Genstat®

Command language

www.vsni.co.uk

VSNi

An Introduction to the Genstat Command Language (18th Edition)

by Roger Payne, Darren Murray and Simon Harding

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

Published by: VSN International, 2 Amberside, Wood Lane,
Hemel Hempstead, Hertfordshire HP2 4TP, UK
E-mail: info@genstat.co.uk
Website: http://www.genstat.co.uk/

First published 2010, for GenStat *for Windows* 13th Edition
This edition published 2015, for Genstat *for Windows* 18th Edition

Contents

Introduction

Genstat is a statistical system with a comprehensive system of menus providing all the standard (and many non-standard) analyses. At first sight, it looks like a standard Windows application. However, if you look more closely, you will find that the menus are defining the analyses by writing *scripts* in Genstat's command language. These scripts are saved in Genstat's Input log to give you a full and complete audit trail. More importantly, though, you can write your own scripts to do something new or non-standard, or even just to save time or automate repetitive tasks.

Once you start to write your own programs, you may want to keep them to use again in future. The most convenient way of doing this is to form them into *procedures*. The use of a Genstat procedure looks exactly the same as the use of one of the standard Genstat directives. You can thus extend and customize Genstat for your own special requirements.

So by learning the command language, you can unlock the full power of Genstat, improve your productivity and extend the scope of your analyses.

This *Introduction* was written to provide the notes for VSN's short course on the Genstat command language, but it can be used equally well as a self-learning tool:

- it explains the basic rules that apply to all Genstat commands, and shows how you can use them alongside the menus in Genstat *for Windows*;
- it shows where to find help information about individual commands, or example scripts illustrating various types of analysis;
- it explains the data structures that Genstat uses to store data;
- it tells you how to define statistical models and calculations;
- it explains how you can write your own procedures.

However, it does not attempt to cover everything! Chapter 10 explains where to find further information. You will see that there are many other Genstat Guides, which can be accessed from within Genstat *for Windows* by selecting sub-options of the Genstat Guides option of the Help menu on the Genstat menu bar. For commands, the most useful are the two *Guides to Genstat*. Part 1 gives the formal definition of the Genstat language and syntax. It then describes (in detail, with examples) the facilities for input and output, calculations, manipulation, programming and graphics. Part 2 covers Genstat's extensive statistical facilities.

1 Menus and commands

In this chapter you will learn:

- how Genstat's menus use commands;
- the basic rules of the Genstat syntax.

1.1 Menus

When you start Genstat on a PC running MS Windows, it starts up two processes. The first, known as the Genstat *Client*, controls the Windows interface for Genstat. The client collects information from you and sends it to the second process, which is known as the Genstat *Server*. This runs in the background, performing calculations, and returning information back for the client to display.

To illustrate the concepts, we shall calculate summary statistics for iron measurements in some soil samples. The data file `Iron.dat` is available in Genstat's Data folder (or directory), together with all the other data files that we use in this Guide. The menus that open files always start in your *working directory*. To make the Data folder your working directory, click on Tools on the menu bar and then on the Working Directory line, as shown in Figure 1.1.

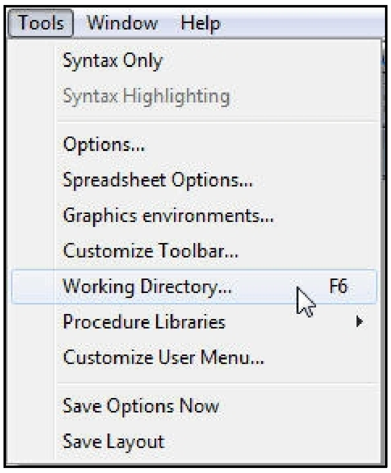


Figure 1.1

This opens the Working Directory menu shown in Figure 1.2. You can now click on the Add Data Folder button, and then on OK, to make the folder your working directory.

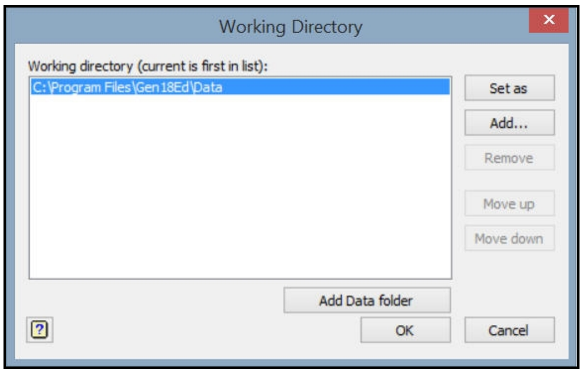

Working Directory
Working directory (current is first in list):
C:\Program Files\Gen18Ed\Data
Set as
Add...
Remove
Move up
Move down
Add Data folder
OK
Cancel

Figure 1.2

The file `Iron.dat` is an ASCII file, that is, an ordinary text file. So we click on the ASCII file sub-option of the Load option of the Data menu on the menu bar (Figure 1.3) to open the menu to Read Data From ASCII File (Figure 1.4).

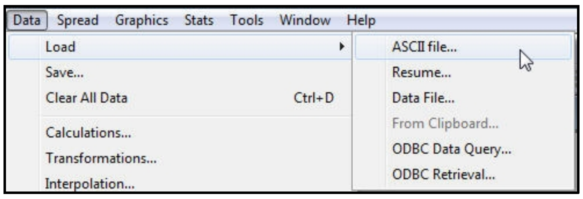
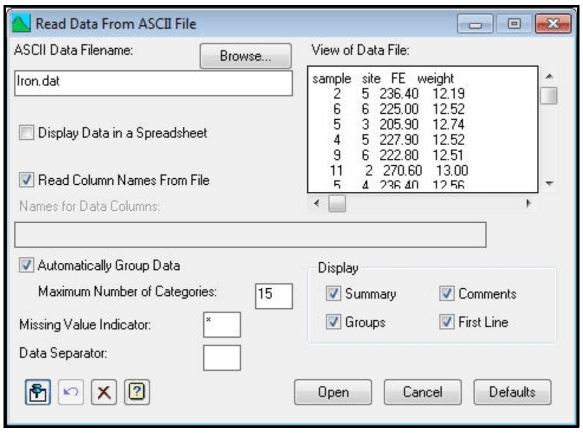


Figure 1.3

The menu has a Browse button that you can use to locate the file, or you can type the file name into the field at the top of the left-hand side of the menu. (Notice that we do not need to type the path of the file, as we have made the Data folder our default directory.) Once this has been done, the box on the right-hand side of the menu shows the first few lines of the file.



ASCII Data Filename: Iron.dat

[ ] Display Data in a Spreadsheet

[✓] Read Column Names From File

Names for Data Columns:

[✓] Automatically Group Data

Maximum Number of Categories: 15

Missing Value Indicator: *

Data Separator:

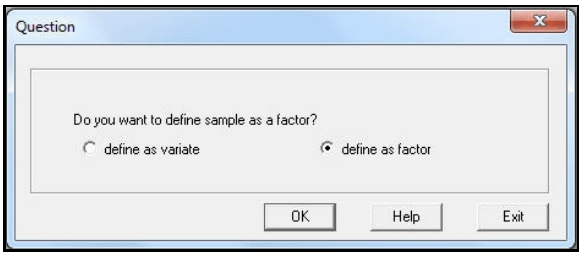
Display: [✓] Summary [✓] Comments [✓] Groups [✓] First Line

View of Data File:

```
sample  site   FE    weight
    2     5  236.40   12.19
    6     6  225.00   12.52
    5     3  205.90   12.74
    4     5  227.90   12.52
    9     6  222.80   12.51
   11     2  270.60   13.00
    5     4  236.40   12.56
```

Figure 1.4

Files like these are easiest to read if the data variables are arranged in columns, with the names of the variables in the first line. We can then check the box to Read Column Names from File, instead of having to type the names into the menu (notice that the box Names for Data Columns is greyed out when the box is checked). Figure 1.4 shows that the file has data for four variables: `sample`, `site`, `FE` and `weight`. We have also checked the box to Automatically Group Data, and entered 15 as the Maximum Number of Categories.



Do you want to define sample as a factor?

( ) define as variate [✓ define as factor]

Figure 1.5

When we click on Open, Genstat counts the number of distinct values taken by each variable and, if that is less than 15, it pops up a menu so that we can choose whether or not to make that variable a *factor*. In the iron data, `sample` has only 12 different values. To make it a factor we leave the radio button in Figure 1.5 at its default setting, define as factor, and click on OK. The variable `site` has only six values, and we can make that into a factor in a similar way.

In Figure 1.4 we checked the boxes for all the display categories, to generate the output below.

First line of data with no missing values

```
        2         5    236.40     12.19
```

Summary

The file C:/Program Files/Gen18ed/Data/Iron.dat is assumed to contain 4 structure(s), with one value for each structure on each record.

The structure identifiers read from the file are: sample, site, FE and weight.

Occurrence of distinct values of sample

```
                          Count
          category
                 1           12
                 2           12
                 3           12
                 4           12
                 5           12
                 6           12
                 7           12
                 8           10
                 9           12
                10           10
                11           10
                12           10


 Occurrence of distinct values of site

                          Count
          category
                 1           24
                 2           22
                 3           24
                 4           18
                 5           24
                 6           24


The file contains 136 values for each of the following structures:

        Identifier        Type      Missing
            sample      factor            0
              site      factor            0
                FE     variate            0
            weight     variate            0
```

The data record iron measurements made from a set of soil samples. The first two variables categorize the samples: each value of `site` records the code number of the laboratory that carried out the analysis, and `sample` contains the number (from 1 to 12) of the originating soil sample that was given to the laboratory to analyse. The other two variables are numbers: the `FE` stores the measured parts per million (ppm) of iron, and `weight` stores the weights of soil that were analysed. Lists of numbers like these are stored in *variates*. Genstat has thus defined four *data structures* to store the information.

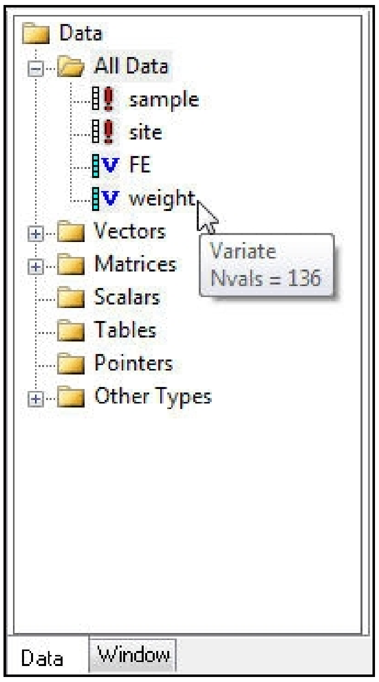


Figure 1.6

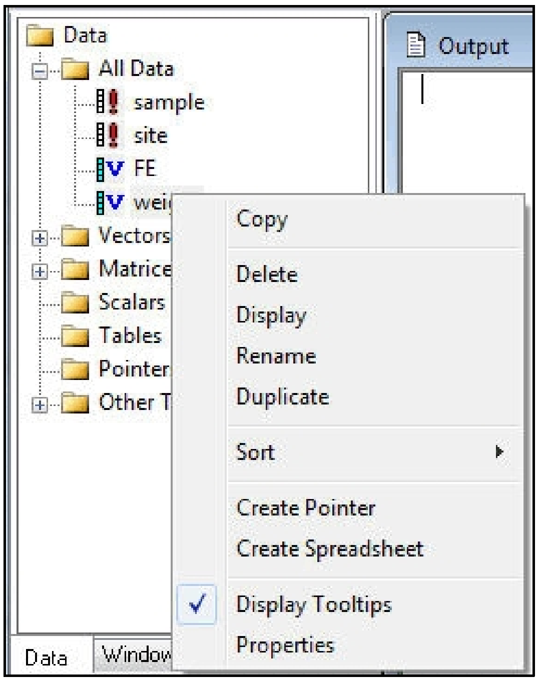


Figure 1.7

You can use the Data View pane to see what data structures are currently stored inside Genstat. The tree allows you to select the types of data structure that you want to list on the right-hand side. In Figure 1.6, we have opened All Data. The *variates* are identified by a blue "v", and the *factors* are identified by the red exclamation mark. Genstat provides a range of data structures that are convenient for different types of data, but these two are the most common. Notice that, as you rest the mouse on the name of a structure, a small window appears with information about its attributes. These *tool tips* are controlled by the right-mouse menu (Figure 1.7), which can be obtained by making a right-mouse click on any of the structures. In addition the menu allows you, for example, to delete, rename or display (i.e. print in the Output window) structures. You can also drag and drop structures from Data View onto most of Genstat's other menus.

The statistical menus in Genstat *for Windows* are accessed from the Stats menu on the menu bar (Figure 1.8). Here we have selected the Summary Statistics sub-option of the Summary Statistics option, which opens the Summary Statistics menu shown in Figure 1.9.

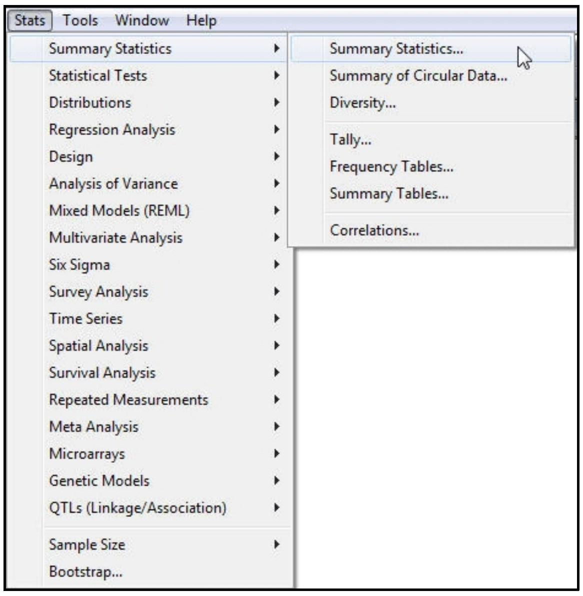
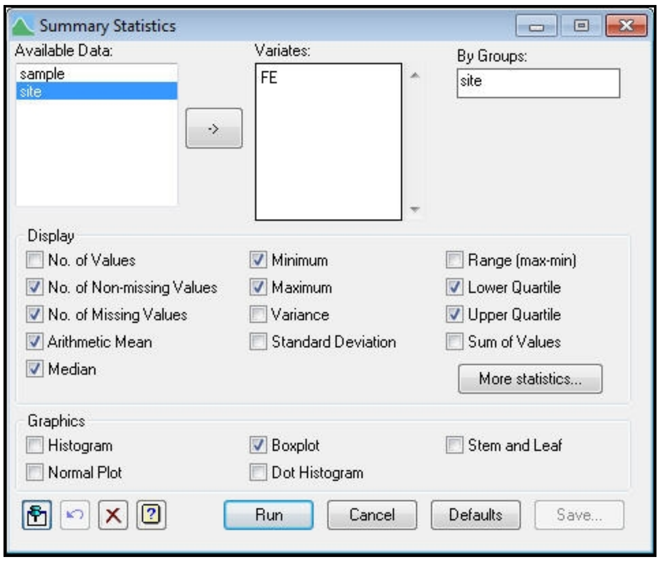


Figure 1.8

The menu displays summary statistics describing the contents of variates, and can also produce some useful graphs. We enter the variate `Fe` into the Variates box, enter the factor `site` into the By Groups box, and click on the Run button.

Genstat prints summary statistics for each site, in turn, in the Output window as shown below. It also starts another process, known as the Genstat *Graphics Viewer*, to display six *boxplots*, one for each of the laboratories (or sites).


Summary Statistics

Available Data: sample, site
Variates: FE
By Groups: site

Display
- [ ] No. of Values
- [✓] No. of Non-missing Values
- [✓] No. of Missing Values
- [✓] Arithmetic Mean
- [✓] Median
- [✓] Minimum
- [✓] Maximum
- [ ] Variance
- [ ] Standard Deviation
- [ ] Range (max-min)
- [✓] Lower Quartile
- [✓] Upper Quartile
- [ ] Sum of Values

Graphics
- [ ] Histogram
- [ ] Normal Plot
- [✓] Boxplot
- [ ] Dot Histogram
- [ ] Stem and Leaf

Figure 1.9

Summary statistics for FE: site 1

```
   Number of observations =   24
 Number of missing values =   0
                     Mean =   289.6
                   Median =   289.1
                  Minimum =   269.5
                  Maximum =   308.2
           Lower quartile =   282.1
           Upper quartile =   295.6
```

Summary statistics for FE: site 2

```
     Number of observations =   22
   Number of missing values =   0
                       Mean =   274.2
                     Median =   273.2
                    Minimum =   262.6
                    Maximum =   283.1
             Lower quartile =   270
             Upper quartile =   280.1
```

Summary statistics for FE: site 3

```
     Number of observations =   24
   Number of missing values =   0
                       Mean =   216.3
                     Median =   212.8
                    Minimum =   200.6
                    Maximum =   252.7
             Lower quartile =   208.7
             Upper quartile =   218.4
```

Summary statistics for FE: site 4

```
     Number of observations =   18
   Number of missing values =   0
                       Mean =   239.5
                     Median =   238.1
                    Minimum =   232.5
                    Maximum =   255.7
             Lower quartile =   236.5
             Upper quartile =   239.4
```

Summary statistics for FE: site 5

```
     Number of observations =   24
   Number of missing values =   0
                       Mean =   234.9
                     Median =   234.6
                    Minimum =   222.7
                    Maximum =   251.6
             Lower quartile =   230.2
             Upper quartile =   237.1
```

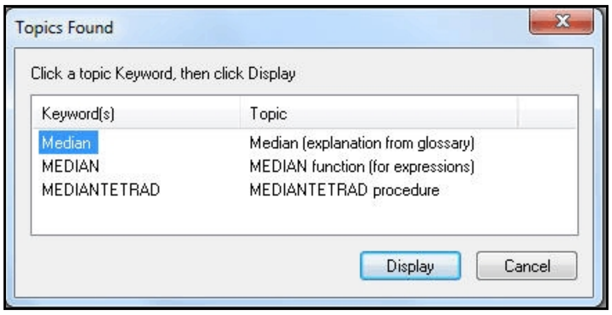
Summary statistics for FE: site 6

```
     Number of observations =   24
   Number of missing values =   0
                       Mean =   225.5
                     Median =   224.2
                    Minimum =   215.3
```

```
       Maximum =   238.6
Lower quartile =   221.8
Upper quartile =   229.1
```

The output is labelled using standard statistical terminology but, if you are unsure about any of the words or phrases, you can use Genstat's context-sensitive help. This links to the VSNi Knowledge Base, opening the relevant page in your web browser. Put the cursor into the word of interest, or into the first word of the phrase of interest, and press the F1 key. For example, if you put the cursor into the word "quartile" and press F1, Genstat opens the Quartile topic, which contains the information: "The lower quartile is the value *l* such that 25% of a sample are less than *l*. Similarly, the upper quartile is the value *u* such that 25% of a sample are greater than *u*."

Sometimes there is more than one potentially relevant topic. Genstat then provides a menu so that you can select the one that seems most appropriate. The menu for the word "median" is shown in Figure 1.10. Selecting Median (explanation from glossary) produces the definition: "Median is the value that divides a sample into two equally sized groups."


Topics Found

Click a topic Keyword, then click Display

Keyword(s)	Topic
Median	Median (explanation from glossary)
MEDIAN	MEDIAN function (for expressions)
MEDIANTETRAD	MEDIANTETRAD procedure

Figure 1.10

Figure 1.11 shows the graph, displayed in a separate window by Genstat's Graphics Viewer. You can zoom the display using the slider on the toolbar, or by holding the left mouse button and moving the mouse up and down. If your mouse has a centre button, you can move the display within the window by moving the mouse with that button held down. Alternatively, you can use the scroll bars at the bottom and on the right-hand side of the screen.

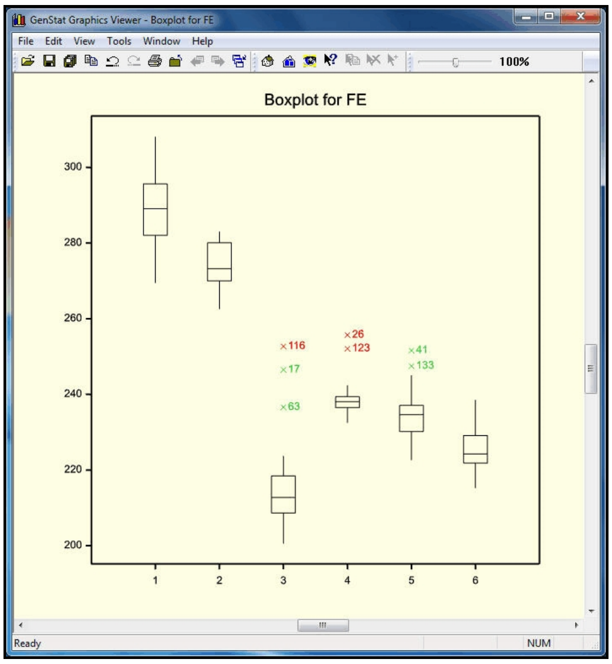


Figure 1.11

These are *schematic* boxplots. Each one has a central box spanning the inter-quartile range of the data from a laboratory (so that 50% of the observations lie inside the box) with a horizontal bar drawn across the box at the median. There are whiskers extending from each box to the most extreme data values within inner *fences* that are defined to be at a distance of 1.5 times the interquartile range beyond the quartiles, or at the furthest data value if that is smaller. Points that lie beyond the whisker are regarded as *outliers*, and are plotted as crosses with labels identifying their unit number. Points that lie beyond outer fences, defined to be at a distance of three times the interquartile range beyond the quartiles, are regarded as *far outliers* and plotted in red. Ordinary outliers are plotted in green.

The display shows that Laboratories 1 and 2 are producing consistently higher results than the rest, and Laboratory 3's results are generally lower.

1.2 Practical

Select the Data File sub-option of the Load option of the Data menu. Move from the directory (i.e. folder) containing the Genstat executable program to the `Data` directory and set this as the working directory.

Load the file `Sales.gsh` and open the Data View pane to see what data structures it contains.

Use the Summary Statistics menu to display the minimum, maximum and mean sales, and to plot a boxplot of the sales in each town.

Print the skewness and kurtosis of the sales figures, and use the context-sensitive help to obtain some information about what these represent.

1.3 Commands

To calculate summary statistics, and display the graphs in Section 1.1, the client has formed a script of commands, and sent them to the server to be executed.

You can see these commands in the Input Log. Depending on how your copy of Genstat has been set up, the Output window may also contain a listing of the commands interspersed with the output. You can control whether this happens, by using the Options menu, which can be opened from the Tools menu on the Genstat menu bar, as shown in Figure 1.12.

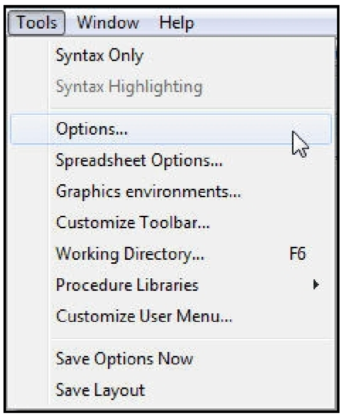
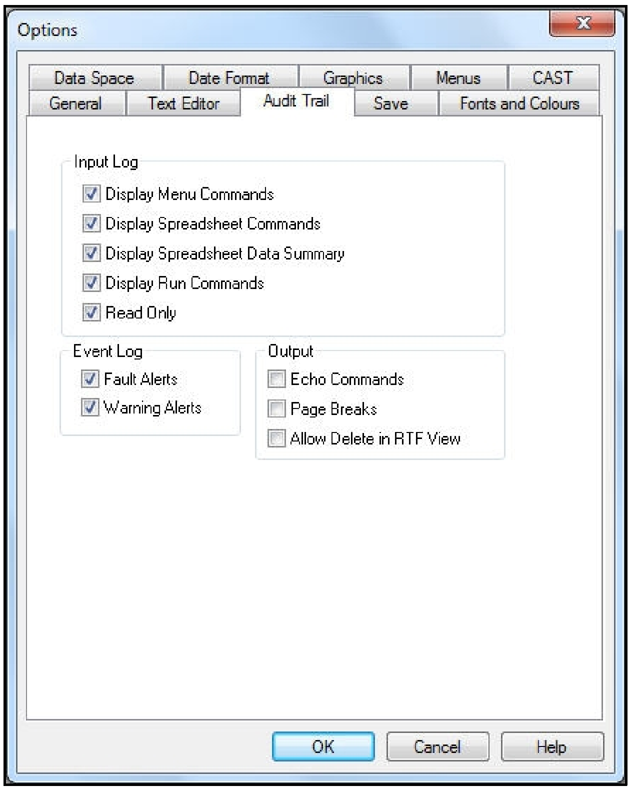


Figure 1.12

The Options menu has tabs to control many different aspects of the way that Genstat runs.

The General tab allows you to stop the Start Page appearing when you open Genstat, and to control aspects like the number of files that are saved for the recent file list.

The Audit trail tab (Figure 1.13) controls what appears in the Input Log, the Event Log and the Output window. The Echo Commands box is unchecked in Figure 1.13. So the commands are not displayed in Output window.



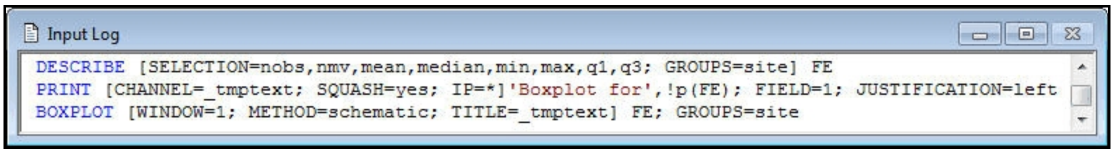
Input Log
- [✓] Display Menu Commands
- [✓] Display Spreadsheet Commands
- [✓] Display Spreadsheet Data Summary
- [✓] Display Run Commands
- [✓] Read Only

Event Log
- [✓] Fault Alerts
- [✓] Warning Alerts

Output
- [ ] Echo Commands
- [ ] Page Breaks
- [ ] Allow Delete in RTF View

Figure 1.13

The contents of the Input Log after drawing the boxplot are in Figure 1.14. There are three commands here: to print the summary statistics (`DESCRIBE`), to produce a title for the boxplot (`PRINT`), and to draw the boxplot (`BOXPLOT`). Notice that Genstat has used a private, temporary structure `_tmptext` to contain the title. Structures like this always have names that begin with the underscore character _, to distinguish them from your own data structures. Previous commands to set the working directory and to load data into Genstat from the file have scrolled up above those shown in the figure.



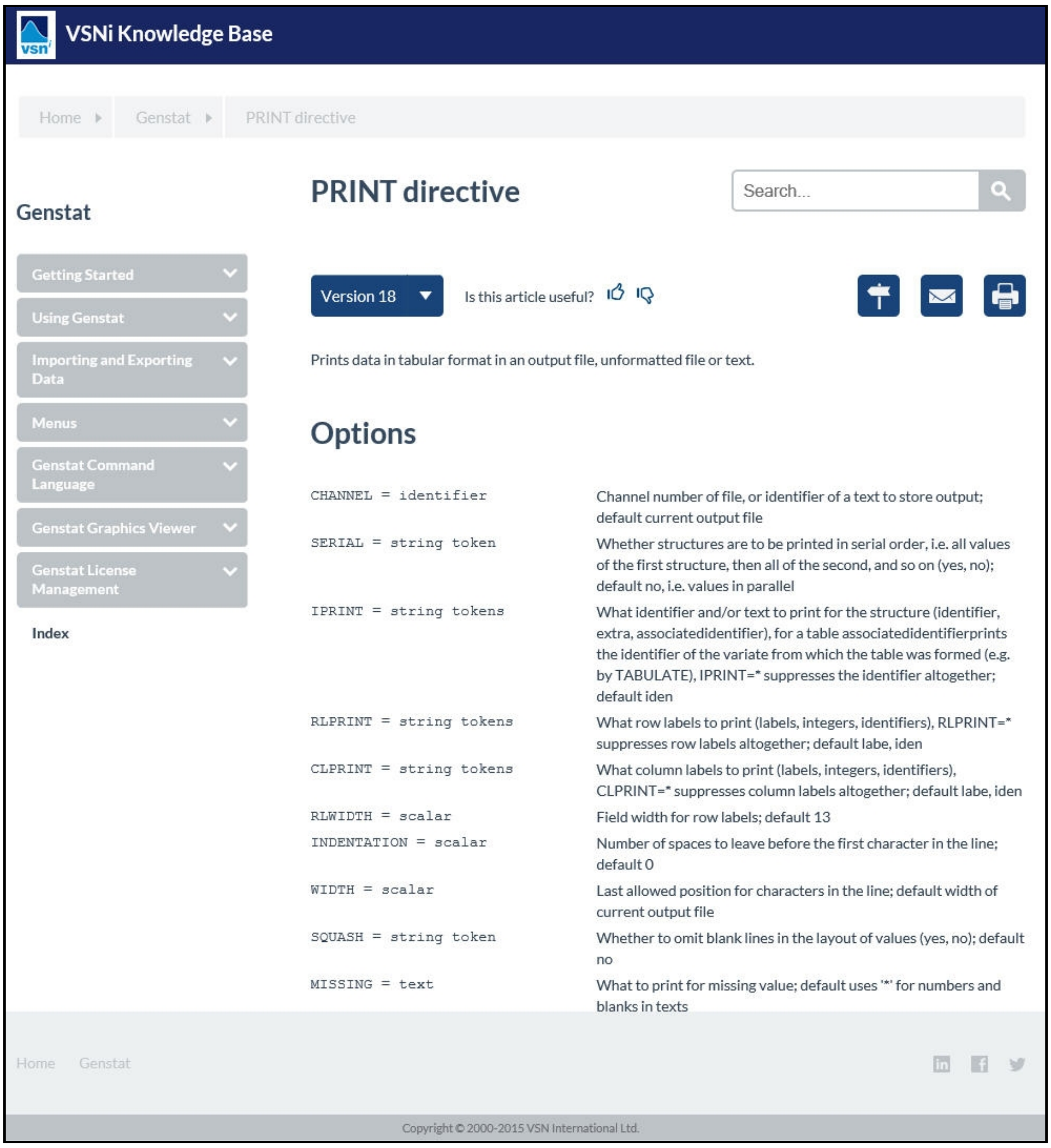
```
DESCRIBE [SELECTION=nobs,nmv,mean,median,min,max,q1,q3; GROUPS=site] FE
PRINT [CHANNEL=_tmptext; SQUASH=yes; IP=*]'Boxplot for',!p(FE); FIELD=1; JUSTIFICATION=left
BOXPLOT [WINDOW=1; METHOD=schematic; TITLE=_tmptext] FE; GROUPS=site
```

Figure 1.14

All Genstat commands have a common form of *syntax*: in other words, there are some general rules that apply to all the commands that you give. We introduce the basic syntax here to get you started. Full details are given in Chapter 3. There are two types of commands: *directives* are the basic commands of the Genstat language while *procedures* are extensions of the language, using programs written in the Genstat language itself. Genstat has a standard library of procedures that are loaded automatically as needed. Directives and procedures both obey identical rules, so you do not need to know which you are using.

Genstat can use different colours to identify the various components of the command. Here, for example, the names of the commands are in blue. This *syntax highlighting* of the current window can be controlled by checking or unchecking the Syntax Highlighting line of the Tools menu (see Figure 1.12).

The context-sensitive help works on the names of commands as well as items in output. So, we can put the cursor into PRINT in the Input log, and press the F1 key to open the VSNi Knowledge Base at the page for `PRINT`.



PRINT directive

Prints data in tabular format in an output file, unformatted file or text.

Options

`CHANNEL = identifier`	Channel number of file, or identifier of a text to store output; default current output file
`SERIAL = string token`	Whether structures are to be printed in serial order, i.e. all values of the first structure, then all of the second, and so on (yes, no); default no, i.e. values in parallel
`IPRINT = string tokens`	What identifier and/or text to print for the structure (identifier, extra, associatedidentifier), for a table associatedidentifierprints the identifier of the variate from which the table was formed (e.g. by TABULATE), IPRINT=* suppresses the identifier altogether; default iden
`RLPRINT = string tokens`	What row labels to print (labels, integers, identifiers), RLPRINT=* suppresses row labels altogether; default labe, iden
`CLPRINT = string tokens`	What column labels to print (labels, integers, identifiers), CLPRINT=* suppresses column labels altogether; default labe, iden
`RLWIDTH = scalar`	Field width for row labels; default 13
`INDENTATION = scalar`	Number of spaces to leave before the first character in the line; default 0
`WIDTH = scalar`	Last allowed position for characters in the line; default width of current output file
`SQUASH = string token`	Whether to omit blank lines in the layout of values (yes, no); default no
`MISSING = text`	What to print for missing value; default uses '*' for numbers and blanks in texts

Figure 1.15

The top of the page, shown in Figure 1.15, shows some of the options of `PRINT`, and tells us that it is a directive that allows you to print data to an output file. This includes the Output window, which is regarded as the primary output "file" by the server. (Note that this does not send data or results to a printer: to do that you can select the Print option from the File menu in the menu bar.)

We will use `PRINT` to illustrate the basic rules of the Genstat syntax.

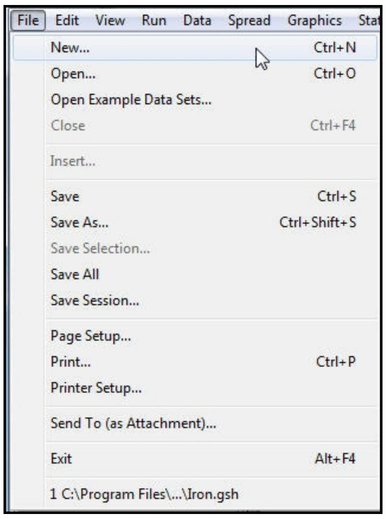


Figure 1.16

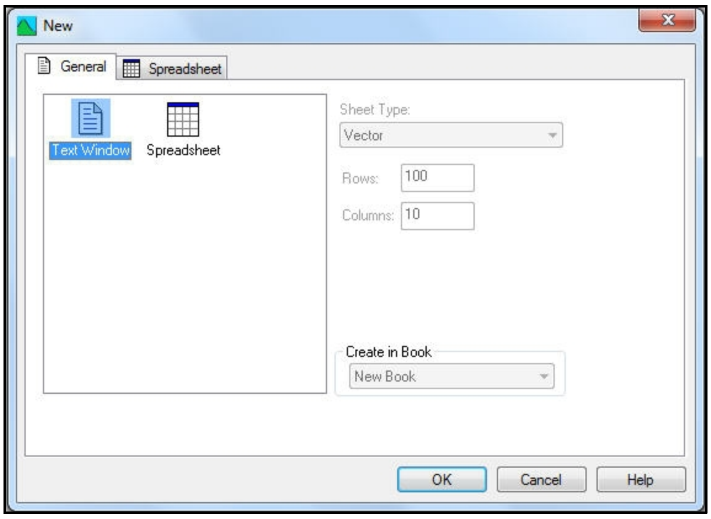


Figure 1.17

To give commands directly, it is best to open a new window in which to construct them. This is done by clicking on File in the menu bar and selecting New, as shown in Figure 1.16. This generates the menu shown in Figure 1.17, allowing you to choose what type of new window you want. Selecting Text Window and clicking on OK gives you a new, empty, window which will become the current window. You can type, for example, the simple command

```
PRINT 1
```

to display a single set of data: the number 1.

When constructing commands in a window, you can use the usual keys for typing and deleting characters, and moving about the window. You can also switch between Insert and Overwrite mode by pressing the Insert key, and the Status bar will display, with Ins or Ovr, which mode you are in at any time.

This is a trivial exercise, of course, but it serves to show how commands work. To get Genstat to execute this command, leave the cursor at the end of the line (that is, just after the `1`) and select the Run menu from the menu bar. Select Submit Line, as shown in Figure 1.18, and the command will be executed. Alternatively, you can use the "short-cut" `Ctrl+L`, by pressing the `L` key while holding down the `Ctrl` key.

The resulting output is displayed in the Output window, as shown in Figure 1.19.

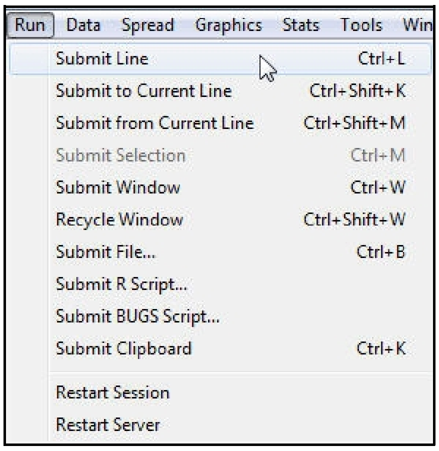
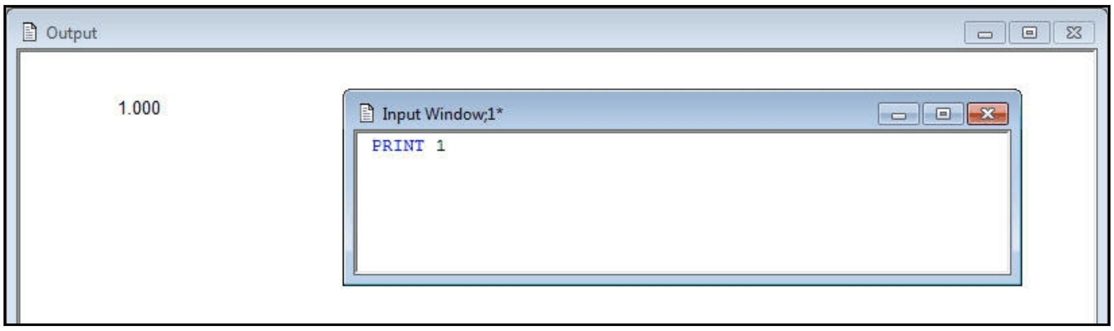


Figure 1.18



```
1.000
```

```
PRINT 1
```

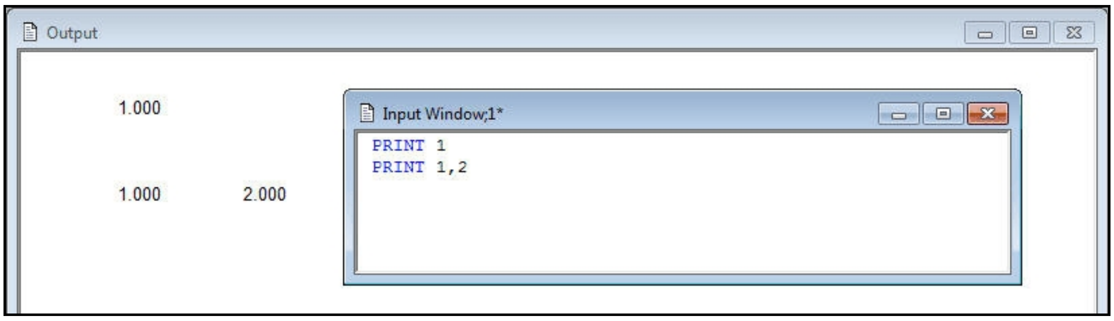
Figure 1.19

This is not a particularly useful operation of course, because you already know what the set of data is, and because it consists only of a single number; however, this will quickly be generalized. In the meantime, you can see that the directive name, `PRINT`, is like a command verb which instructs Genstat to do something, and the number `1` is like the object of the command. All directives, and procedures, work like this, though not all directive names are actually verbs in the English language. The object is called the *primary parameter* of the command.

The `PRINT` directive, like all others, works with sets of data. You can make it work with several sets of data at once by giving a *list*; for example, the command

```
PRINT 1,2
```

has two sets, each containing one number, as shown in Figure 1.20.



```
1.000

1.000         2.000
```

```
PRINT 1
PRINT 1,2
```

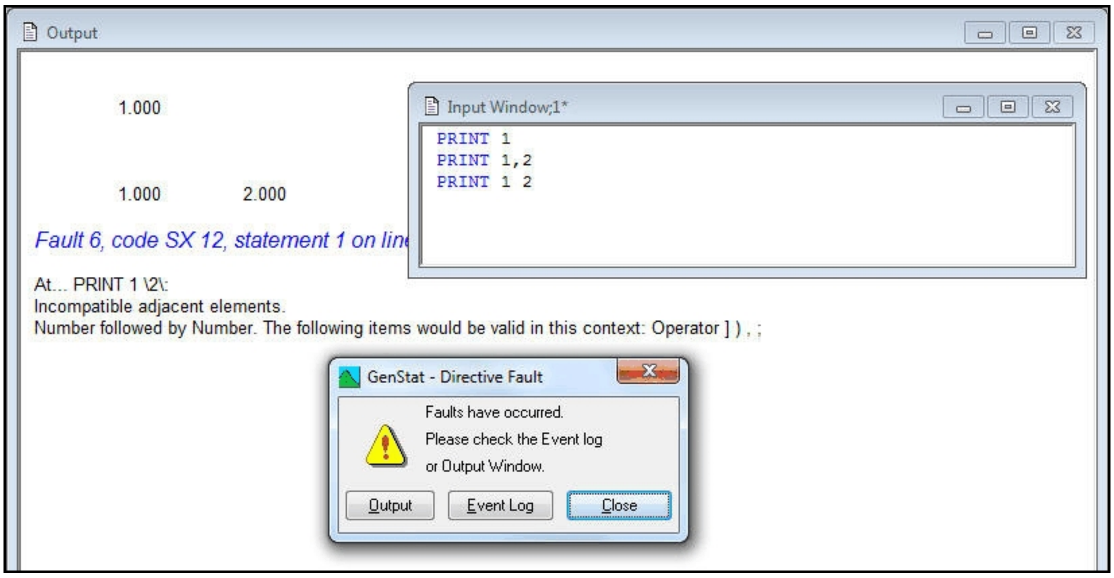
Figure 1.20

In Genstat, lists are always constructed using commas. You must not use just spaces; for example, the command

```
PRINT 1 2
```

would be faulted, because the space may be an accident, and you may have meant

```
PRINT 12
```

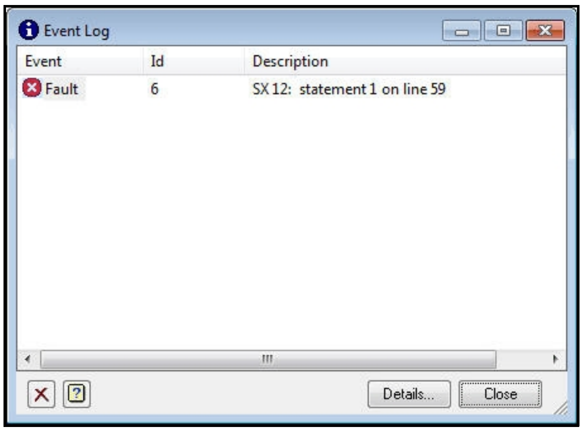


```
1.000

1.000          2.000

Fault 6, code SX 12, statement 1 on lin

At... PRINT 1 \2\:
Incompatible adjacent elements.
Number followed by Number. The following items would be valid in this context: Operator ] ) , ;
```

```
PRINT 1
PRINT 1,2
PRINT 1 2
```

Figure 1.21

Genstat draws attention to mistakes like this by popping up a Fault box, as shown in Figure 1.21. It prints a brief explanatory message in the Output window, and records the fact that a fault has occurred in the a separate window called the Event Log (see Figure 1.22). You can click on the Output button to go to the fault in the Output window, or on the Event Log button to open the Event Log. If you do open the Event Log, you can click on the line of any event to get Genstat to take you to that fault or warning in the Output window.



Event	Id	Description
Fault	6	SX 12: statement 1 on line 59

Figure 1.22

You can, however, use spaces as well as commas. So the following command is acceptable:

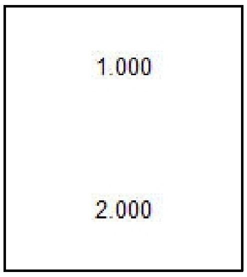
```
PRINT 1 ,    2
```

You will have noticed that `PRINT` commands lay out the data in a tabular form, choosing an appropriate number of decimal places for numbers. By default, a single number is displayed with four significant digits. Also, sets of data with compatible shape are laid out in *parallel*: that is, side-by-side. If you do not want this default display, there several *options* to modify it.

For example, the command

```
    PRINT [SERIAL=yes] 1,2
```

displays the two numbers in *serial* rather than in parallel: that is, one number by itself, and then the other, as shown in Figure 1.23.



```
1.000

2.000
```

Figure 1.23

Most Genstat directives and procedures have options like this to control the way in which the operations are done. They must always be given in square brackets following the directive or procedure name and preceding the parameters, if any. Options have the form *name=setting*, where here the name is SERIAL and the setting is yes. Settings can be words, as here, or numbers. If you set several options, you must separate them with a semi-colon, as in

```
    PRINT [SERIAL=yes; INDENTATION=10] 1,2
```

This command would indent the output by 10 characters, so that if you arrange to send the display to a printer, you could rely on having a clear margin on the paper, perhaps for binding.

The CHANNEL option of PRINT was used in the Input Log, to put the output into the Genstat *text* structure _tmptext.

Most Genstat directives and procedures also have *auxiliary parameters* which control the way the command works. For example, the command

```
    PRINT 1,2; DECIMALS=0,1
```

```
1          2.0
```

Figure 1.24

gives the output shown in Figure 1.24.

The effect of the DECIMALS parameter is to specify how many decimal places to display for each set of data. The essential difference between an option and an auxiliary parameter is that an option specifies a modification once and for all for the command: an auxiliary parameter specifies a modification that may be different for each of the sets of data in turn. The setting of the DECIMALS parameter above, 0,1, is matched item by item with the setting of the primary parameter, 1,2. This distinction applies to all Genstat commands.

The setting of an auxiliary parameter is otherwise like that of an option, with the form *name=setting*, and the semi-colon separator is needed between successive parameters. The primary parameter itself has a name, except when there are no auxiliary parameters. So you could actually give the command:

```
    PRINT STRUCTURE=1,2; DECIMALS=0,1
```

However, if you specify the primary parameter first in a command, its name can always be omitted.

You can abbreviate the names of directives to the first four characters. Names of procedures in the standard Genstat library are defined so that they too can be abbreviated to four characters. Names of options and parameters can also be abbreviated to four characters, and sometimes further. The full abbreviation rules are described in Section 3.3.

So far, we have used the very simplest sets of data, consisting of a single number each. Most practical work is done with series of numbers, like those in Section 1.2. For example, we can display the values in the variate called FE by simply giving the command:

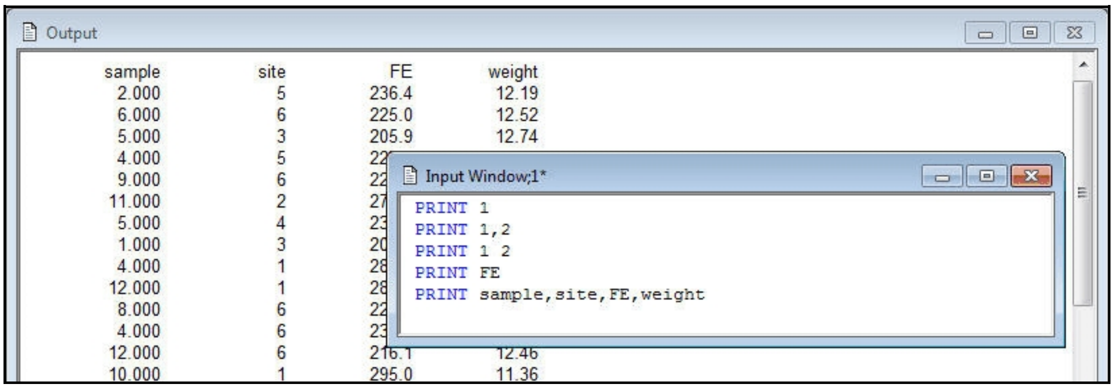
```
PRINT FE
```

The name of a data structure is known as its *identifier* in Genstat. These can consist of up to 32 letters or digits, but they must start with a letter. Case is significant, so the identifier `FE` is different to `fe`. We have used capital letters for this identifier but lower case for the others, like `sample`; however, you may find it easier to stick to all lower-case or all upper-case for your identifiers, at least while you get started with the system.

The `PRINT` command works on all types of Genstat data structures, so you can probably guess that the following command would display all the data that was loaded in Section 1.2.

```
PRINT sample,site,FE,weight
```

Part of the display is shown in Figure 1.25.



```
    sample        site          FE      weight
     2.000           5       236.4       12.19
     6.000           6       225.0       12.52
     5.000           3       205.9       12.74
     4.000           5       22
     9.000           6       22
    11.000           2       27
     5.000           4       23
     1.000           3       20
     4.000           1       28
    12.000           1       28
     8.000           6       22
     4.000           6       23
    12.000           6       216.1       12.46
    10.000           1       295.0       11.36
```

```
PRINT 1
PRINT 1,2
PRINT 1 2
PRINT FE
PRINT sample,site,FE,weight
```

Figure 1.25

Values can be assigned to data structures in many ways; for example, by loading data from a file, by saving the results of an analysis, or by doing calculations (Chapter 2). In Genstat, calculations are specified in *expressions*, and are most often carried out with the `CALCULATE` directive. As a simple example, we can calculate the amount of iron in each tested quantity of soil by multiplying the weight by the concentration of iron:

```
CALCULATE iron = FE * weight
```

The weights of soil were in grams, so the resulting weights of iron will be in micrograms.

This `CALCULATE` command is more powerful than it looks. In fact, 136 separate calculations are done here: Genstat knows to do this because `FE` and `weight` have been defined to be variates with 136 values. So each of the 136 values stored in `FE` is multiplied by the corresponding value in `weight` and the results stored successively in a new variate called `iron`.

We now know enough to make simple changes to the command scripts that the menus produce. For example, suppose we wanted to produce an ordinary boxplot instead of a schematic one. This is not a choice provided by the Summary Statistics menu (Figure 1.9). However, we can copy the `BOXPLOT` statement from the Input log, edit it in a text window to become

```
BOXPLOT [WINDOW=1; TITLE=_tmptext] FE; GROUPS=site
```

and run it using the Submit Line option of the Run menu. We shall explore these possibilities further in the later chapters.

A full description of the syntax is in Chapter 3, which shows all the options of `PRINT`, as well as its parameters.

1.4 Practical

Edit the script of commands produced in Practical 1.2 to plot an ordinary boxplot of the sales in each town. Edit it again to plot a single boxplot for all the data (i.e. with no groups).

The price charged for each item sold in the data set `Sales.gsh`, used in Practical 1.2, was 2.99. Calculate the amount received on each day in each of the towns, and use the `PRINT` directive to display day, town and amount received, in columns, in the Output window.

2 Data structures

In this chapter you will learn about:

- the `LIST` directive, which tells you what data structures are currently in store;
- how Genstat uses data structures to store your data and results;
- the most commonly-used data structures – variates, factors and texts;
- how to form factors from factors or texts;
- the other data structures – scalars, matrices, tables etc;
- how to obtain example programs from the chapter about data structures in the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*.

In Chapter 1 we introduced two different types of data structures, and showed how these can be read into Genstat from an ASCII (text) file. When data are read into Genstat they are stored within a central data pool in the Genstat server. Information about the current data structures can be viewed in the Data View pane, as shown in Figure 1.6. Alternatively, you can use the `LIST` directive.

```
LIST
```

on its own lists all the data structures that are currently stored. Alternatively, you can specify the types of structures that you want to list, for example

```
LIST variate,factor
```

to list the variates and factors.

An important aspect of Genstat is that the data structures are used not only to supply input to the analysis commands, but also to store output from analyses. Genstat has the aim that anything that can be printed in an analysis should also be savable in a suitable data structure, so that it can be used as input to some other command. Thus, for example, there are directives `AKEEP`, `RKEEP` and `VKEEP` to save output from analysis of variance, regression and `REML` analyses. You might want to do this merely to have more control over the way in which the output is printed. However, it also allows you to write some very powerful programs, using the methods described later, in Chapter 9.

2.1 Variates, factors and texts

The first data structure that we introduced was a *variate*, which stores a list of numerical values. The length (or number of values) of a variate is fixed, and two variates of different lengths cannot be used in the same calculation (unless you are calculating summary statistics from them).

The second data structure was called a *factor*. A factor is a special data structure within Genstat for defining an allocation of units into discrete groups. Each group can be represented with a numerical value known as a *level*, and/or a textual value known as a *label*. The groups are also allocated "ordinal" values, represented by the integers 1 upwards, which indicate the ordering of the levels and/or labels. So the ordinal values show the order in which the levels or labels of the factor will be displayed, for example when the factor is used to define a dimension of a table.

To illustrate some of the ways in which factors can be used, we shall use a data set from Chapter 3 of the *Guide to Anova and Design in Genstat*. The data are stored in the spreadsheet file `Nematode.gsh`, which can be accessed most easily by using the Example Data Sets menu. To open the menu, we click on File on the menu bar, and select the Open Example Data Sets option, as shown in Figure 2.1.

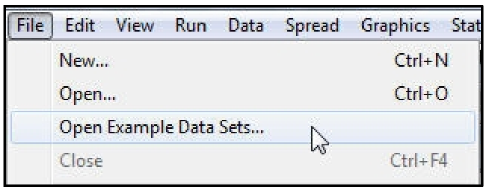
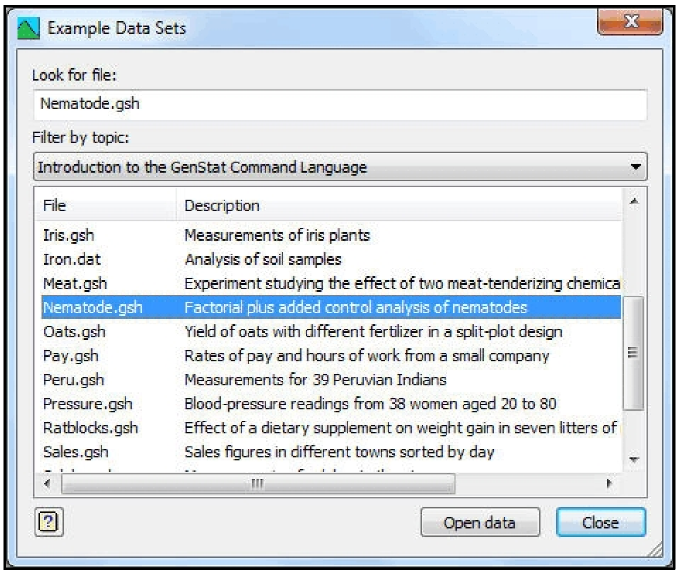


Figure 2.1

In the menu (Figure 2.2), it is easier to find the relevant file if you set the Filter by topic drop-down list to Introduction to the Genstat Command Language. We can then select the file, `Nematode.gsh`, and click on the Open data button.

Alternatively, we can click on the Open option of the File menu on the menu bar, which opens a standard Windows menu to select and open a file. (Exercise: try this and see what happens.)



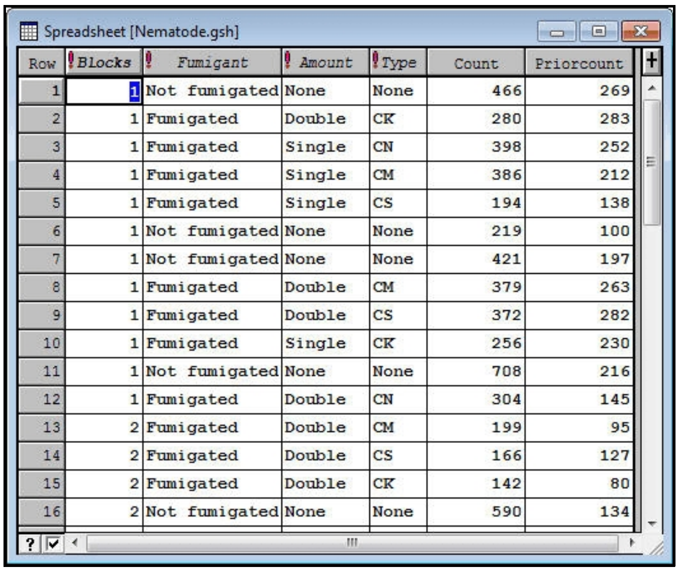
Look for file: Nematode.gsh

Filter by topic: Introduction to the GenStat Command Language

File	Description
Iris.gsh	Measurements of iris plants
Iron.dat	Analysis of soil samples
Meat.gsh	Experiment studying the effect of two meat-tenderizing chemica
Nematode.gsh	Factorial plus added control analysis of nematodes
Oats.gsh	Yield of oats with different fertilizer in a split-plot design
Pay.gsh	Rates of pay and hours of work from a small company
Peru.gsh	Measurements for 39 Peruvian Indians
Pressure.gsh	Blood-pressure readings from 38 women aged 20 to 80
Ratblocks.gsh	Effect of a dietary supplement on weight gain in seven litters of
Sales.gsh	Sales figures in different towns sorted by day

Figure 2.2

Whichever method we use to open the file, it will be displayed within the client as a Genstat spreadsheet, as shown in Figure 2.3.



Row	! *Blocks*	! *Fumigant*	! *Amount*	! *Type*	Count	Priorcount
1	1	Not fumigated	None	None	466	269
2	1	Fumigated	Double	CK	280	283
3	1	Fumigated	Single	CN	398	252
4	1	Fumigated	Single	CM	386	212
5	1	Fumigated	Single	CS	194	138
6	1	Not fumigated	None	None	219	100
7	1	Not fumigated	None	None	421	197
8	1	Fumigated	Double	CM	379	263
9	1	Fumigated	Double	CS	372	282
10	1	Fumigated	Single	CK	256	230
11	1	Not fumigated	None	None	708	216
12	1	Fumigated	Double	CN	304	145
13	2	Fumigated	Double	CM	199	95
14	2	Fumigated	Double	CS	166	127
15	2	Fumigated	Double	CK	142	80
16	2	Not fumigated	None	None	590	134

Figure 2.3

The data are from a randomized-block experiment that studied treatments to control nematodes. Each row of the spreadsheet has data from one of the plots of the experiment. Factors in the spreadsheet are highlighted by putting a red exclamation mark on the left-hand side of the column name, and putting the name into italics.

The factor `Blocks` defines the block to which each plot belonged. So this does not needs any labels, and no levels other than the ordinal numbers 1, 2, ...

The factor `Fumigant` indicates whether or not the plot was fumigated. So this has two labels, `'Not fumigated'` and `'Fumigated'`, but the levels are again the same as the ordinal values (1 and 2).

The factor Amount, however, has both labels and non-standard levels. To see these, we put the cursor into one of the spreadsheet cells in the Amount column, and click on the Edit levels and labels sub-option of the Factor option of the Spread menu, as shown in Figure 2.4.

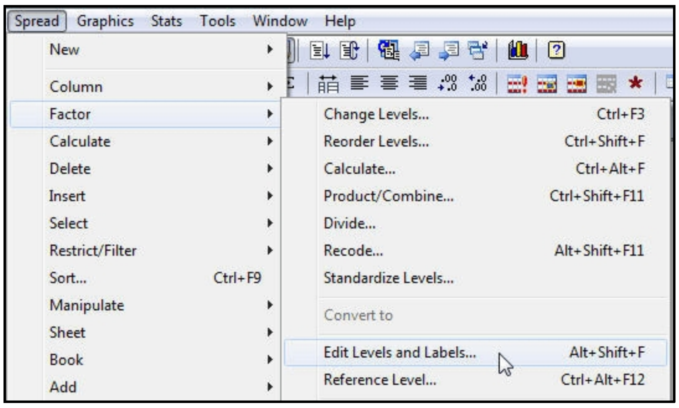
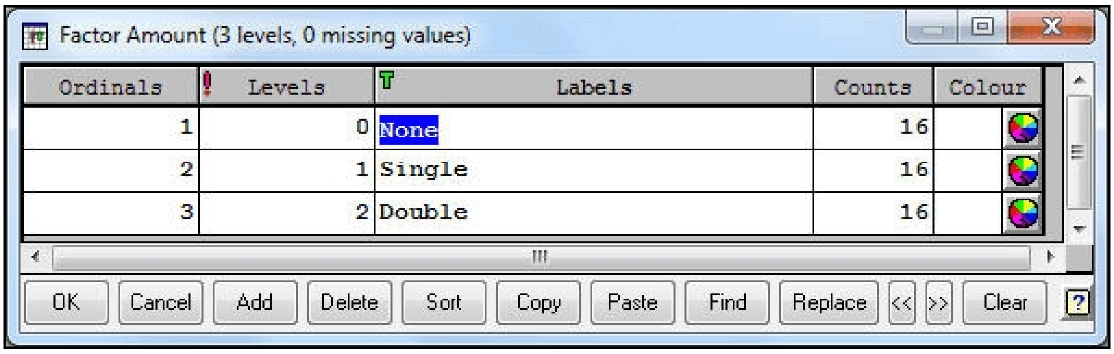


Figure 2.4


Factor Amount (3 levels, 0 missing values)

Ordinals	Levels	Labels	Counts	Colour
1	0	None	16	●
2	1	Single	16	●
3	2	Double	16	●

Figure 2.5

The levels are 0, 1 and 2, matching the descriptions given by the labels (see Figure 2.5). Genstat uses the labels in printed output, and the levels for calculations.

Variates and factors are two of Genstat's *vector* data structures. The third is the *text*, which is a list of textual strings. To show a text, we can use the Change Sheet or Column menu to convert Fumigant into a text.

To open the menu, we click on the Convert sub-option of the Column option of the Spread menu, as shown in Figure 2.6.

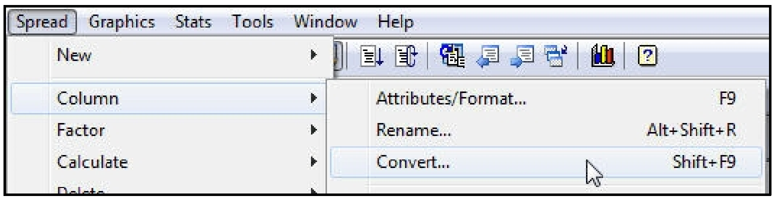


Figure 2.6

We then change the Column Type of Fumigant to Text, and click on OK, as shown in Figure 2.7.

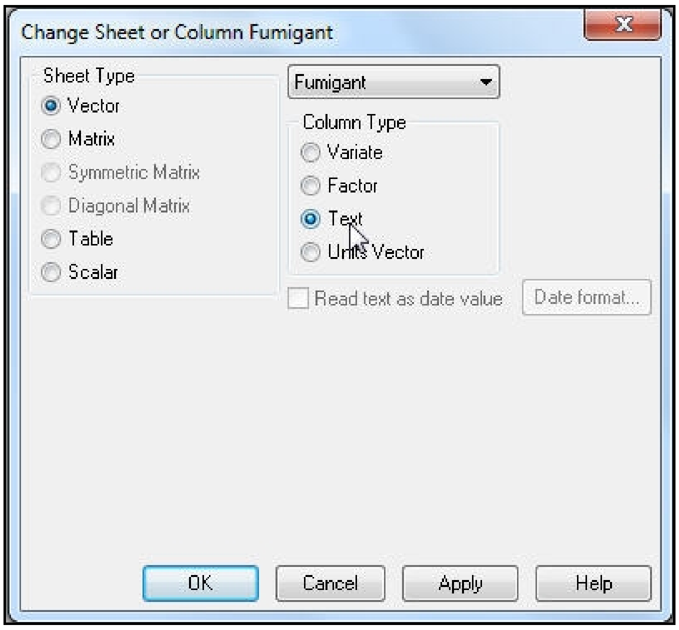
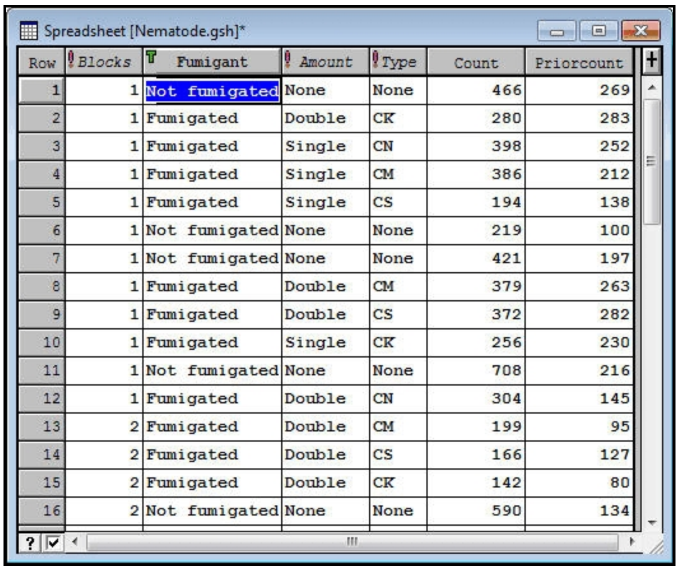


Figure 2.7

Text columns have the letter T shown in green on the left-hand side of the column name (see Figure 2.8).



Row	Blocks	Fumigant	Amount	Type	Count	Priorcount
1	1	Not fumigated	None	None	466	269
2	1	Fumigated	Double	CK	280	283
3	1	Fumigated	Single	CN	398	252
4	1	Fumigated	Single	CM	386	212
5	1	Fumigated	Single	CS	194	138
6	1	Not fumigated	None	None	219	100
7	1	Not fumigated	None	None	421	197
8	1	Fumigated	Double	CM	379	263
9	1	Fumigated	Double	CS	372	282
10	1	Fumigated	Single	CK	256	230
11	1	Not fumigated	None	None	708	216
12	1	Fumigated	Double	CN	304	145
13	2	Fumigated	Double	CM	199	95
14	2	Fumigated	Double	CS	166	127
15	2	Fumigated	Double	CK	142	80
16	2	Not fumigated	None	None	590	134

Figure 2.8

2.2 Practical

Open the spreadsheet file Iris.gsh using either of the methods described in Section 2.1.

Use the LIST directive to display the names of all the data structures that have been loaded into the Genstat server.

Look at the help page for LIST, and find out how to display only the identifiers of the variates that are in store. (Hint: put the cursor into LIST, and press the F1 key to open the help at that page.)

2.3 Forming factors from variates or texts

We can use the Change Sheet or Column menu (Figure 2.7) to convert variates or texts into factors. Alternatively, you can form a new factor with the Form Groups menu, which can be opened by clicking on the Form Groups (Factors) option of the Data menu on the menu bar. This menu uses the GROUPS directive.

In the simplest form of GROUPS, you specify the identifier of the variate or text using the first parameter, DATA, and the identifier for the new factor using the FACTOR parameter. GROUPS then forms a factor with a level for every distinct value of the variate or text. So, we could form a new factor Fumfac from the text Fumigant, with the command

```
GROUPS Fumigant; FACTOR=Fumfac
```

You can set option REDEFINE=yes if you want to change the variate or text itself to become a factor (any setting of the FACTOR parameter is then ignored). so we could convert Fumigant back into a factor with the command

```
GROUPS [REDEFINE=yes] Fumigant
```

Alternatively, you can divide the values of the variate or text into groups to be represented by the factor. You can use the LIMITS option to specify the range of values for each group. The limits vector is a text or a variate, depending whether the factor is being defined from a variate or a text; its values specify boundaries for the ranges. The BOUNDARIES option controls whether these are regarded as upper or lower boundaries; by default BOUNDARIES=lower. You can also ask GROUPS itself to set limits that will partition the units into groups of nearly equal size. You should then specify the NGROUPS option and leave the LIMITS parameter unset. (If you give both LIMITS and NGROUPS, then NGROUPS is ignored.)

If you are defining a factor from a variate VECTOR, the LMETHOD option controls how the levels vector is formed, with the following settings:

median	forms the levels from the median of the units in each group (default);
minimum	forms them from the minimum value in each group;
maximum	form them from the maximum value;
limit	uses the corresponding value from the LIMITS variate, if available, otherwise it takes the minimum value if BOUNDARIES=lower, or takes the maximum value if BOUNDARIES=upper;
given	uses the values supplied (in a variate) by the LEVELS parameter.

With any of the settings median, minumum, maximum or limit, you can use the LEVELS parameter to specify a variate to store the levels that are produced; this can be done even if no factor is being formed, that is if no identifier is supplied for the factor by the FACTOR list. Finally, if you set LMETHOD=*, no levels are formed and any existing levels of the factor will be retained if they are still appropriate; otherwise the levels will be the integers 1 upwards. With any of these settings, you can use the LABELS parameter to specify labels for the factor.

Similar rules apply if you have a text VECTOR except that LMETHOD then governs how the labels are defined for the factor, and LEVELS can be used to specify its levels. The CASE option controls whether the case of the letters in the text strings is important. So, for example, if you set CASE=ignored the strings 'April' and 'april' will be put into the same group. With the default, CASE=significant, they would form different groups.

When the levels are formed from a LIMITS variate, there will be one group with no corresponding limit. If BOUNDARIES=upper, the extra group is above the final limit. The level assigned to that group is then the value that is the same distance above the final limit as the distance between the final limit and the last but one limit. If BOUNDARIES=lower, the extra group is below the first limit, and its level is given the value that is the same distance below the first limit as the distance between the first and second limits. The situation is similar with a LIMITS text, but the label for the extra group is always the single-character string '-'. If you would prefer to have an exact correspondence between the level and the limits, you can set option OMITUNBOUNDED=yes to omit the "unbounded" extra group. Any units beyond the final upper limit, or below the initial lower limit, are then given missing values.

The LDIRECTION option controls the ordering of the levels (for a variate VECTOR) or the labels (for a text VECTOR) when LMETHOD is set to median, minimum or maximum. By default, they are sorted into ascending order, but you can set LDIRECTION=given to take them in the order in which they occur in the VECTOR. This may be useful, for example, if a text vector contains the names of days or of months in calendar order.

You can set the DECIMALS option to request that the values of a variate VECTOR be rounded to a particular number of decimal places before the groups are formed: for example DECIMALS=0 would round each value to the nearest integer.

2.4 Other data structures

There are many other data structures available within Genstat, each with appropriate attributes. A single numerical value is stored within a *scalar*. A two dimensional array of data is contained in a *matrix*, and the two specialized forms of matrices (*symmetric* or *diagonal*) can also be used. Numerical results of cross tabulations or analyses are stored in *tables* that are indexed by a number of classifying factors. *Pointers* store references to (i.e. "point" to) sets of other data structures. A *dummy* stores a reference to a single data structure. *Expressions* define numerical calculations, and *formulae* define statistical models.

Full details about all of Genstat's data structures are given in Chapter 2 of the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*. To open this within Genstat, click on the Syntax and Data Management sub-option of the Genstat Guides

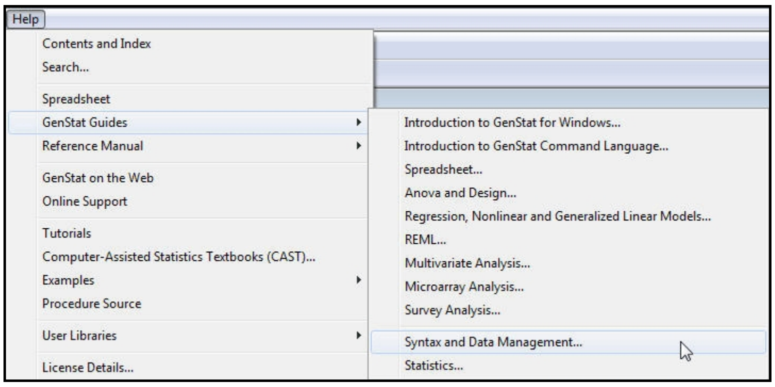


Figure 2.9

option of the Help menu on the menu bar, as shown in Figure 2.9.

The Guide explains the syntax of the directives that "declare" (i.e. define) the various types of data structure, and has example programs to show their use. They can be accessed using one of the examples menus, opened by clicking on the Syntax and Data Management sub-option of the Examples option of the Help menu on the menu bar (Figure 2.10).

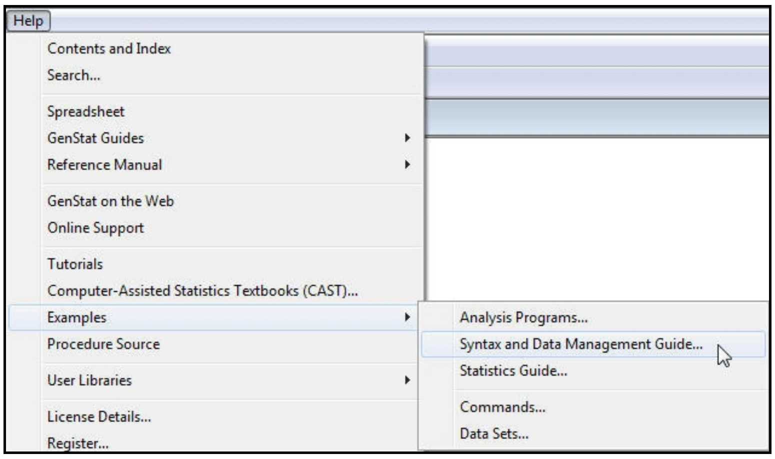
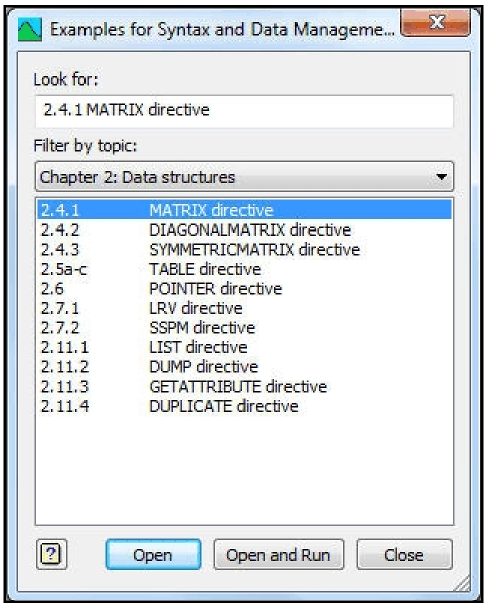


Figure 2.10

You can filter by chapter to make it easier to find the example you want. You can then click on Open to open it in a text window, or Open and Run to open it and then run it (see Figure 2.11).



2.4.1	MATRIX directive
2.4.2	DIAGONALMATRIX directive
2.4.3	SYMMETRICMATRIX directive
2.5a-c	TABLE directive
2.6	POINTER directive
2.7.1	LRV directive
2.7.2	SSPM directive
2.11.1	LIST directive
2.11.2	DUMP directive
2.11.3	GETATTRIBUTE directive
2.11.4	DUPLICATE directive

Figure 2.11

2.5 Practical

Open and run Example 2.4.1 from the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*. Notice that you can define the numbers of rows or columns using a vector (like a variate or a text), or using a pointer, instead of giving a scalar integer. If you do this, the values of the vector or pointer are used as labels when the matrix is printed.

3 Syntax

In this chapter you will learn the details of the Genstat syntax, including:

- Genstat commands, and the roles of their options and parameters;
- how names of directives, procedures, options and parameters can be shortened;
- how to repeat a command;
- how to use the substitution symbol # to insert the values of a data structure into a command;
- how to use progressions, pre- and post-multipliers to specify lists more efficiently;
- suffixed identifiers and pointers;
- unnamed data structures;
- the model formulae that are used to define statistical models.

3.1 Syntax of commands

Genstat commands, or *statements* as we prefer to call them, all have the form:

statement-name [*option-settings*] *parameter-settings* :

The *statement-name* gives the name of the directive or procedure that is to be used, and the terminating colon (`:`) can be omitted if the statement is at the end of a line. Conversely, if you want to continue a statement onto the next line, you can end the line with a continuation symbol `\`.

Parameters specify *parallel* lists of arguments for the directive or procedure. For example

```
PRINT STRUCTURE=name,pay,hours,rate; DECIMALS=0,2,0,2
```

would print `name` and `hours` with no decimal places, and `pay` and `rate` with two. The list for the first parameter of the directive or procedure must be the longest; for `PRINT` this is the parameter `STRUCTURE`. Other parameters provide ancillary information and they will be *recycled* if they are shorter than the first. So, for example, you could write just

```
PRINT STRUCTURE=name,pay,hours,rate; DECIMALS=0,2
```

A warning is printed if any of the other parameters is longer than the first.

Options specify settings that apply to all the (parallel) sets of parameters. For example, if you were to put

```
PRINT [CHANNEL=2] STRUCTURE=name,pay,hours,rate; \
  DECIMALS=0,2
```

then `name`, `pay`, `hours`, and `rate` would all be printed to the output file on channel 2 (with their attendant numbers of decimal places). Most options have *default* values, chosen to be those most often required, and so usually they need not be specified. For example, here are all the options of `PRINT`.

`CHANNEL` = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
`SERIAL` = *string token*	Whether structures are to be printed in serial order, i.e. all values of the first structure, then all of the second, and so on (`yes`, `no`); default `no`, i.e. values in parallel

`IPRINT` = *string tokens*	What identifier and/or text to print for the structure (`identifier`, `extra`, `associatedidentifier`), for a table `associatedidentifier` prints the identifier of the variate from which the table was formed (e.g. by `TABULATE`), `IPRINT=*` suppresses the identifier altogether; default `iden`
`RLPRINT` = *string tokens*	What row labels to print (`labels`, `integers`, `identifiers`), `RLPRINT=*` suppresses row labels altogether; default `labe`, `iden`
`CLPRINT` = *string tokens*	What column labels to print (`labels`, `integers`, `identifiers`), `CLPRINT=*` suppresses column labels altogether; default `labe`, iden
`RLWIDTH` = *scalar*	Field width for row labels; default 13
`INDENTATION` = *scalar*	Number of spaces to leave before the first character in the line; default 0
`WIDTH` = *scalar*	Last allowed position for characters in the line; default width of current output file
`SQUASH` = *string token*	Whether to omit blank lines in the layout of values (`yes`, `no`); default `no`
`MISSING` = *text*	What to print for missing value; default `'*'`
`ORIENTATION` = *string token*	How to print vectors or pointers (`down`, `across`); default `down`, i.e. down the page
`ACROSS` = *scalar* or *factors*	Number of factors or list of factors to be printed across the page when printing tables; default for a table with two or more classifying factors prints the final factor in the classifying set and the notional factor indexing a parallel list of tables across the page, for a one-way table only the notional factor is printed across the page
`DOWN` = *scalar* or *factors*	Number of factors or list of factors to be printed down the page when printing tables; default is to print all other factors down the page
`WAFER` = *scalar* or *factors*	Number of factors or list of factors to classify the separate "wafers" (or slices) used to print the tables; default 0
`PUNKNOWN` = *string token*	When to print unknown cells of tables (`present`, `always`, `zero`, `missing`, `never`); default `pres`
`UNFORMATTED` = *string token*	Whether file is unformatted (`yes`, `no`); default `no`
`REWIND` = *string token*	Whether to rewind unformatted file before printing (`yes`, `no`); default `no`
`WRAP` = *string token*	Whether to wrap output that is too long for one line onto subsequent lines, rather than putting it into a subsequent "block" (`yes`, `no`); default `no`
`STYLE` = *string token*	Style to use for an output file (`plaintext`, `formatted`); default `*` uses the current style of the channel
`PMARGIN` = *string tokens*	Which margins to print for tables (`full`, `columns`, `rows`, `wafers`); default `full`
`OMITMISSINGROWS` = *string tokens*	Whether to omit rows of tables that contain only missing values (`yes`, `no`); default `no`
`VSPECIAL` = *scalar* or *variate*	Special values to be modified in the output
`TSPECIAL` = *text*	Strings to be used for the special values; must be set if

VSPECIAL is set

As you have seen from the examples in Section 1.3, most of these do not usually need to be specified. However, they provide considerable flexibility of output when you want it, particularly for printing multi-way structures such as tables. Some parameters also have defaults. Here are the parameters of PRINT.

STRUCTURE = *identifiers*	Structures to be printed
FIELDWIDTH = *scalars*	Field width in which to print the values of each structure (a negative value -*n* prints numbers in E-format in width *n*); if omitted, a default is determined (for numbers, this is usually 12; for text, the width is one more character than the longest line)
DECIMALS = *structures*	Number of decimal places for numerical data structures, a scalar if the same number of decimals is to be used for all values of the structure, or a data structure of the same type and size to use different numbers of decimals for each value; if omitted or set to a missing value, a default is determined which prints the mean absolute value to 4 significant figures
CHARACTERS = *scalars*	Number of characters to print in strings
SKIP = *scalars* or *variates*	Number of spaces to leave before each value of a structure (* means newline before structure)
FREPRESENTATION = *string tokens*	How to represent factor values (labels, levels, ordinals); default is to use labels if available, otherwise levels
JUSTIFICATION = *string tokens*	How to position values within the field (right, left, center, centre); if omitted, right is assumed
MNAME = *string tokens*	Name to print for table margins (margin, total, nobservd, mean, minimum, maximum, variance, count, median, quantile); if omitted, "Margin" is printed
DREPRESENTATION = *scalars* or *texts*	Format to use for dates and times (stored in numerical structures)
HEADING = *texts*	Heading to be used for vectors printed in columns down the page; default is to use the information requested by the IPRINT option
TLABELS = *texts*	If this is specified for a table STRUCTURE, the values of the table are interpreted as references to lines within the TLABELS text that are to be printed instead of the values of the table itself

Again, we have been able to obtain very acceptable output in previous examples using the default settings, such as 12 for the FIELDWIDTH for variates, and right for JUSTIFICATION.

The settings of options and parameters are either lists, expressions, or formulae, and each setting should be separated from the next (if any) by a semi-colon (`;`). A list is a sequence of items, each separated from the next by a comma. *Expressions* are used to define calculations, as discussed in Chapter 5, and *formulae* to define statistical models, as discussed in Section 3.9.

Lists may be of numbers, as with the `VALUES` option in

```
VARIATE [VALUES=0,5,14,2.3E1,3,8,0,2,8] rain
```

Numbers in Genstat can be represented in either ordinary or "scientific" format.

There are also lists of textual strings, which are used for one of two purposes. Some options and parameters expect you to specify *string tokens* chosen from a defined list of possibilities: for example, the `SERIAL` option can be set to either `yes` or `no`. These options and parameters all have default settings. You can use the special character hash (`#`) to represent the default of an option if you want to add additional settings. So, we could put

```
RLPRINT=#,integers
```

instead of

```
RLPRINT=labels,integers,identifiers
```

as the default is

```
RLPRINT=labels,identifiers
```

String tokens can always be abbreviated to four characters. So, to save space, you will see that we only specify four for the defaults in the syntax definitions. (The full rules for abbreviations like these are given in Section 3.3.)

Other options and parameters, like the `VALUES` option of the `TEXT` directive, allow you to specify arbitrary strings: for example

```
TEXT [VALUES='Last Sunday',Monday,Tuesday,Wednesday,\
  Thursday,Friday,Saturday,Sunday] day
```

In general, these arbitrary strings are enclosed in single quotes (`'`). However, in a string list, the quotes can be omitted provided the string starts with a letter, and then contains only letters and/or digits. A *letter* in Genstat is any of the capital letters `A` to `Z`, or the lower case letters `a` to `z`, or the underscore character (_), or percent (`%`), while a *digit* is one of the numerical characters `0` up to `9`. (Note, though, the string tokens in Genstat are defined so that they always start with a letter and contain only letters and/or digits, so you will never need to put them inside quotes.)

If you have a line break inside a quoted string, Genstat terminates the current string, and then begins a new one on the next line. So

```
'These are the last
words I have to say'
```

is the same as

```
'These are the last','words I have to say'
```

If you want to continue a string on the next line, you must give a continuation character (`\`).

You can include any characters inside a quoted string but, if you want to include a single quote (`'`), a double quote (`"`) or a continuation character (`\`), you must put them twice. Otherwise a single quote on its own would terminate the string, a double quote would begin a comment (so all the following characters would be ignored until the next double quote), and a continuation character would continue the string onto the next line.

Finally, there are lists of identifiers, as in the `STRUCTURE` parameter of `PRINT`: for example

```
PRINT STRUCTURE=name,pay,hours,rate
```

An identifier is the name given to a data structure. It must start with a letter and then contain only letters or digits. Only 32 characters are stored, so `satotal_software_sales_in_January_2008` will not be distinguished from `total_software_sales_in_January_2009` (both will be stored as `total_software_sales_in_January_`). However, upper case is distinguished from lower case, so `SALES` and `Sales` are *not* the same. (It is possible to change this using the `SET` directive, but this is not customary. You can also use `SET` to request that Genstat stores and checks only the first eight characters of identifiers, as was the case in the Fourth and earlier Editions.) Identifiers can also have suffixes, enclosed in square brackets: for example `x[2]` or `employee['grade'].` These are explained in Section 3.7.

Any list may contain missing values, each represented by an asterisk (`*`). These represent unknown (or unset) information.

You can put comments anywhere in a statement, or between two statements. Comments begin and end with the double-quote character (`"`); between the quotes you can type anything you like.

3.2 Making lists more compact

The values of any data structure can be substituted into a list of the appropriate type, using the *substitution symbol* hash (`#`). So, for example, values of variates can be substituted into number lists:

```
VARIATE [VALUES=1,2,3,4] x
VARIATE [VALUES=#x,#x] xx
```

gives `xx` eight values (1,2,3,4,1,2,3,4).

Similarly, values of texts can be substituted into lists of strings. For example, in

```
TEXT [VALUES=data,errors] pde
READ [PRINT=#pde] day,rain
```

the `PRINT` option of `READ` is given the setting `data,errors`.

Lists of numbers that increase or decrease in a regular way can be represented conveniently in Genstat as *progressions*. These have the general form

x, y `...` z

where x is the first number, y the second number, and z the final limit. You can put spaces anywhere within this construct *except* within the sequence of three dots (`...`). The progression generates a sequence of numbers:

$x,\ \ x + s,\ \ x + 2s$ as far as $x + ks$

where s is the difference between y and x (so $x + s = y$) and k is the largest integer such that $x + ks$ does not go beyond z. The step can be either positive or negative and need not be an integer. If the step is 1 or -1, the second number y can be omitted. For example

```
1...5                     =   1,2,3,4,5
5...1                     =   5,4,3,2,1
2,4...10                  =   2,4,6,8,10
2,(4...10)                =   2,4,5,6,7,8,9,10
```

Notice that the progression in the final example must be placed in brackets to make it clear that the second number has been omitted.

Lists of numbers, strings, or identifiers that are repeated in a regular way can be compacted using *multipliers*.

A *pre-multiplier* precedes a bracketed list and repeats the individual elements of the list, in turn, a specified number of times. For example

```
3(1,2)                    =   1,1,1,2,2,2
```

Post-multipliers come after a bracketed list of numbers and repeat the entire list, en bloc, the specified number of times. For example

```
(day,temp,rain)2          =   day,temp,rain,day,temp,rain
```

They may be combined. For example

```
2((1...3)2,4)             =   1,1,2,2,3,3,1,1,2,2,3,3,4,4
```

You can use scalars as pre-multipliers or post-multipliers but you must also use a substitution symbol

```
SCALAR [VALUE=3] nplot
FACTOR [LEVELS=4; VALUES=#nplot(1),(2,3,4)#nplot] block
```

gives `block` the values

```
1,1,1, 2,3,4, 2,3,4, 2,3,4
```

3.3 Practical

Use the `VARIATE` directive to define variates `a` - `g` holding the following sets of numbers:

`a`	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
`b`	12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
`c`	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23
`d`	2.2, 3.4, 4.6, 5.8, 7, 8.2, 9.4, 10.6, 11.8, 13, 14.2, 15.4
`e`	1.5, 1.5, 9, 9, 11, 11, 3.3, 3.3, 6, 6, 2, 2
`f`	1.5, 9, 11, 3.3, 6, 2, 1.5, 9, 11, 3.3, 6, 2
`g`	1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4

Print their values in parallel, in the Output window, with enough decimal places to display all the significant figures of each one.

3.4 Abbreviation rules

Names of directives, options, parameters, and functions (see Section 5.1) are known as *system words*, and these can always be abbreviated to four characters. If more than the minimum number of characters is given for any system word, they will be checked as far as the 32nd; characters from the 33rd onwards are ignored. Names of procedures in the standard Genstat library are defined so that they too can be abbreviated to four characters. Names of user-defined procedures are required to differ only in the first eight characters (although Genstat does give a warning if you define a name whose first four characters are the same as an existing procedure). So, unless you are using procedures only from the standard Genstat library, it may be safer to give eight characters for their names.

Option and parameter names can usually be abbreviated even further. For every directive or procedure, an implicit order is defined for its options and for its procedures. For PRINT, as shown above, the ordering of the parameters is **S**TRUCTURE, **F**IELDWIDTH, **D**ECIMALS, **C**HARACTERS, **SK**IP, **FR**EPRESENTATION, **J**USTIFICATION, **M**NAME, **DR**EPRESENTATION, **H**EADING and **T**LABELS. The rule is that you need specify only sufficient letters to distinguish each parameter from the parameters that occur before it in this (implicit) list. Above we have printed the minimum form for each one in bold letters; so for example you can abbreviate FIELDWIDTH to F, as this is preceded only by STRUCTURE, but SKIP requires the two letters SK. However, as already mentioned, if you are uncertain about the ordering for a particular directive, it is always sufficient to specify four characters.

This abbreviation rule also applies to string tokens. Thus, for example,

```
READ [PRINT=data,errors] day
```

can be written as

```
READ [PRINT=d,e] day
```

(The READ directive, which reads values into data structures, is described in Section 4.1.)

An option or parameter name may be omitted altogether, along with its accompanying equals sign, if Genstat can deduce it from the position of the setting within the statement. In the statement

```
PRINT [CHANNEL=2; SERIAL=yes] \
  STRUCTURE=name,pay,hours,rate; DECIMALS=0,2
```

you can leave out CHANNEL= as CHANNEL is the first option of PRINT: unless you say otherwise (by giving the option name explicitly), Genstat assumes that the first option setting in a statement is for the first option in the implicit ordering for the directive (or procedure). Similarly, as STRUCTURE is the first parameter of PRINT, you can also omit STRUCTURE=, to obtain

```
PRINT [2; SERIAL=yes] name,pay,hours,rate; DECIMALS=0,2
```

For subsequent options (and parameters), Genstat looks to see which option (or parameter) comes after the current one in the implicit ordering. Thus, after CHANNEL Genstat would expect by default to find SERIAL; as our statement also has SERIAL straight after CHANNEL, this can be omitted too.

```
PRINT [2; yes] name,pay,hours,rate; DECIMALS=0,2
```

You can include *null* settings in a statement by typing nothing (other than spaces or comments or continuation symbols) between two semi-colons. If you insert a null setting for FIELDWIDTH, Genstat will then be able to deduce that our third parameter setting is for DECIMALS and so you can simply put

```
PRINT [2; yes] name,pay,hours,rate; ; 0,2
```

Again this does require you to know the implicit ordering for the directive or procedure, but it can save a great deal of typing as you get to know Genstat better.

3.5 Repeating a statement

The *repetition symbol* `&` provides a very convenient way of repeating a statement. It terminates the previous statement, if necessary, and then repeats the name of the statement together with any options that were set. So, for example, you could type the statements

```
READ [CHANNEL=2] year  :  READ [CHANNEL=2] day,temp
```

as

```
READ [CHANNEL=2] year  &  day,temp
```

You can also modify the options by including further settings after `&`. Thus

```
READ [CHANNEL=2] year
READ [CHANNEL=2; SERIAL=yes] day,temp
READ [CHANNEL=3; SERIAL=yes] sunshine,windspeed
```

can be simplified to

```
READ [CHANNEL=2] year
& [SERIAL=yes] day,temp
& [CHANNEL=3] sunshine,windspeed
```

3.6 Practical

Below we show a rather verbose Genstat program, stored as `Shop.gen`. Run the program to see what output it gives.

```
TEXT [NVALUES=11] branch
VARIATE [NVALUES=11] sales01,sales02
VARIATE [NVALUES=11] frontage,depth
READ [PRINT=data,errors,summary] branch, sales01, sales02,\
     frontage,   depth
Ashford         4741100  496700  25  33
Bradford        3386800  350100  21  32
Chelmsford       645800  395200  15  22
Dartford        2381200  298900  12  28
Fordingbridge   1379600  412000  12  25
Guildford       2727300  234700  16  26
Hereford        2993300  358500  14  32
'Milford Haven' 3409000  460600  18  24
Oxford          4752400  439100  15  30
Stafford        4117400  473700  16  28
Twyford          942500  294900  12  16 :
CALCULATE [PRINT=*] sales01 = sales01/100
CALCULATE [PRINT=*] sales02 = sales02/100
CALCULATE [PRINT=summary] allsales = sales01 + sales02
CALCULATE [PRINT=summary] sale_pm2 = \
  allsales / 2  / (frontage * depth)
PRINT [SERIAL=no] branch,frontage, depth,allsales, sale_pm2;\
   DECIMALS=0,0,0,0,2; JUSTIFIC=left,right,right,right,right
```

Modify the program to become as compact as possible, by using abbreviations but not by removing spaces and new lines. Run it again to check that it still works.

3.7 Suffixed identifiers and pointers

Lists of data structures can be stored in a Genstat *pointer* structure to save having to type the list in full every time it is used. For example

```
POINTER [VALUES=rain,temp,windspeed] vars
VARIATE #vars
READ [CHANNEL=2] #vars
PRINT #vars; DECIMALS=2,1,2
```

defines `rain`, `temp`, and `windspeed` to be variates, and then reads and prints their values. When none of the structures in the list is itself a pointer, the substitution symbol (`#`) works in the same way as with variates and texts. If, however, there are pointers in the list, they too are substituted, as are any pointers to which they point.

You can also refer to the elements of pointers using *suffixes*. For example, you can refer to `rain` either using its own identifier, or as the first element of `vars` by using the *suffix* `[1]`: so

`vars[1]`	is	`rain`
`vars[2]`	is	`temp`
`vars[3]`	is	`windspeed`

Furthermore, you can put a list within the brackets:

`vars[3,1]`	is	`windspeed,rain`.

Also, you can put a null list to mean all the available suffixes of the pointer:

`vars[]`	is	`rain,temp,windspeed`.

Identifiers like `vars[1]`, `vars[2]`, and `vars[3]` are called *suffixed* identifiers and, in fact, you can use these even without defining the identifier of the pointer explicitly. Whenever a suffixed identifier is used, Genstat automatically sets up a pointer for the unsuffixed part of the identifier if it does not already exist. Furthermore the pointer will usually be extended automatically (whether it has been set up by you or by Genstat) if you later use a new suffix, like `vars[93]` for example. If, however, you do not want this to happen, you should define the pointer explicitly and set option `FIXNVALUES=yes`. For example

```
POINTER [VALUES=length,width,height; FIXNVALUES=yes]\
        dimensions
```

The `SUFFIXES` option of the `POINTER` directive allows you to specify the required suffixes for pointers that are defined explicitly. Notice that the suffixes do not need to be a contiguous list, nor need they run from one upwards. For example

```
VARIATE [VALUES=1990,1991,1992,1993] suffs
POINTER [NVALUES=4; SUFFIXES=suffs] profit
```

defines `profit` to be a pointer of length four, with suffixes 1990 to 1993.

You could actually omit the `NVALUES` option here as Genstat can determine the length of the pointer by counting the number of values. However, by supplying a text instead of a scalar for `NVALUES` you can define *labels* for the suffixes of the pointer. The length of the text defines the number of values of the pointer, and its values give the labels. For example

```
TEXT [VALUES=name,salary,grade] labs
```

```
POINTER [NVALUES=labs] employee
```

would allow you to refer to `employee['name']`, `employee['salary']`, and so on. Lower and upper-case labels are distinguished, unless you set option `CASE=ignored` in the `POINTER` statement. You can also set option `ABBREVIATE=yes`, to allow the labels to be abbreviated. (By default, `CASE=significant` and `ABBREVIATE=no`.) So, if you had specified

```
POINTER [NVALUES=labs; CASE=ignored; ABBREVIATE=yes]\
        employee
```

you would be able to refer to `employee['name']`, for example, as `employee['n']` or `employee['Name']`.

3.8 Unnamed data structures

It can be wasteful and tedious to set up a structure explicitly when it is needed only once in a program. So Genstat allows you to define an *unnamed structure* instead. Some particularly useful types of unnamed structures are described below.

The unnamed scalar is just a number. Whenever Genstat expects the identifier of a scalar, a number may be given instead. In fact, you have been using this type of unnamed structure already: the statement

```
VARIATE [NVALUES=10] x
```

is equivalent to

```
SCALAR [VALUE=10] n
VARIATE [NVALUES=n] x
```

However, the converse, that you can use a scalar instead of a number, is not always true. The exception is with pre-multipliers and post-multipliers when, as mentioned in Section 3.7, the scalar must be preceded by the substitution symbol (`#`).

The other forms all have a common style: they start with an exclamation mark, then a type code, and then a list enclosed in round brackets. An unnamed variate takes the form

```
!V( list of numbers )
```

where the letter `V` can be in either upper or lower case, or equivalently

```
!( list of numbers )
```

For example

```
GRAPH rain; !(1...8)
```

plots `rain` against day number, 1 to 8.

The unnamed text takes one of two forms. If the text has a single value, then the value can be placed within quotes (`'`) and used instead of the identifier of the text. For example:

```
GRAPH [TITLE='Rainfall during my holiday'] rain; !(1...8)
```

If there are several values, the form is

```
!T( list of strings )
```

where the letter `T` can be in either upper or lower case. For example:

```
PRINT !t(Sat,Sun,Mon,Tues,Wed,Thur,Fri,Sat),rain
```

The unnamed pointer has the form

```
!P( list of identifiers )
```

where, again, the letter `P` can be in either upper or lower case.

There are also unnamed expressions (type code E), and unnamed formulae (type code F). Finally, type code S provides another way to specify an unnamed scalar.

Unnamed structures are particularly useful when assigning values to several structures. The directives that are used to declare data structures all have a parameter as well as an option to specify the values of the structures that are defined. (This is one of the few places in the Genstat language where a directive has both an option and a parameter with the same name.) So you can use the `VALUES` parameter of `VARIATE` to specify different sets of values for `x` and `y`, as follows:

```
VARIATE x, y; VALUES=!(1,2,3), !(4,5,6)
```

`x` now contains the values 1, 2, and 3, and `y` contains 4, 5, and 6.

3.9 Practical

The first of April 2000 was a Saturday. Using unnamed structures, write a `PRINT` statement to print the day numbers in the month, with the day of the week and the week number side by side:

```
1  Saturday                               1
2  Sunday                                 2
3  Monday                                 2
```

Using the unnamed structures from your program, create a variate to hold the day numbers, a variate to store the week number and a text to store the day of the week. Create a pointer using the `POINTER` directive to point to the three data structures, and print the contents of the pointer to the Output window.

3.10 Formulae to define statistical models

Statistical models for analyses like regression, analysis of variance and the analysis of linear mixed models by `REML` are defined in Genstat by *model formulae*. Many of the menus of Genstat *for Windows* define these automatically. However, to use commands or the more advanced menus you will need to specify your own formulae.

In its simplest form, a model formula is a list of *model terms*, linked by the operator "+". Each term defines a set of parameters to be fitted in a statistical model. For example, if we have factors `N` and `S`, the formula

```
N + S
```

specifies two terms representing the main effects of the factors.

You can use commas in lists of factors or variates instead of pluses. So, for example, we could specify a multiple linear regression involving variates `X` and `Y` by either

```
X + Y
```

or

```
X, Y
```

(see Section 6.3 for examples).

Higher-order terms, like interactions, are specified as series of factors separated by dots, but their precise meaning depends on which other terms the formula contains, as we explain below.

The other operators provide ways of specifying a formula more succinctly, and of representing its structure more clearly.

The *crossing operator* `*` is used to specify factorial structures. The formula

```
N * S
```

is used to specify the two-way analysis of variance described in Section 7.1, where the factors represent amounts of nitrogen and sulphur treatments applied in a field experiment. This is expanded to become the formula

```
N + S + N.S
```

which has three terms: `N` for the nitrogen main effect, `S` for the main effect of sulphur, and `N.S` for the nitrogen by sulphur interaction. Higher-order terms like `N.S` represent all the joint effects of the factors `N` and `S` that have not been removed by earlier terms in the formula. Thus here it represents the interaction between nitrogen and sulphur as both main effects have been removed. See Section 3.1 of the *Guide to Anova and Design in Genstat* for a more detailed explanation.

The other most-commonly used operator is the *nesting operator* (`/`). This occurs most often in the block formulae that define the random terms for an analysis of variance. For example, the formula

```
block /  plot
```

is expanded to become the formula

```
block + block.plot
```

As the formula contains no "main effect" for `plot`, the term `block.plot` would represent *plot-within-block* effects (that is the differences between individual plots after removing any overall similarity between plots that belong to the same block). Section 7.4 uses this model to analyse a randomized block design.

A formula can contain more than one of these operators. The three-factor factorial model

```
A * B * C
```

becomes

```
A + B + C + A.B + A.C + B.C + A.B.C
```

and the nested structure

```
block / wplot / subplot
```

which occurs as the block model of a split-plot design (Section 7.7) becomes

```
block + block.wplot + block.wplot.subplot
```

They can also be mixed in the same formula. For example, the factorial-plus-added-control study in Section 3.5 of the *Guide to Anova and Design in Genstat* has treatment structure

```
Fumigant / ( Amount * Type )
```

which expands to

```
Fumigant + Fumigant.Amount + Fumigant.Type +
Fumigant.Amount.Type
```

In general, if `l` and `m` are two model formulae:

```
l * m   =   l + m + l.m

l / m   =   l + fac(l).m
```

(where `l.m` is the sum of all pairwise dot products of a term in `l` and a term in `m`, and `fac(l)` is the dot product of all factors in `l`). For example:

```
(A + B) * (C + D) = (A + B) + (C + D) + (A + B).(C + D)

                  = A + B + C + D + A.C + A.D + B.C + B.D

(A + B)/C = A + B + fac(A + B).C  =  A + B + A.B.C
```

The dot, plus, star and slash operators cover most situations, but there are also a few more-specialized operators. This is the full list, in order of precedence i.e. in the order in which they are evaluated if they occur together in the same formula:

(1)	`,` (comma)	used within lists of factors or variates,
(2)	`.` (dot)	defines higher-order terms,
(3)	`//` (double slash)	indicates pseudo-factor relationships (see *Guide to the Genstat Command Language, Part 2 Statistics*, Section 4.7.3),
(4)	`/` (slash)	defines nested structures,
(5)	`*` (star)	defines factorial structures,
(6)	`+` (plus)	adds terms into a formula,
	`-` (minus)	removes terms from a formula,
	`-/` (minus slash)	removes higher-order terms (e.g. `m -/ F` removes any higher-order term involving `F` from the formula `m`),
	`-*` (minus star)	removes terms and higher-order terms (e.g. `m -* F` removes `F` and any higher-order term involving `F` from the formula `m`).

Within each class, operations are done from left to right within the formula.

There are also several functions that can be used to define contrasts like polynomials. These are explained in the *Guide to the Genstat Command Language, Part 2 Statistics*, Sections 3.4 and 4.5.

4 Input and output

In this chapter you will learn about:

- the READ directive, which reads data values from lines of text into any Genstat data structure;
- the FILEREAD procedure, which reads data from a text file into variates, factors or texts;
- the SPLOAD procedure, which reads Genstat spreadsheets;
- the FSPREADSHEET procedure, which writes Genstat spreadsheets;
- the IMPORT procedure, which reads data from other spreadsheets, statistical systems, databases etc;
- the EXPORT procedure, which writes data to files for other spreadsheets, statistical systems, databases etc;
- the CAPTION directive, which provides titles for output.

4.1 Reading data from text files

Genstat can read many different types of data file. These include text, spreadsheet and database files, as well as files in the formats of many other statistical systems. In Genstat *for Windows*, data files can be read most easily by using the Open option of the File menu, as mentioned in Section 2.1. (This is the standard Windows™ approach.) Opening a spreadsheet or database file in this way loads the data into a Genstat spreadsheet within the client for viewing and/or editing. When you change the focus away from the spreadsheet window, the client passes the data across to the server by writing scripts of commands that use the READ directive. So, for example, READ was used when the data were read from the spreadsheet file Nematode.gsh in Chapter 2.

These two commands read data values into a variate Sepal_Length.

```
VARIATE [NVALUES=150] Sepal_Length
READ Sepal_Length
5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1 5.7 5.1
5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7
4.8 5.4 5.2 5.5 4.9 5.0 5.5 4.9 4.4 5.1
5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0
7.0 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2
5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6
5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7
5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5
5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7
6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2
6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0
6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2
7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9
6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9 :
```

The VARIATE command defines Sepal_Length to be a variate with 150 values. In fact this is not strictly necessary as, if you ask READ to read data into data structure that has not yet been declared, it will declare the data structure by default to be a variate, and define its length from the number of values that are read. Many Genstat directives make *default declarations* like this. However, it will always be safer to declare your data structures explicitly. Genstat can then check for mistakes. In the simplest form of READ, the data values come immediately after the `READ` statement, and are terminated by a semi-colon. The following example reads data into a 3×3 symmetric matrix `S`, and then prints it.

```
  2  SYMMETRICMATRIX [ROWS=3] S
  3  READ [PRINT=data,errors] S
  4  1.4 3.8 4.1 7.0 2.5 4.6 :
  5  PRINT S; DECIMALS=1

                              S
              1             1.4
              2             3.8            4.1
              3             7.0            2.5            4.6
                              1              2              3
```

Notice that we have set the `PRINT` option of `READ` to "echo" the lines of data values from the input, and to print details of any errors. The default for the option is `PRINT=errors,summary`, where the `summary` setting gives a summary of the data that are read. The layout of the values in the input line is completely free-format. It does not need to match the shape of the data structure.

`READ` handles many different styles and formats. It can read several structures at once, and can take data from external files. It can also read data from unformatted files and Genstat text structures. You can find full details in Subsections 3.1.2 - 3.1.13 of the *Guide to the Genstat Command Language Part 1, Syntax and Data Management*.

However, if your data are vectors (factors, variates or texts) with their values arranged in columns in an external file, it is easier to use the `FILEREAD` procedure. `FILEREAD` is used by Read Data From ASCII File menu, which is one of the sub-options of the Load option of the Data menu on the menu bar of the Genstat client. So it was used to read the iron measurements in Section 1.1. The settings that we selected in the menu in Figure 1.4 generated the command

```
FILEREAD [PRINT=summary,groups,comments,firstline;\
  NAME='C:/Program files/Gen17ed/Data/Iron.dat';\
  MISSING='*'; SEPARATOR=' '; MAXCATEGORY=15;\
  IMETHOD=read] FGROUPS=check
```

The `PRINT` option selects the output to display. Here we have asked for all the four possibilities:

- to print summaries of the values read into each data structure,
- to give details of groups (i.e. about factors and their levels),
- to display any comments in the file occurring before the data, and
- to echo the first line of data in the file.

The `NAME` option gives the name of the file. Remember that Genstat uses the character `\` as the continuation character, to continue a command onto the next line. So, if you want to include a `\` character in a string, you need to specify it twice, as explained in Section 3.1. To simplify this in strings that represent file names, Genstat allows you to use the character `/` instead of `\`. So we can put

```
'C:/Program files/Gen18ed/Data/Iron.dat'
```

instead of

```
'C:\\Program files\\Gen18ed\\Data\\Iron.dat'
```

The MISSING option specifies the character used in the file to represent missing values. In fact the character * is the default, so this option could have been omitted. Similarly, the SEPARATOR option, which specifies the character that is used to separate data values, could have been omitted, as the default is the space character.

Setting option IMETHOD=read tells FILEREAD to read the names of the data structures from the file. With the default, IMETHOD=supply, they must be specified using the IDENTIFIER parameter (which is the first parameter of the procedure). So the command would then have become

```
FILEREAD [PRINT=summary,groups,comments,firstline;\
  NAME='C:/Program files/Gen18ed/Data/Iron.dat';\
  MISSING='*'; SEPARATOR=' '; MAXCATEGORY=15]
  sample,site,FE,weight; FGROUPS=check
```

(omitting the parameter name as it is the first parameter). When the names are read from the file, the IDENTIFIER parameter is set by default to the identifiers that are found there.

The FGROUPS parameter allows structures to be turned automatically into factors. The default setting is check. If you are running Genstat interactively, FILEREAD will then prompt for a decision about any structure where the number of distinct values is less than or equal to the setting supplied by the MAXCATEGORY option. If you are running Genstat in batch, all structures with these few distinct values become factors automatically. FGROUPS can also be set to form or leave to tell Genstat explicitly whether each structure should or should not be defined automatically as a factor.

Columns that are not defined to be factors are defined as either texts or variates, according to whether FILEREAD finds a number or character string for that data structure in the first record with no missing values that it finds in the file. However, you can use the REPRESENTATION parameter to tell FILEREAD explicitly whether a data structure should contain numbers or characters (and you must always do this if there are no units without missing values).

There are several options that are not shown in our example. These are required less often, but we explain them briefly below.

The END option specifies the string that will be used to denote the end of the data. The default is the colon character (:). The end of the file will also terminate the data, so you need specify this only if you want to stop reading data before the end.

The COMMMENTSYMBOLS option can be set to a list of single characters, in quotes. If any of these characters is found at the start of a record, before any data has been read, that record will be treated as a comment. By default, the double-quote symbol is the only comment symbol, but it must appear at the start of every record to be treated as a comment.

The SKIP option allows records at the start of the file to be skipped altogether. It can be set either to the number of records to be skipped, or to a string, indicating that all records are to be skipped up to and including the first record containing that string.

The ISAVE option can be set to a pointer to store the identifiers read from the file (if IMETHOD=read) or supplied interactively (if IMETHOD=supply). Also, the IMETHOD option has an additional setting none. With this setting, the identifiers of the data structures can still be defined using the IDENTIFIER parameter. Alternatively, if the ISAVE option is set but IDENTIFIER is unset, the data structures will be set up as elements of the ISAVE pointer. If both ISAVE and IDENTIFIER are unset, FILEREAD just reports on the contents of the file.

4.2 Practical

The file Bacteria.dat in the Data folder contains the lines

```
"Counts of bacteria and crop type, Jan-Feb 2008"
 18    pea
117    pea
 21 cereal
  7    pea
176 cereal
 85 cereal
244 cereal
  4    pea
 55 cereal
  8    pea
 73 cereal
  *    pea
  3    pea
  4    pea
 40 cereal
198 cereal
123    pea
 17    pea
 74 cereal
  3    pea
  2    pea
  5    pea
  0 cereal
  4    pea
  2    pea
 10 cereal
  2 cereal
  4 cereal
  3    pea
  1    pea
  * cereal
  2 cereal
  4    pea
  1    pea
  4 cereal
 15 cereal
  1 cereal
 12 cereal
  4    pea
```

* cereal

Use FILEREAD to read the data values into a variate Counts, and a factor Crop. Produce some summary statistics for the counts in each crop.

Now look at the file Bacteri2.dat, also in the Data folder. (Hint: you can open this file in a text window in the Genstat client, by using the Open option of the File menu.) How would the FILEREAD command need to change to read the data from this file instead?

4.3 Reading data from spreadsheet files

Genstat spreadsheet files can be read very easily using the SPLOAD directive. For example, we could have read the spreadsheet file Nematode.gsh in Chapter 2 with the command

```
SPLOAD 'C:/Program files/Gen18ed/Data/Nematode.gsh'
```

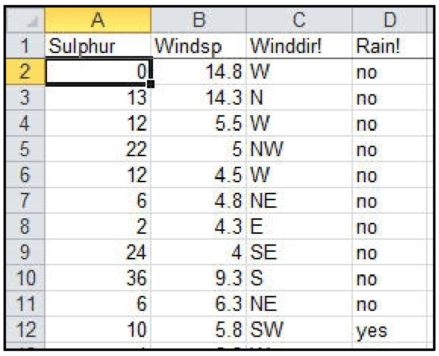
The data would then have been read straight in to the server, and no longer have been displayed in the client. This can be very useful if you have a very large spreadsheet, as it can be time consuming to take the data through the client to the server. (In fact, the client will use SPLOAD itself, instead of READ, if you select the Using fast load (Save and Close) sub-option of the Update option of the Spread menu.)

The first parameter, FILE, gives the filename. There is also a SHEETNAME parameter that can be used to specify which sheet to read from a multi-paged spreadsheet (see the *Guide to the Genstat Spreadsheet* for more information). Finally, there is an ISAVE parameter that can be set to a pointer to save the identifiers of the data structures that have been read.

You will not usually need to set any of the options; for details see the on-line help.

Other spreadsheet files can be read using IMPORT. This is a very powerful procedure, which can also, for example, read files from other statistical systems, image files and GIS files, as well as CSV (i.e. comma-delimited) files. Here we will look only at its use for reading variates, factors or texts from Excel spreadsheet files.

As with text files, it is easiest to read data from an Excel file if the data values are arranged in columns with the identifier for each column in the first row. As an example, Figure 4.1 shows part of the file Sulphur.xls in the Data folder. This has four columns, with identifiers Sulphur, Windsp, Winddir and Rain. The exclamation marks after the names of Winddir and Rain tell IMPORT that these must be formed into factors.



	A	B	C	D
1	Sulphur	Windsp	Winddir!	Rain!
2	0	14.8	W	no
3	13	14.3	N	no
4	12	5.5	W	no
5	22	5	NW	no
6	12	4.5	W	no
7	6	4.8	NE	no
8	2	4.3	E	no
9	24	4	SE	no
10	36	9.3	S	no
11	6	6.3	NE	no
12	10	5.8	SW	yes

Figure 4.1

As the file contains all the information necessary to define what data structures are to be read, the command is very simple. We need supply only the file name, using the first parameter (FILE). So the command is

```
IMPORT 'C:/Program files/Gen18ed/Data/Sulphur.xls'
```

By default IMPORT prints a "catalogue" of the data structures that have been read, shown below. You can suppress this by setting option PRINT=*.

```
Loading Spreadsheet File

Catalogue of file C:\Gen18ed\Data\Sulphur.xls

Sheet Title: Sheet1
Description: Data read from d:\Program Files Gen18ed\Data\Sulphur.xls [GENSTAT
DATA]A2:D115
Sheet Type: vector
   Index          Type     Nval   Name
       1       variate      114   Sulphur
       2       variate      114   Windsp
       3        factor      114   Winddir
       4        factor      114   Rain
```

This shows that IMPORT has read the data by first forming a temporary Genstat spreadsheet file, and then importing that into the server. You can save the Genstat spreadsheet by supplying a name (and path) for the file, using the OUTFILE option. You could also set the option METHOD=create, if you want to form the Genstat spreadsheet without then reading the data into the server.

Notice that IMPORT has automatically detected the range of cells that contain the data values. It can do this in Sulphur.xls as all the other cells are empty. In more complicated situations, you can use the CELLRANGE parameter to define the range of cells. This is specified in a string using the standard Excel conventions. In Sulphur.xls the top-left cell is A1, and the bottom-right cell is D115. So we would have put CELLRANGE='A1:D115'.

As in SPLOAD, the SHEETNAME parameter specifies which sheet to read if the file has more than one, and the ISAVE parameter can again be set to a pointer to save the identifiers of the data structures that have been read.

If the identifiers are not in the file, you must set option IMETHOD=supply, and set the COLUMN parameter to a text containing the names. You can put an exclamation mark at the end of a name, as before, if you want the column to be formed into a factor. You can also put a hash character (#) at the end if you want it to be a variate, or a dollar ($) if you want it to be a text. In fact you can supply these final characters, on their own, to force the column type even if the names are in the file.

So another way to read Sulphur.xls would be with the commands

```
TEXT   [VALUES=Sulphur,Windsp,'Winddir!','Rain!'] Cols
IMPORT 'C:/Program files/Gen18ed/Data/Sulphur.xls';\
       CELLRANGE='A2:D115'; COLUMNS=Cols
```

Details of the other options and parameters of IMPORT are in the on-line help.

4.4 Practical

The file Traffic.xls is an Excel data file with one worksheet called counts storing one set of data in the area B3:D43. Use the IMPORT procedure to read the data into Genstat, converting day and month to factors. Examine the distribution of the counts using a histogram. (Hint: use the Summarize Contents of Variates menu).

4.5 Exporting data to files

Genstat spreadsheet files can be created using the `FSPREADSHEET` procedure. For example, we could create a spreadsheet file `Ispread.gsh`, containing the iron measurements from Section 4.1, by using the command

```
    FSPREADSHEET [OUTFILE='Ispread.gsh'] sample,site,FE,weight
```

The `OUTFILE` option specifies the name (and path) of the file. If this is omitted when Genstat is running interactively, the spreadsheet is opened as a window in the client. (It can then be saved, later, to a file using the Save or Save as sub-options of the File menu on the menu bar.) The data structures to put into the spreadsheet are specified by the first parameter of the procedure, `DATA`. Other options and parameters allow you, for example, to add a page to a multi-page spreadsheet, to protect columns by making them read-only, to specify a title, or to save analysis commands. Full details are in the on-line help.

Data can be saved in other types of file using the `EXPORT` procedure. For example, we could create an Excel file `Ispread.xls`, containing the iron measurements from Section 4.1, by using the command

```
    EXPORT [OUTFILE='Ispread.xls'] sample,site,FE,weight
```

The `OUTFILE` option again specifies the name (and path) of the file; the file extension `xls` tells `EXPORT` that we want to create an Excel file. The available extensions are: `.XLS` for Excel, `.XLSX` for Excel 2007, `.WQ1` for Quattro, `.ODS` for Open Office Spreadsheet, `.DBF` for dBase, `.FMT` for Gauss, `.SDD` for SPlus, `.RDA` for R, `.TPT` for SAS transport, `.TAB` for tab delimited text, `.WOR` for Instat, `.MAT` for MatLab, `.ARFF` for Weka Attribute, `.TXT` for plain ASCII text, `.CSV` for comma delimited text, `.TAB` for tab delimited text, `.HTM` for a HTML table, `.RTF` for Word Rich text format, `.GSH` for Genstat spreadsheet, `.GWB` for Genstat work book, and `.BMP`, `.EMF`, `.GIF`, `.JPG`, `.TIF`, `.PNG` or `.PSD` for an image file. An image file can be created either from single matrix containing RGB colour values, or three columns of variates or factors columns (specifying x-coordinates, y-coordinates and RGB colour values), or five columns of variates or factors columns (specifying x-coordinates, y-coordinates and red, green and blue colour values). The coordinate (0, 0) corresponds to the top left corner of the image, and the y-values increase as you move down the image.

The `SHEETNAME` option allows you to specify the name of the sheet to add to an Excel file, rather than using the default `'Genstat data'`. The name should only contain letters, numbers and spaces.

The `METHOD` option controls how `EXPORT` behaves when asked to overwrite an existing file. The available settings are `add`, `append`, `overwrite`, `prompt` and `fail`, with a default of `prompt` when running interactively, and `fail` when running in batch.

The data structures to put into the spreadsheet are again specified by the first parameter of the procedure, `DATA`.

The `COLUMNS` parameter can specify names for the columns. By default, if you are saving variates, factors or texts, their identifiers are used. The setting is a text with a single line except for a matrix, where it should have a line for each column and also an extra initial line if the matrix has row labels.

The `PLAINNAMES` option allows you to suppress the additional type information that Genstat adds by default to the column names (`!` for factors, etc). Alternatively, you can set option `NONAMES=yes` to suppress the names altogether.

These are the main options and parameters that you are likely to need. Again there are others that allow you, for example, to add a page to a spreadsheet, to specify a title, or to save analysis commands. Full details are in the on-line help.

4.6 Practical

Save the bacteria data from Practical 2.2 in an Excel file.

4.7 Database files

You can load into Genstat from an ODBC database using the `DBIMPORT` procedure, and save data from Genstat into an ODBC database using the `DBEXPORT` procedure. Details are in the on-line help.

4.8 Custom output and captions

There are several commands that you can use to construct your own output. By default, in Genstat *for Windows*, this will appear in the Output window. Alternatively, you can use the `OPEN` directive to open an external file for output, and the `OUTPUT` directive to send subsequent output there. Also, many commands have a `CHANNEL` option that lets you divert output temporarily; for example, see `PRINT` in Section 3.1. We will not discuss this further here, but you can find the details in Sections 3.3 and 3.4 of the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management.*

The main command for output is `PRINT`, which was described in Section 3.1. This can display the contents of any Genstat data structure. It can choose appropriate styles and formats automatically, or you can set your own. For example

```
PRINT Sulphur,Windsp,Winddir,Rain
```

prints the variates and factors in the pollution data set from Section 4.3, in columns down the page. The field width is 12 characters, and the numbers of decimal places for the variates is chosen so that their mean of the absolute values of each variate would be displayed to four significant figures.

```
  3  PRINT Sulphur,Windsp,Winddir,Rain

       Sulphur      Windsp     Winddir        Rain
          0.00       14.80           W          no
         13.00       14.30           N          no
         12.00        5.50           W          no
         22.00        5.00          NW          no
         12.00        4.50           W          no
          6.00        4.80          NE          no
          2.00        4.30           E          no
         24.00        4.00          SE          no
         36.00        9.30           S          no
          6.00        6.30          NE          no
         10.00        5.80          SW         yes
          4.00        8.30           W         yes
          3.00       16.00          SW         yes
          7.00       15.80           W          no
          2.00       16.00          SW         yes
```

3.00	16.70	W	yes
5.00	9.50	W	no
6.00	9.80	W	yes
13.00	12.00	W	yes
49.00	4.80	N	no
26.00	2.70	W	no
6.00	6.50	SW	no
3.00	13.50	SW	yes
6.00	6.00	S	yes
8.00	10.50	W	yes
4.00	5.30	S	no
6.00	18.00	S	yes
5.00	8.50	W	yes
3.00	15.00	SW	yes
3.00	22.70	SW	no
10.00	*		yes
7.00	8.50	NW	no
3.00	8.30	SW	no
1.00	14.30	SW	no
4.00	15.00	SW	no
5.00	10.50	S	no
3.00	13.80	S	no
3.00	8.50	S	no
3.00	6.00	SE	no
5.00	16.50	S	yes
3.00	7.30	SE	no
1.00	9.80	NE	yes
6.00	7.30	NE	yes
5.00	5.50	NE	yes
5.00	6.00	E	no
6.00	11.30	E	no
11.00	8.80	E	no
2.00	8.50	SE	no
3.00	8.30	SE	no
3.00	14.50	SE	yes
2.00	9.00	S	yes
3.00	11.50	S	yes
7.00	10.00	SW	no
3.00	11.30	SW	no
3.00	18.50	SW	yes
5.00	16.00	W	no
29.00	8.30	W	no
14.00	13.00	NW	yes
15.00	13.50	NW	no
9.00	10.00	NW	no
17.00	9.00	NE	no
4.00	7.30	N	no
7.00	5.80	E	no
14.00	8.50	SE	no
4.00	6.00	SE	yes
5.00	8.00	SW	no
3.00	7.50	S	yes
3.00	10.50	SW	yes
4.00	11.00	SW	yes
4.00	16.00	W	no
2.00	13.50	SW	yes
5.00	20.20	SW	no
3.00	11.50	W	yes

```
         4.00          15.00              W            no
         5.00          10.30             SW           yes
        33.00           7.80              W           yes
        28.00           6.30              N           yes
        13.00           6.80             SW            no
         5.00          10.00             SE           yes
        26.00           5.50             SE            no
         4.00          17.00              N           yes
         8.00          19.20              N           yes
         9.00          10.00             NE           yes
        36.00           6.00             NE            no
         7.00           8.80             NW            no
        29.00           9.30             NE           yes
        11.00           6.50              W            no
        12.00           6.80             SW            no
        26.00           8.80              W            no
        21.00           8.30             NW            no
        13.00           5.30              E            no
         9.00          10.30             SW            no
        24.00          13.30              S           yes
        19.00           7.00              E           yes
        14.00          20.50             NE           yes
        28.00          17.50             NE            no
        20.00          14.80             NW           yes
        43.00          18.50             NW            no
        20.00           8.30              N            no
        25.00           0.50              N            no
         1.00           7.80             NW           yes
        16.00           3.50              E            no
        31.00           4.50             NW            no
        38.00           3.70             SE           yes
        11.00           8.00              W            no
         5.00          11.50             SW           yes
         5.00           5.80              S           yes
         4.00          14.80             SW            no
        14.00          13.30              S           yes
         3.00          10.80              S           yes
         3.00          15.00             SW           yes
         7.00          17.20             SW           yes
         2.00          12.00             SW            no
         2.00          18.00             SW           yes
```

As you can see, the sulphur figures are exact integers, and the wind speeds have been recorded to only one decimal place. So it might have been more sensible to put

```
PRINT Sulphur,Windsp,Winddir,Rain; DECIMALS=0,1,*,*
```

to remove the unnecessary trailing zeros. `DECIMALS` is ignored when textual values, like the factor labels, are being printed. So we could have specified anything for the third and fourth items of the list.

Notice that factor labels are printed by default when these are available. You can print levels instead by setting the `FREPRESENTATION` parameter. This would print levels for `Winddir`, and labels for `Rain`

```
PRINT Sulphur,Windsp,Winddir,Rain; DECIMALS=0,1,*,*;\
      FREPRESENTATION=*,*,levels,labels
```

`FREPRESENTATION` also has a setting `ordinals`, which produces the ordinal level numbers 1, 2 etc. `FREPRESENTATION` is ignored for variates, so we could equally well have typed `levels`, `labels` or `ordinals` for the first two elements of the list.

This describes only a few of the parameters of `PRINT`. A full description, with many examples, is in Sections 3.2.1 and 3.2.2 of the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*.

It might seem that a good way of producing a caption, for example *Next analysis*, would be to put to display the string `'Next analysis'` using `PRINT` i.e.

```
PRINT 'Next analysis'
```

A better way, though, is to use the `CAPTION` directive, which reproduces the same caption styles that are used by the standard Genstat directives. The contents of the caption are supplied by the first parameter, `TEXT`. The `STYLE` parameter specifies a string to indicate the caption style:

`plaintext`	ordinary text,
`stress`	text to be emphasized,
`minor`	a minor caption signifying a sub-section in the output,
`major`	a major caption signifying a section in the output,
`meta`	a meta-caption to group several sections of output,
`note`	a "note" to the user, and
`status`	a "status" message.

The `PFIRST` option controls what happens before the caption is displayed, It has settings:

`page`	to start the caption on a new page,
`dots`	to precede it by a line of dots (or a horizontal "rule" if the output is in rich text, and
`outprint`	to give either dots or a new page according to the setting for the current output channel (see the `OUTPUT` directive).

The example below displays `'Next analysis'` as a major title in Genstat's rich-text style of output, using the default font settings.

```
   4  CAPTION 'Next analysis'; STYLE=major
```

Next analysis

In Genstat's plain-text output, it would look like this

```
   4  CAPTION 'Next analysis'; STYLE=major

Next analysis
=============
```

The rich-text ouput changes automatically to take account of any changes to the font settings for major captions that the user might have selected using the Fonts and Colours tab of the Options menu in Genstat *for Windows* (Figure 1.13).

The rich-text output also recognises a sub-syntax, introduced by the special character tilde (`~`), that allows you to generate Greek characters and mathematical symbols. In plain-text output, character versions are given instead. For example, the string `'~{alpha}'` is printed as α in rich-text, but as `alpha` in plain-text. So, again you do not need to worry about how the user has configured the Output window. The rules can be found in Section 1.4.2 of the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management* or in the on-line help for `PRINT`.

The `PAGE` directive moves to the top of a new page, and the `SKIP` directive can generate blank lines. For example

```
SKIP [FILETYPE=output] 3
```

gives three blank lines. We need to set option `FILETYPE=output`, as the default action for `SKIP` is to skip lines in an input file.

`PRINT`, `CAPTION`, `PAGE` and `SKIP` should handle most of the straightforward situations. Genstat's commands for text manipulation can be used to construct more complicated types of output; details are in Section 4.7 of the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*.

4.9 Practical

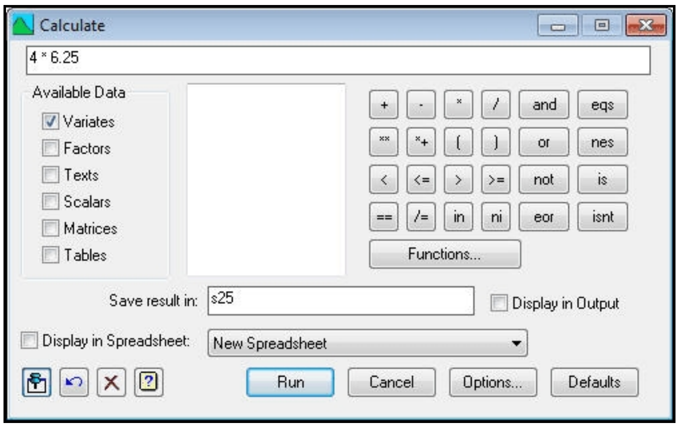
Display the caption *Counts of bacteria on pea and cereal crops*, and then print the bacteria data from Practical 2.2, using zero decimal places for the counts.

5 Calculations and manipulation

In this chapter we introduce the facilities for calculations and data handling in Genstat, including:

- the CALCULATE directive, which provides the very general calculator used by the Genstat Calculate menu;
- arithmetic calculations;
- relational and logical tests (and how Genstat represents their *true* or *false* results);
- functions for transforming and summarizing your data;
- the RESTRICT directive and SUBSET procedure, which allow you to operate on subsets of your data values;
- the SORT directive which allows you to reorder data values.

5.1 Calculations



4 * 6.25
Variates ✓
Save result in: s25
Display in Spreadsheet: New Spreadsheet

Figure 5.1

Calculations are done in Genstat *for Windows* using the Calculate menu, which is obtained by selecting Calculations from the Data menu on the menu bar. This has buttons for all the usual arithmetic operators, as well as some less familiar operations, which we will describe later in this chapter.

A full description of the menu is in Section 2.12 of the *Introduction to Genstat for Windows*. Here we will just use the menu to multiply two numbers together. For example, we put the cursor into the window at the top of the menu, type 4, click the button for the operator *, and then type 6.25. You can display the result in the output window by checking the Display in Output box, or you can save the results in a data structure by typing its identifier into the Save Result In box (or you can do both). In Figure 5.1 we have typed S25 for the identifier. This will be defined as a scalar data structure (since the calculation has generated a single number as its result), storing the value 25. Clicking on Run produces the following output.

```
  2  DELETE [REDEFINE=yes] S25
  3  CALCULATE S25=4 * 6.25
  4  PRINT S25

              S25
            25.00
```

The output is echoing the commands that have been written by the menu to do the calculation and print the result. The `DELETE` directive in line 2 deletes `S25` in case it had already been used in this run of Genstat. The option setting `REDEFINE=yes` deletes any pre-existing definition of `S25`, so that it can now be defined as a scalar. (If the option had been omitted, the command would merely delete any values stored by `S25`). In line 3, the calculation is done by the `CALCULATE` directive, and in line 4 the results are printed.

`CALCULATE` has a single parameter which supplies an expression to define what calculation is to be done, and where the results are to be stored. As the directive has only one parameter, no parameter name is defined in the syntax. So there is no possibility of confusion between the equals character in the parameter setting (i.e. *parameter-name=expression*) and the equals character in its use within the expression (e.g. `S25 = 4 * 6.25`).

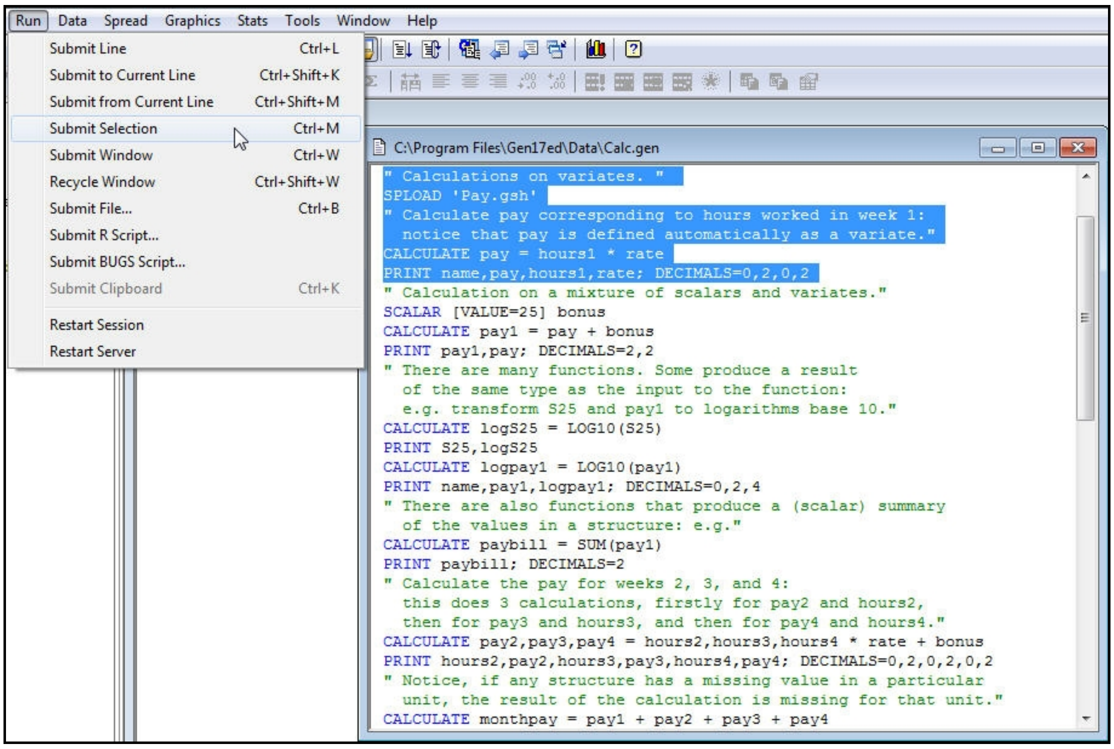
All the usual arithmetic operators are available:

`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division
`**`	exponentiation (for example, `X**2` stands for X^2)

There are also functions and logical tests, as we describe later.

`CALCULATE` can operate on any numerical data structure. In the example above 4 and 6.25 were (unnamed) scalars. Notice that `CALCULATE` automatically declares the structure to hold the results if it has not been declared already (or if it has just been deleted!). Thus, as the calculation above produces a single number, `S25` is automatically declared as a scalar.

To illustrate the commands for calculations and manipulation, we will use a Genstat program in the file `Calc.gen` (again in the Data folder). Instead of submitting the commands one at a time, as in Section 1.3 (see Figure 1.18) we will submit several lines at once. We first highlight the lines with the mouse, in the usual way. Then click on the Submit Selection option of the Run menu on the menu bar, as shown in Figure 5.2. Notice that this option has the short-cut key `Ctrl-M` (i.e. press the `M` key while holding down the `Ctrl` key).



```
" Calculations on variates. "
SPLOAD 'Pay.gsh'
" Calculate pay corresponding to hours worked in week 1:
  notice that pay is defined automatically as a variate."
CALCULATE pay = hours1 * rate
PRINT name,pay,hours1,rate; DECIMALS=0,2,0,2
" Calculation on a mixture of scalars and variates."
SCALAR [VALUE=25] bonus
CALCULATE pay1 = pay + bonus
PRINT pay1,pay; DECIMALS=2,2
" There are many functions. Some produce a result
  of the same type as the input to the function:
  e.g. transform S25 and pay1 to logarithms base 10."
CALCULATE logS25 = LOG10(S25)
PRINT S25,logS25
CALCULATE logpay1 = LOG10(pay1)
PRINT name,pay1,logpay1; DECIMALS=0,2,4
" There are also functions that produce a (scalar) summary
  of the values in a structure: e.g."
CALCULATE paybill = SUM(pay1)
PRINT paybill; DECIMALS=2
" Calculate the pay for weeks 2, 3, and 4:
  this does 3 calculations, firstly for pay2 and hours2,
  then for pay3 and hours3, and then for pay4 and hours4."
CALCULATE pay2,pay3,pay4 = hours2,hours3,hours4 * rate + bonus
PRINT hours2,pay2,hours3,pay3,hours4,pay4; DECIMALS=0,2,0,2,0,2
" Notice, if any structure has a missing value in a particular
  unit, the result of the calculation is missing for that unit."
CALCULATE monthpay = pay1 + pay2 + pay3 + pay4
```

Figure 5.2

Most practical calculations are done on whole series of numbers, stored in variates in Genstat. To show what can be done, the program uses some administrative data from a small company, recording rates of pay and hours of work over a four-week period. Line 6 uses `SPLOAD` to load the data, as explained in Section 4.3. Line 9 then calculates the wages for the first week.

```
  5  " Calculations on variates. "
  6  SPLOAD 'Pay.gsh'
```

Loading Spreadsheet File

Catalogue of file Pay.gsh

```
Sheet Type: vector
    Index          Type       Nval    Name
        2          text         10    name
        3       variate         10    rate
        4       variate         10    hours1
        5       variate         10    hours2
        6       variate         10    hours3
        7       variate         10    hours4
```

Note: Missing indices are used by unnamed or system structures. These store ancillary information, for example factor labels.

```
   7  " Calculate pay corresponding to hours worked in week 1:
  -8    notice that pay is defined automatically as a variate."
   9  CALCULATE pay = hours1 * rate
  10  PRINT name,pay,hours1,rate; DECIMALS=0,2,0,2
```

```
     name          pay        hours1          rate
   Clarke       337.50            45          7.50
    Innes       318.75            51          6.25
    Adams       600.00            40         15.00
    Jones       287.50            46          6.25
      Day       250.00            40          6.25
     Grey       275.00            44          6.25
  Edwards       352.50            47          7.50
    Baker       315.00            42          7.50
     Hill       400.00            40         10.00
   Foster       410.00            41         10.00
```

The calculation takes place for every unit of the variates: so the pay for `Foster` is the appropriate value for `hours1` (`41`) multiplied by the corresponding value of `rate` (`10.00`).

Scalars can be included in the obvious way. in line 13, the scalar `bonus` is added to every unit of the variate `pay`.

```
11  " Calculation on a mixture of scalars and variates."
12  SCALAR [VALUE=25] bonus
13  CALCULATE pay1 = pay + bonus
14  PRINT pay1,pay; DECIMALS=2,2

         pay1          pay
       362.50       337.50
       343.75       318.75
       625.00       600.00
       312.50       287.50
       275.00       250.00
       300.00       275.00
       377.50       352.50
       340.00       315.00
       425.00       400.00
       435.00       410.00
```

We can also include ordinary numbers. By putting the amount of the bonus into the scalar structure we can use it again (without having to remember its exact value) later in the program. But if it were not needed later, we could simply put

```
CALCULATE pay1 = pay + 25
```

Genstat provides many functions for use in expressions. A concise description of each one, in alphabetical order, is in Section 4.2 of the *Reference Manual, Part 1 Summary*, which can be accessed by clicking on the Summary sub-option of the Reference Manual option of the Help menu on the menu bar. Alternatively, a more detailed description is in Section 4.2 of the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*, which can be accessed by clicking on the Syntax and Data Management sub-option of the Genstat Guides option of the Help menu on the menu bar.

Many functions are *transformations*. These produce a result that is the same type of structure as the *argument* of the function. One example is the `LOG10` function, which transforms numbers to logarithms with base 10. So, in line 18, the command

```
CALCULATE logS25 =  LOG10(S25)
```

gives a scalar result as `S25` is a scalar, whereas the command

```
CALCULATE logpay1 =  LOG10(pay1)
```

in line 20 generates a variate with 10 values, to match `pay1`.

```
 15  " There are many functions. Some produce a result
-16    of the same type as the input to the function:
-17    e.g. transform S25 and pay1 to logarithms base 10."
 18  CALCULATE logS25 = LOG10(S25)
 19  PRINT S25,logS25

            S25          logS25
          25.00           1.398

  20  CALCULATE logpay1 = LOG10(pay1)
  21  PRINT name,pay1,logpay1; DECIMALS=0,2,4

     name          pay1       logpay1
   Clarke        362.50        2.5593
    Innes        343.75        2.5362
    Adams        625.00        2.7959
    Jones        312.50        2.4949
      Day        275.00        2.4393
     Grey        300.00        2.4771
  Edwards        377.50        2.5769
    Baker        340.00        2.5315
     Hill        425.00        2.6284
   Foster        435.00        2.6385
```

These are some of the most useful transformations.

Mathematical transformations

`COS(x)`	cosine of `x`, for `x` in radians.
`EXP(x)`	exponent of `x`: e^x.
`LOG(x)`	natural logarithm of `x`, for `x` > 0.
`LOG10(x)`	logarithm to base 10 of `x`, for `x` > 0.
`SIN(x)`	sine of `x`, for `x` in radians.
`SQRT(x)`	square root of `x`, for `x` ≥ 0.

Arithmetical transformations

`ABS(x)`	absolute value of `x`: $\lvert x \rvert$.
`INTEGER(x)`	integer part of `x`: [`x`].
`MODULO(x; y)`	modulus of `x` to base `y`.
`ROUND(x)`	nearest integer to `x`.

Statistical transformations

`ANGULAR(p)`	the angular transformation: for a percentage `p` ($0<p<100$), forms $x = (180/\pi) \times \arcsin(\mathrm{sqrt}(p/100))$.

`IANGULAR(x)`	the inverse of the angular transformation (result in percentages).
`ILOGIT(x)`	the inverse of the logit transformation (result in percentages).
`LOGIT(p)`	takes the logit transformation log(`p`/(100−`p`)) of the percentages `p` (0 < `p` < 100).
`NED(p)`	gives the Normal Equivalent Deviate, that is the value `x` that leaves a proportion `p` (0 < `p` < 1) to the left of it under the standard Normal curve; this is another transformation for percentage data when expressed as proportions: Probit transformation = `NED+5`.
`NORMAL(x)`	the Normal probability integral: gives the probability that a random variable with a standard Normal `N`(0,1) distribution is less than `x`; this is the inverse of `NED`.

Re-ordering and combining transformations

`CUMULATE(x)`	forms the cumulative sum of the values in `x`; that is, x_1, x_1+x_2, $x_1+x_2+x_3$, and so on.
`DIFFERENCE(x; s)`	forms the differences of the values in `x`; that is, x_i-x_{i-s}. If `s` is omitted, first differences are formed, as for `s`=1
`REVERSE(x)`	takes the values in `x` in reverse order.
`SHIFT(x; s)`	shifts the values in `x` by `s` places (to the right or left according to the sign of `s`). This is not a circular shift, so some positions lose their values and are given missing values.

Other functions produce a scalar summary of all the values in a structure. For example, we can use the `SUM` function to calculate the total pay bill for all the employees.

```
 22  " There are also functions that produce a (scalar) summary
-23    of the values in a structure: e.g."
 24  CALCULATE paybill = SUM(pay1)
 25  PRINT paybill; DECIMALS=2

        paybill
        3796.25
```

Here are some of the other summary functions like `SUM`.

`MAXIMUM(x)`	maximum of the values in `x`.
`MEAN(x)`	mean of the values in `x`.
`MEDIAN(x)`	median of the values in `x`.
`MINIMUM(x)`	minimum of the values in `x`.
`NMV(x)`	number of missing values in `x`.
`NOBSERVATIONS(x)`	number of observations (that is, non-missing values) in `x`.
`NVALUES(x)`	number of values, including missing values, of `x` (that is, the length of `x`).
`SUM(x)`	sum of the values in `x`.

`TOTAL(x)`	synonym of `SUM`.
`VARIANCE(x)`	variance of the values in `x`.

Expressions can contain lists, to specify that the same calculation is to be done for several sets of structures. For example, we can calculate the wages for the next three weeks in one command:

```
CALCULATE pay2,pay3,pay4 = \
  hours2,hours3,hours4 * rate + bonus
```

This has the same effect as the three commands

```
CALCULATE pay2 = hours2 * rate + bonus
CALCULATE pay3 = hours3 * rate + bonus
CALCULATE pay4 = hours4 * rate + bonus
```

Notice that, if any of the lists on the right-hand side of the expression is shorter than the list on the left-hand side, the list is re-used. So the value of `Bonus` is used for all three weeks. To take a more complicated example

```
CALCULATE X,Y,Z = A,B,C + 1,2
```

is the same as the three calculations

```
CALCULATE X = A + 1
CALCULATE Y = B + 2
CALCULATE Z = C + 1
```

However, the lists on the right-hand side must not be longer than the list on the left-hand side. This re-using of lists is general in Genstat, as explained in Section 3.1.

```
 26  " Calculate the pay for weeks 2, 3, and 4:
-27    this does 3 calculations, firstly for pay2 and hours2,
-28    then for pay3 and hours3, and then for pay4 and hours4."
 29  CALCULATE pay2,pay3,pay4 = hours2,hours3,hours4 * rate + bonus
 30  PRINT hours2,pay2,hours3,pay3,hours4,pay4; DECIMALS=0,2,0,2,0,2

        hours2          pay2        hours3          pay3        hours4          pay4
            41        332.50            43        347.50            42        340.00
            46        312.50            48        325.00            44        300.00
            40        625.00             *             *            40        625.00
            44        300.00            47        318.75            42        287.50
            42        287.50            43        293.75             *             *
            45        306.25            42        287.50            43        293.75
             *             *            46        370.00            43        347.50
            47        377.50            44        355.00            45        362.50
            40        425.00            40        425.00           400       4025.00
            40        425.00            42        445.00            41        435.00
```

Notice that a *missing* number of hours generates a *missing* result for pay. This illustrates a general rule in calculations that, if any of the structures involved in a calculation has a missing value in a particular unit, the result of the calculation will be missing for that unit. So, when we now calculate the total pay for each employee over the four weeks, we obtain a missing result if the pay is unknown for any of the four weeks.

```
 31  " Notice, if any structure has a missing value in a particular
-32    unit, the result of calculation is missing for that unit."
 33  CALCULATE monthpay = pay1 + pay2 + pay3 + pay4
 34  PRINT pay1,pay2,pay3,pay4,monthpay; DECIMALS=2

          pay1          pay2          pay3          pay4      monthpay
        362.50        332.50        347.50        340.00       1382.50
        343.75        312.50        325.00        300.00       1281.25
        625.00        625.00             *        625.00             *
        312.50        300.00        318.75        287.50       1218.75
        275.00        287.50        293.75             *             *
        300.00        306.25        287.50        293.75       1187.50
        377.50             *        370.00        347.50             *
        340.00        377.50        355.00        362.50       1435.00
        425.00        425.00        425.00       4025.00       5300.00
        435.00        425.00        445.00        435.00       1740.00
```

If you want to replace a missing value, you can use the function `MVREPLACE`. This has two arguments, which are separated from each other by a semi-colon. The first specifies the identifier of the data structure with the missing values, and the second supplies the values that are to replace them. In our example we might assume that a value would be missing if an employee had not been present during the week concerned, so we should replace it by zero.

```
 35  " Replace the missing values by the value 0 using MVREPLACE."
 36  CALCULATE pay2,pay3,pay4 = MVREPLACE( pay2,pay3,pay4; 0,0,0 )
 37  CALCULATE monthpay = pay1 + pay2 + pay3 + pay4
 38  PRINT pay1,pay2,pay3,pay4,monthpay; DECIMALS=2

          pay1          pay2          pay3          pay4      monthpay
        362.50        332.50        347.50        340.00       1382.50
        343.75        312.50        325.00        300.00       1281.25
        625.00        625.00          0.00        625.00       1875.00
        312.50        300.00        318.75        287.50       1218.75
        275.00        287.50        293.75          0.00        856.25
        300.00        306.25        287.50        293.75       1187.50
        377.50          0.00        370.00        347.50       1095.00
        340.00        377.50        355.00        362.50       1435.00
        425.00        425.00        425.00       4025.00       5300.00
        435.00        425.00        445.00        435.00       1740.00
```

Expressions can also involve relational and logical tests. These produce the value 1 if the result is *true*, and 0 if it is *false*. For example, we can use the greater-than operator (`>`) to set up a variate of 0s and 1s according to whether staff are recorded as working less than or greater than 100 hours in the fourth week.

```
CALCULATE odd4 = hours4 > 100
```

We could then use the result in the `MVINSERT` function to place a missing value in the `monthpay` variate, since we believe this record must be wrong. This function also has two arguments. The first is the identifier of the structure with values that need changing, and the second is a variate of 0s and 1s indicating which values are to become missing values. So the following command takes the values of `monthpay,` and inserts a missing value whenever the corresponding value of `odd4` is non-zero, storing the results back in `monthpay`:

```
CALCULATE monthpay = MVINSERT(monthpay; odd4)
```

However, the calculation does not have to be done in two stages: in general, the arguments of functions in Genstat expressions can themselves be expressions, as illustrated below.

```
 39  CALCULATE odd4 = hours4 > 100
 40  PRINT odd4,hours4; DECIMALS=0

          odd4        hours4
             0            42
             0            44
             0            40
             0            42
             *             *
             0            43
             0            43
             0            45
             1           400
             0            41

 41   " MVINSERT puts a missing value into monthpay when hours4>100."
 42   CALCULATE monthpay = MVINSERT(monthpay; hours4>100)
 43   PRINT monthpay,hours4; DECIMALS=2,0

      monthpay        hours4
       1382.50            42
       1281.25            44
       1875.00            40
       1218.75            42
        856.25             *
       1187.50            43
       1095.00            43
       1435.00            45
             *           400
       1740.00            41
```

The available operators for relational tests are as follows:

`==` or `.EQ.`	equality
`>=` or `.GE.`	greater than or equal to
`>` or `.GT.`	greater than
`<=` or `.LE.`	less than or equal to
`<` or `.LT.`	less than
`/=` or `<>` or `.NE.`	not equal to
`.IN.`	inclusion: `X.IN.Vals` gives result true for each value of `X` that is equal to any one of the values of `Vals`
`.NI.`	non-inclusion: the opposite of `.IN.`

There are also some logical operators that can be useful to combine the results of expressions involving relational operators.

`.AND.`	and: `a.AND.b` true if both `a` and `b` are true
`.EOR.`	either or: `a.EOR.b` is true if either `a` or `b`, but not both, is true
`.OR.`	or: `a.OR.b` is true if either `a` or `b` is true
`.NOT.`	not: `.NOT.a` is true for `a` untrue

The precedence of the operators (that is, the order in which they are evaluated if there are several different ones in an expression) is much the same as you would expect from ordinary arithmetic; see Section 1.6.2 of the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*. However, in case of any doubt, it is safest to use brackets – the expression inside a pair of brackets is always evaluated first. So, for example

```
CALCULATE A = 5 * 2 + 3
```

gives `A` the value 13, but

```
CALCULATE A = 5 *(2 + 3)
```

gives `A` the value 25.

Genstat text structures can be used in expressions, but only with the set inclusion operators `.IN.` and `.NI.` (see above), or the string operators `.EQS.` (equality) and `.NES.` (inequality). For example, the expression

```
Text1 .EQS. Text2
```

compares the string in each unit (or line) of `Text1` with that in the corresponding unit of `Text2`, giving the result *true* if they are identical; while

```
Text1 .NES. Text2
```

gives the result *true* if they differ.

When a factor occurs on the right-hand side of an expression, Genstat usually works with its levels. The exception is when the factor occurs as the first operand of the operators `.IN.` or `.NI.` and the second operand is a text; the factor labels are then used instead. A factor can also occur on the left-hand side of an expression and receive the results of a calculation; an error is reported if any of the resulting values is not one of the levels of the factor. Two functions are provided especially for factors: `NLEVELS(F)` gives the number of levels of the factor `F`, and `NEWLEVELS(F; V)` forms a variate from the factor `F`, using variate `V` to define values for the levels.

For example, the factor Amount in Section 2.1 had three levels, 0, 1 and 2 (see Figure 2.5). The program below uses the numbers 0, 1.2 and 2.4 instead.

```
VARIATE [VALUES=0,1.2,2.4] Newvals
CALCULATE Newamounts = NEWLEVELS(Amount; Newvals)
```

5.2 Practical

Details are given below of numbers of personal computers sold by a shop in the months of 2001 and the prices charged; the data are available in the spreadsheet file `Computer.gsh`. Calculate and print the amount received from PC sales in each month, and the total received over the whole year.

```
"month number  price"
January    12     999
February    8    1150
March      21    1150
April      18    1250
May         7    1250
June        5    1250
July        6    1250
August     18    1099
September   5    1250
October    17    1250
November   13    1250
December   31    1150
```

5.3 Subsets of data values

When dealing with a large set of data, you often need to be able to select a subset of values to study, either temporarily or for the remainder of a session. For example, with the pollution data in Section 4.3, we might want to concentrate just on the rainy days and draw a picture of the distribution of sulphur measurements.

One convenient way of doing this in Genstat is to use the `RESTRICT` directive. This allows you to define a *restriction* on a vector (i.e. on a variate, factor or text). Once this has been done, most commands that work with that vector will operate only on the restricted set of units. The on-line help will tell you whether a particular command, that operates on vectors, does take notice of restrictions. For example, the help page for `PRINT` says

> *You can restrict any vector (variate, factor or text) to specify that only a subset of its units should be printed. When printing in series the vectors can be restricted to different subsets; but with parallel printing any restriction is applied to all the vectors (and any pointers) so, if more than one vector is restricted, they must all be restricted in the same way.*

This illustrates a general convention, that a restriction on a vector will apply to all vectors that are involved in the same operation. Also that, if an operation involves more than one restricted vector, all the restrictions must be to the same set of units. Notice, though, that the convention is applied in a sensible way. If you print several vectors in series, these are printed separately, one at a time, and so they can be restricted in different ways. If you print them in parallel, they are printed side-by-side, so any restriction must apply to all of them.

For example, we shall use `RESTRICT` to print a list of the staff who worked less than 42 hours in the first week:

```
" Restrict name to those working less than 42 hours in week 1."
RESTRICT name; CONDITION = hours1<42
PRINT name
```

```
   name
  Adams
    Day
   Hill
 Foster
```

The general form is:

```
RESTRICT list of vectors; CONDITION=logical expression
```

where a *vector* can be a variate, text, or factor. The logical expression must supply a variate of 0s and 1s, specifying which values of the vectors are to be selected in the same way as values are specified in the `MVINSERT` function in the previous section. In fact, the values do not have to be just 0 or 1: any non-zero value is taken to mean that a value is to be in the subset.

An advantage of `RESTRICT` is that does not discard the restricted units. So you can change the restriction to look at some other set of units, or you can cancel the restriction by specifying `RESTRICT` with no condition.

```
" Cancel the restriction on name."
RESTRICT name
PRINT name,hours1; DECIMALS=0,0
```

```
   name      hours1
 Clarke          45
 Innes           51
  Adams          40
  Jones          46
    Day          40
   Grey          44
Edwards          47
  Baker          42
   Hill          40
 Foster          41
```

The `.IN.` and `.NI.` operators are particularly useful with `RESTRICT`. For example, we could print the details of hours worked by Adams and Day by putting

```
TEXT [VALUES=Adams,Day] AD
RESTRICT Hours; CONDITION=Name.IN.AD
PRINT Name,Hours
```

or the details just of Adams by

```
RESTRICT Hours; CONDITION=Name.IN.'Adams'
PRINT Name,Hours
```

```
 50  " Another example: use of the .IN. operator
-51    to print details of Adams and Day "
 52  TEXT [VALUES=Adams,Day] AD
 53  RESTRICT hours1; CONDITION=name.IN.AD
 54  PRINT name,hours1

      name            hours1
     Adams             40.00
       Day             40.00

  55   " or just the details of Adams "
  56   RESTRICT hours1; CONDITION=name.IN.'Adams'
  57   PRINT name,hours1

      name            hours1
     Adams             40.00

  58   RESTRICT hours1
```

However, if we had specified

```
     RESTRICT Hours; CONDITION=Name.EQS.'Adams'
```

Genstat would have reported a fault as it would be unable to do a line-by-line comparison of the text `Names` (which has 10 values) and `'Adams'` (which has only one).

In Genstat *for Windows*, you can also apply restrictions using sub-options of the Restrict/Filter option of the Spread menu on the menu bar. These then send an appropriate `RESTRICT` command to the Genstat server.

You can use the `SUBSET` procedure if you do want to store a subset of the units (or if you want to use a command that does not recognise `RESTRICT`). Below we form a text `name42`, and a variate `pay42`, with the names and pay in week 1 of the employees that worked less than 42 weeks.

```
 59  " Form a variate with the pay in week 1, and a text with names
-60    of the subset that worked less than 42 hours in week 1."
 61  SUBSET [CONDITION= hours1<42] name,pay1; NEWVECTOR=name42,pay42
 62  PRINT name42,pay42

name42          pay42
 Adams          625.0
   Day          275.0
  Hill          425.0
Foster          435.0
```

The subset is again defined by a logical expression, which must now be specified by the `CONDITION` option; units with *true* values (non-zero and non-missing) for the condition are included in the subset, others are omitted.

Subsets can be formed for factors, texts and variates. Relevant attributes will also be transferred across to the new structures but, if the subset excludes some of the levels of a factor, a new reduced set of levels (and labels) can be requested by setting option

SETLEVELS=yes.

The original vectors are specified by the OLDVECTOR parameter and identifiers for the vectors to contain the subsets are specified by the NEWVECTORS parameter. If NEWVECTORS is not set, the OLDVECTORS are redefined to store the subsets instead of their original values.

In Genstat *for Windows*, you can form subsets using the Subset menu, which can be opened by clicking on the Subset option of the Data menu on the menu bar. This uses the SUBSET procedure, and allows you to form a new vector for the subset. However, you can do this for only one vector at a time.

Alternatively, you can first apply a restriction to the required subset, using sub-options of the Restrict/Filter option of the Spread menu on the menu bar, and then clicking on the Restricted rows sub-option of the Delete option of the Spread menu. This deletes the unwanted rows, and then uses READ to read the subset into Genstat. You can form the subset for several vectors at once. However, the subset replaces their original values.

5.4 Practical

Using the data on sales of personal computers in Practical 5.2, print the names of the months when fewer than 10 were sold. Calculate the total amount received in just those months. Do you get the same answer by using RESTRICT and SUBSET?

5.5 Sorting data

The SORT directive allows you to reorder the units of a list of vectors according to one or more index vectors. The first parameter (OLDVECTOR) specifies the vectors whose values are to be sorted, and NEWVECTOR supplies structures to store the sorted values. First we sort the names and monthly pay values from the example in Section 5.1 into alphabetic order of names.

```
 63  " Sort alphabetically."
 64  SORT [INDEX=name] monthpay,name; NEWVECTOR=sortpay,sortname
 65  PRINT sortname,sortpay; DECIMALS=0,2

 sortname        sortpay
    Adams        1875.00
    Baker        1435.00
   Clarke        1382.50
      Day         856.25
  Edwards        1095.00
   Foster        1740.00
     Grey        1187.50
     Hill              *
    Innes        1281.25
    Jones        1218.75
```

The index vectors can be texts (as above), factors, or variates. They can be specified using the INDEX option, as above, but if this is omitted, Genstat uses the first old vector. We now sort the names and monthly pay into descending order of the amount paid. By default, values are sorted into ascending order, but this can be changed by setting option

DIRECTION to descending as shown below.

```
" Sort into descending order of monthly pay."
SORT [DIRECTION=descending] monthpay,name; NEWVECTOR=sortpay,sortname
PRINT sortname,sortpay; DECIMALS=0,2
```

```
 sortname       sortpay
     Hill             *
    Adams       1875.00
   Foster       1740.00
    Baker       1435.00
   Clarke       1382.50
    Innes       1281.25
    Jones       1218.75
     Grey       1187.50
  Edwards       1095.00
      Day        856.25
```

Notice that any missing values are placed first, in ascending or descending sorting.

If the NEWVECTOR parameter is omitted, the sorted values are placed into the structures in the OLDVECTOR parameter themselves, thereby losing the original ordering. We show this below, at the same time as demonstrating that you can use more than one index vector; the values are sorted first according to the rates of pay, and then, where several people have the same rate, according to alphabetic order of names.

```
" Sort into alphabetic order within (ascending) pay rates."
SORT [INDEX=rate,name] OLDVECTOR=name,rate,monthpay
PRINT name,rate,monthpay; DECIMALS=0,2,2
```

```
     name          rate      monthpay
      Day          6.25        856.25
     Grey          6.25       1187.50
    Innes          6.25       1281.25
    Jones          6.25       1218.75
    Baker          7.50       1435.00
   Clarke          7.50       1382.50
  Edwards          7.50       1095.00
   Foster         10.00       1740.00
     Hill         10.00             *
    Adams         15.00       1875.00
```

5.6 Practical

Using the data on sales of personal computers in Practical 5.2, sort (and print) the months according firstly to the number of PC's sold in each one, and then according to the profits made in each.

5.7 Other manipulation facilities

There are many other facilities for data manipulation in Genstat, which we do not have to space to describe here. We list below some of the more useful with references, in brackets, to the relevant sections of the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*:

`APPEND`	appends values of a list of vectors of the same type (4.4.4)
`STACK`	combines several data sets by "stacking" the corresponding vectors (4.4.5)
`UNSTACK`	splits vectors into individual vectors according to levels of a factor (4.4.6)
`EQUATE`	copies values between sets of data structures (4.3.1)
`SETRELATE`	compares the sets of values in two data structures (4.3.2)
`SETCALCULATE`	performs Boolean set calculations on the contents of vectors and pointers (4.3.3)
`TXCONSTRUCT`	forms a text structure by appending or concatenating values of scalars, variates, texts, factors or pointers; allows the case of letters to be changed or values to truncated and reversed (4.7.2)

The best summary of all the data manipulation commands is in Section 1.7 of the *Guide to the Genstat Command Language, Part 2 Statistics*. A more detailed description, with examples, is in Chapter 4 of the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*.

6 Regression

This chapter introduces the commands for fitting regression models in Genstat, and shows how they are used by the menus. We start with *simple linear regression*, where a straight line is fitted to represent the relationship between two variables; one variable is considered as the *response variable* (or *y-variate*) and the model predicts its mean value given the value of the other, *explanatory variable* (or *x-variate*). We then show how you can fit regressions with several explanatory variables, and assess which ones are really needed in the model. Finally, we show how to fit *parallel* and *non-parallel regressions* when you have an explanatory factor as well as an x-variate.

So, in this chapter you will learn about:

- the `MODEL` directive, which defines the response variate;
- the `TERMS` directive, which defines the maximal (i.e. most complicated) model, and is useful when you want to study a sequence of regression models;
- the `FIT` directive, which does the analysis;
- how `MODEL`, `TERMS` and `FIT` are used by the Linear Regression menu to fit a simple linear regression i.e. one with a single explanatory variate;
- how to use the `ADD`, `DROP`, `STEP` and `TRY` directives to modify a regression model when you have several explanatory variates;
- how you can include factors in your regression model, to study parallel and non-parallel regression lines.

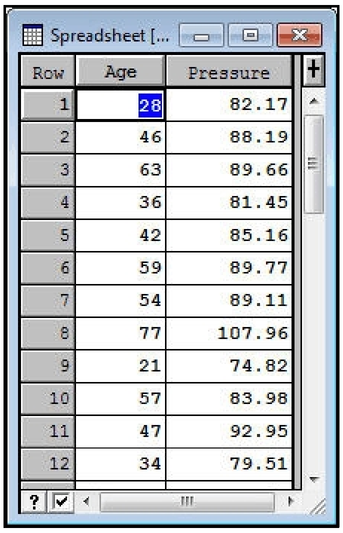
If your main interest is in how to use the Genstat commands rather than how to use Genstat's regression facilities, you may want to skip Sections 6.3 onwards. We will use simple linear regression (from Section 6.1) again in Section 8.1, when we discuss how you can mix commands with menus. However, we will not make any further use of multiple linear regressions (Section 6.3) or parallel regressions (Section 6.5).

A comprehensive description of the regression menus is given in the *Guide to Regression, Nonlinear and Generalized Linear Models in Genstat*, while full details of the commands and the underlying statistical theory are in the *Guide to the Genstat Command Language, Part 2 Statistics*, Chapter 3. These can be accessed from within Genstat *for Windows* by selecting sub-options of the Genstat Guides option of the Help menu on the menu bar.

6.1 Simple linear regression

Spreadsheet file `Pressure.gsh` (Figure 6.1) contains recordings of the blood-pressure of a sample of 38 women whose ages range from 20 to 80. The file can be opened from within Genstat using the Example Data Sets menu, as explained in Section 2.1.

We can read the data into Genstat using `SPLOAD`, as explained in Section 4.3.



Row	Age	Pressure
1	28	82.17
2	46	88.19
3	63	89.66
4	36	81.45
5	42	85.16
6	59	89.77
7	54	89.11
8	77	107.96
9	21	74.82
10	57	83.98
11	47	92.95
12	34	79.51

Figure 6.1

```
  2  SPLOAD 'Pressure.gsh'

Loading Spreadsheet File

Catalogue of file Pressure.gsh

Sheet Type: vector
     Index          Type        Nval    Name
         1        variate         38    Age
         2        variate         38    Pressure
```

Figure 6.2 shows a plot of pressure against age, drawn using the command

```
DGRAPH Pressure; Age
```

(alternatively, this could be done through the graphics wizard, by selecting 2D Scatter Plot option of the Graphics menu on the menu bar, as explained in Section 3.1 of the *Introduction to Genstat for Windows*). This suggests that there is a linear relationship between blood-pressure and age. We will quantify this by a linear regression model, which specifies a *line of best fit* or a *regression line* between the points on the graph. It is natural here to assume that the blood-pressure is *responding*

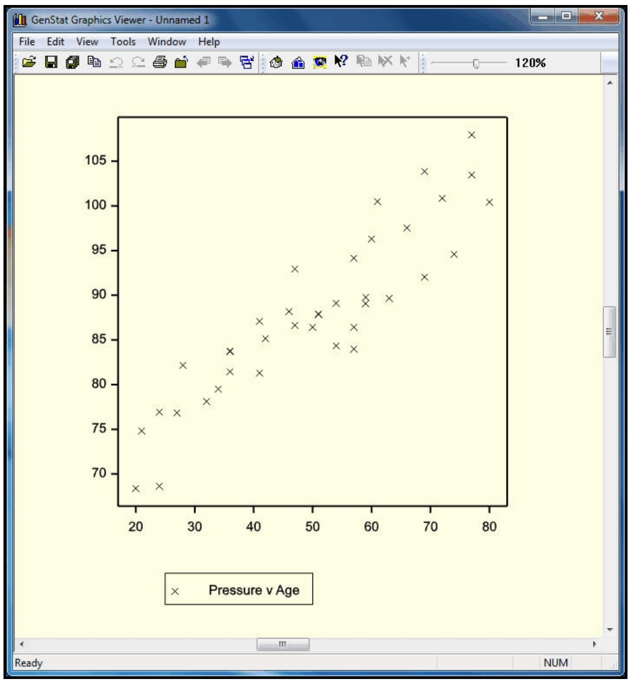


Figure 6.2

to increasing age, so we will fit a line or model to predict blood-pressure from age. The equation of the line is

$$pressure_i = a + b \times age_i + e_i$$

where a can be interpreted as the intercept of the regression line, b as its slope and e_i as the error, or vertical distance of the ith point from the line. A regression analysis produces estimates of the *parameters* a and b of this model, and also of the variance of the variable e which is often of as much interest as the parameters. Further information about method of estimation, and of the assumptions that are necessary, is given in Chapter 1 of the *Guide to Regression, Nonlinear Models and Generalized Linear Models*.

In this section we will show how the Linear Regression menu uses the regression commands to fit models like this. The menu, shown in Figure 6.3, is opened by clicking on the Linear Models sub-option of the Regression Analysis option of the Stats menu on the menu bar. For a simple linear regression like this, you just need to put the name of the response variate Pressure into the Response Variate box, and the name of the explanatory variate Age into the Explanatory Variate box. If we now click on Run, the menu fits the regression using the commands

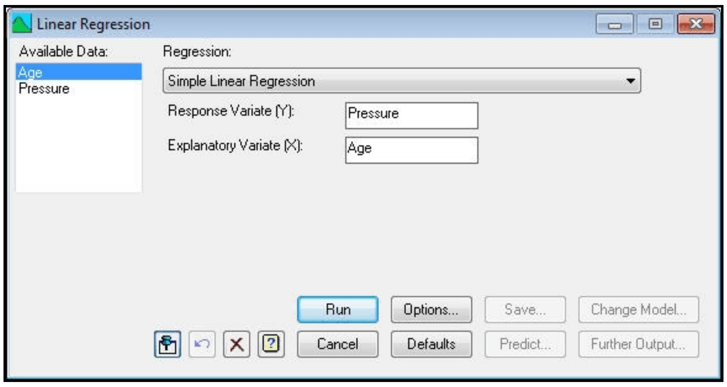
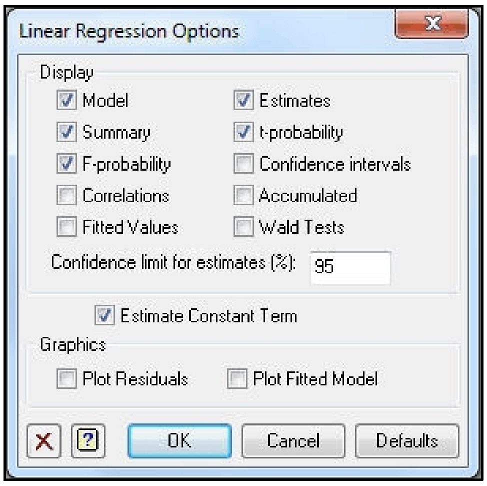

Regression: Simple Linear Regression
Response Variate (Y): Pressure
Explanatory Variate (X): Age

Figure 6.3

```
"Simple Linear Regression"
MODEL Pressure
TERMS Age
FIT [PRINT=model,summary,estimates; CONSTANT=estimate;\
    FPROB=yes; TPROB=yes] Age
```

The MODEL statement gives the identifier of the variate that contains the values of the response variable (this is the variable whose behaviour is to be modelled).

TERMS is relevant when you want to explore a sequence of regression models. It defines the *maximal model*, that is the most complicated model that you may want to fit. Genstat can then define a common set of units for the regression (omitting any units that are have missing values in the response or explanatory variates), and carry out some initial calculations to make the process more efficient. TERMS is not really necessary here, as we have only one model to fit. We will need it, however, in the Section 6.3 when we investigate several explanatory variates.


Linear Regression Options

Display

- [✓] Model
- [✓] Estimates
- [✓] Summary
- [✓] t-probability
- [✓] F-probability
- [ ] Confidence intervals
- [ ] Correlations
- [ ] Accumulated
- [ ] Fitted Values
- [ ] Wald Tests

Confidence limit for estimates (%): 95

- [✓] Estimate Constant Term

Graphics

- [ ] Plot Residuals
- [ ] Plot Fitted Model

Figure 6.4

The FIT statement specifies the explanatory variate, and produces the analysis. When you are using the menus, the options of the FIT statement are defined by the boxes that are checked in the Linear Regression Options menu (which is obtained by clicking on the Options button of the Linear Regression menu). Figure 6.4 shows the settings that were used for the FIT statement above.

The PRINT option controls the output from FIT, and is set by some of the Display boxes. The string tokens model, summary and estimates are actually the defaults for PRINT, so the option could have been omitted. For clarity, though, the menus will usually set all the options that they control.

Options FPROB (short for FPROBABILITY) and TPROB (short for TPROBABILITY) are set to yes when the F-probability and T-probability boxes, respectively, are checked. When FPROB=yes, Genstat provides probabilities for variance ratios (see the *Summary of analysis* below). When TPROB=yes, Genstat provides probabilities for t-statistics (see the *Estimates of parameters*). The default for both options is no.

The CONSTANT option is defined by the Estimate Constant Term box, and specifies whether the *constant* or intercept (*a* in the equation above) is included in the model. The setting CONSTANT=estimate is the default. The alternative setting, omit, would constrain the fitted line to pass through the origin (that is, the response will be zero when the explanatory is zero). However, this may be unwise if the data are close to the origin, as the analysis would still be based on the assumptions that the variability about the line is constant for the whole range of the data, and that the relationship is linear right down to the origin.

The output is shown below.

```
 4  "Simple Linear Regression"
 5  MODEL Pressure
 6  FIT Age
```

Regression analysis

```
        Response variate:  Pressure
            Fitted terms:  Constant, Age
```

Summary of analysis

```
Source              d.f.          s.s.          m.s.        v.r.
Regression             1        2647.7       2647.69      169.73
Residual              36         561.6         15.60
Total                 37        3209.3         86.74

Percentage variance accounted for 82.0
Standard error of observations is estimated to be 3.95.
```

Estimates of parameters

```
Parameter              estimate            s.e.         t(36)
Constant                  63.04            2.02         31.27
Age                      0.4983          0.0382         13.03
```

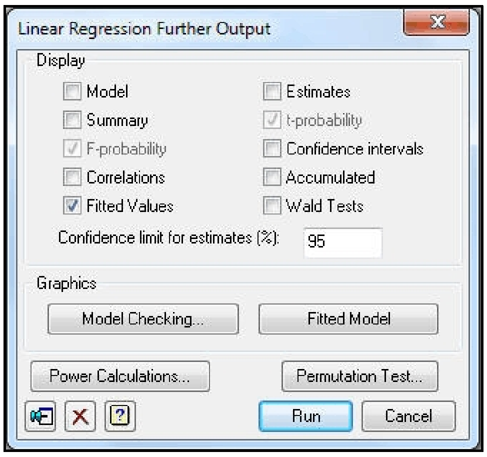
The model setting of the PRINT option gives a description of the model, listing the response variable and the fitted terms: these are the explanatory variable and the constant (or intercept) term.

The *Summary of Analysis*, produced by the summary setting of the PRINT option, contains an analysis of variance to assess the regression. The column headed *m.s.* (mean square) shows how much of the variance of the observations can be explained by the linear dependence on age (Regression), and how much is left over (Residual). The variance ratio (*v.r.*) is the ratio of the mean squares, and may be used to test formally whether there is a significant linear relationship. As the menu has set option FPROB (i.e. FPROBABILITY) to yes, a column of probabilities is included. The heading "F pr." reminds you that the probabilities are calculated by assuming that the variance ratio has an F distribution. A variance ratio as large as the one in this analysis indicates a significant relationship at the 0.1% level of significance (corresponding to probability 0.001).

The *percentage variance accounted for* is a summary of how much of the variability of this set of response measurements can be explained by the fitted model. It is the difference between residual and total mean squares expressed as a percentage of the total mean square. When expressed as a proportion rather than a percentage, this statistic is called the *adjusted* R^2; it is not quite the same as R^2, the squared coefficient of correlation. The adjustment takes account of the number of parameters in the model compared to the number of observations.

The estimates setting displays the estimates of the parameters in the model. So, for example, you can see that blood-pressure rises on average by 0.4983 units for each year of age, with a standard error of 0.0382. The corresponding t-statistic is 13.03 with 36 degrees of freedom. As the menu has set option TPROB (i.e. TPROBABILITY) to yes, a column of probabilities is included. The probability is less than 0.001, indicating that there is a significant association between pressure and age, as we might expect from the graph in Figure 6.2.

You can obtain further output from the analysis by using the Linear Regression Further Output menu, which is obtained by clicking on the Further Output button of the Linear Regression menu. In Figure 6.5, we are asking to print the fitted values. This uses the `RDISPLAY` directive, which has options `PRINT`, `FPROBABILITY` and `TPROBABILITY`, just like those of `FIT`. The output below is generated by the `fittedvalues` setting of the `PRINT` option.



☐ Model
☐ Summary
✓ F-probability
☐ Correlations
✓ Fitted Values
☐ Estimates
✓ t-probability
☐ Confidence intervals
☐ Accumulated
☐ Wald Tests

Confidence limit for estimates (%): 95

Figure 6.5

```
  7  RDISPLAY [PRINT=fittedvalues]

Regression analysis

Fitted values and residuals

                                              Standardized
Unit              Response     Fitted value       residual    Leverage
1                    82.17            77.00           1.36       0.072
2                    88.19            85.97           0.57       0.028
3                    89.66            94.44          -1.24       0.042
4                    81.45            80.98           0.12       0.045
5                    85.16            83.97           0.31       0.032
6                    89.77            92.44          -0.69       0.034
7                    89.11            89.95          -0.22       0.028
8                   107.96           101.41           1.74       0.095
9                    74.82            73.51           0.35       0.105
10                   83.98            91.45          -1.92       0.031
11                   92.95            86.46           1.67       0.027
12                   79.51            79.99          -0.12       0.050
13                   87.86            88.46          -0.15       0.026
14                   76.85            76.50           0.09       0.076
15                   76.93            75.00           0.51       0.090
16                   87.09            83.47           0.93       0.034
17                   97.55            95.93           0.42       0.050
18                   92.04            97.43          -1.41       0.060
19                  100.85            98.92           0.51       0.072
20                   96.30            92.94           0.87       0.036
21                   86.42            87.96          -0.39       0.026
22                   94.16            91.45           0.70       0.031
23                   78.12            78.99          -0.23       0.057
24                   89.06            92.44          -0.87       0.034
25                   94.58            99.92          -1.41       0.080
26                  103.48           101.41           0.55       0.095
27                   81.30            83.47          -0.56       0.034
28                   83.71            80.98           0.71       0.045
29                   68.38            73.01          -1.24       0.111
30                   86.64            86.46           0.05       0.027
```

```
31                      87.91      88.46      -0.14    0.026
32                      86.42      91.45      -1.29    0.031
33                     103.87      97.43       1.68    0.060
34                      83.76      80.98       0.72    0.045
35                      84.35      89.95      -1.44    0.028
36                      68.64      75.00      -1.69    0.090
37                     100.50      93.44       1.82    0.038
38                     100.42     102.91      -0.67    0.111

Mean                    87.95      87.95       0.00    0.053
```

The *fitted values* are the values predicted by the model for each observation; that is, $a + b \times x_i$. Instead of displaying the *simple residuals*, e_i, these values have been divided by their standard error: the resulting *standardized residuals* should be like observations from a Normal distribution with unit variance, if the assumptions made in this analysis are valid. The *leverage* values indicate how influential each observation is: a large value indicates that the fit of the model depends strongly on that observation.

The statistics that are printed in the *Summary of analysis* are controlled by the `SELECTION` option of the `FIT` and `RDISPLAY` directives. The default for an ordinary linear regression is to give the percentage variance accounted for and the standard error of an individual observation (see output from the `FIT` statement earlier in this section). There are several alternative statistics, but these cannot be selected from the regression menus. However, you can give your own `RDISPLAY` command after fitting the regression by the menu. For example,

```
    RDISPLAY [PRINT=summary; SELECTION=%ss]
```

to print the percentage of the total sum of squares accounted for by the regression. Alternatively, you could copy the `FIT` statement from the Input log, and edit it in a new text window to rerun the original analysis, but with a different summary.

```
    FIT [PRINT=model,summary,estimates; CONSTANT=estimate;\
        FPROB=yes; TPROB=yes; SELECTION=%ss] Age
```

Clicking on the Fitted Model button in the Linear Regression Further Output menu displays the fit graphically. With a simple linear regression, like this, we do not need to choose the explanatory variate to use for the x-axis. So there is no subsequent menu; the plot just appears, as shown in Figure 6.6.

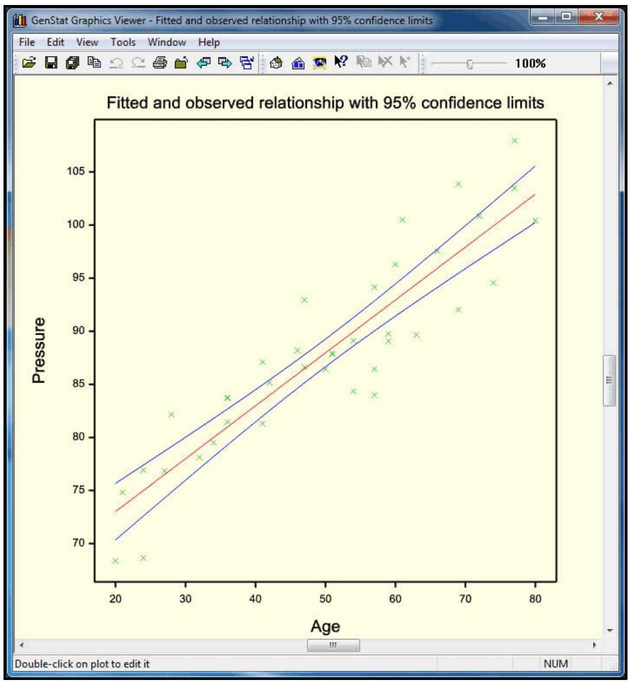


Figure 6.6

The menu uses the `RGRAPH` procedure, with the statement

```
RGRAPH [CIPLOT=yes]
```

The option `CIPLOT=yes` provides 95% confidence limits for the fitted line. You can change the probability for the confidence interval using the `CIPROBABILITY` option (but not in the regression menus).

Clicking on the Model Checking button in the Linear Regression Further Output menu produces the Model Checking menu (Figure 6.7). This uses the `RCHECK` procedure to plot various graphs to check the assumptions of the analysis visually. The first parameter, `YSTATISTIC`, indicates whether you want to plot residuals (the default), leverage values or *Cook's statistics* (a combination of the residual and leverage information); this is set by the Display in Graph boxes in the menu. The second parameter `XSTATISTIC`, which is set by the Type of Graph boxes, chooses the type of graph. Full details are in the on-line help, but the default setting `composite` covers the main requirements. This produces a composite display of four of the available graphs:

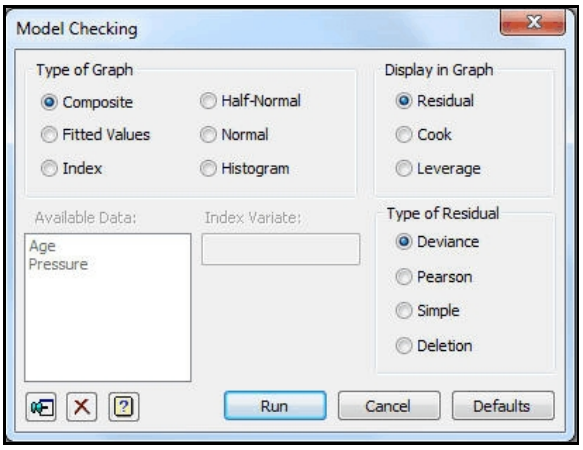


Figure 6.7

- a histogram of the residuals, so that you can check that the distribution is symmetrical and reasonably Normal,
- a plot of residuals against fitted values, so that you can check whether the residuals are roughly symmetrically distributed with constant variance,

- a *Normal plot* which plots the ordered residuals against Normal distribution statistics – if they lie roughly on a straight line, the residuals are roughly Normally distributed, and
- a *half-Normal plot* which does the same for the absolute values of the residuals, and can be more useful for small sets of data.

So to assess the fit of the linear regression of pressure on age, we need only give the statement

```
RCHECK
```

The resulting plots are in Figure 6.8. These indicate that the variance seems unrelated to the size of the observation, but that the distribution seems to be more constrained than the Normal: the largest residuals are a little smaller than would be expected from a Normal distribution. Experience shows the analysis is robust to small departures from Normality. However, we should be cautious in interpreting the F-statistics and t-statistics (which rely on the assumption of Normality), if the histogram looks very non-Normal.

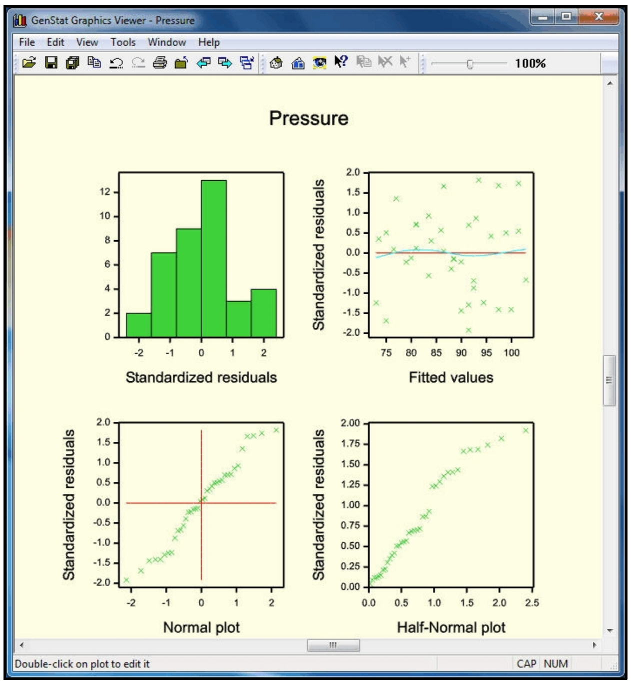
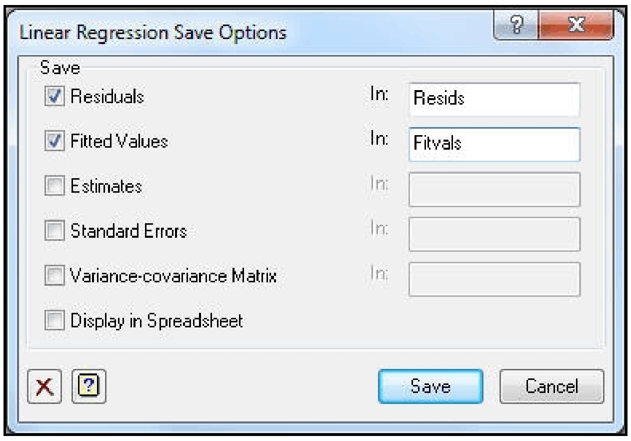


Figure 6.8

One of Genstat's strengths is that virtually anything that can be printed as output from an analysis can also be saved in a suitable Genstat data structure. So you can use regression, for example, as just one part of a more general analysis.

To save output using the Genstat menus, you click on the Save button in the Linear Regression menu (Figure 6.3) to open the Linear Regression Save Options menu shown in Figure 6.9. The menu accesses the most commonly-needed components. The `RKEEP` directive, which it uses, is far more comprehensive. For example, the menu settings in Figure 6.9 will generate the statement below.


Linear Regression Save Options

Save

☑ Residuals	In:	Resids
☑ Fitted Values	In:	Fitvals
☐ Estimates	In:	
☐ Standard Errors	In:	
☐ Variance-covariance Matrix	In:	
☐ Display in Spreadsheet		

Figure 6.9

```
RKEEP RESIDUALS=Resids; FITTEDVALUES=Fitvals
```

This uses the RESIDUALS and FITTEDVALUES parameters of RKEEP to save the residuals and fitted values in variates Resids and Fitvals. Other parameters of RKEEP can, for example, save leverages and standard errors of the fitted values:

```
RKEEP RESIDUALS=Resids; FITTEDVALUES=Fitvals;\
      LEVERAGES=Levs; SEFITTEDVALUES=Sefitvals
```

Full details are in the *Guide to the Genstat Command Language, Part 2 Statistics*, Section 3.1.4.

In the remainder of this chapter we will concentrate on the regression commands, rather than the menus. However, you can find descriptions of how to use the menus for the later analyses, in Chapter 1 of the *Guide to Regression, Nonlinear and Generalized Linear Models in Genstat*.

6.2 Practical

An absorptiometer was used to measure the absorption of light passing through suspensions that contained different numbers of cells. It was intended to estimate the number of cells in future suspensions by the rapid light absorption method, so it was decided to fit a regression of cell counts on light absorption. The data are available in the spreadsheet file Absorb.gsh, where *X* is the absorptiometer reading and *Y* the cell count (10^8/ml). This example comes from *Experimentation in Biology* by Ridgman (1975, Blackie, Glasgow).

Load the data into Genstat and fit a linear regression of cell count on absorptiometer reading.

Produce a graphical display of the regression.

Copy the FIT statement from the Input log, edit it to suppress the warnings about the leverages of the points, and run the command again. (Hint: use the NOMESSAGES option of FIT.)

6.3 Multiple linear regression

In multiple linear regression you have several explanatory variables. You can fit these by specifying a list of variates, rather than a single variate, as the parameter of the FIT directive. However, there is an extra problem, that you need to decide which ones are needed in the model. So you need to be able to explore models, comparing alternative variables or sets of variables, as well as to display and check the model that you finally select. Section 1.11 of the *Guide to Regression, Nonlinear and Generalized Linear Models in Genstat* shows how to do this using the Change Models menu. Below we show the equivalent commands..

We illustrate this approach with a short set of data from a production plant, on page 352 of *Applied Regression Analysis* by Draper & Smith (1981, Wiley, New York). Information was collected over 17 months on variables possibly associated with water usage: the average temperature, the amount of production, the number of operating days and the number of employees. The data are available in the spreadsheet file Water.gsh. (So we can load them into Genstat using SPLOAD in the usual way.)

Before you start to explore different regression models, it is best to use the TERMS directive to define the *maximal model*, that is the most complicated model that you may want to fit. Genstat can then define a common set of units for the regression (omitting any units that are have missing x- or y-values), and carry out some initial calculations to make the process more efficient. Here we may want to include any of the variates `Employ`, `Opdays`, `Product` or `Temp`. So we set the parameter of `TERMS` (which defines the maximal model) to list them all; see line 4. We then start by fitting just the constant (line 5).

```
  2  SPLOAD 'Water.gsh'
```

Loading Spreadsheet File

```
Catalogue of file Water.gsh

Sheet Type: vector
    Index          Type       Nval    Name
        1       variate         17    Employ
        2       variate         17    Opdays
        3       variate         17    Product
        4       variate         17    Temp
        5       variate         17    Water

   3  MODEL Water
   4  TERMS Employ,Opdays,Product,Temp
   5  FIT
```

Regression analysis

```
        Response variate:  Water
            Fitted terms:  Constant
```

Summary of analysis

```
Source              d.f.          s.s.          m.s.         v.r.
Regression             0         0.000             *
Residual              16         3.193        0.1995
Total                 16         3.193        0.1995

Percentage variance accounted for 0.0
Standard error of observations is estimated to be 0.447.
```

Message: the following units have large standardized residuals.

```
             Unit        Response        Residual
               16           4.488            2.73
```

Estimates of parameters

```
Parameter             estimate            s.e.      t(16)
Constant                 3.304           0.108      30.49
```

You can use the following directives to modify a regression model:

- `ADD` adds extra terms,

- `DROP` drops terms,
- `SWITCH` adds terms (if absent), or drops them (if already present),
- `TRY` tries potential changes one at a time and then reinstates the original model, and
- `STEP` selects the best terms to add or drop.

They all have a `PRINT` option, like that of `FIT`, to control output. They each have a single parameter to specify the modifications to the model.

Next we use `TRY` to help decide which explanatory variate to fit first. The parameter lists the explanatory variates to try.

```
  6  TRY Employ,Opdays,Product,Temp

Changes investigated by TRY

Change                                d.f.        s.s.        m.s.
+ Employ                                 1       0.545       0.545
+ Opdays                                 1       0.025       0.025
+ Product                                1       1.270       1.270
+ Temp                                   1       0.261       0.261
Residual of initial model               16       3.193       0.200
```

The `PRINT` option of `TRY` has a setting `changes` (in addition to those `FIT`), which acts as the default, and summarizes the effect of each change. The table of changes shows their degrees of freedom, sums of squares and mean squares. For comparison, it also has a line showing the residual from the initial model.

Here it is clear that the best variate to add is `Product`. So we add `Product`, and then use `TRY` to help decide which explanatory variate to fit next.

```
  7  ADD [FPROBABILITY=yes; TPROBABILITY=yes] Product

Regression analysis

     Response variate:  Water
         Fitted terms:  Constant, Product

Summary of analysis

Source          d.f.         s.s.         m.s.       v.r.    F pr.
Regression         1        1.270       1.2702       9.91    0.007
Residual          15        1.922       0.1282
Total             16        3.193       0.1995

Change            -1       -1.270       1.2702       9.91    0.007

Percentage variance accounted for 35.8
Standard error of observations is estimated to be 0.358.

Message: the following units have large standardized residuals.
```

```
                Unit         Response          Residual
                  16            4.488              2.31
```

Message: the following units have high leverage.

```
                Unit         Response          Leverage
                   2            2.828              0.27
                   3            2.891              0.25
```

Estimates of parameters

```
Parameter               estimate           s.e.        t(15)      t pr.
Constant                   2.273          0.339         6.71     <.001
Product                   0.0799         0.0254         3.15     0.007
```

```
    8   TRY Employ,Opdays,Product,Temp
```

Changes investigated by TRY

```
Change                                        d.f.            s.s.            m.s.
+ Employ                                         1           0.563           0.563
+ Opdays                                         1           0.078           0.078
- Product                                       -1          -1.270           1.270
+ Temp                                           1           0.289           0.289
Residual of initial model                       15           1.922           0.128
```

The messages in the summary from `ADD` warn us about one large residual, and two months with high leverage. So if this was our final model, we would have to be careful in interpreting the results if we suspected that these two months were special in some way.

Continuing, though, the possible changes are to add `Employ`, `Opdays` or `Temp`, or to drop `Product`. So the changes are positive for `Employ`, `Opdays` and `Temp`, and negative for `Product`. With positive changes, the aim is to pick the variate that has the largest mean square, as these represent additional variation that could be included in the model. With negative changes, the aim is to pick the variate that has the smallest mean square, as these represent variation that could be removed from the model. As we have only just added `Product`, as the best change from the previous model, it is not sensible to take it straight out again, and in our subsequent use of change we will omit the variates that are already in the model. The best variate to add is `Employ`, and we will do that next.

```
  9  ADD [FPROBABILITY=yes; TPROBABILITY=yes] Employ
```

Regression analysis

```
        Response variate:   Water
            Fitted terms:   Constant, Product, Employ
```

Summary of analysis

```
Source             d.f.          s.s.        m.s.      v.r.    F pr.
Regression            2         1.833     0.91664      9.44    0.003
Residual             14         1.359     0.09710
Total                16         3.193     0.19954

Change               -1        -0.563     0.56310      5.80    0.030

Percentage variance accounted for 51.3
Standard error of observations is estimated to be 0.312.
```

Message: the following units have high leverage.

```
            Unit      Response      Leverage
               1         3.067          0.55
              16         4.488          0.39
```

Estimates of parameters

```
Parameter              estimate          s.e.    t(14)   t pr.
Constant                   4.60          1.01     4.55   <.001
Product                  0.2034        0.0559     3.64   0.003
Employy                -0.02157       0.00896    -2.41   0.030
```

```
  10  TRY Opdays,Temp
```

Changes investigated by TRY

```
Change                                 d.f.         s.s.         m.s.
+ Opdays                                  1        0.168        0.168
+ Temp                                    1        0.184        0.184
Residual of initial model                14        1.359        0.097
```

The next variate to add is `Temp`.

```
 11  ADD [FPROBABILITY=yes; TPROBABILITY=yes] Temp
```

Regression analysis

```
        Response variate:  Water
            Fitted terms:  Constant, Product, Employ, Temp
```

Summary of analysis

```
Source             d.f.          s.s.        m.s.      v.r.    F pr.
Regression            3         2.017     0.67248      7.44    0.004
Residual             13         1.175     0.09040
Total                16         3.193     0.19954

Change               -1        -0.184     0.18417      2.04    0.177

Percentage variance accounted for 54.7
Standard error of observations is estimated to be 0.301.
```

```
Message: the following units have high leverage.
             Unit        Response        Leverage
                1           3.067            0.59
```

```
Estimates of parameters

Parameter            estimate         s.e.     t(13)   t pr.
Constant                 3.87         1.10      3.51   0.004
Product                0.1933       0.0544      3.56   0.004
Employ               -0.01968      0.00874     -2.25   0.042
Temp                  0.00804      0.00563      1.43   0.177
```

Finally we add `Opdays`.

```
  12  ADD [PRINT=estimates,accumulated; FPROBABILITY=yes; TPROBABILITY=yes]\
  13      Opdays
```

```
Regression analysis

Estimates of parameters

Parameter            estimate         s.e.     t(12)   t pr.
Constant                 6.36         1.31      4.84   <.001
Product                0.2117       0.0455      4.65   <.001
Employ               -0.02182      0.00728     -3.00   0.011
Temp                  0.01387      0.00516      2.69   0.020
Opdays                -0.1267       0.0480     -2.64   0.022

Accumulated analysis of variance

Change                   d.f.         s.s.         m.s.       v.r.    F pr.
+ Product                   1      1.27017      1.27017      20.49    <.001
+ Employ                    1      0.56310      0.56310       9.08    0.011
+ Temp                      1      0.18417      0.18417       2.97    0.110
+ Opdays                    1      0.43139      0.43139       6.96    0.022
Residual                   12      0.74380      0.06198

Total                      16      3.19263      0.19954
```

The `accumulated` setting of the `PRINT` option produces an "accumulated" analysis of variance table that summarizes all the changes that have been made. Notice that all the additions were significant apart from `Temp`. However, the t-statistic for the regression coefficient for `Temp`, in the table of parameter estimates, is significant showing that `Temp` *is* needed in the final model. (The t-statistics show the effect of fitting each variate after every other one.) This illustrates an important aspect of multiple regression: the explanatory variates are often correlated. So the order in which you fit them can be important. To reinforce this point, we use `DROP` to remove `Temp` from the model.

```
 14  DROP [PRINT=accumulated; FPROBABILITY=yes] Temp
```

Regression analysis

Accumulated analysis of variance

```
Change                   d.f.         s.s.         m.s.      v.r.    F pr.
+ Product                   1      1.27017      1.27017     20.49    <.001
+ Employ                    1      0.56310      0.56310      9.08    0.011
+ Temp                      1      0.18417      0.18417      2.97    0.110
+ Opdays                    1      0.43139      0.43139      6.96    0.022
Residual                   12      0.74380      0.06198
- Temp                     -1     -0.44780      0.44780      7.22    0.020

Total                      16      3.19263      0.19954
```

The F statistic for remove `Temp` from the model assesses the difference between a model containing `Employ`, `Opdays`, `Product` and `Temp`, and one containing only `Employ`, `Opdays` and `Product`. So it shows the effect of adding `Temp` to the model last. The probability of 0.02 is the same as the t-probability, verifying that `Temp` really is needed in the model. (Mathematically, an F statistic for an explanatory variate is the square of the t-statistic for its estimate, so the two probabilities must be the same.)

With this example the conclusion is clear: all the explanatory variates are needed in the model. However, with other data set, you may need to try several orders of fitting, and you may end up with more than one plausible model. The `RSEARCH` procedure provides ways of doing this automatically, including the ability to try all possible subsets of explanatory variates. A full description and example is given in Section 3.2.8 of the *Guide to the Genstat Command Language, Part 2 Statistics*.

6.4 Practical

Spreadsheet file `Peru.gsh` (in the Data folder) contains a data set recording blood pressure and physical characteristics of some Peruvian Indians (see McConway, Jones & Taylor 1999, *Statistical Modelling using GENSTAT*, Arnold, London, Section 6.2). The aim is to see whether blood pressure, `sbp`, can be explained effectively by regression models involving the physical variables. Use the `TRY` directive to build a model containing up to two variables.

Can that model be improved by adding further variables?

6.5 Regression with groups

This section introduces the types of model that you can fit when you have factors in a regression model. Suppose you have one explanatory factor and one explanatory variate. You may then want to see how the regression line for the explanatory variate is the same within all the groups defined by the factor. Or perhaps the slope is the same for all the groups but the intercepts differ. Or perhaps the lines have different slopes and different intercepts. In this section we shall use commands to do the analysis. Section 1.15 of the

Guide to Regression, Nonlinear and Generalized Linear Models shows how to do this with menus.

We shall illustrate the possibilities using the sulphur pollution data in spreadsheet file `Sulphur.xls`. from Section 4.3. First, we fit a simple linear regression on the wind speed.

```
  2  IMPORT 'C:/Program Files/Gen18ed/Data/Sulphur.xls'
```

Loading Spreadsheet File

```
Catalogue of file C:\USERS\ROGER\APPDATA\LOCAL\TEMP\GENSTAT\G150D52628.GWB

Sheet Title: Sheet1
Description: Data read from C:\Program Files\Gen18ed\Data\Sulphur.xls [GENSTAT
DATA]A2:D115
Sheet Type: vector
    Index              Type      Nval    Name
        1           variate       114    Sulphur
        2           variate       114    Windsp
        3            factor       114    Winddir
        4            factor       114    Rain

    3   MODEL Sulphur
    4   FIT [FPROBABILITY=yes; TPROBABILITY=yes] Windsp
```

Regression analysis

```
        Response variate:   Sulphur
            Fitted terms:   Constant, Windsp
```

Summary of analysis

```
Source               d.f.          s.s.          m.s.       v.r.    F pr.
Regression              1          935.        934.52       9.49    0.003
Residual              111        10932.         98.48
Total                 112        11866.        105.95

Percentage variance accounted for 7.0
Standard error of observations is estimated to be 9.92.
```

Message: the following units have large standardized residuals.

```
             Unit       Response        Residual
               20          49.00            3.57
               98          43.00            3.88
```

Message: the following units have high leverage.

```
             Unit       Response        Leverage
               30           3.00           0.075
               72           5.00           0.051
               95          14.00           0.054
              100          25.00           0.051
```

Estimates of parameters

```
Parameter             estimate           s.e.     t(111)     t pr.
Constant                 17.03           2.33       7.32     <.001
Windsp                  -0.636          0.207      -3.08     0.003
```

```
  5  RCHECK
```

The model checking plots produced by the RCHECK procedure in line 5 show a very skewed distribution of residuals (see Figure 6.10). So, we shall try transforming the sulphur measurements to logarithms, forming a new variate LogSulphur, containing the logarithms (to base 10) of the sulphur values.

The first value of Sulphur is zero, so the logarithm cannot be calculated. Invalid calculations like these do not cause Genstat to fail. Instead it gives a warning, and sets the result in that unit to a missing value.

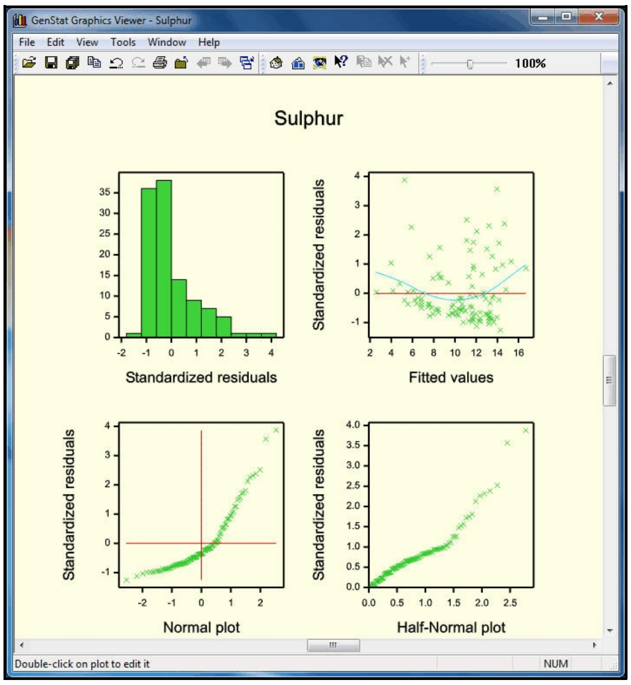


Figure 6.10

```
 6  CALCULATE LogSulphur = LOG(Sulphur)
```

Warning 1, code CA 7, statement 1 on line 6

```
Command: CALCULATE LogSulphur = LOG10(Sulphur)
Invalid value for argument of function.
The first argument of the LOG function in unit  1 has the value 0.0000
```

```
  7  MODEL LogSulphur
  8  TERMS Windsp + Rain + Windsp.Rain
  9  FIT [FPROBABILITY=yes; TPROBABILITY=yes] Windsp
```

Regression analysis

```
        Response variate:  LogSulphur
            Fitted terms:  Constant + Windsp
```

Summary of analysis

```
Source              d.f.          s.s.          m.s.       v.r.     F pr.
Regression             1          1.50        1.4952      10.35     0.002
Residual             110         15.89        0.1445
Total                111         17.39        0.1567

Percentage variance accounted for 7.8
Standard error of observations is estimated to be 0.380.
```

Message: the following units have large standardized residuals.

```
             Unit       Response       Residual
               98          1.633           2.68
```

Message: the following units have high leverage.

```
             Unit       Response       Leverage
               30          0.477          0.076
               72          0.699          0.052
               95          1.146          0.055
              100          1.398          0.051
```

Estimates of parameters

```
Parameter          estimate          s.e.     t(110)     t pr.
Constant             1.1066        0.0892      12.41     <.001
Windsp             -0.02557       0.00795      -3.22     0.002
```

```
  10  RCHECK
```

The residual plot, in Figure 6.11, shows a much more symmetric distribution of residuals, with no evidence that the variance is changing with the size of the sulphur measurement. The plot does show up the imprecise recording of the sulphur measurements as integers: the apparent diagonal lines of points correspond to sulphur measurements with equal values.

We will now fit a sequence of regression models, to assess how the regression changes according to whether or not it was raining. Again we will use the `TERMS` directive to define the most complicated model that we may want to fit.

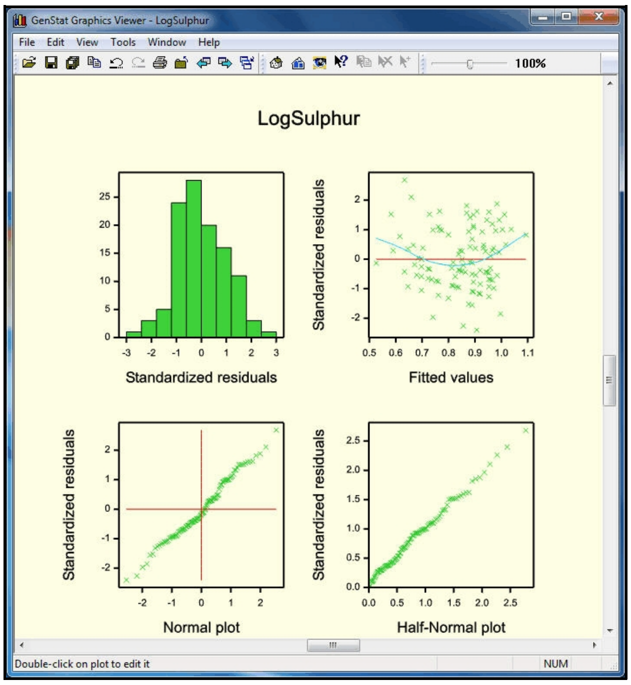

GenStat Graphics Viewer - LogSulphur
File Edit View Tools Window Help
100%
LogSulphur
Standardized residuals
Fitted values
Normal plot
Half-Normal plot
Double-click on plot to edit it
NUM

Figure 6.11

The parameter of `TERMS`, and the regression-fitting directive (`FIT`, `ADD` etc.) is actually a *model formula*. A full definition of model formulae is given in

Section 3.9, so we give only a brief description here. However, they enable much more complicated models to be defined than the simple and multiple linear regressions in Sections 6.1 and 6.3.

In its simplest form, a model formula is a list of *model terms* separated by the operator `+`. Each model term specifies a set of parameters in a statistical model. It may be a single variate (representing a regression coefficient) or a factor (representing a set of main effects). Alternatively, it may consist of several variates and/or factors separated by the operator dot (`.`), and define a *higher-order* term like an interaction. Commas can be used instead of pluses if the model terms are all single variates (and/or single factors). So we could have put

```
    TRY Employ + Opdays + Product + Temp
```

instead of

```
    TRY Employ,Opdays,Product,Temp
```

in Section 6.3.

The other operators provide ways of specifying a formula more succinctly, and of representing its structure more clearly. Crossed (or factorial) relationships can be specified by the star operator (`*`). The formula

```
    Windsp * Rain
```

in line 8, is expanded by Genstat automatically to become

```
    Windsp + Rain + Windsp.Rain
```

So we could have achieved exactly the same effect by specifying

```
    TERMS Windsp + Rain + Windsp.Rain
```

in line 8.

The regression directives all have an option, `FACTORIAL`, which can be used to specify the maximum order (that is, number of factors and/or variates) in the terms to be fitted in the analysis; the default is 3.

Fitting `Windsp` in line 9 has given a single regression line, as usual. Adding the main effect of the factor `Rain`, in line 10 below, will change the model so that there is a different intercept (i.e. constant) for each level of `Rain`. Finally, adding the interaction `Windsp.Rain`, in line 13, will give a different slope (i.e. regression coefficient) for each level of `Rain`. So this will carry out an *analysis of parallelism*, going from a single line, to parallel lines and then finally to non-parallel lines.

```
 11  ADD [FPROBABILITY=yes; TPROBABILITY=yes] Rain
```

Regression analysis

```
      Response variate:   Logsulphur
          Fitted terms:   Constant + Windsp + Rain
```

Summary of analysis

```
Source           d.f.          s.s.          m.s.       v.r.    F pr.
Regression          2          1.89        0.9442       6.64    0.002
Residual          109         15.50        0.1422
Total             111         17.39        0.1567

Change             -1         -0.39        0.3933       2.77    0.099
```

```
Percentage variance accounted for 9.2
Standard error of observations is estimated to be 0.377.
```

Message: the following units have high leverage.

```
            Unit        Response        Leverage
              30           0.477           0.102
              72           0.699           0.073
```

Estimates of parameters

```
Parameter            estimate            s.e.     t(109)     t pr.
Constant               1.1235          0.0891      12.62     <.001
Windsp               -0.02193         0.00818      -2.68     0.008
Rain yes              -0.1240          0.0745      -1.66     0.099
```

```
Parameters for factors are differences compared with the reference level:
                     Factor   Reference level
                       Rain   no
```

```
  12   RGRAPH
```

The effect of rainfall is represented here by the difference between dry and wet days: that is, by comparing level `yes` of the factor `Rain` to its *reference level* `no`. (By default the reference level is the first level of the factor, but the Column Attributes spreadsheet menu allows you to choose other levels.) So the model is

$$Logsulphur = a + b \times Windsp$$

for dry days, and

$$Logsulphur = a + d + b \times Windsp$$

for wet days.

The model thus consists of two *parallel* regression lines (Figure 6.12). The estimates show that rainfall decreases the sulphur on average by 25% (antilog(–0.1240) = 75%), but this effect is not statistically significant there is still a large amount of unexplained variation in the sulphur measurements. This version of the model is very convenient if you want to make comparisons with the reference level (which may, for example, represent a standard set of conditions or treatment). However, we show later in this section how you can obtain the alternative version with a parameter in the model for each intercept.

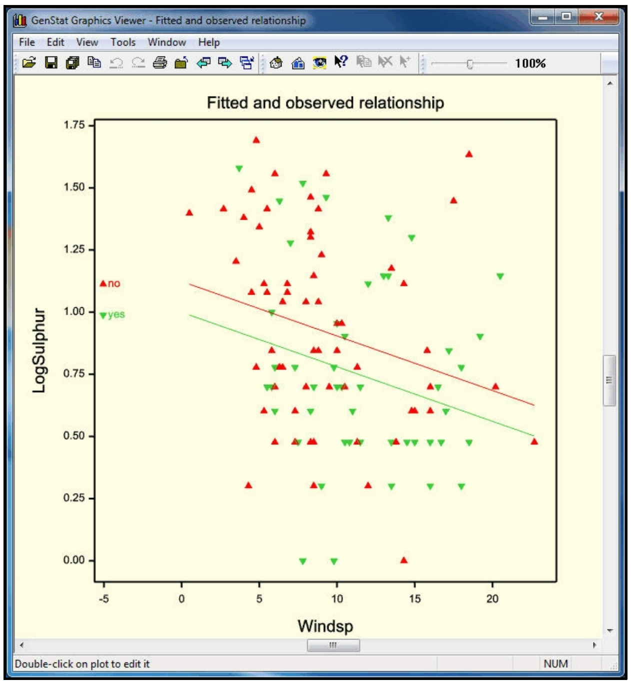


Figure 6.12

We can see whether the linear effect of wind speed is different in the two categories of rainfall by adding the interaction `Windsp.Rain`.

```
 13  ADD [FPROBABILITY=yes; TPROBABILITY=yes] Windsp.Rain
```

Regression analysis

```
        Response variate:  Logsulphur
            Fitted terms:  Constant + Windsp + Rain + Windsp.Rain
```

Summary of analysis

```
Source          d.f.        s.s.        m.s.      v.r.    F pr.
Regression         3        1.92      0.6402      4.47    0.005
Residual         108       15.47      0.1432
Total            111       17.39      0.1567

Change            -1       -0.03      0.0323      0.23    0.636

Percentage variance accounted for 8.6
Standard error of observations is estimated to be 0.378.
```

Message: the following units have large standardized residuals.

```
            Unit      Response      Residual
              98         1.633          2.61
```

Message: the following units have high leverage.

```
            Unit      Response      Leverage
              30         0.477         0.160
              72         0.699         0.112
```

```
          95          1.146        0.111
         104          1.580        0.093
```

Estimates of parameters

```
Parameter                 estimate         s.e.     t(108)    t pr.
Constant                     1.153        0.109      10.57    <.001
Windsp                     -0.0252       0.0107      -2.36    0.020
Rain yes                    -0.208        0.193      -1.08    0.283
Windsp.Rain yes             0.0079       0.0167       0.47    0.636

Parameters for factors are differences compared with the reference level:
                Factor   Reference level
                  Rain   no
```

```
  14  RGRAPH
```

The output now shows the slope of the regression for dry days, titled `Windsp`, and the difference in slopes between wet and dry, titled `Windsp.Rain yes`. So again we can see immediately that the difference between the slopes is small and not significant. The graph of the fitted model is shown in Figure 6.13.

We can use the accumulated analysis of variance table to help decide how complicated a model is needed. We could have done this by setting the `PRINT` option in the `ADD` statement (line 13) like this:

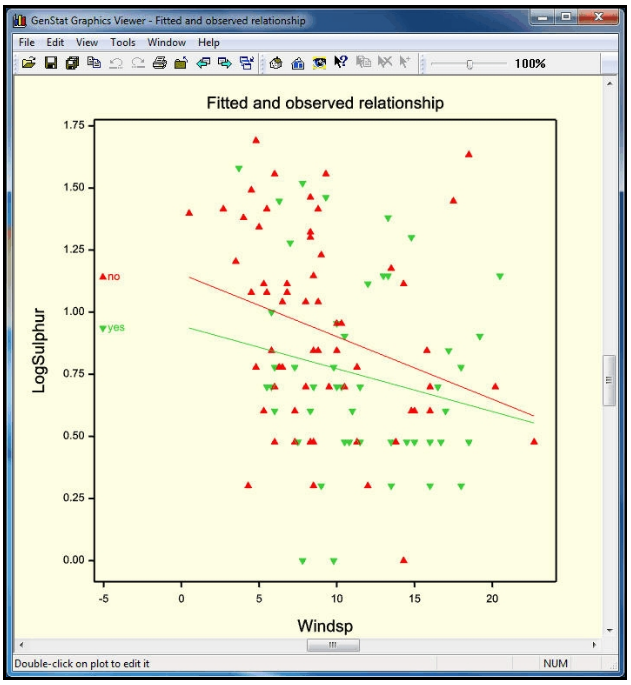

GenStat Graphics Viewer - Fitted and observed relationship
File Edit View Tools Window Help
100%
Fitted and observed relationship
LogSulphur
Windsp
no
yes
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-5
0
5
10
15
20
Double-click on plot to edit it
NUM

Figure 6.13

```
ADD [PRINT=#,accumulated; FPROBABILITY=yes; TPROBABILITY=yes] \
    Windsp.Rain
```

Remember that the special character `#` can be used to represent the default setting, as explained in Section 3.1. Alternatively (as below) we can use `RDISPLAY`.

```
15  RDISPLAY [PRINT=accumulated; FPROBABILITY=yes]
```

Regression analysis

Accumulated analysis of variance

```
Change                          d.f.         s.s.         m.s.      v.r.    F pr.
+ Windsp                           1       1.4952       1.4952     10.44    0.002
+ Rain                             1       0.3933       0.3933      2.75    0.100
+ Windsp.Rain                      1       0.0323       0.0323      0.23    0.636
Residual                         108      15.4677       0.1432

Total                            111      17.3884       0.1567
```

Here a Common line (in fact, a simple linear regression) would be enough. However, we will refit the parallel lines to illustrate how to get a parameter for each intercept, rather than parameters for differences from the reference level.

```
 16  TERMS [FULL=yes] Windsp + Rain + Windsp.Rain
 17  FIT [CONSTANT=omit; FPROBABILITY=yes; TPROBABILITY=yes] \
 18      Windsp + Rain
```

Regression analysis

```
        Response variate:  Logsulphur
            Fitted terms:  Windsp + Rain
```

Summary of analysis

```
Source          d.f.           s.s.           m.s.       v.r.    F pr.
Regression         2           1.89         0.9442       6.64    0.002
Residual         109          15.50         0.1422
Total            111          17.39         0.1567

Change            -1          -0.39         0.3933       2.77    0.099

Percentage variance accounted for 9.2
Standard error of observations is estimated to be 0.377.
```

Message: the following units have high leverage.

```
            Unit        Response        Leverage
              30           0.477           0.102
              72           0.699           0.073
```

Estimates of parameters

```
Parameter             estimate             s.e.     t(109)     t pr.
Windsp                -0.02193          0.00818      -2.68     0.008
Rain no                 1.1235           0.0891      12.62     <.001
Rain yes                 1.000            0.109       9.14     <.001
```

The option `Full=yes` in the `TERMS` statement (line 16) ensures that Genstat allocates a parameter to every level of each factor in the model (otherwise it excludes their reference levels). In line 17, we set option CONSTANT=omit to omit the constant. So the analysis now has an (i.e. constant) for each level of Rain. The t-statistics now assess whether each of those constants is zero, rather than their difference. We could fit a regression coefficient for each level of Rain by also omitting Windsp i.e.

```
FIT [CONSTANT=omit; FPROBABILITY=yes; TPROBABILITY=yes]\
    Rain + Windsp.Rain
```

6.6 Practical

Spreadsheet file Calcium.gsh (in the Data folder) contains a data set, discussed in Sections 4.6 and 6.4 of McConway, Jones & Taylor (1999, *Statistical Modelling using GENSTAT*, Arnold, London), which contains information about mortality in 61 towns. The aim is to study how this relates to the calcium concentration in the drinking water supply.

Fit a linear regression of mortality on calcium. Are the assumptions satisfied better by transforming mortality to logarithms?

Perform a simple linear regression with groups to see how the relationship is affected by region.

6.7 Other regression facilities

In addition to the linear regression models, described in this chapter, Genstat covers many other types of regression model, including smoothing splines and locally weighted regression. It also has commands and menus for other regression analyses including

- all-subsets regression,
- screening tests,
- polynomial regression,
- nonlinear curves (both standard and user-defined),
- split-line regression,
- generalized linear models (for non-Normal data like counts and proportions),
- ordinal regression,
- generalized additive models,
- generalized linear mixed models,
- hierarchical generalized linear models,
- regression trees and forests, and
- quantile regression.

Menus for some of these are described in the *Guide to Regression, Nonlinear and Generalized Linear Models in Genstat*. All the main commands are described, with examples, in the *Guide to the Genstat Command Language, Part 2 Statistics*, Chapter 3. (These Guides can both be accessed from within Genstat *for Windows* by selecting sub-options of the Genstat Guides option of the Help menu on the menu bar.)

The other regression commands are in the *Genstat Reference Manual, Part 3 Procedure Library PL26* (to open, select the Procedure Library sub-option of the Genstat Reference Manual option of the Help menu on the menu bar).

7 Analysis of variance

Genstat has very comprehensive facilities for analysis of variance. In this chapter, we introduce the main commands, and show how they are used by the menus. However, we do not attempt to cover all the possibilities, nor to describe the underlying statistical theory. A full description of the commands (and theory) is in the *Guide to the Genstat Command Language, Part 2 Statistics*, Chapter 4, while the menus (and theory) are described in the *Guide to Anova and Design in Genstat*. These can be accessed from within Genstat *for Windows* by selecting sub-options of the Genstat Guides option of the Help menu on the menu bar.

Analyses with one or two treatment factors, and optionally also a block factor, can be analysed very easily by the `A2WAY` procedure, which is used by the One- and two-way Anova menu (see Sections 1.3 and 3.1 of the *Guide to Anova and Design in Genstat*). If that is all that you want you should read its description in the *Genstat Reference Manual, Part 3 Procedure Library PL25* or the on-line help.

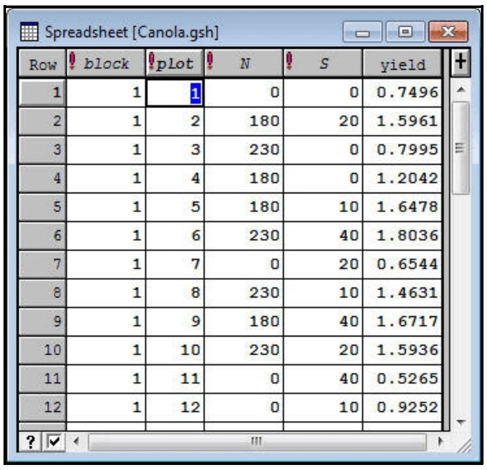
In this chapter, we describe the `ANOVA` directive, which is used by the general Analysis of Variance menu, and handles a much wider set of situations. Technically it analyses data from *generally balanced* designs. These include most of the commonly occurring experimental designs such as randomized blocks, Latin squares, split plots and other orthogonal designs, as well as designs with balanced confounding, like balanced lattices and balanced incomplete blocks. `ANOVA` can itself detect whether or not a design can be analysed, giving diagnostic `AN-1` if the design is unbalanced; so if you are not sure whether or not a particular design is analysable, you can run it through `ANOVA` to check. Unbalanced designs with a single error term can be analysed using the `AUNBALANCED` procedure instead of `ANOVA`; see Section 4.1.10 of the *Guide to the Genstat Command Language, Part 2 Statistics*. Alternatively, if you have several error terms, you can use the `REML` directive; the associated commands are described in Chapter 5 of the *Guide to the Genstat Command Language, Part 2 Statistics*, and the menus are in the *Guide to REML In Genstat*.

So, in this chapter, you will learn about:

- the `TREATMENTSTRUCTURE` directive, which defines the treatment (or fixed) terms for the analysis;
- the `BLOCKSTRUCTURE` directive, which defines the "underlying structure" of the data or, equivalently, the error terms;
- the `ANOVA` directive, which does the analysis;
- how `BLOCKSTRUCTURE`, `TREATMENTSTRUCTURE`, `COVARIATE` and `ANOVA` are used by the Analysis of Variance menu;
- how to use the `BLOCKSTRUCTURE` directive to analyse a randomized-block design;
- how to use the `BLOCKSTRUCTURE` directive to analyse a split-plot design (this information, in Section 7.7 and 7.8, can be skipped if you are interested only in the simpler designs).

7.1 Designs with a single error term

The spreadsheet file Canola.gsh, in the Data folder, contains data from a field experiment to examine the effects of sulphur and nitrogen fertilizers on the yield of canola; see Figure 7.1. So there are two treatment factors, which have been called S and N. The experiment used a randomized-block design, so there is also a factor, here called block, to indicate the block to which each of the experimental plots belonged. The factor plot numbers the plots in each block.

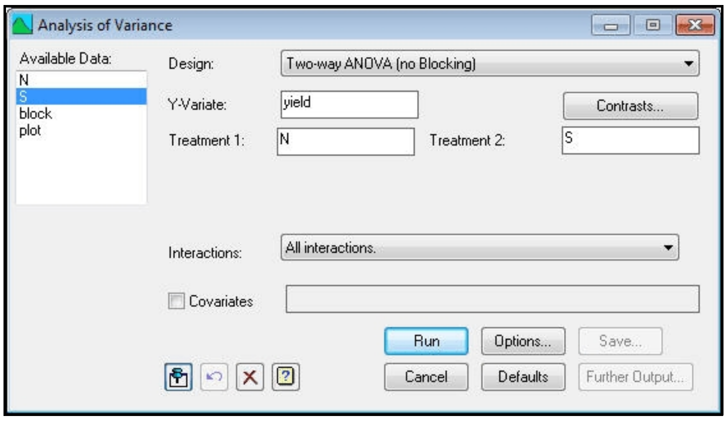

Spreadsheet [Canola.gsh]

Row	block	plot	N	S	yield
1	1	1	0	0	0.7496
2	1	2	180	20	1.5961
3	1	3	230	0	0.7995
4	1	4	180	0	1.2042
5	1	5	180	10	1.6478
6	1	6	230	40	1.8036
7	1	7	0	20	0.6544
8	1	8	230	10	1.4631
9	1	9	180	40	1.6717
10	1	10	230	20	1.5936
11	1	11	0	40	0.5265
12	1	12	0	10	0.9252

Figure 7.1

Initially, we shall ignore the structure of the design, and treat the data as though they came from a completely-randomized design (i.e. one where the treatments were applied to the plots completely at random). So the model has a single error term, representing the underlying random variability of the plots.

In this section we will analyse the design using the general Analysis of Variance menu, and explain the commands that it uses. You open the menu (Figure 7.2) by clicking on the General sub-option of the Analysis of Variance option of the Stats menu on the menu bar. The Design box at the top of the menu is a drop-down list containing many of the standard analyses. The menu customizes itself to provide the appropriate controls for each analysis, and writes Genstat statements to define the models to be fitted. Here we have selected Two-way ANOVA (no Blocking) and so, as well as a box for the y-variate, there are boxes for the two treatment factors (here N and S). If we click on Run, the menu writes the following script:


Analysis of Variance

Available Data: N, S, block, plot

Design: Two-way ANOVA (no Blocking)

Y-Variate: yield

Treatment 1: N

Treatment 2: S

Interactions: All interactions

Covariates

Figure 7.2

```
"Two-way ANOVA (no Blocking)."
BLOCK "No Blocking"
TREATMENTS N*S
COVARIATE "No Covariate"
ANOVA [PRINT=aovtable,information,means; CONTRASTS=7;\
      PCONTRASTS=7; FPROB=yes; PSE=diff] yield
```

Before we can use ANOVA to analyse the data, we must first define the model that is to be fitted. Potentially this has three parts, which can be given in any order

The BLOCKSTRUCTURE directive (or BLOCK for short) defines the "underlying structure" of the design or, equivalently, the *error* terms for the analysis. Its parameter is a model formula (usually called the *block formula*), and it has no options. In simple cases like this, where there is only a single error term, the directive can be omitted. However, the model definitions carry over from one analysis to another unless you restart the session or server (see the Run menu on the menu bar) or you clear all data (see the Data menu). So the menu has included it in the script, with a null setting to cancel any previous definition.

The TREATMENTSTRUCTURE directive specifies the treatment (or *systematic*, or *fixed*) terms for the analysis. Again its parameter is a model formula (usually called the *treatment formula*), and it has no options.

A full definition of model formulae is given in Section 3.9, so we give only a brief description here. In its simplest form, a *model formula* is a list of *model terms* separated by the operator +. Each model term specifies a set of parameters in a statistical model. It may be a single factor (representing a set of main effects). Alternatively, it may consist of several factors separated by the operator dot (.), and define a *higher-order* term like an interaction. The other operators provide ways of specifying a formula more succinctly, or of representing its structure more clearly.

Factorial (or crossed) relationships can be specified by the star operator (*). Here we have a two-way factorial structure and so, in line 3 below, we define

```
TREATMENTSTRUCTURE N * S
```

This is expanded by Genstat automatically to become

```
TREATMENTSTRUCTURE N + S + N.S
```

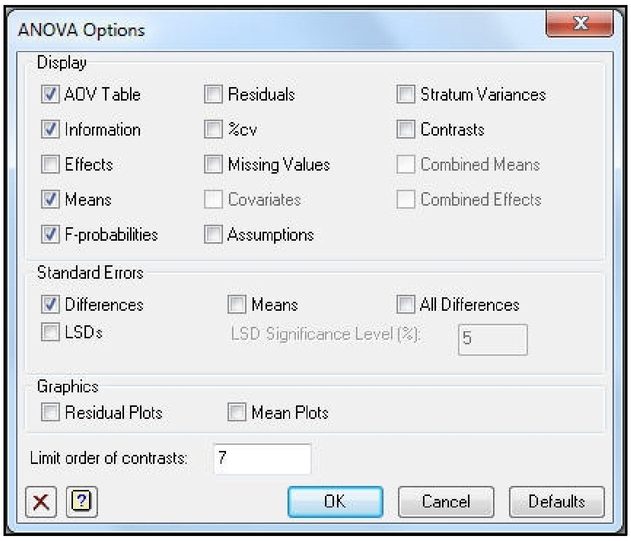
The meanings of terms like N.S depend on context: they represent all the joint effects of the factors in the term that have not been fitted already by earlier terms in the model. Here we have fitted the main effects of N and S, and so N.S represents their interaction.

The ANOVA directive has an option, FACTORIAL, which can be used to specify the maximum order (that is, number of factors) in the treatment terms to be fitted in the analysis. The default is 3, so it does not need to be set in the script above.

The COVARIATE directive specifies the covariates if an analysis of covariance is required. This is not discussed here, but details can be found in Section 3.6 of the *Guide to Anova and Design in Genstat* or Section 4.3 of the *Guide to the Genstat Command Language, Part 2 Statistics*. Here there are no covariates. So the Covariates box in the menu has been left unchecked, and the script contains a null COVARIATE statement to cancel any earlier definition.

Once the model has been defined, the ANOVA directive can be used to perform the analysis of variance. Its first parameter specifies the response or y-variates. The menu allows you to analyse only one variate at a time, but the command allows you to list several. If a y-variate contains missing values, these are estimated in the analysis and the degrees of freedom are adjusted (see the *Guide to the Genstat Command Language, Part 2 Statistics*, Section 4.4). When you list several y-variates, a unit will be treated as missing if it missing in any y-variate or any covariate. So, if the y-variates have different sets of missing units, you may prefer to analyse them in separate statements.

The options of the ANOVA statement are defined by boxes in the ANOVA Options menu (Figure 7.3), which is obtained by clicking on the Options button of the Analysis of Variance menu. Most of the Display boxes are used to define the PRINT option of ANOVA; this can be set to a list of string tokens to select the output to be printed. The exception is the F-probabilities box, which sets the option FPROB (i.e. FPROBABILITY); this controls whether probabilities are given for the variance ratios in the analysis-of-variance table.


ANOVA Options

Display: [✓ AOV Table] [✓ Information] Effects [✓ Means] [✓ F-probabilities] Residuals %cv Missing Values Covariates Assumptions Stratum Variances Contrasts Combined Means Combined Effects

Standard Errors: [✓ Differences] LSDs Means All Differences LSD Significance Level (%): 5

Graphics: Residual Plots Mean Plots

Limit order of contrasts: 7

Figure 7.3

The most commonly used settings of PRINT are:

aovtable	analysis-of-variance table,
information	details of large residuals, non-orthogonality and any aliasing in the model,
covariates	estimated coefficients and standard errors of any covariates,
effects	tables of effects,
residuals	tables of residuals,
contrasts	estimated coefficients of polynomial or other contrasts,
means	tables of means,
%cv	coefficient of variation, and
missingvalues	estimated missing values.

The default is aovtable, information, covariates, means, missingvalues.

In the ANOVA statement above PRINT=aovtable,information,means, giving the output shown below.

```
  2  SPLOAD 'Canola.gsh'
```

Loading Spreadsheet File

```
Catalogue of file Canola.gsh

Sheet Title:
Data imported from Genstat Spreadsheet: C:\Program Files\Gen18ed\Data\Canola.gsh
 on: 16-Mar-2014 12:53:48

Sheet Type: vector
    Index           Type        Nval    Name
        1         factor          36    block
        2         factor          36    plot
        4         factor          36    N
        6          factor        36   S
        7          variate       36   yield
```

Note: Missing indices are used by unnamed or system structures. These store ancillary information, for example factor labels.

```
   3  TREATMENTSTRUCTURE N * S
   4  ANOVA [FPROBABILITY=yes] yield
```

Analysis of variance

```
Variate: yield

Source of variation            d.f.        s.s.        m.s.      v.r.   F pr.
N                                 2     4.59223     2.29611     42.56   <.001
S                                 3     0.97720     0.32573      6.04   0.003
N.S                               6     0.64851     0.10808      2.00   0.105
Residual                         24     1.29476     0.05395
Total                            35     7.51269
```

Message: the following units have large residuals.

```
*units* 17                                             0.527         s.e.  0.190
*units* 22                                            -0.405         s.e.  0.190
```

Tables of means

```
Variate: yield

Grand mean  1.104

         N          0         180         230
                0.601       1.313       1.398

         S          0          10          20          40
                0.829       1.155       1.167       1.266

         N          S           0          10          20          40
         0                  0.560       0.770       0.524       0.552
       180                  0.894       1.289       1.525       1.545
       230                  1.032       1.404       1.454       1.700
```

Standard errors of differences of means

```
Table                            N             S             N
                                                             S
rep.                            12             9             3
d.f.                            24            24            24
s.e.d.                      0.0948        0.1095        0.1896
```

The aovtable setting of the PRINT option produces the analysis-of-variance table. This has a line for each of the three *treatment terms*: N represents the main effect of nitrogen, that is the overall way in which yield responds to nitrogen. Similarly S represents the main effect of sulphur, while N.S represents the interaction between nitrogen and sulphur. The interaction assesses the way in which the effect of nitrogen on yield differs according to the amount of sulphur or, equivalently, the way in which the sulphur effect differs according to the amount of nitrogen. If there is no interaction, we could decide on the best amount of nitrogen to apply without needing to consider how much sulphur will be used (and how much sulphur to use without needing to think about the amount of nitrogen). A more detailed explanation of the meaning of interactions is given in Section 3.1 of the *Guide to Anova and Design in Genstat*.

The information setting of the PRINT option provides details of any large residuals, non-orthogonality and aliasing in the model. As there are none in this analysis, nothing is printed.

The means setting of the PRINT option gives a table of means for every treatment term in the analysis of variance. The Standard Errors boxes in the ANOVA Options menu control the types of standard error that accompany the tables, by setting the PSE option in the ANOVA statement. In Figure 7.3 the Differences box is checked, and the other boxes are unchecked. So the statement contains the setting PSE=diff. Remember that string tokens like differences can always be abbreviated to four characters, so this is equivalent to putting PSE=differences. As this is the default setting, the option could have been omitted. For clarity, though, menus will usually set all the options that they control. Other boxes correspond to the setting means which produces standard errors of means, and LSD which produces least significant differences. The significance level to use in the calculation of the least significant differences can be changed from the default of 5% using the LSDLEVEL option; this corresponds to the box to the right of the LSDs box (which is greyed-out unless the the LSDs box is checked). The All Differences box is more complicated: it uses the AKEEP directive to save all the differences in a symmetric matrix, and then the PRINT directive to print them. If you leave all the Standard Errors boxes unchecked, the menu will set PSE=* and no standard errors will be printed.

The CONTRASTS and PCONTRASTS options could have been omitted as they are not relevant here. They may be needed if you are fitting contrasts of treatment terms. These are described in Section 3.2 of the *Guide to Anova and Design in Genstat* or Section 4.5 of the *Guide to the Genstat Command Language, Part 2 Statistics*.

The other options and parameters of ANOVA are described in the *Guide to the Genstat Command Language, Part 2 Statistics*, Section 4.1.2.

You can obtain further output from the analysis by using the ANOVA Further Output menu, which is obtained by clicking on the Further Output button of the Analysis of Variance menu. In Figure 7.4, we are asking to print the fitted values. This uses the ADISPLAY directive, which has options PRINT, FPROBABILITY and PSE just like those of ANOVA. The output below is generated by the effects setting of their PRINT options. These are the estimates of the parameters in the *linear model* that has been fitted:

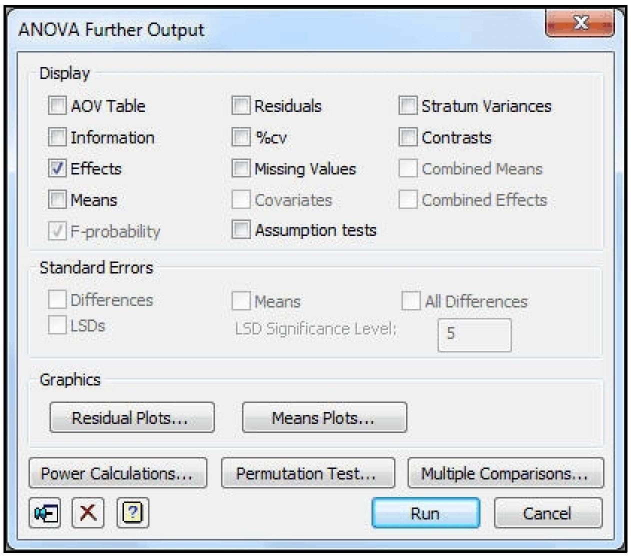


Figure 7.4

$$y_{ijk} = \mu + n_j + s_k + ns_{jk} + \varepsilon_{ijk}$$

in which the parameters

n_j represent the *main effect* of nitrogen (N),
s_k represent the *main effect* of sulphur (S), and
ns_{jk} represent the *interaction* between nitrogen and sulphur (N.S).

These all arise from the treatment model, whereas the grand mean μ and the residuals ε_{ijk} are included automatically. (For further details about linear models for factorial designs see the *Guide to the Genstat Command Language, Part 2 Statistics*, Section 4.1.)

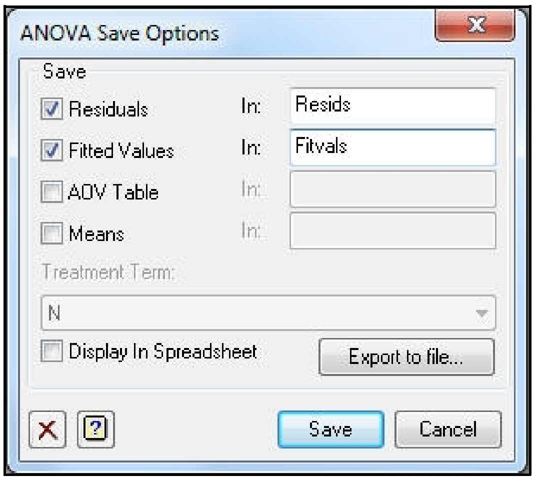
```
  5  ADISPLAY [PRINT=effects]
```

Tables of effects

```
Variate: yield

N effects,  e.s.e. 0.0670,  rep. 12

          N          0        180        230
                -0.503      0.209      0.294

S effects,  e.s.e. 0.0774,  rep. 9

          S          0         10         20         40
                -0.276      0.051      0.063      0.162

N.S effects,  e.s.e. 0.1341,  rep. 3

          N          S          0         10         20         40
          0                 0.234      0.118     -0.141     -0.211
        180                -0.144     -0.075      0.148      0.071
        230                -0.090     -0.044     -0.007      0.141
```

The Residual Plots button of the ANOVA Further Output menu uses the APLOT procedure to display diagnostic plots of the residuals; this is described in Section 7.4. The Means Plots button uses the AGRAPH procedure to display tables of means; see Section 7.6. The other buttons use the APOWER, APERMTEST and AMCOMPARISON procedures, which are explained in Sections 4.11.3, 4.1.6 and 4.1.8 of the *Guide to the Genstat Command Language, Part 2 Statistics*.

To save output using the Genstat menus, you click on the Save button in the Analysis of Variance menu (Figure 7.2) to open the ANOVA Save Options menu shown in Figure 7.5. The menu accesses the most commonly-needed components. The AKEEP directive, which it uses, is far more comprehensive. For example, the menu settings in Figure 7.5 will generate the statement below.


ANOVA Save Options

Save
- [✓] Residuals In: Resids
- [✓] Fitted Values In: Fitvals
- [ ] AOV Table In:
- [ ] Means In:

Treatment Term: N

- [ ] Display In Spreadsheet

Figure 7.5

```
AKEEP RESIDUALS=Resids; FITTEDVALUES=Fitvals
```

This uses the RESIDUALS and FITTEDVALUES options of RKEEP to save the residuals and fitted values in variates Resids and Fitvals. The parameters of AKEEP can save information, like tables of means, for individual terms in the analysis. Full details are in the *Guide to the Genstat Command Language, Part 2 Statistics*, Section 4.6.

In the remainder of this chapter we will concentrate on the analysis of variance commands, rather than the menus. However, you can find descriptions of how to use the menus for the later analyses, in Sections 3.1 and 5.1 of the *Guide to Anova and Design in Genstat*, or Sections 6.5 and 6.8 of the *Introduction to Genstat for Windows*.

7.2 Practical

An experimenter conducted a trial with insecticides for killing ants. Five types of insecticide were used on each of three types of bait. The experimenter measured the time from the release of a colony of ants to when the bait was picked up. Each combination of bait and insecticide was used three times, the order of the observations being decided entirely at random. The data are available in file Ant.gsh in the Data folder. Analyse the experiment.

7.3 Randomized-block designs

The randomized-block design is perhaps the simplest type of designed experiment. In these designs, the experimental units are grouped together into sets known as *blocks* with the aim that units in the same block will be more similar than units in different blocks. Each block contains the same number of replicates of each treatment combination (usually one of each), and the allocation of the treatments is randomized independently within each block. In our example, there is a factor called `block` to indicate the "block" of land to which each plot belonged. In other examples the blocking factor might represent different litters of animals, or different days on which the experiment was conducted, and so on.

In the analysis, the aim is to estimate and remove the between-block differences so that the treatment effects can be estimated more precisely. The conventional way of analysing these designs, which can be seen in many text books, can be achieved in Genstat simply by putting the block factor (here called `block`) at the start of the treatment formula, as we have done in line 6 below.

```
TREATMENTSTRUCTURE block + N * S
```

The variance ratio for `block` compares the variability of the blocks of land with the variability of the individual plots within each block, and its value of 3.44 shows that it was worthwhile using the design in this experiment. This can be confirmed also by the fact that the mean square for the Residual has decreased from 0.054 to 0.045. (The Residual line now represents the random variability of the experimental plots after removing block differences as well as the effects of the treatments.) So the standard errors of differences of means will also be smaller; see the final analysis in this section.

```
   6  TREATMENTSTRUCTURE block + N * S
   7  ANOVA [PRINT=aov; FPROBABILITY=yes] yield

Analysis of variance

Variate: yield

Source of variation          d.f.        s.s.       m.s.     v.r.   F pr.
block                           2     0.30850    0.15425     3.44   0.050
N                               2     4.59223    2.29611    51.22   <.001
S                               3     0.97720    0.32573     7.27   0.001
N.S                             6     0.64851    0.10808     2.41   0.061
Residual                       22     0.98625    0.04483
Total                          35     7.51269
```

This method of including the block factor in the treatment model works well for straightforward designs like the randomized-block design (and if this is as complicated as your designs are likely to become, you can omit the rest of this section). However, it is not satisfactory in more complicated situations like the balanced-incomplete-block or split-plot designs. Moreover, the analysis that is obtained does not reflect the real structure of the design, for example that `Block` is a random term and not a fixed term like the treatment main effects and interaction.

Consequently, the `BLOCKSTRUCTURE` directive is provided to allow you to define the *underlying structure* of the design, and thus the random (or error) terms that should occur in the analysis. The randomized-block design has an underlying structure of units nested within blocks. In the field experiment, the factor `block` indicates the block to which each plot belongs and the factor `plot` identifies the plots within each block, and so we can specify the block structure as follows:

```
BLOCKSTRUCTURE block / plot
```

This expands to give two model terms

```
    block  +  block.plot
```

each of which now defines a *stratum* in the analysis-of-variance table. The `Block` stratum contains the variation between blocks, and the `block.plot` stratum contains the variation between the plots within each block.

```
 8  TREATMENTSTRUCTURE N * S
 9  BLOCKSTRUCTURE block / plot
10  ANOVA [FPROBABILITY=yes] yield
```

Analysis of variance

```
Variate: yield

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

block stratum                   2    0.30850    0.15425     3.44

block.plot stratum
N                               2    4.59223    2.29611    51.22   <.001
S                               3    0.97720    0.32573     7.27   0.001
N.S                             6    0.64851    0.10808     2.41   0.061
Residual                       22    0.98625    0.04483

Total                          35    7.51269
```

Message: the following units have large residuals.

```
block 1 plot 3                                   -0.349        s.e.   0.166
block 2 plot 5                                    0.532        s.e.   0.166
block 2 plot 10                                  -0.400        s.e.   0.166
```

Tables of means

```
Variate: yield

Grand mean  1.104

         N          0        180        230
                0.601      1.313      1.398

         S          0         10         20         40
                0.829      1.155      1.167      1.266

         N          S          0         10         20         40
         0                 0.560      0.770      0.524      0.552
       180                 0.894      1.289      1.525      1.545
       230                 1.032      1.404      1.454      1.700
```

Standard errors of differences of means

```
Table                    N          S          N
                                               S
rep.                    12          9          3
d.f.                    22         22         22
s.e.d.              0.0864     0.0998     0.1729
```

For the randomized-block design it may seem that the change has involved no more than a relabelling of the analysis-of-variance table. Here all the treatment terms are estimated in the *bottom stratum* `Block.Plot`. The advantage of the use of the `BLOCKSTRUCTURE` directive becomes clearer when there are treatments that are estimated in one of the higher strata, as we shall see in Section 7.7.

7.4 Plots of residuals

Procedure `APLOT` provides up to four types of plots of residuals so that you can check the assumptions of the analysis.

The plots are selected using the `METHOD` parameter, with settings: `fitted` for residuals versus fitted values, `normal` for a Normal plot, `halfnormal` for a half-Normal plot, `histogram` for a histogram of residuals, `absresidual` for a plot of the absolute values of the residuals versus the fitted values, and `index` for a plot against an "index" variable (specified by the `INDEX` option). The default is
`METHOD=fitted,normal,halfnormal,histogram`.

The residuals and fitted values are accessed automatically from the structure specified by the `SAVE` option of `APLOT`. If the `SAVE` option is not set, they are taken from the `SAVE` structure of the last y-variate to have been analysed by `ANOVA`. By default, simple residuals are plotted, but you can set option `RMETHOD=standardized` to plot standardized residuals instead.

Figure 7.6 shows the default plots from the analysis of the canola data in Section 7.3.

Chapter 4 of the *Guide to Anova and Design on Genstat* has more information about the assumptions. Further details about `APLOT` can be found in the *Guide to the Genstat Command Language, Part 2 Statistics*, Section 4.1.4.

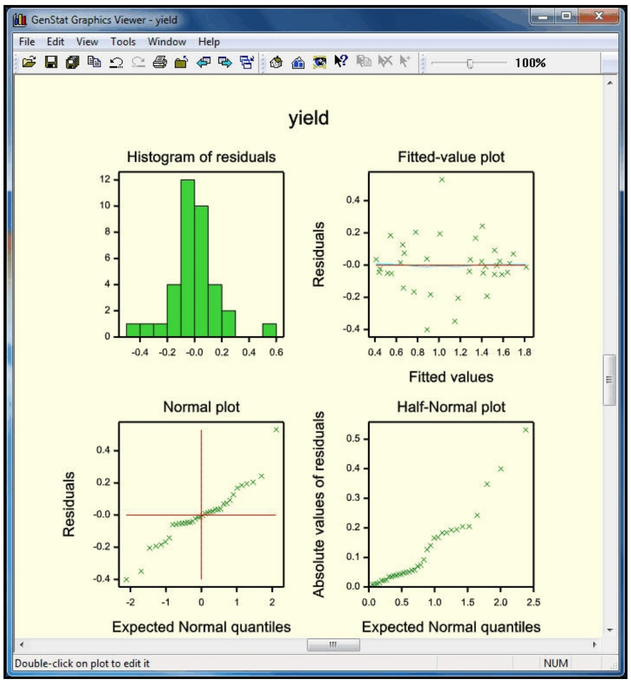

GenStat Graphics Viewer - yield
File Edit View Tools Window Help
100%
yield
Histogram of residuals
Fitted-value plot
Residuals
Fitted values
Normal plot
Half-Normal plot
Absolute values of residuals
Expected Normal quantiles
Expected Normal quantiles
Double-click on plot to edit it
NUM

Figure 7.6

7.5 Practical

Seven litters each of five rats were used in a randomized-block design (with litters as blocks) to study the effects of different diets on the gain in weight of rats. Analyse the data, in file `Ratblocks.gsh`, to see whether there are any differences between the diets.

Plot and assess the residuals.

7.6 Plots of means

Procedure `AGRAPH` can be used to plot tables of means from an `ANOVA` analysis. If none of its options or parameters are specified, `AGRAPH` plots the first two-way table of means in the most recent `ANOVA` (or for the first one-way table if there were no two-way tables). Alternatively, you can plot means from an earlier analysis, by using the `SAVE` option of `AGRAPH` to specify its save structure (saved using the `SAVE` option of the `ANOVA` command that performed the analysis).

Usually, each mean is represented by a point. However, with high-resolution plots, the `METHOD` option can be set to `lines` to draw lines between the points, or `data` to draw just the lines and then also plot the original data values, or `barchart` to plot the means as a barchart, or `splines` to plot the points together with a smooth spline to show the trend over each group of points. The `DFSPLINE` specifies the degrees of freedom for the splines; if this is not set, 2 d.f. are used when there are up to 10 points, 3 if there are 11 to 20, and 4 for 21 or more. The `GRAPHICS` option controls whether a high-resolution or a line-printer graph is plotted; by default `GRAPHICS=high`.

The `PSE` option specifies the type of error bar to be plotted with the means, with settings:

`differences`	average standard error of difference;
`lsd`	average least significant difference;
`means`	average effective standard error for the means;
`allmeans`	plots plus and minus the effective standard error around every mean.

The `LSDLEVEL` option sets the significance level (%) to use for the least significant differences (default 5). The `allmeans` setting is often unsuitable for plots other than barcharts when there are `GROUPS`, as the plus/minus e.s.e. bars may overlap each other.

You can define the table of means to plot explicitly, by specifying its classifying factors using the `XFACTOR`, `GROUPS`, `TRELLISGROUPS` and `PAGEGROUPS` parameters. The `XFACTOR` parameter defines the factor against whose levels the means are plotted. With a multi-way table, there will be a plot of means against the `XFACTOR` levels for every combination of levels of the other factors classifying the table. The `GROUPS` parameter specifies factors whose levels are to be included in a single window of the graph. So, for example, if you specify

```
AGRAPH [METHOD=line] XFACTOR=A; GROUPS=B
```

`AGRAPH` will produce a plot of the means in a single window with factor `A` on the x-axis, and a line for each level of the factor `B`. You can set `GROUPS` to a pointer to specify several factors to define groups. For example

```
POINTER [VALUES=B,C] Groupfactors
AGRAPH [METHOD=line] XFACTOR=A; GROUPS=Groupfactors
```

to plot a line for every combination of the levels of factors `B` and `C`. Similarly, the `TRELLISGROUPS` option can specify one or more factors to define a trellis plot. For example,

```
AGRAPH [METHOD=line] XFACTOR=A; GROUPS=B; TRELLISGROUPS=C
```

will produce a plot for each level of `C`, in a trellis arrangement; each plot will again have factor `A` on the x-axis, and a line for each level of the factor `B`. Likewise, the `PAGEGROUPS` parameter can specify factors whose combinations of levels are to be plotted on different pages. So

```
AGRAPH [METHOD=line] XFACTOR=A; GROUPS=B; PAGEGROUPS=C
```

will produce a plot for each level of `C`, but now on separate pages. Multi-way tables can plotted even if the corresponding model term was not in the `ANOVA` analysis. For example you can plot a two-way table even if the analysis contained only the main effects of the two factors; however, the lines will then all be parallel and no standard errors or LSDs can be included.

The `NEWXLEVELS` parameter enables different levels to be supplied for `XFACTOR` if the existing levels are unsuitable. If `XFACTOR` has labels, these are used to label the x-axis unless you set option `XFREPRESENTATION=levels`.

The `TITLE`, `YTITLE` and `XTITLE` parameters can supply titles for the graph, the y-axis and the x-axis, respectively.

In Figure 7.7, we have used the command

```
AGRAPH [METHOD=line] S; GROUPS=N
```

to plot mean yields, as points joined by lines, against the level of sulphur for each level of nitrogen.

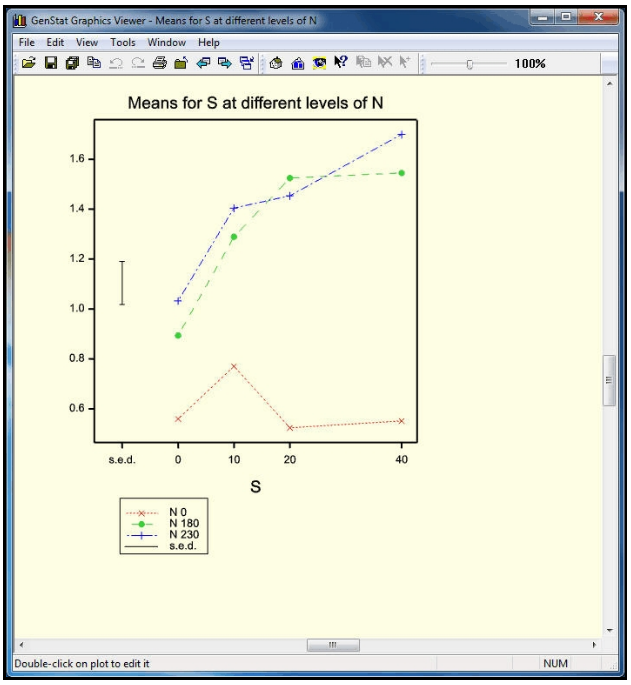

GenStat Graphics Viewer - Means for S at different levels of N
File Edit View Tools Window Help
100%
Means for S at different levels of N
1.6
1.4
1.2
1.0
0.8
0.6
s.e.d.
0
10
20
40
S
N 0
N 180
N 230
s.e.d.
Double-click on plot to edit it
NUM

Figure 7.7

7.7 Split-plot designs

V3 N3	V3 N2	V3 N2	V3 N3
V3 N1	V3 N0	V3 N0	V3 N1
V1 N0	V1 N1	V2 N0	V2 N2
V1 N3	V1 N2	V2 N3	V2 N1
V2 N0	V2 N1	V1 N1	V1 N2
V2 N2	V2 N3	V1 N3	V1 N0
V3 N2	V3 N0	V2 N3	V2 N0
V3 N1	V3 N3	V2 N2	V2 N1
V1 N3	V1 N0	V1 N2	V1 N3
V1 N1	V1 N2	V1 N0	V1 N1
V2 N1	V2 N0	V3 N2	V3 N3
V2 N2	V2 N3	V3 N1	V3 N0
V2 N1	V2 N2	V1 N2	V1 N0
V2 N3	V2 N0	V1 N3	V1 N1
V3 N3	V3 N1	V2 N3	V2 N2
V3 N2	V3 N0	V2 N0	V2 N1
V1 N0	V1 N3	V3 N0	V3 N1
V1 N1	V1 N2	V3 N2	V3 N3

We now show how to analyse split-plot designs. These designs were devised originally for agricultural experiments where some of the factors can be applied to smaller plots of land than others. Here there are two treatment factors: three different varieties of oats (labelled `V1`, `V2` and `V3` on the plan), and four levels of nitrogen (labelled `N0` to `N3`). Because of limitations on the machines for sowing seed, different varieties cannot conveniently be applied to plots as small as those that can be used for the different rates of fertilizer. So the design was set up in two stages. First of all, the blocks were each divided into three plots of the size required for the varieties, and the three varieties were randomly allocated to the plots within each block (exactly as in the randomized blocks design). Then each of these plots, or *whole-plots* as they are usually known, was split into four *sub-plots* (one for each rate of nitrogen), and the allocation of nitrogen was randomized independently within each whole-plot. Data from the experiment are in the spreadsheet file `Oats.gsh`, in the Data folder.

The design has sub-plots nested within whole-plots, which are themselves nested within the blocks: that is,

```
BLOCKSTRUCTURE Blocks/Wplots/Subplots
```

This expands to

```
Blocks + Blocks.Wplots + Blocks.Wplots.Subplots
```

giving strata for variation between blocks, between whole-plots within the blocks, and for sub-plots within the whole-plots (within blocks).

Just as in the randomized block design, the blocks all contain the same sets of treatments, and so no treatments are estimated in the `Blocks` stratum. But varieties, which were applied to whole-plots, are estimated in the `Blocks.Wplots` stratum.

The variance ratio for varieties is calculated by dividing the `Variety` mean square by the `Blocks.Wplots` residual mean square. It is easy to see that this is the correct thing to do. When we look to see whether the varieties differ we are really trying to answer the question: "Do the yields from the three sets of whole-plots, on the first of which the variety Victory was grown, on the second Golden rain, and on the third Marvellous, differ by more than the amount that we would expect for any three randomly chosen sets of whole-plots?". (Technically, variety is said to be *confounded* with whole plots.) The terms for `Nitrogen`, which was applied to sub-plots, and for the `Variety.Nitrogen` interaction are both estimated in the stratum for sub-plots within whole-plots (`Blocks.Wplots.Subplots`).

```
  2  SPLOAD 'Oats.gsh'
```

Loading Spreadsheet File

```
Catalogue of file Oats.gsh

Sheet Type: vector
    Index        Type      Nval    Name
        1      factor        72    blocks
        2      factor        72    wplots
        3      factor        72    subplots
        6      factor        72    variety
        9      factor        72    nitrogen
       10     variate        72    yield

Note: Missing indices are used by unnamed or system structures. These
store ancillary information, for example factor labels.

   3  TREATMENTS variety * nitrogen
   4  BLOCK blocks / wplots / subplots
   5  ANOVA [FPROBABILITY=yes] yield
```

Analysis of variance

```
Variate: yield

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

blocks stratum                     5      15875.3       3175.1      5.28

blocks.wplots stratum
variety                            2       1786.4        893.2      1.49    0.272
Residual                          10       6013.3        601.3      3.40

blocks.wplots.subplots stratum
nitrogen                           3      20020.5       6673.5     37.69    <.001
variety.nitrogen                   6        321.8         53.6      0.30    0.932
Residual                          45       7968.8        177.1

Total                             71      51985.9


Message: the following units have large residuals.

blocks 1                                                 31.4          s.e.   14.8
```

Tables of means

```
Variate: yield

Grand mean  104.0

    variety          Victory    Golden rain      Marvellous
                        97.6          104.5           109.8

  nitrogen         0 cwt      0.2 cwt      0.4 cwt      0.6 cwt
                    79.4         98.9        114.2        123.4

        variety    nitrogen         0 cwt      0.2 cwt      0.4 cwt      0.6 cwt
        Victory                      71.5         89.7        110.8        118.5
    Golden rain                      80.0         98.5        114.7        124.8
     Marvellous                      86.7        108.5        117.2        126.8
```

Standard errors of differences of means

```
Table                          variety       nitrogen        variety
                                                             nitrogen
rep.                                24             18              6
s.e.d.                            7.08           4.44           9.72
d.f.                                10             45          30.23
Except when comparing means with the same level(s) of
variety                                                         7.68
d.f.                                                              45
```

The standard errors accompanying the tables of means also take account of the stratum where each treatment term was estimated.

The `variety` s.e.d. of 7.08 = √(2×601.3/24) is based on the residual mean square for `blocks.wplots`, while that for `nitrogen` (4.44 = √(2×177.1/18)) is based on that for `blocks.wplots.subplots`. The `variety` × `nitrogen` table is more interesting. There are two s.e.d.'s according to whether the two means to be compared are for the same variety. If they are, then the sub-plots from which the means are calculated will all involve the same set of whole-plots, so any whole-plot variability will cancel out, giving a smaller s.e.d. than for a pair of means involving different varieties.

Finally notice that this time the Information output category has generated a message noting that block 1 has a large residual compared to the residuals of the other five blocks. In this instance, the message can be taken as confirming the success of the choice of the blocks: that is, that the yields of the plots in block 1 are consistently higher than those in the other blocks. Large residuals in the `block.wplot.subplot` stratum, however, might indicate possibly aberrant values.

Split-plot designs occur not only in field experiments, but also in animal trials (where, for example, the same diet may need to be fed to all the animals in a pen but other treatments may be applied to individual animals), or in industrial experiments (where different processes may require different sized batches of material), or even in cookery experiments. There can also be more than one treatment factor applied to any size of unit.

7.8 Practical

In an experiment to study the effect of two meat-tenderizing chemicals, the two (back) legs were taken from four carcasses of beef and one leg was treated with chemical 1 and the other with chemical 2. Three sections were then cut from each leg and allocated (at random) to three cooking temperatures, all 24 sections (4 carcasses × 2 legs × 3 sections) being cooked in separate ovens. The table below shows the force required to break a strip of meat taken from each of the cooked sections (the data are also in the file `Meat.gsh` in the Data folder). Analyse the experiment, and plot the means against the temperatures.

Leg			1			2	
Carcass	Section	Chemical	Temp	Force	Chemical	Temp	Force
1	1	1	2	5.5	2	3	6.3
	2	1	3	6.5	2	1	3.5
	3	1	1	4.3	2	2	4.8
2	1	2	1	3.2	1	3	6.2
	2	2	3	6.0	1	2	5.0
	3	2	2	4.7	1	1	4.0
3	1	2	1	2.6	1	2	4.6
	2	2	2	4.3	1	1	3.8
	3	2	3	5.6	1	3	5.8
4	1	1	3	5.7	2	2	4.1
	2	1	1	3.7	2	3	5.9
	3	1	2	4.9	2	1	2.9

7.9 Other facilities for anova and design

The commands for analysis of variance are described, in detail with examples, in the *Guide to the Genstat Command Language, Part 2 Statistics*, Chapter 4. This chapter also covers Genstat's extensive facilities for designing experiments. The REML facilities for analysing unbalanced designs, are described in Chapter 5. The equivalent menus are described in the *Guide to Anova and Design* and the *Guide to REML*. All these Guides can be accessed by clicking on sub-options of the Genstat Guides option of the Help menu (see Figure 10.1).

8 Using commands with menus or spreadsheets

In this chapter we discuss some of the ways in which you can use commands alongside the Genstat menus in order to extend the facilities that the menus offer, or simply to repeat the same analysis more conveniently with a different data set.

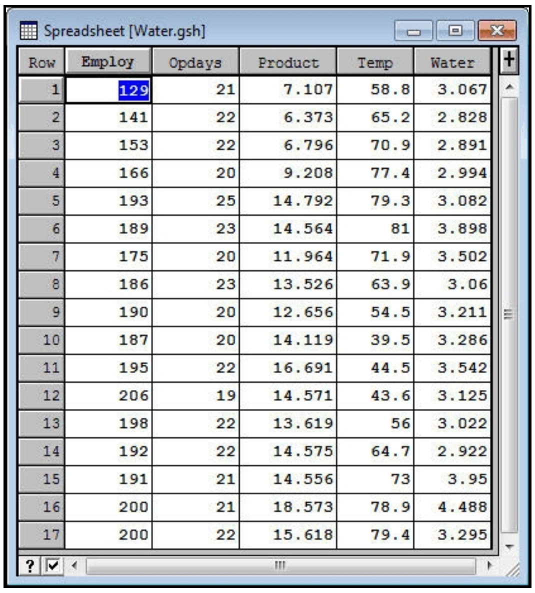
So, you will learn how to:

- modify scripts from the Input Log;
- include analysis commands in a spreadsheet;
- use a `FOR` loop to repeat a sequence of commands, each time operating on a different data set.

8.1 Commands from menus

The menus in Genstat provide convenient ways of generating analyses. As we have seen, in earlier chapters, they operate by generating commands which are executed by the Genstat server. Most users will have Genstat's options set to record all these commands in the Input Log. (This is controlled by the boxes in the Audit Trail tab of the Options menu as explained in Section 1.3). If you have kept a full record (with the menu set as in Figure 1.13), you can recreate the same analyses later. First select the Input Log as the active window by clicking on the Input Log line of the Window menu on the menu bar. Then save it in a file using the Save menu obtained by clicking on either Save or Save As in the File menu on the menu bar. When you rerun Genstat, open the file using the Select Input File menu (obtained by clicking on Open in the File menu on the menu bar). Then click on Submit Window in the Run menu on the menu bar.

You may also want to copy commands from the Input Log into another text window and adapt them. We illustrate this with the set of data, showing water use at a production plant, that we analysed in Section 6.3. Information was collected over 17 months on variables possibly associated with water usage: the average temperature, the amount of production, the number of operating days and the number of employees. The data are available in the spreadsheet file `Water.gsh` (Figure 8.1).



Row	Employ	Opdays	Product	Temp	Water
1	129	21	7.107	58.8	3.067
2	141	22	6.373	65.2	2.828
3	153	22	6.796	70.9	2.891
4	166	20	9.208	77.4	2.994
5	193	25	14.792	79.3	3.082
6	189	23	14.564	81	3.898
7	175	20	11.964	71.9	3.502
8	186	23	13.526	63.9	3.06
9	190	20	12.656	54.5	3.211
10	187	20	14.119	39.5	3.286
11	195	22	16.691	44.5	3.542
12	206	19	14.571	43.6	3.125
13	198	22	13.619	56	3.022
14	192	22	14.575	64.7	2.922
15	191	21	14.556	73	3.95
16	200	21	18.573	78.9	4.488
17	200	22	15.618	79.4	3.295

Figure 8.1

First we fit a simple linear regression for Water with a single explanatory variable, `Product`, using the Linear Regression menu as shown in Figure 8.2. (See Section 6.1 for more details about the menu.) This generates the output below.

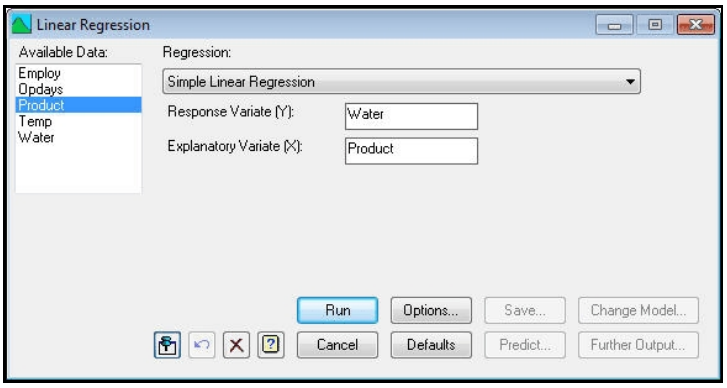


Figure 8.2

```
Regression analysis

          Response variate:   Water
              Fitted terms:   Constant, Product


Summary of analysis

Source              d.f.          s.s.          m.s.       v.r.    F pr.
Regression             1         1.270        1.2702       9.91    0.007
Residual              15         1.922        0.1282
Total                 16         3.193        0.1995

Percentage variance accounted for 35.8
Standard error of observations is estimated to be 0.358.

Message: the following units have large standardized residuals.
             Unit         Response         Residual
               16            4.488             2.31

Message: the following units have high leverage.
             Unit         Response         Leverage
                2            2.828             0.27
                3            2.891             0.25


Estimates of parameters

Parameter            estimate            s.e.       t(15)     t pr.
Constant                2.273           0.339        6.71     <.001
Product                0.0799          0.0254        3.15     0.007
```

The script of commands that has been executed to produce the analysis can be found at the end of the Input Log.

```
"Simple Linear Regression"
MODEL Water
TERMS Product
FIT [PRINT=model,summary,estimates; CONSTANT=estimate;\
        FPROB=yes; TPROB=yes] Product
```

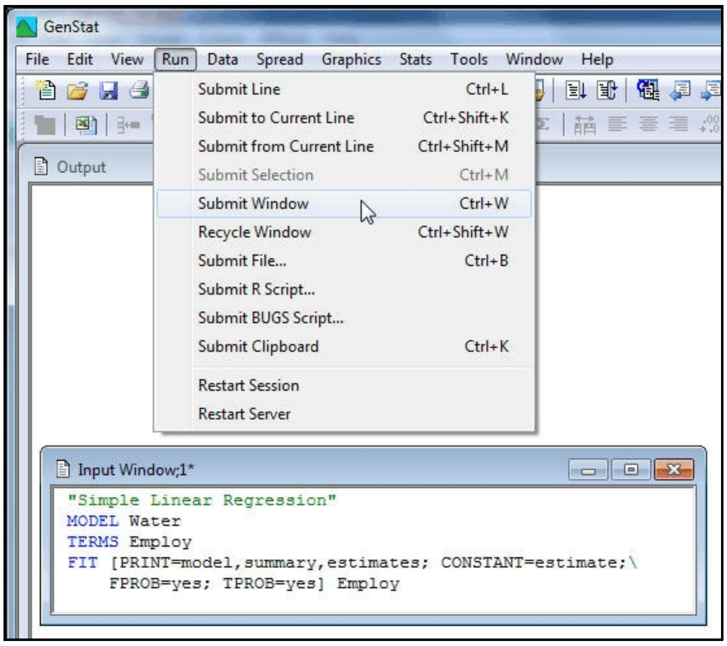
As we explained in Section 6.1, MODEL has specified the response variate, TERMS has defined the most complicated model that may be fitted, and FIT has fitted a linear regression with the explanatory variate Product. Note that we have reformatted the FIT command from the way in which it may appear within the log, so that each line fits within the width of the page. (Remember that the character \ indicates that the command continues onto the next line.)

If you want to repeat the analysis with another data set, you may find it easier to edit the script instead of rerunning the menus. (Here there is only one menu to change, but in more complicated analyses you may have used several.)

To do a regression on the variate Employ instead of Product, we could open a new text window (see Section 1.3), copy the script there, and edit it to become.

```
"Simple Linear Regression"
MODEL Water
TERMS Employ
FIT [PRINT=model,summary,estimates; CONSTANT=estimate;\
    FPROB=yes; TPROB=yes] Employ
```

Then run the commands using one of the methods provided by the Run menu on the menu bar (see Section 1.3). Here as these are the only commands in the window, it would be easiest to click on the line Submit Window, as shown in Figure 8.3.



```
"Simple Linear Regression"
MODEL Water
TERMS Employ
FIT [PRINT=model,summary,estimates; CONSTANT=estimate;\
    FPROB=yes; TPROB=yes] Employ
```

Figure 8.3

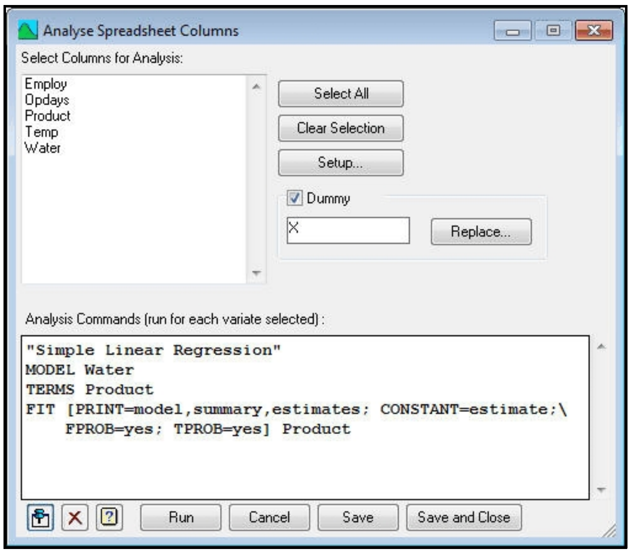
8.2 Practical

Use the Example Data Sets menu to open the spreadsheet file Peru.gsh. Fit a linear regression of sbp on height (using the menus). Save the regression commands to a text window, and modify the program to regression on age instead. Execute the program using the Run menu.

8.3 Commands to analyse a spreadsheet

Suppose we have decided that the right way to analyse this data set is by simple linear regression. (Note that this is for illustration purposes only, as we actually need a multiple linear regression, fitting all the explanatory variates, as we discovered in Section 6.3.) It might then be useful to put the commands into the Analyse Spreadsheet Columns menu for the spreadsheet `Water.gsh`. First we need to make `Water.gsh` the active window. Then we select the Analysis line within the Sheet section of the Spread menu on the menu bar.

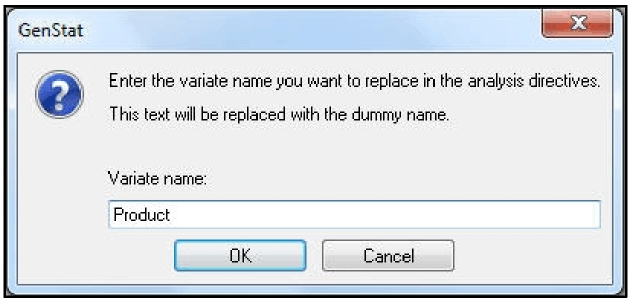
The lower half of the resulting menu (Figure 8.4) has a section into which you can type or paste commands to analyse the data. The menu provides flexibility by allowing you to perform the analysis, in turn, for several columns. The columns can be selected by highlighting them in the Select Columns for Analysis window of the menu. You refer to them in the analysis commands using the *dummy* whose name is given in the Dummy window. A *dummy* is a data structure that contains the identifier of another structure. When a command containing a dummy is executed, the dummy is replaced by the identifier that it currently contains. So, by using a dummy, you can conveniently change the data structure on which the command operates. In Figure 8.4 the dummy is called `X`.


Analyse Spreadsheet Columns
Select Columns for Analysis:
Employ
Opdays
Product
Temp
Water
Select All
Clear Selection
Setup...
Dummy
X
Replace...
Analysis Commands (run for each variate selected):
Run
Cancel
Save
Save and Close

```
"Simple Linear Regression"
MODEL Water
TERMS Product
FIT [PRINT=model,summary,estimates; CONSTANT=estimate;\
    FPROB=yes; TPROB=yes] Product
```

Figure 8.4

The commands in the lower half of the window were pasted straight from the `Input Log` (again with some reformatting to ensure that they do not need to scroll beyond the right-hand side of the window). So, they have `Product` as the explanatory variate. To set the commands to refer to an arbitrary explanatory variate (denoted by the dummy `X`) you click on the Replace button, fill in the resulting menu (Figure 8.5), and click on OK.


GenStat

Enter the variate name you want to replace in the analysis directives.
This text will be replaced with the dummy name.

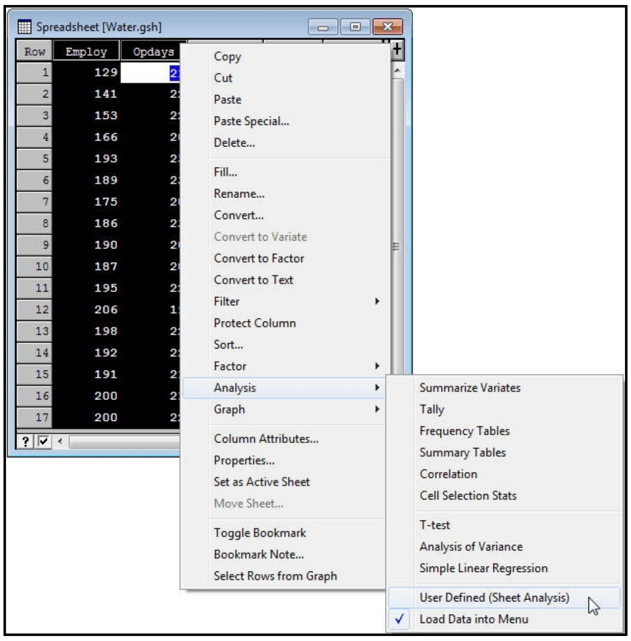
Variate name:
Product

Figure 8.5

The commands then become

```
"Simple Linear Regression"
MODEL Water
TERMS X
FIT [PRINT=model,summary,estimates;\
  CONSTANT=estimate; FPROB=yes; TPROB=yes] X
```

To run the commands with explanatory variates `Employ` and `Opdays`, for example, you can highlight their lines in the Select Columns for Analysis window and then click on Run. Alternatively, click on the Save and Close button to store the commands with the sheet. You can then run the analysis at any time: first highlight the columns by clicking on their names at the top of the spreadsheet; then either select the Sheet Analysis line in the Run menu on the menu bar, or make a right-mouse click on the spreadsheet and select User Defined (Sheet Analysis) from the Analysis section of the resulting menu (Figure 8.6). If, as in Figure 8.6, the line Load data into menu is checked, Genstat will open the Analyse Spreadsheet Columns menu with `Employ` and `Opdays` already selected in the Select Columns for Analysis window, ready for you to click on Run. If it is unchecked, Genstat simply runs the analyses.



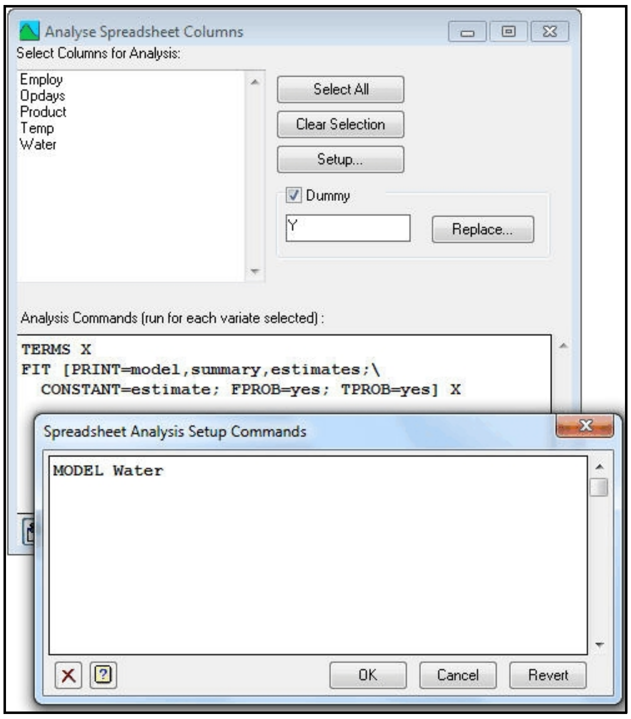
Row	Employ	Opdays
1	129	2
2	141	2
3	153	2
4	166	2
5	193	2
6	189	2
7	175	2
8	186	2
9	190	2
10	187	2
11	195	2
12	206	1
13	198	2
14	192	2
15	191	2
16	200	2
17	200	2

Figure 8.6

The initial line

```
MODEL Water
```

is common to all the analyses. We can arrange that this command is executed only once (and thus improve efficiency) by moving it to the Spreadsheet Analysis Setup Directives menu (Figure 8.7). This menu is obtained by clicking on the Setup button of the Analyse Spreadsheet Columns menu. After moving the line, you click on the OK button on the Spreadsheet Analysis Setup Directives menu, and then on the Save button of the Analyse Spreadsheet Columns menu.


Analyse Spreadsheet Columns
Select Columns for Analysis:
Employ
Opdays
Product
Temp
Water
Select All
Clear Selection
Setup...
✓ Dummy
Y
Replace...
Analysis Commands (run for each variate selected):
Spreadsheet Analysis Setup Commands
OK
Cancel
Revert

```
TERMS X
FIT [PRINT=model,summary,estimates;\
  CONSTANT=estimate; FPROB=yes; TPROB=yes] X
```

```
MODEL Water
```

Figure 8.7

8.4 Practical

Paste the linear regression program from Practical 8.2 into the Analyse Spreadsheet Columns menu. Put dummy X into the program instead of height or age. Run the analysis with some other x-variates.

8.5 Repeating a sequence of commands (FOR loops)

The commands that are executed with the spreadsheet analysis commands for the columns Employ and Opdays can, as usual, be found in the Input Log.

```
"Analysis of Data in Spreadsheet: Water.gsh"
MODEL Water
FOR X = Employ,Opdays
    "Simple Linear Regression"
    TERMS X
    FIT [PRINT=model,summary,estimates;\
        CONSTANT=estimate; FPROB=yes; TPROB=yes] X
ENDFOR
```

After the initial "setup" line to define Water as the dependant variate (and a comment to introduce the analysis), the TERMS and FIT lines are applied to the variates Employ and Opdays using a *for* loop. This is introduced by a FOR directive, and terminated by an ENDFOR directive. The parameters of FOR take the form: dummy = list of identifiers. Here we have X = Employ,Opdays so the contents of the loop are executed twice. On the first time, X is set to Employ and on the second it is set to Opdays.

If FOR has more than one parameter, the dummies change in parallel. So, if we had an additional dependent variate, Coffee say, we could put

```
FOR Y = Coffee,Water; X = Employ,Opdays
    MODEL Y
    TERMS X
    FIT [PRINT=model,summary,estimates;\
        CONSTANT=estimate; FPROB=yes; TPROB=yes] X
ENDFOR
```

to perform a regression for Coffee with explanatory variate Employ, and then one for Water with explanatory variate Opdays. So the two dummies, Y and X, pass through their lists in parallel. If the second list, or any other subsequent list, is shorter than the first list it is "recycled": that is, the dummy starts the list again each time it reaches the end until the first list has finished. For example,

```
FOR Y = Coffee,Water,Biscuits; X = Employ,Opdays
    MODEL Y
    TERMS X
    FIT [PRINT=model,summary,estimates;\
        CONSTANT=estimate; FPROB=yes; TPROB=yes] X
ENDFOR
```

would perform a regression for Coffee with explanatory variate Employ, then one for Water with explanatory variate Opdays, and finally one for Biscuits with explanatory variate Employ.

Further information about FOR, and the other programming facilities that Genstat offers, can be found in Chapter 9.

8.6 Practical

Modify the program that you wrote in the text window during Practical 8.2, and include a `FOR` loop to do regressions of

- `sbp` on `forearm`,
- `weight` on `age`, and
- `sbp` on `chin`.

Do you need to type `sbp` more than once? Run the program and examine the output.

9 Programs and procedures

The standard Genstat directives provide a wide range of standard analyses, as well as some more unusual techniques. However, the Genstat language has many of the facilities of a computing language too. These can be useful even if you merely want to repeat the same analysis several times (Section 9.1). They also allow you to write general programs, and to add new commands to the language, as *procedures* (Section 9.5).

So, in this chapter you will learn:

- more about how to use `FOR` loops to repeat sequences of commands;
- how to use Block-if structures to choose which set of statements to execute with each data set;
- how to exit from part of a program;
- how to form your programs into procedures.

9.1 FOR loops (recap)

Section 8.5 introduced the Genstat `FOR` loop, which repeats a sequence of statements several times. The loop was constructed by the spreadsheet analysis commands when we analysed the data in `Water.gsh`.

```
"Analysis of Data in Spreadsheet: Water.gsh"
MODEL Water
FOR X = Employ,Opdays
    "Simple Linear Regression"
    TERMS X
    FIT [PRINT=model,summary,estimates;\
        CONSTANT=estimate; FPROB=yes; TPROB=yes] X
ENDFOR
```

The loop starts with a `FOR` statement, and ends with an `ENDFOR` statement. The parameters of `FOR` take the form: dummy = list of identifiers. Here we have `X = Employ,Opdays` so the contents of the loop are executed twice. On the first time, `Y` is set to `Employ` and on the second it is set to `Opdays`.

In Section 8.5 we also explained that, if `FOR` has more than one parameter, the dummies change in parallel. If the second list, or any subsequent list, is shorter than the first list it is “recycled”: that is, the dummy starts the list again each time it reaches the end until the first list has finished. For example,

```
FOR Y = Coffee,Water,Biscuits; X = Employ,Opdays
    MODEL Y
    TERMS X
    FIT [PRINT=model,summary,estimates;\
        CONSTANT=estimate; FPROB=yes; TPROB=yes] X
ENDFOR
```

would perform a regression for `Coffee` with explanatory variate `Employ`, then one for `Water` with explanatory variate `Opdays`, and finally one for `Biscuits` with explanatory variate `Employ`.

There is also an alternative form of loop, with no parameters. Instead you can specify the number of times to execute the loop by `NTIMES` option, and obtain the number of each "pass" though the loop using the `INDEX` option. For example

```
  2  SPLOAD 'Water.gsh'

Loading Spreadsheet File

Catalogue of file Water.gsh

Sheet Type: vector
   Index        Type      Nval    Name
       1     variate        17    Employ
       2     variate        17    Opdays
       3     variate        17    Product
       4     variate        17    Temp
       5     variate        17    Water

   3  POINTER [VALUES=Employ,Opdays,Product,Temp] Vars
   4  CALCULATE Nvars = NVALUES(Vars)
   5  FOR [NTIMES=Nvars; INDEX=i]
   6    CALCULATE Corr = CORRELATION(Water; Vars[i])
   7    &         Nobservations = NOBS(Water + Vars[i])
   8    &         Abscorr = ABS(Corr)
   9    PRCORRELATION [NOBSERVATIONS=Nobservations] \
  10                  Abscorr; CUPROBABILITY=Prob
  11    CALCULATE Prob = 0.5 * Prob
  12    PRINT Corr,Prob
  13  ENDFOR

        Corr         Prob
      0.4132      0.02480


        Corr         Prob
    -0.08883       0.1836


        Corr         Prob
      0.6307     0.001658


        Corr         Prob
      0.2858      0.06655
```

On the first pass through the loop, the scalar `i` from the `INDEX` option of `FOR` has the value one. So line 6 calculates the correlation between `Water` and the first element of the `Vars` pointer, namely `Employ`. On the second pass, the correlation is between `Water` and `Opdays`, and so on.

In line 7, `NOBS` is a summary function that gives the number of observations (i.e. non-missing values). Remember from Section 5.1 that a unit in

```
Water + Vars[i]
```

will be missing if it is missing in either `Water` or `Vars[i]`. So `Nobservations` in line 7 will be the number of units that available to calculate the correlation in line 6. This is then used in line 9 by the `PRCORRELATION` procedure when it calculate the cumulative upper probability `Prob` of the absolute correlation `Abscorr`. The next line multiplies the probability by 0.5 to take account of the fact that we are doing a two-sided test.

In this and later examples we have indented the contents of the loop by two spaces. This is to make it easier to read, and is not required by Genstat.

9.2 Block-if structures

The directives `IF`, `ELSIF`, `ELSE`, and `ENDIF` can be used to select between alternative sets of statements. In the simplest case, we can use this to control whether or not a particular set of statements is executed. For example

```
IF Prob .LE. 0.05
  PRINT 'Significant correlation'
ENDIF
```

Genstat evaluates the logical condition in the `IF` statement and then, if it is true, executes the statements between `IF` and `ENDIF`; otherwise, if the condition is untrue, it skips those statements and continues with whatever comes after `ENDIF`. (Logical expressions are described in Section 5.1.)

By including `ELSIF` statements, we can construct a more complicated block-if structure.

```
IF Prob .LE. 0.001
  PRINT 'Probability < 0.1%'
ELSIF Prob .LE. 0.01
  PRINT 'Probability < 1%'
ELSIF Prob .LE. 0.05
  PRINT 'Probability < 5%'
ENDIF
```

`ELSIF` statements also each have a logical expression. To execute the block-if structure, Genstat looks at the result of the expression in the `IF` statement, and then each `ELSIF` in turn, and executes the statements introduced by the `IF` or `ELSIF` that contains the first true result. You can also supply some statements to be executed if none are true; these are introduced by an `ELSE` statement. For example,

```
IF Prob .LE. 0.001
  PRINT 'Probability < 0.1%'
ELSIF Prob .LE. 0.01
  PRINT 'Probability < 1%'
ELSIF Prob .LE. 0.05
  PRINT 'Probability < 5%'
ELSE
  PRINT 'Not significant'
ENDIF
```

You can have any number of control structures (like loops or block-if structures) within other control structures. We now put a block-if structure within the original loop from Section 9.1.

```
FOR [NTIMES=Nvars; INDEX=i]
  CALCULATE Corr = CORRELATION(Water; Vars[i])
  &         Nobservations = NOBS(Water + Vars[i])
  &         Abscorr = ABS(Corr)
  PRCORRELATION [NOBSERVATIONS=Nobservations] \
              Abscorr; CUPROBABILITY=Prob
  CALCULATE Prob = 0.5 * Prob
```

```
 IF Prob .LE. 0.001
   PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 0.1%';\
         FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
 ELSIF Prob .LE. 0.01
   PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 1%';\
         FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
 ELSIF Prob .LE. 0.05
   PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 5%';\
         FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
 ELSE
   PRINT [IPRINT=*] 'Correlation',Corr,'Not significant';\
         FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
 ENDIF
ENDFOR
```

```
Correlation    0.413    Probability < 5%


Correlation   -0.089    Not significant


Correlation    0.631    Probability < 1%


Correlation    0.286    Not significant
```

Setting option `IPRINT=*` in the `PRINT` statements stops the identifiers of the data structure `Corr` being printed.

9.3 Exit from a control structure

There are no labels and jumps in Genstat; however, the `EXIT` directive can be used to break out of control structures like loops and block-if structures. The `CONTROLSTRUCTURE` option indicates the type of control structure that is to be exited (with default `FOR`). The `NTIMES` option specifies how many of them to exit (default 1). The `REPEAT` option controls whether you exit from a loop altogether (the default, `REPEAT=no`) or whether you go to `ENDFOR` and then repeat the loop with the next set of parameters (`REPEAT=yes`). `EXIT` has a logical expression as its parameter to control whether or not the exit takes place; if the expression is omitted, the exit always takes place.

9.4 Practical

Re-open the spreadsheet file `Peru.gsh` from Practical 8.2, and set up a pointer containing all the columns. (Hint: in Genstat *for Windows*, you can do this by clicking on the Create Pointer sub-option of the Sheet option of the Spread menu.)

Write a loop to print the mean and variance of every variate. (Hint: use the `MEAN` and `VAR` functions.)

Add an if-block so that you use 4 decimals if variance<10, 3 decimals if 10<variance<100, 2 if 100<variance<1000, or otherwise to use default decimals.

Execute the program using the Run menu.

9.5 Procedures

Sets of frequently required statements can be formed into procedures. As you have seen, the use of a Genstat procedure looks exactly the same as the use of one of the standard Genstat directives.

Genstat has a library of standard procedures which is attached automatically to Genstat whenever it is run. Similarly there can be a local library which will also be attached automatically, and you can form and attach libraries of your own. When Genstat meets a statement with a name that it does not recognize as a directive nor as the name of a procedure already in store, it will look for a procedure of that name in the libraries: firstly in your own libraries, then in the local library (if any), and then in the standard procedure library.

The standard library contains procedures contributed not only by the writers of Genstat but also by knowledgeable Genstat users from many application areas and countries. It is controlled by an Editorial Board, who check that the procedures are useful and reliable, and maintain standards for the documentation. For details see the page, *Procedure Library: Instructions for Authors*, in the on-line help.

Full information about procedures and procedure libraries is given in the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*, Section 5.3. Here we show a simple example to illustrate the main ideas.

Procedures start with a `PROCEDURE` statement to define the procedure name. The options and parameters of the procedure are then defined, using the `OPTION` and `PARAMETER` directives. The statements that perform the actions of the procedure come next, followed by an `ENDPROCEDURE` statement.

Below, we define a procedure called `CORTEST` to calculate and test correlations between two variables `Y` and `X`.

The procedure name is supplied by the parameter of `PROCEDURE`, in a quoted string (see line 6). The first eight characters of the name must not be the same as any other procedure that is currently accessible within Genstat; otherwise Genstat assumes that you are replacing that procedure. If the first four characters are the same as those of another procedure, Genstat will give a warning. Programs that have abbreviated the name to four characters would then be ambiguous.

The `PARAMETER` option of `PROCEDURE` indicates whether the settings in any list specified for the parameters of the procedure are to be taken individually, by calling the procedure several times, or whether they should be processed together. The difference between these alternatives can be illustrated by considering the directives `ANOVA` and `PRINT`. For example, with

```
ANOVA Height,Weight; RESIDUALS=Hres,Wres
```

Genstat will first do an analysis with the values in the `Height` variate and store the resulting residuals in the variate `Hres`; it then analyses `Weight` and stores the residuals in `Wres`. This action corresponds to the default setting `PARAMETER=dummy`. Inside the procedure, each parameter is represented by a dummy data structure with the same name as the parameter. This will point to each item of the list in turn, like the parameters of a `FOR` loop (Section 9.1). Conversely, in the statement

```
PRINT Height,Hres
```

the values of `Height` and `Hres` are printed together down the page, but this is possible only if `PRINT` is able to access both variates simultaneously. In a procedure this can be done by setting `PARAMETER=pointer`. Each parameter is then represented by a pointer, storing the complete list of settings.

The procedure `CORTEST` has two parameters, `Y` and `X`, which are defined in line 7. The `PARAMETER` option is not been set in the `PROCEDURE` statement, so these are represented inside the procedure by the dummies `Y` and `X`. No options have been defined.

The statements to be executed when `CORTEST` is used are in lines 8-28 (the blank lines in 8 and 28 are included to make the procedure easier to read, and are not required by Genstat). The dummies `Y` and `X` refer to data structures in the outer program. All the other data structures in lines 8-28 are *local* within the procedure. They have no connection with data structures in the outer program (even those that might have the same identifier). They are created when the procedure runs, and deleted when it ends.

The `ENDPROCEDURE` statement is in line 29. There is then a `FOR` loop, in lines 31-33, that uses the procedure to calculate the same correlations as in Section 9.2.

```
  2  SPLOAD 'Water.gsh'

Loading Spreadsheet File

Catalogue of file Water.gsh

Sheet Type: vector
    Index          Type       Nval    Name
        1       variate         17    Employ
        2       variate         17    Opdays
        3       variate         17    Product
        4       variate         17    Temp
        5       variate         17    Water

   3  POINTER [VALUES=Employ,Opdays,Product,Temp] Vars
   4  CALCULATE Nvars = NVALUES(Vars)
   5
   6  PROCEDURE 'CORTEST'
   7  PARAMETER 'Y','X'
   8
   9  CALCULATE Corr = CORRELATION(Y; X)
  10  &          Nobservations = NOBS(Y + X)
  11  &          Abscorr = ABS(Corr)
  12  PRCORRELATION [NOBSERVATIONS=Nobservations] \
  13                Abscorr; CUPROBABILITY=Prob
  14  CALCULATE Prob = 0.5 * Prob
  15  IF Prob .LE. 0.001
  16    PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 0.1%';\
  17          FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
  18  ELSIF Prob .LE. 0.01
  19    PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 1%';\
  20          FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
  21  ELSIF Prob .LE. 0.05
  22    PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 5%';\
  23          FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
  24  ELSE
  25    PRINT [IPRINT=*] 'Correlation',Corr,'Not significant';\
  26          FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
  27  ENDIF
  28
  29  ENDPROCEDURE
```

```
30
31  FOR [NTIMES=Nvars; INDEX=i]
32    CORTEST Y=Water; X=Vars[i]
33  ENDFOR
```

```
Correlation      0.413      Probability < 5%


Correlation      -0.089     Not significant


Correlation      0.631      Probability < 1%


Correlation      0.286      Not significant
```

The PARAMETER and OPTION directives have several parameters that allow you to define attributes to be checked. For example, we could redefine the PARAMETER statement above to become

```
PARAMETER 'Y','X','Corr','Prob'; \
  MODE=p;              "default - expects an identifier list   "\
  TYPE=2('variate','scalar'); \
  SET=2(yes,no);       "X & Y must be set (but not Corr & Prob)"\
  INPUT=2(yes,no);     "X & Y only for input (but not ...)     "\
  DECLARED=2(yes,no);  "X & Y must have been declared (but ...)"\
  PRESENT=2(yes,no);   "X & Y must have values (but not ...)   "\
  COMPATIBLE=*,'nvalues',*,* "X must have same no. values as Y"
```

The comments on the right-hand side of each line explain what is being checked by each parameter. A full explanation of all the possibilities is in the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*, Section 5.3.2.

Options and parameters can also be used to save output from the procedure. Below we have defined new parameters Corr to save the correlation, and Prob to save the probability. (So Corr and Prob will no longer be local to the procedure.) A complication now, is that the program will fail with an *unset dummy* diagnostic unless the dummies Corr and Prob are set. It would be tedious to require users of the procedure to set them if they do not want to save the results that are calculated. So we use the ASSIGN directive in line 17 to set them to local data structures if they have not been set in the procedure call.

```
ASSIGN [METHOD=preserve] corr,prob; POINTER=Corr,Prob
```

ASSIGN sets either dummies or elements of pointers (hence the apparently confusing name, POINTER, for the second parameter). So here we are assigning local data structures corr and prob to Corr and Prob, but setting the option METHOD=preserve requests that any existing values of the dummies are preserved. So, this acts as a "safety net" in case these parameters are not set when CORTEST is called..

A further issue is that we have chosen to assign local structures whose names differ from the dummies only by being completely in lower case rather than having an initial capital letter. If you are writing a procedure for anyone to use, you need to realise that the user can use the CASE option of the SET directive to request that the case of identifiers be ignored. We guard against this by using SET in line 16 to request that the case be significant. However, it is bad practice to change the user's own programming environment, and so we set option `RESTORE=case` in the `PROCEDURE` statement in line 1 to ensure that the case sensitivity is restored to its original condition after the procedure has run. `SET` allows many different aspects of the Genstat environment to be modified (details are in the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*, Section 5.6.1), and most of these can be reset by `RESTORE`. Alternatively, you can save the environment settings using `GET` (*Guide to the Genstat Command Language, Part 1 Syntax and Data Management*, Section 5.6.2), and restore them yourself explicitly using `SET`.

```
  2  SPLOAD 'Water.gsh'
```

Loading Spreadsheet File

```
Catalogue of file Water.gsh

Sheet Type: vector
   Index         Type     Nval   Name
       1      variate       17   Employ
       2      variate       17   Opdays
       3      variate       17   Product
       4      variate       17   Temp
       5      variate       17   Water
```

```
   3  POINTER [VALUES=Employ,Opdays,Product,Temp] Vars
   4  CALCULATE Nvars = NVALUES(Vars)
   5
   6  PROCEDURE [RESTORE=case] 'CORTEST'
   7  PARAMETER 'Y','X','Corr','Prob'; \
   8    MODE=p;              "default - expects an identifier list   "\
   9    TYPE=2('variate','scalar'); \
  10    SET=2(yes,no);       "X & Y must be set (but not Corr & Prob)"\
  11    INPUT=2(yes,no);     "X & Y only for input (but not ...)     "\
  12    DECLARED=2(yes,no); "X & Y must have been declared (but ...)"\
  13    PRESENT=2(yes,no);   "X & Y must have values (but not ...)   "\
  14    COMPATIBLE=*,'nvalues',*,* "X must have same no. values as Y"
  15
  16  SET [CASE=significant]
  17  ASSIGN [METHOD=preserve] corr,prob; POINTER=Corr,Prob
  18
  19  CALCULATE Corr = CORRELATION(Y; X)
  20  &          Nobservations = NOBS(Y + X)
  21  &          Abscorr = ABS(Corr)
  22  PRCORRELATION [NOBSERVATIONS=Nobservations] \
  23                Abscorr; CUPROBABILITY=Prob
  24  CALCULATE Prob = 0.5 * Prob
  25  IF Prob .LE. 0.001
  26    PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 0.1%';\
  27          FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
  28  ELSIF Prob .LE. 0.01
  29    PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 1%';\
  30          FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
  31  ELSIF Prob .LE. 0.05
  32    PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 5%';\
  33          FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
  34  ELSE
  35    PRINT [IPRINT=*] 'Correlation',Corr,'Not significant';\
  36          FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
  37  ENDIF
```

```

ENDPROCEDURE

FOR [NTIMES=Nvars; INDEX=i]
  CORTEST Y=Water; X=Vars[i]
ENDFOR
```

```
Correlation      0.413     Probability < 5%


Correlation      -0.089    Not significant


Correlation      0.631     Probability < 1%


Correlation      0.286     Not significant
```

Our final refinement to `CORTEST` is to define a `PRINT` option, to illustrate how you can define options (or parameters) that have string tokens as their settings.

```
OPTION 'PRINT';\
  MODE=t;          "values will be textual strings"\
  VALUES=!t(correlation,nobservations); "defines allowed tokens"\
  DEFAULT='corr';  "defines the default when not set"\
  LIST=yes         "whether more than one setting can be given"
```

Options (or parameters) with `MODE=t` (i.e. text) expect strings as their settings, rather than the identifiers that are expected with the default of `MODE=p` (i.e. pointer). The `VALUES` parameter defines the string tokens that are allowed. These are supplied in an unnamed text structure. If you are writing a procedure for others to use, you should ensure that they all differ from each other in the first four characters. When you call the procedure, Genstat will recognise abbreviations of the tokens, and pass their full forms into the procedure (in the same case of letters that you have used for `VALUES`). The `DEFAULT` parameter supplies the default values (and notice that these too can be abbreviated). The `LIST` parameter indicates whether or not the user is allowed to specify more than one token at a time for a mode-t option. If you specify `LIST=yes` with a mode-p option, the option will be supplied as a pointer instead of a dummy. The `PARAMETER` directive does not have a `LIST` parameter (you are not allowed to specify more than one token for a mode-t parameter).

```
SPLOAD 'Water.gsh'
```

Loading Spreadsheet File

Catalogue of file Water.gsh

```
Sheet Type: vector
   Index        Type      Nval    Name
       1     variate        17    Employ
       2     variate        17    Opdays
       3     variate        17    Product
       4     variate        17    Temp
       5     variate        17    Water
```

```
POINTER [VALUES=Employ,Opdays,Product,Temp] Vars
CALCULATE Nvars = NVALUES(Vars)

PROCEDURE [RESTORE=case] 'CORTEST'
OPTION 'PRINT';\
  MODE=t;             " values will be textual strings        "\
  VALUES=!t(correlation,nobservations);"defines allowed tokens"\
  DEFAULT='corr';     " defines the default when not set      "\
  LIST=yes            " whether >1 setting can be given       "
PARAMETER 'Y','X','Corr','Prob'; \
  MODE=p;             "default - expects an identifier list   "\
  TYPE=2('variate','scalar'); \
  SET=2(yes,no);      "X & Y must be set (but not Corr & Prob)"\
  INPUT=2(yes,no);    "X & Y only for input (but not ...)     "\
  DECLARED=2(yes,no); "X & Y must have been declared (but ...)"\
  PRESENT=2(yes,no);  "X & Y must have values (but not ...)   "\
  COMPATIBLE=*,'nvalues',*,* "X must have same no. values as Y"

SET [CASE=significant]
ASSIGN [METHOD=preserve] corr,prob; POINTER=Corr,Prob

CALCULATE Corr = CORRELATION(Y; X)
&         Nobservations = NOBS(Y + X)
&         Abscorr = ABS(Corr)
PRCORRELATION [NOBSERVATIONS=Nobservations] \
              Abscorr; CUPROBABILITY=Prob
CALCULATE Prob = 0.5 * Prob

IF SUM(!t(correlation,nobservations) .IN. PRINT)
  CAPTION 'Correlation test'; STYLE=minor
  IF 'correlation' .IN. PRINT
    IF Prob .LE. 0.001
      PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 0.1%';\
            FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
    ELSIF Prob .LE. 0.01
      PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 1%';\
            FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
    ELSIF Prob .LE. 0.05
      PRINT [IPRINT=*] 'Correlation',Corr,'Probability < 5%';\
            FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
    ELSE
      PRINT [IPRINT=*] 'Correlation',Corr,'Not significant';\
            FIELD=11,7,20; DECIMALS=3; JUSTIFICATION=left
    ENDIF
  ENDIF
  IF 'nobservations' .IN. PRINT
```

```
49      PRINT   [IPRINT=*] 'Number of observations:',Nobservations;\
50              DECIMALS=0; JUSTIFICATION=left
51    ENDIF
52  ENDIF
53
54  ENDPROCEDURE
55
56  CORTEST [PRINT=*] Y=Water; X=Employ; CORR=Cor; PROB=Pr
57  PRINT   Cor,Pr
```

```
        Cor           Pr
     0.4132      0.02480
```

```
58  CORTEST Y=Water; X=Employ
```

Correlation test

```
Correlation        0.413        Probability < 5%
```

In line 31, `SUM(!t(correlation,nobservations).IN.PRINT)` will be non-zero (i.e. true) if `PRINT` contains either of the two tokens. So the this expression checks whether any printed output is required. In line 32, we use the `CAPTION` directive to display a title for the analysis (see Section 4.8 for an explanation). In line 56, we verify that setting `PRINT=*` suppresses the printed output, and that we can use the `CORR` and `PROB` parameters to save the correlation and probability. In line 58 we call the procedure with its default output.

You should now be able to write simple procedures of your own, but to find out more, you can read the Section 5.3 of the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management*. The Genstat commands that are used to form a procedure library are described in Section 3.8.1 of that *Guide*. Alternatively, the Procedure option of the Tools menu on the menu bar of the Genstat client has sub-options that allow you to form a procedure library, and arrange for libraries that are stored in a Genstat add-in folder to be attached automatically when you run Genstat. (There are two add-in folders: the "system" folder is alongside the `Bin` folder that stores the Genstat executable program; the location of your own "user" add-in folder is defined by a box on the General tab of the Options menu.) The Genstat client also has a *resource language* that enables you to write your own menus for your new procedures. For details, see the Genstat resource language topic in the Knowledge Base. The new menus then appear as sub-options of a new option, User, on the menu bar.

9.6 Practical

Convert your program from Practical 9.4 into a procedure, with a parameter `DATA` to supply the variate, and parameters `MEAN` and `VARIANCE` to save the mean and variance.

Add a `PRINT` option to control the printed output.

9.7 Other useful commands

Below we mention some of the other Genstat commands that may be useful when you are writing a procedure. The numbers in brackets are references to sections in the *Guide to the Genstat Command Language, Part 1 Syntax and Data Management.*

To declare (i.e. define) data structures, the main directives are

`SCALAR`	scalar (2.2.1)
`VARIATE`	variate (2.3.1)
`TEXT`	text (2.3.2)
`FACTOR`	factor (2.3.3)
`MATRIX`	rectangular matrix (2.4.1)
`DIAGONALMATRIX`	diagonal matrix (2.4.2)
`SYMMETRICMATRIX`	symmetric matrix (2.4.3)
`TABLE`	table (2.5)
`DUMMY`	dummy (2.2.2)
`POINTER`	pointer (2.6)

Others are described in later sections of Chapter 2. If you want to redefine an existing structure as a different type, you must first delete its existing attributes (including its type) by using the `DELETE` with option setting `REDEFINE=yes`.

Alternatively, to rename a data structure

`RENAME`	renames a data structure, to give it a new identifier (2.10.2)

To define a data structure with exactly the same attributes as an existing one

`DUPLICATE`	forms new data structures with attributes taken from an existing structure (2.10.3)
`PDUPLICATE`	duplicates a pointer, with all its components (2.10.4)

To "convert" values of data structures by putting them into data structures with a different type

`CALCULATE`	can form matrices using functions like `DIAGONAL`, `COLBIND`, `ROWBIND` etc (4.2.4), and can form variates from matrices by using qualified identifiers (4.1.6)
`EQUATE`	copies values between sets of data structures (4.3.1)
`VTABLE`	forms a variate and a set of classifying factors from a table (i.e. converts a table into a data matrix)

To obtain details about data structures

`CALCULATE`	can provide information about sizes etc using functions like `NVALUES`, `NVRESTRICTED`, `NVRESTRICTED`, `NMV`, `NLEVELS`, `NROWS`, `NCOLUMNS` etc (4.2.2)
`GETATTRIBUTE`	accesses attributes of data structures such as their types, sizes etc (2.11.3)

For output

`PRINT`	prints data in tabular form to an output file or a

	text (3.2.1, 3.2.2 and 3.7)
`CAPTION`	prints various types of caption and title (3.2.3)
`PAGE`	moves to the top of the next page of an output file (3.2.4)
`SKIP`	skips lines of input or output files (3.3.3)
`DECIMALS`	sets the number of decimals for a structure, using its round-off
`CONCATENATE`	concatenates together lines of text vectors (4.7.1)
`TXCONSTRUCT`	forms a text structure by appending or concatenating values of scalars, variates, texts, factors or pointers; allows the case of letters to be changed or values to truncated and reversed (4.7.2)
For checking	
`FAULT`	allows you to issue a standard Genstat fault, warning or message (5.4.1)
`SETRELATE`	compares the sets of values in two data structures (4.3.2)
`SETCALCULATE`	performs Boolean set calculations on the contents of vectors and pointers (4.3.3)
To run other programs from inside Genstat	
`SUSPEND`	suspends execution of Genstat to carry out commands in the operating system (5.7.1)

This is used by the procedure `BXGENSTAT` to run WinBUGS or OpenBUGS, and by the procedure `RXGENSTAT` to run R.

10 Further information

Genstat has companion guides to this one, which can be accessed (in pdf format) by clicking on the appropriate sub-option of the Genstat Guides option of the Help menu (Figure 10.1). The guides at the top (Introduction to Survey Analysis) describe the use of Genstat's menus for various types of task. However, they also give useful background information and (where relevant) explain the underlying statistical theory. So, for example, you would find the *Guide to Anova and Design* a helpful supplement to Chapter 8 of this Introduction. Here we aim to tell you only how to use the commands for analysis of variance, In contrast, the *Guide to Anova and Design*, which is based on VSN's two-day course on the subject, shows many different types of design, and explains the underlying statistics. It also covers the analysis of unbalanced designs, using either the regression or the `REML` algorithms.

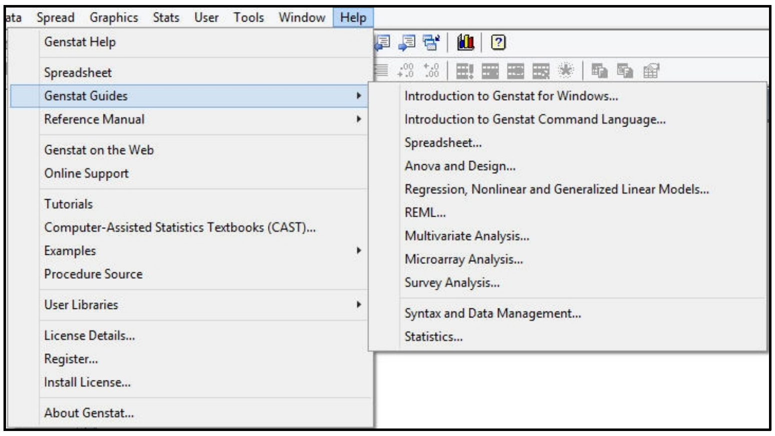


Figure 10.1

The aim of this Introduction is to give you the necessary skills to write commands and programs, and the confidence to extend your knowledge from the other documentation when you want to do something that we have not covered. The best place to look for more information about the commands is in the bottom section of the Genstat Guides option, which opens the two *Guides to Genstat*. Part 1 (Syntax and Data Management) gives the formal definition of the Genstat language and syntax. It then describes (in detail, with examples) the facilities for input and output, calculations, manipulation, programming and graphics. Part 2 (Statistics) covers Genstat's extensive statistical facilities, again with examples. Two particularly useful chapters are

- Chapter 7 of Part 1, which gives a summary of the statistical facilities, with cross references to the corresponding sections in Part 2, and
- Chapter 1 of Part 2 ,which gives a summary of the syntax and of the contents of Part 1, again with cross references.

If you want an overview of Genstat's commands, these chapters are the best place to look, and we do not attempt to duplicate them in this Introduction!

Another useful source of information is the three-part Reference Manual, which can be accessed from the Reference Manual option of the Help menu (Figure 10.2). The Directives and Procedure Library sub-options collate the help pages on the directives and

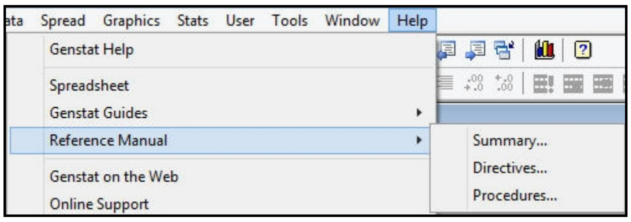


Figure 10.2

procedures, respectively, into pdf documents. These have the advantage that you can carry out searches, or print information, on several commands at once. The Summary sub-option collates the options and parameters of all the directives and procedures (in alphabetic order), and lists all the functions.

Index