Procedures not only allow convenient reuse of a program by the original author, they also allow programs to be made available to other users, thereby providing a very powerful and convenient means of extending the facilities available in Genstat.

6. References

References accompanying procedures mentioned above are to the Genstat Procedure Library help information (see Section 4): 1987 refers to Library Release 1.2/1, 1988 to Release 1.3/1 and 1989 to Release 1.3/2. With Releases 1.3/1 and 1.3/2, procedure `LIBMANUAL` allows a manual to be produced that collates all the help information for these Library. Printed copies of the 1.3/2 Manual will be available from NAG later in 1989. Other reference:

[1] Payne, R.W. and Digby, P.G.N.
Genstat 5 Procedure Library Instructions for Authors.
Genstat Newsletter 20, pp. 31-40, 1987.

Appendix 1. The Genstat 5 Procedure Library Editorial Committee

As of 1 June 1989, the Committee is as follows:

R W Payne (Chairman),
P G N Digby (Secretary),
G M Arnold (Minutes Secretary),
B A Alm, J Bryan-Jones, M J Campbell, T J Cole, A W A Murray, D Ratcliff, J H Roger, C Triggs, K I Trinder, G Tunnicliffe Wilson and A J Weekes.

Appendix 2. Information About Help for Procedures in the Genstat Procedure Library

```
***** Procedure LIBEXAMPLE *****

LIBEXAMPLE accesses examples and source code of Genstat 5 Library procedures


*** Description ***

LIBEXAMPLE allows you to obtain an example of the use of any procedure in the
Genstat 5 Procedure Library, also to access the source code of any procedure,
so that you can see how it works, or modify it. The examples and source are
stored in a backing-store file which is attached to backing-store channel 4;
any file already attached to that channel is closed and a warning is printed.

The procedures for which you want examples or source code are specified using
the PROCEDURE parameter. Each procedure name should be given in a quoted
string, and either all in capital letters or all in lower-case letters. The
EXAMPLE parameter is used to specify the identifier of a text to store each
example. The example can then be run (as a macro) using the operator ##. The
SOURCE parameter specifies the identifier of a text to store the source code.
For example
  LIBEXAMPLE 'PERCENT'; EXAMPLE=%Ex
  ##%Ex
would put an example of how to use PERCENT into the text %Ex, and then run it.

No options.  Parameters: PROCEDURE, EXAMPLE, SOURCE.


*** Options ***

None.


*** 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
```

```
***** Procedure LIBHELP *****

LIBHELP provides help information about Genstat 5 Library procedures


*** Description ***

LIBHELP provides information about procedures in the Genstat 5 Procedure
Library. The information is stored in a backing-store file which is attached
to backing-store channel 4; any file that is already attached to that channel
is closed and a warning is printed.

LIBHELP has one parameter, called PROCEDURE, which you use to indicate the
procedures for which you want information; if PROCEDURE is not specified,
information is given about LIBHELP itself. The name of each procedure should
be given in a quoted string, and either all in capital letters or all in
lower-case letters: for example
  LIBHELP 'LIBINFORM'
will obtain information about the procedure LIBINFORM (you can use LIBINFORM
to find out what procedures and modules are in the Library).

LIBHELP has a single option, called PRINT, with which you specify a list of
strings to indicate what information you want about each procedure. The
possible values, with explanations in brackets, are as follows:
'index' (one-line description),  'description' (full description),
'options' (syntax of the options),  'parameters' (syntax of the parameters),
'method' (description of the method used),  'restrict' (action when arguments
 are restricted),  'calls' (list of procedures called by this procedure),
'similar' (procedures with similar facilities),  'authors' (list of authors),
'errors' (details of any reported errors).

Option: PRINT.  Parameter: PROCEDURE.


*** Options ***

PRINT     = strings Indicates what information is required about each
                    procedure (index, description, options, parameters,
                    method, restrict, calls, similar, authors, errors);
                    default description


*** Parameters ***

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
```

```
***** Procedure LIBINFORM *****

LIBINFORM prints information about the contents of the Procedure Library


*** Description ***

LIBINFORM provides information about the Genstat 5 Procedure Library.
The information is stored in a backing-store file which is attached to
backing-store channel 4; any file that is already attached to that channel
is closed and a warning is printed.

LIBINFORM has one parameter, called MODULE, which you can use to specify that
information is required only for a specified list of modules of the Library.
The name of each module should be given in a quoted string, and either all in
capital or all in lower-case letters. If MODULE is not set, the information is
given for the whole Library.

LIBINFORM has a single option called PRINT with which you specify a list of
strings to indicate what information you want about each module. The possible
values, with explanations in brackets, are as follows:
'contents' (list of procedures in the module or in the Library - see MODULE),
'index' (index lines for the procedures in the module/Library), 'errors' (list
of procedures in the module/Library for which errors have been reported),
'modules' (list of modules in the Library; only given if MODULE is not set).

Option: PRINT.  Parameter: MODULE.


*** Options ***

PRINT    = strings  Indicates what information is required about each module
                    (contents, index, modules, errors); default contents,errors


*** Parameters ***

MODULE   = texts    single-valued texts indicating the modules about which the
                    information is required; if this is not set, information is
                    given about the whole Library
```

```
***** Procedure LIBMANUAL *****

LIBMANUAL prints a 'Manual' containing information about Library procedures


*** Description ***

LIBMANUAL prints all the available information about the procedures in
the Genstat 5 Procedure Library. There is first a header page, with title
and list of index lines giving brief details about the procedures. Then
the full Help information is printed about each of the procedures in turn.
The Help information is stored in a backing-store file which is attached to
backing-store channel 4; any file that is already attached to that channel
is closed and a warning is printed.

By default the 'Manual' is printed to the current output channel. The CHANNEL
option allows you to print the Manual to some other channel; however, if you
do this, you should also set the OLDCHANNEL option, to allow LIBMANUAL to
reset the output channel afterwards.

Options: CHANNEL, OLDCHANNEL.  No parameters.


*** Options ***

CHANNEL    = scalar  Channel to which to print the manual;
                     default is to use the current output channel
OLDCHANNEL = scalar  Channel to return to after printing the manual
                     default is to leave this as set by CHANNEL


*** Parameters ***

None.
```

High-quality Output Following Analysis of a Two-period Cross-over Design

H-U P Hockey
Applied Statistics Research Unit
University of Kent
Canterbury
Kent
United Kingdom CT2 7NF

1. Introduction

The advent of laser printers means that the output from statistical packages can now be inserted directly into statistical reports, provided the format and layout meets required criteria. This is a highly desirable situation to achieve as it eliminates the need for error-prone and expensive transcription from computer output to report.

However, few statistical packages have been designed to give users as much control over the form of tabular and analytical output as they have over the content of that output. It is hoped to demonstrate below that there are sufficient text-processing and other capabilities in Genstat to produce, on high-quality printers, statistical and textual output for direct reproduction or publication.

2. The Procedure P2X2CROSSOVER

Consider Table 1. It shows how the analysis of a two-period, two-treatment cross-over trial might be presented and is, of itself, not particularly interesting. The point is that Table 1 was directly produced by a Genstat 5 program, using a procedure called P2X2CROSSOVER.

This procedure has been written for an ongoing series of two-period cross-over studies, with each study containing at least four, and often many more, variates. During the course of reporting such studies it is often the case that, following data review, the subjects or values in the study database change, and the data needs to be quickly re-analysed and presented. As well as increasing throughput and timeliness in response to changed data, procedures such as P2X2CROSSOVER in a site library can also be used to encourage a uniform style in an organisation's publications.

It is intended that P2X2CROSSOVER should not be used to present results until after sufficient data analysis. Usually, the display output only is written to a secondary output file while the primary output file contains standard Genstat output and the usual residual plots, together with graphs specifically useful for the analysis of two-period cross-over trials. This diagnostic output can be omitted once the emphasis has moved from analysis or re-analysis to display.

3. Statistical Notes on P2X2CROSSOVER

The data analysed in Table 1 is that of Hills and Armitage, [1], augmented by the following fictional observations (three incomplete and one totally missing sequence) in order to show the generality of P2X2CROSSOVER as regards unbalanced and missing data.

Subject	Sequence	Period	Treatment	Dry Nights
30	A	1	Drug	13
30	A	2	Placebo	*
31	B	1	Placebo	10
31	B	2	Drug	*
32	B	1	Placebo	*
32	B	2	Drug	11
33	A	1	Drug	*
33	A	2	Placebo	*

It is irrelevant to P2X2CROSSOVER whether all eight, only the first six, only the three non-missing, or none, of the above observations are included in the input data. In all four instances the same conclusions regarding *Treatment* and *Period* comparisons are made, as the additional data has no within-subject information. It has not been considered necessary or desirable to deal with missing values by deleting incomplete sequences, which are often different for each variate in a study. The principle of inclusion of all observed data is also important for its own sake. (The above discussion assumes the reasons for missing observations are not treatment-related.)

Inclusion of the three non-missing observations affects only the F-test for *Sequence*, and certain cells of the *Treatment* by *Sequence* cross-classification in Table 1, but not the two treatment differences in that table, whose average is the value of the overall treatment contrast. Hence the conclusion for the augmented data is exactly that of Hills and Armitage, apart from rounding error differences.

As the original Hills and Armitage data, which consisted of complete sequences only, is unbalanced, the analysis-of-variance table gives both unadjusted and adjusted sums of squares for *Treatment* and *Period*. Whenever the sequences that are complete are balanced, the analysis-of-variance table has the usual appearance.

P2X2CROSSOVER does not present observed treatment means. They are equivalent to the predicted treatment means only for completely balanced data with no missing values, and otherwise convey little information about the treatment comparison.

4. Using P2X2CROSSOVER

The Appendix shows how a Genstat program using P2X2CROSSOVER might look for a typical study with four variates in total (one of which is not to be presented). It does not show the code for P2X2CROSSOVER, 32 statements of which are simply PRINT statements using the JUSTIFICATION and FIELDWIDTH parameters. A suggestion to improve both the procedure code and the appearance of the calling program is a new PRINT parameter setting, JUSTIFICATION=centre, to centre output within the corresponding print field.

The program of the Appendix uses a related procedure, PCONTRASTSUMMARY, that produces summary output for many variates in the form of Table 2, which displays fictional results. An input parameter to PCONTRASTSUMMARY is a pointer whose values are one-line texts formed by P2X2CROSSOVER, or conceivably by any procedure or set of statements that produce simple contrasts. The texts in each pointer together form the body of tables such as Table 2.

```
              Two-Period Crossover Analysis of Variance of
               the Number of Dry Nights out of 14 Nights
                Experienced by Patients with Enuresis

                             Dry Nights


                        ANALYSIS OF VARIANCE

-------------------------------------------------------------------------------

                                        Sums of   Mean Sums
Source of variation            DF       Squares   of Squares       F          P

-------------------------------------------------------------------------------

Sequence                        1        37.045     37.045      1.811      0.188
Error (between subjects)       30       613.504     20.450

Treatment                       1        68.431     68.431     12.711      0.001
+Period                         1         8.709      8.709      1.618      0.214

Period                          1        18.776     18.776      3.488      0.073
+Treatment                      1        58.364     58.364     10.841      0.003
Error (within subjects)        27       145.360      5.384

-------------------------------------------------------------------------------


                         SUMMARY STATISTICS

   ------------------------------------------------------------------------

                                 Sequence
                         1                                  2

              New Drug then Placebo               Placebo then New Drug
          Period     Period    Difference     Period     Period    Difference
            1          2         (1-2)          1          2         (2-1)

   ------------------------------------------------------------------------

   N        18         17         17           13         13          12
   Mean    8.39       5.29       2.82         7.85       9.08        1.25
   S.D.    3.90       4.25       3.47         2.94       2.75        2.99
   S.E.                          0.841                               0.863

   ------------------------------------------------------------------------


                              CONTRAST

         ------------------------------------------------------------

                                                   95% Confidence
                             Difference   SED   Limits on Difference
                                                  Lower      Upper

         ------------------------------------------------------------

         New Drug - Placebo      2.04    0.619    0.77       3.31

         ------------------------------------------------------------
```

Table 1

```
                        Summary of Statistical Analyses


                          Difference                95% Confidence
                      (Capsule - Solution)      Limits on Difference      P-value
Parameter              Mean Diff.      SED         Lower       Upper
-----------------------------------------------------------------------------------

Area Under Curve         -0.003      0.0625      -0.150       0.145        0.966

Maximum Concentration     1.25       0.380        0.35        2.15         0.013

Elimination Rate          0.234      0.6419      -1.416       1.884        0.731
-----------------------------------------------------------------------------------
```

Table 2

The final part of the Appendix program shows how the PRINT directive, the EXTRA parameter of the VARIATE directive, and the ANOVA directive can be simply combined to produce output that is suitable for many display purposes. Table 3 shows the result of applying similar commands to the augmented Hills and Armitage data, and can be compared to Table 1, which is based on regression analysis.

For non-orthogonal data, ANOVA output with missing values restricted out is essential to show the split of information available from the two error strata. P2X2CROSSOVER assumes most *Treatment* and *Period* information is in the lower stratum.

5. Discussion

It is perhaps just coincidence that the design of output to the default limit of 80 characters per line, necessary for interactive use of Genstat 5, has meant displays such as Table 3 fit nicely onto A4 or quarto paper. Certainly, ANOVA output could be improved for display purposes in ways such as having an INDENTATION option for ANOVA output, and extending the general eight-character limit on factor identifiers and factor labels.

Non-orthogonality or missing values can cause the default ANOVA output to run on to a second page but often the deletion of superfluous blank lines can compress the output to one page. Alternatively, the PRINT option of ANOVA can be used to omit the printing of the information summary or estimated missing values, for display purposes.

While there exist statistical packages that allow considerable graphical manipulation, there are few that allow textual and numerical manipulation as easily as demonstrated above. This is possible in Genstat not only because of its text processing and printing features, but perhaps more importantly, because the results of powerful statistical operations are easily extracted and manipulated. In this respect especially, Genstat has the advantage over competing software. Who knows how much more useful Genstat might become if the specific problems of the production of tailor-made output were more directly addressed by its developers?

6. Reference

[1] Hills, M. and Armitage, P.
The Two-period Cross-over Clinical Trial.
British Journal of Clinical Pharmacology, 8, pp. 7-20, 1979.

```
***** Analysis of variance *****

Variate: Number of dry nights

Source of variation           d.f.       s.s.       m.s.    v.r.  F pr.

Subject stratum
Sequence                         1     37.045     37.045    1.80  0.190
Period                           1      8.767      8.767    0.43  0.519
Treatmnt                         1     32.164     32.164    1.57  0.221
Residual                        28    574.684     20.524

Subject.Period stratum
Period                           1     18.776     18.776    3.49  0.073
Treatmnt                         1     58.364     58.364   10.84  0.003
Residual                        27    145.360      5.384

Total                           60    873.049

***** Information summary *****

Model term                      e.f.  non-orthogonal terms

Subject stratum
  Period                       0.049
  Treatmnt                     0.040  Sequence  Period

Subject.Period stratum
  Period                       0.951  Subject
  Treatmnt                     0.923  Subject  Period

***** Tables of means *****

Variate: Number of dry nights

Grand mean  7.56

 Sequence Drg->Pbo Pbo->Drg
              6.89     8.46
     rep.       35       26

   Period        1        2
              8.12     6.98
     rep.       31       30

 Treatmnt     Drug  Placebo
              8.56     6.52
     rep.       31       30

*** Standard errors of differences of means ***

Table              Sequence      Period    Treatmnt
rep.                unequal     unequal     unequal
s.e.d.                1.173       0.609       0.619

***** Stratum standard errors and coefficients of variation *****

Variate: Number of dry nights

Stratum                    d.f.          s.e.        cv%

Subject                      28             *          *
Subject.Period               27         2.320       30.7
```

Table 3

Appendix

```
JOB 'Presentation of three 2x2 cross-over analyses, for fictional data'

UNITS [NVALUES=18]
FACTOR [LEVELS=9; VALUES=2(1...9)] Subject
FACTOR [LEVELS=2; VALUES=(1,2)9] Period
FACTOR [LABELS=!T('Cap->Sol','Sol->Cap'); VALUES=2(1,2,1,2,1,2,2,1,1)] Sequence
FACTOR [LABELS=!T(Capsule,Solution)] Treatmnt
READ [CHANNEL=2] Treatmnt,auc,cmax,tmax,kel

"Set up texts to denote the variates. (Longname, Shortname and Unitsname will
 be left-justified as below, but including left-hand margin spaces. Some trial
 and error may be necessary for aesthetic centring in the display output.)"

TEXT Longname[1...4]; VALUES= \
   '         Area Under the Curve of Plasma Concentration of Tetrasubacate', \
   '             Maximum Observed Plasma Concentration of Tetrasubacate', \
   '         Time to Maximum Observed Plasma Concentration of Tetrasubacate', \
   '                  Elimination Rate Constant for Tetrasubacate'

& Shortname[1...4]; VALUES= \
   '                                       AUC', \
   '                                      Cmax', \
   '                                      Tmax', \
   '                                       Kel'

& Unitsname[1...3]; VALUES= \
   '                                    (mg.h/l)', \
   '                                     (mg/l)', \
   '                                    (minutes)'

& Parametername[1...4]; VALUES='Area Under Curve','Maximum Concentration', \
   'Time to Maximum Conc.','Elimination Rate'

INPUT [PRINT=*] 3                                       "input the procedures"
SCALAR DisplayChannel; VALUE=2                        "display output channel"
   & LHM; VALUE=5                                     "left-hand margin spaces"

" ********************** Display table production starts ********************* "

P2X2CROSSOVER [CHANNEL=DisplayChannel;          \
           INDENTATION=LHM;                      \
                HEADER='STUDY ASRU-PK443-177';    \
         SUBJECTFACTOR=Subject;                    \
          PERIODFACTOR=Period;                      \
       TREATMENTFACTOR=Treatmnt;                     \
                LABELS=!T(Capsule,Solution);          \
        NOFULLANALYSIS=yes]       DATA=auc,cmax,kel;   \
                           TABLENUMBER=1,2,3;           \
                            PAGENUMBER=10,11,12;         \
                              LONGNAME=Longname[1,2,4];   \
                             SHORTNAME=Shortname[1,2,4];   \
                             UNITSNAME=Unitsname[1,2],*;    \
                         PARAMETERNAME=Parametername[1,2,4]; \
                            SSDECIMALS=6,4,4;                 \
                          MEANDECIMALS=3,2,2;                  \
                    DIFFERENCEDECIMALS=3,2,3;                   \
                           SUMMARYTEXT=SummaryLines[1...3]
```

```
PCONTRASTSUMMARY [CHANNEL=DisplayChannel;                                     \
                  INDENTATION=LHM;                                              \
                       HEADER=*;                                                 \
                       LABELS=!T(Capsule,Solution)] SUMMARYTEXTS=SummaryLines;   \
                                                     TABLENUMBER=2;               \
                                                      PAGENUMBER=5

" ******************** Display table production finished ******************** "


"The following statements can be used to provide additional ANOVA output."

VARIATE AUC; EXTRA= \
   ' (Area Under the Curve of Plasma Concentration of Tetrasubacate, mg.h/l)'
& Cmax; EXTRA= \
   ' (Maximum Observed Plasma Concentration of Tetrasubacate, mg/l)'
& Kel; EXTRA= ' (Elimination Rate Constant)'
CALCULATE AUC,Cmax,Kel = auc,cmax,kel

BLOCKS Subject/Period
TREATMENTS Sequence + Period + Treatmnt

FOR Y=AUC,Cmax,Kel; TableNumber=1...3

  PAGE [CHANNEL=DisplayChannel]
  PRINT [CHANNEL=DisplayChannel; IPRINT=*; INDENTATION=LHM; SQUASH=yes] \
     'APPENDIX'; FIELDWIDTH=70
  PRINT [CHANNEL=DisplayChannel; IPRINT=*; INDENTATION=LHM] 'TABLE', \
     TableNumber; FIELDWIDTH=38,2; DECIMALS=0; JUSTIFICATION=right,left

  RESTRICT Y; CONDITION=Y.NE.(0/0)
  OUTPUT [PRINT=*] DisplayChannel
  ANOVA [PRINT=aovtable,information,means,%cv; FPROBABILITY=yes] Y
  OUTPUT 1
  RESTRICT Y

ENDFOR
STOP
```

A Data Organisation System for Field Experiments

A E Ainsley
A D Todd
AFRC Institute of Arable Crops Research
Rothamsted Experimental Station
Harpenden
Hertsfordshire AL5 2JQ

1. Introduction

Experiments in which many quantities may be measured on several occasions present organisational difficulties. Database systems can deal with the organisation of data, but further analysis often requires the use of statistical packages. In this article we illustrate how backing-store and procedure libraries can be used to make Genstat look like a database program, whilst retaining the more advanced facilities for data management and analysis that are required. The actual method is not described in great detail since the main point is to encourage other users to customise the system to suit their own local situations.

The impetus for this work was provided when we were approached by an experimenter who was going to do a series of field experiments in which he would need to record many variates at regular intervals throughout the growing season. On each occasion several samples would be taken from some, or all, of the plots, the plants would be dissected, and the relevant variates recorded for each one. Thus the data from a single date of each experiment would consist of measurements of many variates on several plants from each of some, or all, of the plots. Sampling had to be such that both the number of plots and the number of plants per plot would vary between occasions. There was clearly a need to organise the data on the computer so that it could easily be accessed for tabulation, graphing, analysis, and so on. The system had to be complete enough so that we could access it from time to time without having to remember too many details, but simple enough for a new user of Genstat.

Our solution to the problem has been to provide two sets of programs, the first of which is used by the statistician/data processor to set up the data-organisation system. The second set of programs is used to access the data and is provided in a procedure library. Thus it is both easy to use, and easy to extend as the need arises. The system is flexible enough to allow the user to:

(a) examine individual variates recorded on a particular date,

(b) examine several variates from a particular date,

(c) examine the same variate over several dates.

In Section 2 we describe briefly how the system is set up, in Section 3 we describe two of the procedures, in Section 4 we describe the organisation of files on a MicroVAX II VMS system, and in Section 5 we illustrate use of the system on an experiment on eyespot of wheat conducted at Rothamsted in 1988. Although originally developed for use with agricultural data a similar approach could be used for many other types of data.

We shall not discuss statistical aspects of the analysis of such complex data here, although there are obviously many statistical issues that arise from the data.

2. Setting up the System

The method of setting up the system is similar to those of Todd [2] and Fenlon [1] and is illustrated in Figure 1. Initially the experimenter provides a field plan of the experiment, containing information on the design and layout, and lists of variates to be recorded and to be stored. The recorded and stored variates are not necessarily the same, as some of the stored variates may be derived from the recorded ones. The lists of variates do not have to be complete at this stage, as it is a simple matter to update the system to include new variates. This can be important if, for example, an unforseen disease attacks the experiment part way through the season. However, it is useful in the initial stages to know what variates to expect.

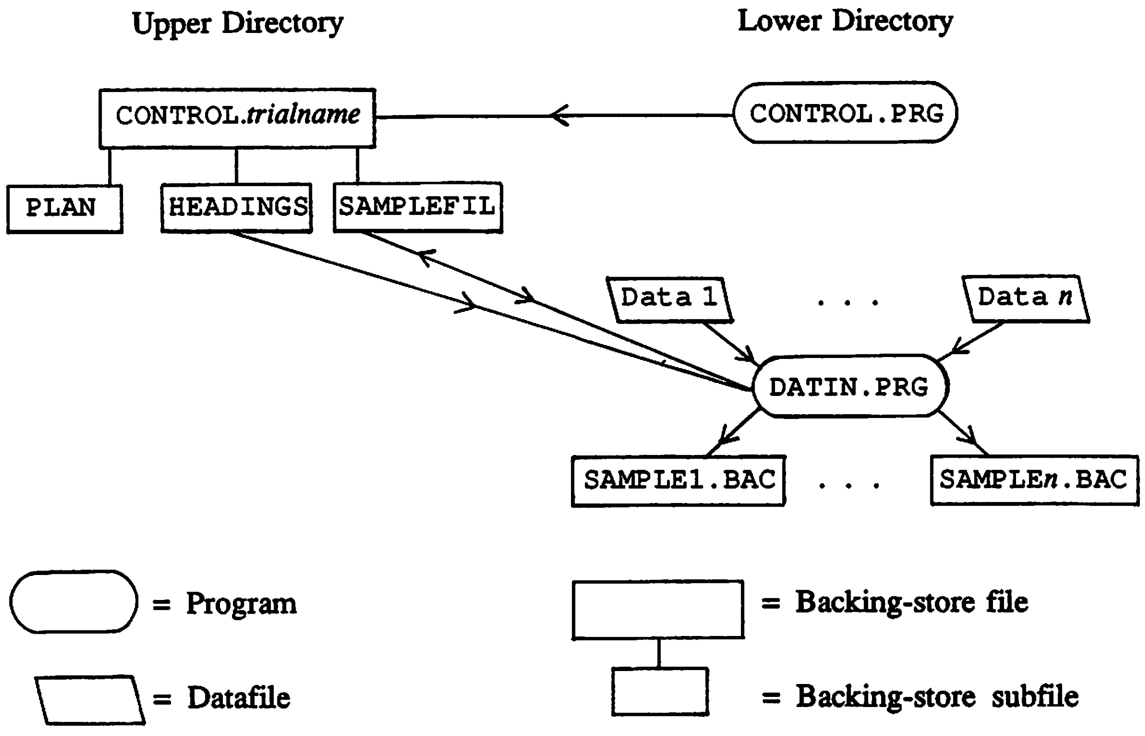

Figure 1

Firstly a trialname code which will be used to identify the experiment is assigned. The plan, in the form of levels of block and treatment factors for each plot, and the lists of recorded and stored variates are then input to the program stored in the file called `CONTROL.PRG`. This program sets up a backing-store file, `CONTROL`.*trialname*, containing three subfiles, `HEADINGS`, `SAMPLFIL` and `PLAN`.

Subfile `HEADINGS` contains the following structures:

`PFULL`	: a pointer containing the identifiers of all variates to be recorded in the experiment,
`PMAP`	: a pointer containing the identifiers of all variates to be stored for the experiment, in the order in which they are to be stored,
`HEAD`	: a pointer containing the headings for the variates to be stored,
`SUFFIX`	: a variate containing the suffixes of `HEAD`.

Subfile `SAMPLFIL` initialises the following structures:

`FILE`	: a pointer that will contain the names of the individual backing-store file for each sample date,
`LABEL`	: a pointer that will contain the heading for each sample date,
`STIND`	: a pointer that will contain a variate for each sample date, indicating which variates are stored at that date,
`SAMSUFF`	: a variate that will contain the suffixes for `HEAD`, `LABEL` and `STIND`.

Subfile `PLAN` contains factors describing the block and treatment structure of the experiment.

The program also produces a reference file, `HEADINGS`.*trialname*, giving the names and codes of all recorded variates, and the names, codes and reference numbers of the stored variates for this experiment.

As the data from each sample is received, it is input to the program stored in the file DATIN.PRG. This program does some limited checking of the data, such as that the input variates belong to the set of recorded variates, and that values of scores are not out of range, derives the variates to be stored and writes them to a backing-store file for that sample, SAMPLE*n*.BAC. SAMPLE*n*.BAC then contains the derived variates that are present in the sample, along with factors that index each plant by plot. The program also updates the subfile SAMPLFIL in CONTROL.*trialname*.

An outline of the programs in CONTROL.PRG and DATIN.PRG is given in the appendix.

3. Accessing the System

At present the library contains procedures to retrieve data from a given sample, to retrieve plan factors, to provide a list of stored variate headings and sample dates, to give tables of which variates are saved for which dates, and to perform simple data reduction, such as forming plot means. This could easily be extended to provide other facilities. The procedures are not intended to be completely fool-proof. Speed of execution can be considerably improved by incorporating only limited checking. Since the users of the system are on the same site as us, and there is no danger of the stored data being contaminated by incorrect use of any of the procedures, we have not included rigorous checking on arguments.

The system is designed so that the user need not know where or how the data is stored. In order to find out what information is currently available for a given experiment, procedure INFO is used. All data is retrieved through procedure SETUP. INFO and SETUP are described in the following sections. Examples of the use of these and other procedures are included in Section 5.

3.1. Procedure INFO

The purpose of this procedure is to provide information on the stored information for a given experiment. By default a table showing which variates are stored for each sample that is currently available is produced. A table containing information on only a subset of the samples can be produced by using the SAMPLENOS parameter. The textual descriptions of the stored variates and available sample dates can be printed using the VARIATEHEADING and SAMPLEHEADING options respectively.

Options

VARIATEHEADING = *string*	If Y, variate headings for all stored variates are printed; default N
SAMPLEHEADING = *string*	If Y, sample headings are printed; default N
VARXSAM = *string*	If Y, table of stored variates by sample is printed; default Y

Parameter

SAMPLENOS = *variate*	Samples to be included in the table; if omitted, all are included

3.2. Procedure SETUP

In its simplest use the user supplies the number of the sample he or she wishes to work on, the variates required and the name of a pointer in which to place the retrieved data; for example:

```
    SETUP [SAMPLENO=1; VARIATENO=!(1,5)] D
```

This will retrieve stored variates 1 and 5 and place them in D[1,5]. A warning is displayed if the user requests a variate that is not stored and a failure occurs if the sample number is out of range. SETUP can also be used to retrieve factors giving the plot and plant that each observation comes from and the text description of each variate. There is an option for linking the headings to the variates and also a failure indicator.

Options

SAMPLENO = *scalar*	Sample required; default 0, i.e. no action
VARIATENO = *variate*	Variate containing the identification numbers of the variates to be retrieved; default *
EXTRA = *string*	Whether the extra attribute should be set for the variates retrieved; default Y
FAIL = *scalar*	Failure indication; default 0, i.e. correct

Parameters

POINTERNAME = *pointer*	To contain retrieved variates; any previous values of the pointer are lost.
PLOTIDENTIFICATION = *factor*	To identify field-plot that each measurement comes from; * means not required
PLANTIDENTIFICATION = *factor*	To identify plant that each measurement comes from; * means not required
HEADINGNAME = *pointer*	To contain text heading for each retrieved variate

4. File Organisation

Using a MicroVAX II VMS system, we arranged the data organisation system so that it is set up in one username. By providing read and execute access, the system can be accessed from other usernames, possibly on other machines in the same VAX cluster. The user needs in his or her username-initialisation file (LOGIN.COM) a symbol which runs a command file to set up a DCL logical which points to the appropriate library. The symbol, called EXP, has one parameter which indicates the experiment name and sets another logical, CONTROL, to point to the backing-store file controlling that experiment. In practice, since we do not need the procedures in the Genstat Procedure Library, we attach our library by redefining the logical GNPRCL. This saves time if directive or procedure names are incorrectly typed. In other applications, if the Genstat Procedure Library is required, the logical for the site library could be overwritten instead.

Since a single user may have several experiments running at any one time, we set up the directory structure so that at one level we have the CONTROL.*trialname* backing-store files for all the experiments. In a separate subdirectory for each experiment, we put the programs CONTROL.PRG and DATIN.PRG along with the backing-store library can either be placed in the upper level directory or in a completely different directory.

5. Example

An experiment on which this system has been used was one designed to examine factors affecting the development of the cereal stem-based disease, eyespot. The experiment was two blocks of a 2×2×3 factorial with the factors being crop (wheat, barley), seed rate (low, high) and inoculum (control, W, R). Over the season samples were taken from the experiment on 20 different dates. Up to 17 variates were recorded on the seedling phase of the experiment (samples 1-6) and 51 on the adult phase (samples 7-20). These were reduced to 17 and 30 variates to be stored respectively. Since the variates in the two phases were quite different the experiment was split into two separate data-access systems. During the seedling phase up to approximately 500 plants were examined on each date, with sampling being destructive. The following example shows a simple use of the system.

```
$ EXP 88RM6S
$ GENSTAT

 Genstat 5  Release 1.3  (Vax/VMS4)
 Copyright 1988, Lawes Agricultural Trust (Rothamsted Experimental Station)

> "Find out definitions of the stored variates and which variates are
>  stored on which dates"
>
> INFO [VARIATEHEADING=Y]

   Definitions of Stored Variate

 HEAD[1]        A) TOTAL NO LEAVES - MAIN SHOOT
    .                       .
    .                       .
 HEAD[10]       J) TOTAL NO. OF MAIN TILLERS
 HEAD[12]       L) NO. OF MAIN TILLERS WITH EYESPOT
    .                       .
    .                       .
 HEAD[24]       X) GROWTH STAGE

   STORED VARIATES BY SAMPLE, 1=STORED, 0=UNSTORED

   VARIATE  SUFFIX        SAMPLE

                         1      2      3      4      5


         A      1        1      1      1      1      0
                            .            .
                            .            .
         J     10        1      1      1      1      1
         L     12        1      1      1      1      1
                            .            .
                            .            .
         U     21        0      0      0      0      1
                            .            .


 > "Retrieve variates of interest for sample 3"
 >
 > SETUP [SAMPLENO=3; VARIATENO=!(10,12,21)] POINTERNAME=D3; PLOTID=PL3; \
 >    HEADINGNAME=H3

   88RM6 SAMPLE3 5/4/88 R81165.DAT

   requested variate not available          21

 > "Form means over the plants in each plot"
 >
 > PLOTMN PLANTVARIATE=D3[]; PLOTVARIATE=DM3[10,12]; PLOTID=PL3; \
 >    HEADINGNAME=H3[10,12]
 > "... for information"
 > DUMP D3[],PL3,DM3[]

  ***** DUMP *****

  Identifier       Type  Length   Values Missing  Ref.No.
      D3[10]    Variate     411  Present       0    -1205
      D3[12]    Variate     411  Present       0    -1203
         PL3     Factor     411  Present       0    -1446
     DM3[10]    Variate      24  Present       0    -1502
     DM3[12]    Variate      24  Present       0    -1503
```

```
> "Retrieve plot and treatment factors (stored on plot basis)"
>
> GETFACT CROP, SEEDRATE, INOCULUM, PLT
>
> PRINT PLT,DM3[]

            PLT      DM3[10]      DM3[12]
              1        4.944       0.6111
              2        5.429       0.7143
              3        2.875       0.8750
              4        3.235       0.9412
              5        5.167       0.5556
              6        5.588       0.0588
              7        3.118       0.8235
              8        4.680       0.1600
              9        5.312       0.6250
             10        4.667       0.5333
             11        3.833       0.5000
             12        3.200       0.6000
             13        4.667       0.2778
             14        5.500       0.2222
             15        3.385       0.7692
             16        2.842       0.5789
             17        3.000       0.5000
             18        5.375       0.3125
             19        2.929       1.0714
             20        4.133       0.3333
             21        6.000       0.5294
             22        5.812       0.3125
             23        3.286       0.7143
             24        5.526       0.0000

> FOR
> "Form treatment means for the retrieved variates"
>    PRINT H3[]
>    TABULATE [PRINT=means; CLASSIFICATION=CROP,SEEDRATE,INOCULUM] DM3[]
> ENDFOR

                H3[10]                           H3[12]
  J) TOTAL NO. OF MAIN TILLERS  L) NO. OF MAIN TILLERS WITH EYESPOT

              INOCULUM     Control                  R                     W
                              Mean       Mean       Mean       Mean       Mean       Mean
       CROP   SEEDRATE
     Barley        Low       5.557     0.0294      5.472     0.5703      5.344     0.4688
                  High       5.246     0.2363      5.048     0.4960      5.333     0.3889
      Wheat        Low       4.400     0.4333      3.082     1.0063      3.130     0.8221
                  High       3.417     0.5000      3.021     0.5895      3.202     0.7689

> STOP

 ******** End of job.  Maximum of 25900 data units used at line 12 (16860 left)
```

6. Conclusion

We have used this system to set up two-years-worth of data for several experiments. After the initial setting up it has proved a very useful and time-saving approach. The user benefits from being able to access his data easily and quickly and not having to rely on someone else to produce simple graphs and tabulations. The statistician benefits from having much of the routine work removed, as well as being able to use the system for any more complex analyses that may be required. Although not applicable here, such a system could be very useful in monitoring progress throughout the season and in planning what to record at future dates.

7. References

[1] Fenlon J.
The Use of Genstat as a Data Organiser for Long-season Data.
Genstat Newsletter 20, pp. 13-19, 1987.

[2] Todd A.D.
The Saving of Punching and Programming Time in the Analysis of Experiments.
Genstat Newsletter 6, PP. 14-17, 1980.

Appendix

Outline of control program in file CONTROL.PRG

```
"  Program to set up headings, datafile names and plan in
   backing-store file CONTROL.88RM6A and print reference information
   in output file HEADINGS.88RM6A"

"  Define codes for recorded variates"
POINTER [VALUES=A,B1,B2,C1,C2,D1,D2,E,F,AGEM,G,H,I,M,Y,XA,AA,BB,CC,GG] PFULL

"  Define codes for stored variates"
& [VALUES=A,B1,B2,C1,C2,D1,D2,E,F,ST] PMAP

"  Define suffixes for stored variates"
VARIATE [VALUES=1...9,12] SUFFIX

"  Set up headings for stored variates"
TEXT [NVALUES=1] HEAD[#SUFFIX]; VALUES=' A) NO. (PRIMARY) SHOOTS PER PLANT', \
   ' B1) NO SHOOTS WITH EYESPOT ON LEAF SHEATHS', etc.

"  Set up headings for recorded variates that are not stored "
TEXT [NVALUES=1] OHEAD[1...12]; VALUES=' AGEM) AGE MAIN SHOOT', etc.

"  Open backing-store file and file for reference printout"
OPEN '[DPROTH.XXX]CONTROL.trialname'; CHANNEL=1; FILETYPE=backing
& '[DPROTH.XXX.trialname]HEADINGS.trialname'; CHANNEL=2; FILETYPE=output

"  Store variate headings in subfile HEADINGS"
STORE [CHANNEL=1; SUBFILE=HEADINGS; METHOD=overwrite] PFULL,PMAP,SUFFIX,HEAD

"  Print headings to reference file"
PRINT [CHANNEL=2] !T('                  Experiment XXX', \
   '                 ****************')
& !T('   Definition of recorded variate names',\
   '   ****************************************')
& [SQUASH=yes; ORIENT=across; IPRINT=*] HEAD[1...9],OHEAD[]; JUSTIFICATION=left
PAGE [CHANNEL=2]
PRINT [CHANNEL=2] !T('   Definition of stored variate names and numbers',\
   '   ************************************************')
& [SQUASH=yes; ORIENT=across] HEAD[]; JUSTIFICATION=left

"  Set up backing-store subfile to contain sample information"
POINTER FILE,LABEL,STIND
VARIATE SAMSUFF
STORE [CHANNEL=1; SUBFILE=SAMPLFILE; METHOD=overwrite] FILE,LABEL,STIND,SAMSUFF

"  Set up block and treatment factors at plot level"

"  Store plan factors in subfile PLAN"
STORE [CHANNEL=1; SUBFILE=PLAN; METHOD=overwrite] BLK,PLT,TREAT
CLOSE 1; FILETYPE=backing
STOP
```

Outline of data-input program in file `DATIN.PRG`

Input is a datafile attached to input channel 2 containing:

`HEADNAME`	– a heading for this sample
`FHEAD`	– the name of the control file for this experiment
`FNAME`	– the name of the backing-store file to store this sample in
`TREAD`	– list of variate codes recorded for this sample in order to be read
Data	

```
"  Program to create backing-store file containing derived variates
   for a single sample"

"  Define variate to indicate what variates are stored"
VARIATE [VALUES=100(0)] INDICATR
POINTER PDATA

"  Initialize texts and scalars"

"  Read the file specification info"

"  Retrieve pointers of variates and suffixes
   PFULL contains codes of all variates ever recorded for this experiment,
   PMAP contains codes of all variates stored in this experiment
   SUFFIX contains suffixes of PMAP"
OPEN FHEAD; CHANNEL=1; FILETYPE=backing
RETRIEVE [CHANNEL=1; SUBFILE=HEADINGS] PFULL,PMAP,SUFFIX
CLOSE 1; FILETYPE=backing

"  Read the rest of the specification info"
READ [CHANNEL=2; END=*] SAMPNO
READ [CHANNEL=2; SETNVALUES=yes] TREAD
POINTER [VALUES=##TREAD] PREAD

"  Read data"

"  Check data values in range expected etc"

"  Derive variates to be stored"

"  Create factors PLOT and SUBPLOT"

"  Fill pointer PDATA with the derived variates, in the order of PMAP,
   and give a message if a variate in PMAP is missing"
CALCULATE NVPMAP = NVALUES(PMAP)
FOR DUM=#PMAP; DUM2=1...NVPMAP
  CALCULATE SUFD = SUFFIX$[DUM2]
  GETATTRIBUTE [ATTRIBUTE=nvalues] DUM; SAVE=PCHK
  IF PCHK['nvalues'].EQ.MV
    PRINT [IPRINT=*] 'DERIVED VARIATE', SUFD, ' WILL NOT BE STORED'; \
       DECIMALS=0; FIELDWIDTH=4
  ELSE
    CALCULATE PDATA[SUFD] = DUM
    & INDICATR$[SUFD] = 1
  ENDIF
ENDFOR

"  Print a summary of each derived variate"
CALCULATE [PRINT=summary] PDATA[] = PDATA[]

"  If everything OK store the data"
OPEN FNAME; CHANNEL=1; FILETYPE=backing
STORE [CHANNEL=1; SUBFILE=SAMPLE[SAMPNO]; METHOD=overwrite] \
   HEADNAME,PLOT,SUBPLOT,PDATA
CLOSE 1; FILETYPE=backing
```

```
"  Define individual sample backing-store filenames"
OPEN FHEAD; CHANNEL=2; FILETYPE=backing
RETRIEVE [CHANNEL=2; SUBFILE=SAMPLFILE; MERGE=yes] FILE,LABEL,STIND,SAMSUFF
TEXT FILE[SAMPNO]; VALUES=FNAME
& LABEL[SAMPNO]; VALUES=HEADNAME
VARIATE STIND[SAMPNO]; VALUES=INDICATR
IF NVALUES(SAMSUFF).EQ.MV
"  First sample for this expt"
  CALCULATE SAMSUFF = SAMPNO
ELSE
  IF SAMPNO.NI.SAMSUFF
    VARIATE [VALUES=#SAMSUFF,SAMPNO] SAMSUFF
    SORT SAMSUFF
  ENDIF
ENDIF
STORE [CHANNEL=2; SUBFILE=SAMPLFILE; METHOD=overwrite] FILE,LABEL,STIND,SAMSUFF
CLOSE 2; FILETYPE=backing

FOR
  PRINT !T('   Backing-store filenames for each sample',\
    '    ****************************************')
  PRINT [ORIENT=across] FILE[],HEADNAME[]; JUSTIFICATION=left
ENDFOR
STOP
```

Fitting a Simple Time-series Model with Genstat

I White
Forestry Commission
Bush Estate
Roslin

Time-series models of the kind familiar to econometricians have recently begun to find favour with a wider audience. For example, Rennols, Carnell and Tee [2] use a time-series model to examine the relationship between rainfall and depth of water table. At time t the rainfall is X_t and the 'true' water table depth (measured down from the ground surface) is Z_t. This is measured as the borehole water depth.

$$Y_t = Z_t + e_t \tag{1}$$

where e_t ($t = 1,2,\ldots$) is a series of constant-variance uncorrelated errors with variance σ^2. The value Z_t depends on previous values and rainfall through the equation

$$Z_t = \lambda Z_{t-1} + H(1-\lambda) - \alpha X_t \tag{2}$$

Repeated substitution for $Z_{t-1}, Z_{t-2}, \ldots,$ leads to the alternative expression

$$Z_t = H + \beta L_t - \alpha M_t \tag{3}$$

where $\beta = Z_0 - H$, $L_t = \lambda^t$, and $M_t = \sum_i \lambda^i X_{t-i}$. The observed series Y_t satisfies the equation

$$Y_t = \lambda Y_{t-1} - \alpha X_t + H(1-\lambda) - V_t \tag{4}$$

where $V_t = e_t - \lambda e_{t-1}$ is a moving-average error process.

Rennols *et al.* point out that ignoring the nature of the error term in (4), and regressing the vector $(y_1, y_2, \ldots, y_T)'$ on regressors $(y_0, y_1, \ldots, y_{T-1})'$ and $(x_1, x_2, \ldots, x_T)'$, leads to inconsistent estimation of the parameter λ. Unaware of this, inexperienced users of statistical packages often use the method, which requires only standard linear regression facilities.

The correct procedure is maximum likelihood applied to equations (1) and (3). In Genstat 5 this is achieved by the `FITNONLINEAR` directive. There are three parameters of interest, λ, H, and α, and one nuisance parameter β which represents the initial state of the process. One parameter, λ, enters in an essentially nonlinear way, but with λ fixed the model is linear in the remaining parameters. Example 1 shows the estimation of four parameters from a pair of series of length 100, artificially generated with $\lambda = 0.9$, $\alpha = 20$, $H = 500$, and $\sigma^2 = 25$. The parameter H is estimated as the constant, and α (with its sign reversed) as the coefficient of M_t. Note the device of writing

$$M_t = \lambda^t \sum_i (X_{t-i} / \lambda^{t-i})$$

An alternative procedure is provided by the time-series facilities for fitting transfer-function models of the form

$$Z_t - \delta_1 Z_{t-1} - \ldots - \delta_p Z_{t-p} = w_0(X_t - c) - w_1(X_{t-1} - c) - \ldots - w_q(X_{t-q} - c)$$

[1, p. 345]. From equations (1) and (2), the output series Y_t is the sum of an error process (here just 'white noise') and a transformation of the X_t series, defined recursively by equation (2), which can be rewritten as

$$Z_t - \lambda Z_{t-1} = -\alpha(X_t - c)$$

where $c = H(1-\lambda)/\alpha$. This is a transfer-function model with $p = 1$, $q = 0$, $\delta_1 = \lambda$, $w_0 = -\alpha$ and a constant term c, of interest here as the daily rainfall amount necessary to maintain complete soil saturation. In Example 2, the model is fitted to the same series as above. The `FIX` option of `ESTIMATE` suppresses the constant usually associated with the output error process e_t, in order to avoid aliasing with the constant c of the transfer-function component, and the option setting `PRIORMETHOD=estimate` ensures that the initial value Z_0 is estimated as an extra parameter (Genstat 5 Reference Manual, 11.5).

References

[1] Box, G.E.P and Jenkins, G.M.
Time Series Analysis, Forecasting and Control.
Holden-Day, Oakland, California, (revised edition) 1976.

[2] Rennols, K., Camell, R. and Tee, V.
A Descriptive Model of the Relationship Between Rainfall and Soil Water Table.
Journal of Hydrology, 47, pp. 103-114, 1980.

Example 1

```
    1  UNITS [NVALUES=100]
    2  READ Rain,Depth

   Identifier   Minimum      Mean   Maximum    Values   Missing
         Rain    0.0000    0.4100    4.0000       100         0     Skew
        Depth     347.2     418.7     474.5       100         0
  104  SCALAR Lamda
  105  VARIATE [VALUES=1...100] T
  106  MODEL Depth
  107  RCYCLE Lamda; INITIAL=0.5; LOWER=0.0; UPPER=1.0; STEPLENGTH=0.01
  108  EXPRESSION [VALUE=L=EXP(T*LOG(Lamda))] E[1]
  109  & [VALUE=M=L*CUMULATE(Rain/L)] E[2]
  110  FITNONLINEAR [CALCULATION=E; SELINEAR=yes] L,M


 110..................................................................


 ***** Nonlinear regression analysis *****

  Response variate: Depth

 *** Summary of analysis ***

                d.f.         s.s.         m.s.
 Regression        3       72582.     24193.97
 Residual         96        2158.        22.48
 Total            99       74740.       754.95

 Percentage variance accounted for 97.0


 *** Estimates of parameters ***

                    estimate         s.e.
 Lamda               0.90158      0.00368
 * Linear
 L                    -97.35         3.47
 M                   -19.631        0.389
 Constant             499.15         2.71
```

Example 2

```
 111  TSM [MODEL=transfer] Tf; ORDERS=!(0,1,0,0); PARAMETERS=!(1,2.5,0.9,-20)
 112  & [MODEL=arima] Error; ORDERS=!(0,0,0)
 113  TRANSFERFUNCTION Rain; TRANSFERFUNCTION=Tf; PRIORMETHOD=estimate
 114  ESTIMATE [FIX=!(0,1,2,3,0,0)] Depth; TSM=Error


114.............................................................................


***** Time-series analysis *****

  Output series: Depth
    Noise model: Error
                        autoregressive    differencing  moving-average
          Non-seasonal               0               0               0
   Input series:
Rain          ; transfer function: Tf
          Non-seasonal               1               0               0

              d.f.     deviance
  Residual      96        2158.


*** Transfer-function model 1 ***

Delay time 0

                     ref.     estimate          s.e.
Transformation          0      1.00000         FIXED
Constant                1       2.5027        0.0707

* Non-seasonal; no differencing

                 lag ref.     estimate          s.e.
Autoregressive     1    2      0.90157       0.00368
Moving-average     0    3      -19.632         0.389

*** Autoregressive moving-average model ***

Innovation variance 22.48

                     ref.     estimate          s.e.
Transformation          0      1.00000         FIXED
Constant                0           0.         FIXED

* Non-seasonal; no differencing
```

Features of the Genstat 5 Language: 2

S A Harding
Statistics Department
AFRC Institute of Arable Crops Research
Rothamsted Experimental Station
Harpenden
Hertfordshire
United Kingdom AL5 2JQ

K I Trinder
NAG Ltd
Wilkinson House
Jordan Hill Road
Oxford
United Kingdom OX2 8DR

1. Introduction

The first article in this series generated a few favourable comments, but no contributions or suggestions for aspects of Genstat 5 that need a little more explanation than the Reference Manual offers. Do let us know if you have any ideas; otherwise the series might be rather short-lived. This article includes some notes on substitution (using ## and #) and a section containing several small items that might nonetheless be useful to some readers.

2. A Warning About Macro Substitution

There is a brief explanation and some examples of macro substitution (using ##) in the Genstat manual (Section 1.7.2, p27). It is not always clear when you are able to use this feature of the language and some users have run into problems when attempting to use ## in more complex programs, particularly when FOR loops or procedures are involved. Reading Section 1.7.2 carefully you will see that '... substitution takes place *immediately* after Genstat reads the statement ...'. Implicit in this description is the action that occurs when ## is found within a FOR loop or procedure: substitution occurs *before* the statement is stored. Thus the following example will not work as hoped:

```
FOR Var=V,W; Opt='LIMITS=Limvar','NGROUPS=5'
  ...
  ...
  HISTOGRAM [##Opt] Var
  ...
  ...
ENDFOR
```

The intention was to have different options for the HISTOGRAM statement on different passes through the loop. Firstly, this can never work as intended as the substitution will occur only once. In fact, when the HISTOGRAM statement is read Opt is still unassigned, so ##Opt produces a null string which results in HISTOGRAM [] Var being stored. When the loop is executed a default HISTOGRAM will be produced each time. The output may be even more confusing in some circumstances: if the FOR statement had the option setting COMPILE=each then Opt would be assigned to the first unnamed text, 'LIMITS=Limvar', and this would be stored as part of the statement so that a limits variate would be used for each histogram. Worse still, if Opt had been used as a dummy in a previous FOR loop, the example above would substitute its previous contents.

The message then is to take care when using macros inside a FOR loop (or procedure). When a structure has no values, ## is effectively ignored, and this can produce unexpected results; we hope to introduce a warning message which will be printed when this occurs.

3. Structure-value Substitution

Related to (though not the same as) macro substitution is structure-value substitution. This is where a structure identifier is replaced by the values of the structure when the stucture name is preceded by a single hash (#). For example, if an option or parameter expects a list of numbers and those numbers are contained in a variate Nplaces, then #Nplaces may be given, as in:

```
VARIATE [VALUES=0,2,2] Nplaces
...
...
PRINT Age,Height,Weight; DECIMALS=#Nplaces
```

Some examples in VARIATE declarations can be found in the next section. Further information and examples are given in Section 1.4.4 of the Reference Manual. Preceding a pointer with # is particularly useful since the elements of the pointer (to the lowest level) are substituted for the pointer; see Section 3.3.4 of the Reference Manual.

One other important point should be made about this type of substitution. Options and parameters that require expressions often expect them to be expression values rather than expression data structures. The same is true for formulae. If data structures (containing the appropriate expressions or formulae) are to be used, then they must be preceded by #. Some examples are given below:

(a)
```
EXPRESSION [VALUE=Capacity=Height*Width*Depth] Volume
...
...
CALCULATE #Volume
```
is equivalent to
```
CALCULATE Capacity=Height*Width*Depth
```

(b)
```
EXPRESSION [VALUE=Sex.IN.'male'] Males
...
...
RESTRICT Sex,Name,Age; CONDITION=#Males
```
is equivalent to
```
RESTRICT Sex,Name,Age; CONDITION=Sex.IN.'male'
```

(c)
```
FORMULA [VALUE=Drug,Dose] Dmodel
...
...
FIT #Dmodel
```
is equivalent to
```
FIT Drug,Dose
```

(d)
```
FORMULA [VALUE=Blocks/Plots/Subplots] Design
...
...
BLOCKSTRUCTURE #Design
```
is equivalent to
```
BLOCKSTRUCTURE Blocks/Plots/Subplots
```

4. Forming Subsets of Vectors

RESTRICT has always provided a very handy facility which allows subsets of vectors to be looked at and analysed. The principle advantages are that the subsets of elements are not duplicated (thus saving space) and that the other elements are not lost and can easily be brought back into use.

There will be times though when it is necessary to form a new vector containing the subset of elements. One possible reason for this would be if the original vector is large and takes up data space required for other use; a subset vector could be formed and the original vector deleted.

To form a subset vector, X say, from a vector, V, an index, J, must first be created with:

```
RESTRICT V; CONDITION=condition; SAVESET=J
RESTRICT V
```

The following four statements are equivalent in their effect and will each form the required vector X:

```
(a) CALCULATE X = ELEMENTS(V; J)
(b) CALCULATE X = V$[J]
(c) VARIATE [VALUES=#V$[J]] X
(d) VARIATE X; VALUES=!(#V$[J])
```

Note that while (a) and (b) are perhaps easier to understand, (c) and (d) are probably more efficient because the declaration of X as a variate is implicit in (a) and (b) whereas (c) and (d) do not require the values of X to be calculated. If the index vector J is no longer required then it could of course be deleted.

If there is so little free space that the subset cannot be formed while the original vector still exists then the following illustrates a way of using the character data space to achieve the same result:

```
RESTRICT V; CONDITION=condition
PRINT [CHANNEL=T; IPRINT=*; SQUASH=yes] V,':'
DELETE V
READ [CHANNEL=T; SETNVALUES=yes] X
DELETE T
```

The PRINT statement will cause the text structure T to be created and the values of the (restricted) vector V and a colon will be written to it. In some cases, it may also be helpful to make use of other options and parameters of PRINT, the FIELDWIDTH and DECIMALS parameters in particular. An external file could well have been used instead of T, although this would have required opening and closing the file for both output and then input.

5. Bits and Pieces

(a) Tidying up the data space

In many situations, Genstat creates temporary unnamed data structures which are deleted at a later stage by an internal tidying up operation. Because this is a (relatively) time-consuming operation, it is only performed in a limited number of places although its effect can be very desirable if there is little free data space. The operation always takes place when the DELETE directive is used and it is sometimes wise to give

```
DELETE
```

with no options or parameters before (and perhaps after) another statement that is likely to require a lot of free data space to ensure that the maximum is available.

(b) Missing values

Although missing values are usually represented by an asterisk (`*`), this notation cannot be used in expressions because of the potential confusion with the multiplication operator. If missing values are required then they must be represented in some other way. One way is to calculate a quantity that is known to result in a missing value, such as zero divided by zero, as in the following example:

```
CALCULATE B=A.NE.0/0
```

which will result in `B` containing zeroes or ones depending on whether the values of `A` are missing or not. A more elegant way is to declare a scalar without giving it a value, as in:

```
SCALAR Mv
```

Since no value is given, Genstat assumes it is missing (note that this assumption is made only for scalar data structures). This scalar can then be used in the job as a constant in much the same way that the value of pi might be put in a scalar and used as a constant. The example above could then be written as:

```
CALCULATE B=A.NE.Mv
```

(c) One-trip `FOR` loops

The article 'A Data Organisation System for Field Experiments' by Anne Ainsley and Alan Todd in this Newsletter contains examples with `FOR` loops where there are no dummies or `NTIMES` options, i.e.

```
FOR
   ...
   ...
ENDFOR
```

That is to say, they are loops that will be passed through exactly one time. This may seem rather pointless at first glance but the advantage is that it allows a block a statements to be all compiled before they are executed. The output from the block of statements will not therefore be broken up by the individual statements as they are printed during compilation. For example:

```
   1  PRINT 'Humpty Dumpty sat on a wall.'


Humpty Dumpty sat on a wall.

   2  & 'Humpty Dumpty had a great fall.'


Humpty Dumpty had a great fall.

   3  FOR
   4    SKIP [FILETYPE=output] 2
   5    PRINT [SQUASH=yes] 'All the King''s horses and all the King''s men,'
   6    & 'Couldn''t put Humpty together again.'
   7    SKIP [FILETYPE=output] 1
   8  ENDFOR


All the King's horses and all the King's men,
Couldn't put Humpty together again.

   9  STOP
```

In fact, such a block of statements is treated in a way similar to that of a block of statements that came between two `'RUN'` directives in Genstat 4.

(d) Printing dummy values

In most places in a Genstat job where a dummy is given, the dummy is replaced by the identifier that is its current value. For example, in the following loop,

```
FOR I=A,B,C
   ...
   ...
   PRINT I
   ...
   ...
ENDFOR
```

it is the values of A, B and C which will actually be printed. If instead the names of the identifiers are required to be printed, rather than their values, then it is necessary to print I as the element of a pointer, as in:

```
PRINT !P(I)
```

(e) Commenting out sections of program

When developing a Genstat program or procedure it is often useful to be able to comment out a section of the code to disable it temporarily. This can easily be done by enclosing the relevant set of statements in quotes ("). However, large sections of code will often contain comments already, in which case this method is not so simple to use. An alternative method is to bracket the required section of code with the statements

```
IF 0
   ...
   ...
ENDIF
```

which will always inhibit execution of the intermediate statements, as zero is interpreted as the logical value *false*.

Corrections to the First Printing of the Genstat 5 Manual

A E Ainsley
AFRC Institute of Arable Crops Research
Rothamsted Experimental Station
Harpenden
Hertfordshire
United Kingdom *AL5 2JQ*

The following corrections were made in the reprinted first edition of the hardback manual and the first printing of the softback edition. Owners of manuals from the first printing, identifiable by there being no reference to the First Edition immediately after the line '©Lawes Agricultural Trust, 1987' on page iv, may care to make the following alterations. Only corrections of fact and sense are included in the following. Negative line numbers are used to denote lines from the bottom of the page and only the main text (excluding examples) is included when counting lines.

Page	**Line**		
viii	4	replace:	Lane 1987
		with:	Lane 1988
14	–13	replace:	... –1.6
		with:	... –0.4
23	15	replace:	COUNTS
		with:	COUNTS
23	21	replace:	MINIMA, MAXIMA
		with:	MINIMA, MAXIMA
58	–4	replace:	scalar 1 matrix 5
		with:	1 scalar 5
67	–1	replace:	default 8
		with:	default *
161	7	replace:	(Sca = Ma$[2;1]
		with:	(Sca = Ma$[2;1])
165	–4	replace:	$i\_s < 1$ or $i\_s > n$
		with:	$i - s < 1$ or $i - s > n$
183	–7	replace:	NGROUP
		with:	NGROUPS
218	–14	replace:	ApA
		with:	A'A
242	–14	replace:	specied
		with:	specified
262	7	replace:	' and ,
		with:	' and .
265	7	replace:	2,1,2
		with:	!(2,1,2)
273	–9	replace:	3,5,7,2
		with:	3,2,7,5
273	–8	replace:	2,6,4,8
		with:	2,8,4,6
277	–3	replace:	2,6,4,8
		with:	2,8,4,6

Page	Line		
283	–3	delete sentence beginning:	
			By default, output is sent ...
292	–12	replace:	YLOWER=0,0.5; YUPPER=0.5,1; XLOWER=0; XUPPER=0.5
		with:	YLOWER=0; YUPPER=0.5,0.4; XLOWER=0,0.5, XUPPER=0.5,1
299	–9	replace five lines beginning:	
			VARIATE [VALUES=1...3] Xpos; DECIMALS=0
		and ending:	
			STYLE=grid
		with:	"Draw a plot with specified contour levels"
299	–1	replace:	7.3.3.
		with:	7.3.3 but with windows rearranged to fit on the page.
308	–13	replace:	$(y_i – f_i)/\surd(w_i \times \text{variance}_i)$
		with:	$(y_i – f_i) \times \surd(w_i/\text{variance}_i)$
308	–9	replace:	matrix *X* by
		with:	matrix *X* and the diagonal matrix of weights *W* by
308	–8	replace:	$l_i = \{X(X'X)^{-1}X'\}_{ii}$
		with:	$l_i = w_i \times \{X(X'WX)^{-1}X'\}_{ii}$
331	–3	replace:	$x_i \times y_i$
		with:	$x_i \times z_i$
351	11	replace:	where ω is
		with:	where ϕ is
353	13	replace:	$\log_e$
		with:	log
353	14	replace:	$\log_e$
		with:	log
359	–3	replace:	$u_i\{X(X'UX)^{-1}X'\}_{ii}$
		with:	$u_i \times w_i \times \{X(X'UWX)^{-1}X'\}_{ii}$
360	2	replace:	in 8.1.1 except that each residual is the signed contribution to the deviance
		with:	in 8.1.1: each residual is the signed square root of the contribution to the deviance
360*	7	replace:	$\text{sign}(y_i – f_i) \times (\text{deviance contribution})_i/\surd\{w_i \times \text{dispersion} \times (1–l_i)\}$
		with:	$\text{sign}(y_i – f_i) \times \surd\{w_i \times (\text{deviance contribution})_i/(\text{dispersion} \times (1–l_i))\}$
360*	15	replace:	$\text{residual}_i = (y_i – f_i)/\surd\{w_i \times \text{dispersion} \times (1–l_i)\}$
		with:	$\text{residual}_i = (y_i – f_i) \times \surd\{w_i/(V(f_i) \times \text{dispersion} \times (1–l_i))\}$
362	9	replace:	$y = a + b \times r^x$
		with:	$y = a + b \times r^x + \varepsilon$
365	19	replace:	$y = a_1 + b \times r^x$
		with:	$y = a_1 + b \times r^x + \varepsilon$
365	21	replace:	$y = a_2 + b \times r^x$
		with:	$y = a_2 + b \times r^x + \varepsilon$

Page	**Line**		
365	28	replace:	$y = a_1 + b_1 \times r^x$
		with:	$y = a_1 + b_1 \times r^x + \varepsilon$
365	30	replace:	$y = a_2 + b_2 \times r^x$
		with:	$y = a_2 + b_2 \times r^x + \varepsilon$
366	1, 3, 11, 12, 13, 14, 16, 17, 18, 20, 21, 22, 29, 31		
		at the end of each line add:	
			$+ \varepsilon$
367	8, 18, 20, 27, 34		
		at the end of each line add:	
			$+ \varepsilon$
368	1, 15, 18, 21		
		at the end of each line add:	
			$+ \varepsilon$
381	–13	replace:	Xr=NPI((
		with:	Xr=NORMAL((
539	–3	replace:	and the past values
		with:	and the later values
539	–2	replace:	series with past values
		with:	series with later values
539	–1	replace:	both past and future values
		with:	both future and past values
541	7	replace:	The number the minimum
		with:	The number *n* is the minimum
548	6	replace:	transformation is 2*N*.
		with:	transformation is *N*.
552	7	replace:	$\phi(B)\{'^d y_t^{(\lambda)} - c\}$
		with:	$\phi(B)\{\nabla^d y_t^{(\lambda)} - c\}$
552	8	replace:	• is the *differencing*
		with:	∇ is the *differencing*
553	15	replace:	$\phi(B)\Phi(B^s)\{'^d \nabla_s^D y_t^{(\lambda)} - c\} = \theta(B)\theta(B^s)a_t$
		with:	$\phi(B)\Phi(B^s)\{\nabla^d \nabla_s^D y_t^{(\lambda)} - c\} = \theta(B)\Theta(B^s)a_t$
553	–9	replace:	$P \neq 0, D \neq 0, Q \neq 0$ and $s \neq 1$.
		with:	$P \geq 0, D \geq 0, Q \geq 0$ and $s \geq 1$.
556	3	delete sentence beginning:	
			If $t_0 < 1$
556	–10	replace:	$\{\sigma_a^m \times$
		with:	$\{\sigma_a^{2m} \times$
557	–16	replace:	$\lvert V \rvert = 1 + (\phi_1 - \theta_1) \times ($
		with:	$\lvert V \rvert = 1 + (\phi_1 - \theta_1)^2 \times ($
558	2	replace:	$- \lvert V \rvert^{1/N}$
		with:	$\lvert V \rvert^{1/m}$

Page	Line		
558	9	replace:	$\left\{\prod_{t=1}^{N} y_t\right\}1/N$
		with:	$\left\{\prod_{t=1}^{N} y_t\right\}^{1/N}$
569	−13	replace:	then calculate the simple
		with:	then calculate the sample
574	−4	replace:	$+(1-\theta_1)$
		with:	$+(1-\theta_1 B)$
671	−10	replace:	NGROUP
		with:	NGROUPS
743	22	in column 1 replace:	
			MV()
		with:	NMV()
744	−23	in column 1 after:	
			PCO 481
		insert line:	PCP 452
747	−6	in column 2 before:	
			TDISPLAY 560
		insert line:	TABULATE 222

N.B. The corrections marked with an * in the Page column were also incorrect in the second printing.

Correction to Genstat Newsletter No. 20

There is a small error in the article 'Prediction on the Scale of the Linear Predictor for Generalized Linear Models' by M S Ridout in Genstat Newsletter No. 20. The procedure `CONVERT` does not distinguish properly between the settings `LINK=canonical` and `LINK=complementaryloglog`. To make it work properly, two lines should be inserted: they are flagged by arrows in the listing below. This error was introduced during editing. We apologise for the inconvenience.

```
  PROCEDURE 'CONVERT'
  PARAMETER 'DISTRIBUTION','LINK','NDIST','NLINK'
     "
      Converts strings DISTRIBUTION and LINK, set as for the MODEL directive,
      to numeric codes NDIST and NLINK defined as follows

         DISTRIBUTION   NDIST                   LINK      NLINK
         --------------------                 ----------------
          Multinomial     1                       Log        1
              Poisson     2                     Logit        2
             Binomial     3                  Identity        3
               Normal     4                Reciprocal        4
                Gamma     5                     Power        5
       Inversenormal      6                    Probit        6
                                           Squareroot        7
                                  Complementaryloglog        8
     "
  CONCATENATE [DIST1] DISTRIBUTION; WIDTH=1
  & [LINK1] LINK; WIDTH=1
→& [LINK2] LINK; WIDTH=1; SKIP=1
  CALC NDIST = SUM((!T(M,m,P,p,B,b,N,n,G,g,I,i) .IN. DIST1) * !(2(1...6)))
  & NLINK = SUM((!T(L,l,I,i,R,r,P,p,S,s,C,c) .IN. LINK1) * !(2(1,3,4,5,7,8)))
  & NLINK = NLINK + (NLINK .IN. !(1,5) .AND. NDIST .EQ. 3)
→& NLINK = NLINK + (NDIST-1-NLINK)*(NLINK.EQ.8 .AND. LINK2.NI.!T(O,o))
  ENDPROCEDURE
```