Module 1

A Primer on SAS for Windows

(Version 9.1)

Certificate in EnvIroStats (Non-Award)

This document is part of an online Certificate in EnviroStats (Non-Award) by the University of Canberra. Course enquiries can be directed to the address below. Expressions of interest in the course can be made online through:

http://aerg.canberra.edu.au/envirostats

Copies of this publication are available from:

The Institute for Applied Ecology
University of Canberra ACT 2601
Australia

Email: **georges@aerg.canberra.edu.au**

Correct citation:

Georges, A. (2002). *Biometry: Statistics for Ecology and Natural Resource Management. Module 1: A Primer on SAS for Windows (Version 9.1).* Flexible Delivery Development Unit, Centre for the Enhancement of Learning, Teaching and Scholarship (CELTS), University of Canberra, ACT 2601, Australia. ISBN: 1 740880269

SPONSORED BY:

Australian Government
Australian Centre for International Agricultural Research

WorldFish CENTER

Australian Government
Fisheries Research and Development Corporation

Materials development team:

Author:	Arthur Georges, 2002, 2006
Instructional designer:	Peter Donnan, 2002
Editor:	Loretta Barnard, 2002
Graphic Design:	Peter Delgado, 2002
Desktop Publishing:	Kristi McDonald, 2004 Sue Bebbington, 2004
FDDU Project Manager:	Deborah Veness, 2002
Dynamic Web Page Design:	TCNI Software Solutions PO Box 47 LATHAM ACT 2615 Australia

First prepared in January, 2002 for Semester 1, 2002.
Reprinted January 2003 for Semester 1, 2003.
Reprinted January 2004 for Semester 1, 2004.
Reprinted November 2004 for Semester 1, 2005.
Revised and reprinted, June 2006
Revised and reprinted, June 2007

Published by Technology & Educational Design Services

(TEDS)
University of Canberra
ACT 2601, AUSTRALIA

Contents

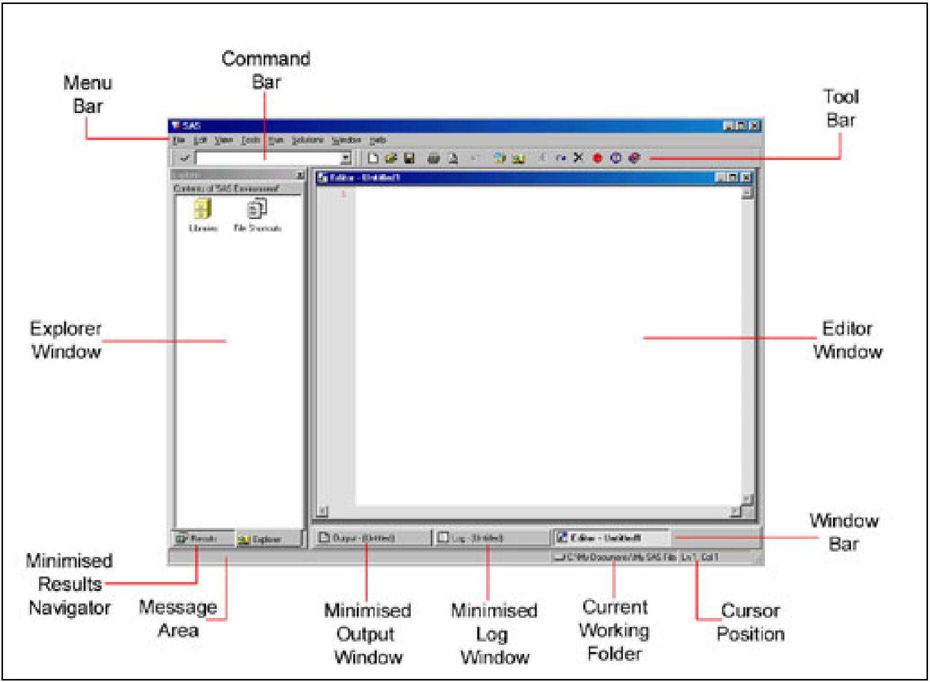
A Primer on SAS for Windows (Version 9.1)

Lesson 1: Key Concepts

The SAS interface

When you first start SAS, five windows are created (Figure 1-1). These windows are used to create and submit programs, to monitor the progress of program execution, to view and navigate the results, and to view and edit data files created during program execution.

Figure 1-1. SAS as it appears when it first starts. Only the EXPLORER and EDITOR windows are visible. The OUTPUT and LOG windows and the RESULTS NAVIGATOR are minimised.



A SAS program is the set of instructions you write to specify the analysis. Your program initially resides in the **EDITOR** window.

When a program is submitted for execution, SAS will:

- access the raw data set and read it into a temporary SAS workfile.
- report on the progress of program execution in the **LOG** window. This information is used for diagnostics and debugging.
- direct the results of the analysis to the **OUTPUT** window or to a **GRAPH** window.

The contents of any one of these windows can be re-directed to a printer or stored on disk for a permanent record.

An **EXPLORER** window allows you to access your SAS workfiles, to view and edit the data therein, to create new workfile libraries, and to delete workfiles and libraries.

A **RESULTS NAVIGATOR** provides a means of quickly navigating through completed analyses.

SAS as a programming language

SAS is a computer package with the combined features of a statistical package, a data entry and retrieval system, and a high level programming language.

As with most computer languages, SAS comprises:

- a set of **keywords** that are combined by the programmer to form statements or instructions to be executed.
- a clearly defined **sequence** in which these instructions are executed.
- a mechanism for **branching**, subject to some condition being met.
- a mechanism for **looping**, that is, repeating sequence of statements while a condition is met until a condition is met, or some fixed number of times.

You will need to become familiar with each of these concepts to use SAS to its full capability. Fortunately, SAS is a high level programming language, and relatively few programming instructions are required to undertake a set task.

Figure 1-2. A SAS Program. Keywords are highlighted in blue. The program is read in sequence one statement at a time from the top. Branching is provided with an IF statement. The DATA step is a loop, terminated by the DATALINES statement, and repeated in sequence for each line in the raw data.

```
DATA FOREST;
  INPUT SPECIES $ DENSITY HEIGHT DIAMETER;
  IF SPECIES="radiata" THEN DO;
    YIELD=DENSITY*3.1416*DIAMETER*HEIGHT*0.85;
  END;
  ELSE DO;
    YIELD=DENSITY*3.1416*DIAMETER*HEIGHT*0.62;
  END;
DATALINES;
radiata 0.42 10.1 0.6
radiata 0.39 8.2 0.45
radiata 0.44 9.3 0.56
radiata 0.44 10.2 0.53
pinaster 0.52 10.6 0.6
pinaster 0.54 8.2 0.55
pinaster 0.49 6.7 0.32
;
```

The raw data

When first using SAS, the data are best arranged as a rectangular block made up of a series of rows (referred to as observations or entities) and fields (referred to as variables or attributes). It makes matters much simpler if the raw data are arranged in this form.

Figure 1-2. The pig-nosed turtle, Carettochelys insculpta*, from Pul Pul billabong in Kakadu National Park [Photo: John Cann].*

The data in Table 1-1 are from a study of the pig-nosed turtle *Carettochelys insculpta* in Kakadu National Park (Figure 1-2) (Georges and Kennett, 1989).

The data are arranged so that each row contains data collected from an individual turtle. The measurements for each turtle are lined up to form columns of values.

The first variable contains the number of the tag attached to the turtle. The second variable contains the sex of each individual, MALE for mature males, FEMALE for mature females. Juvenile individuals cannot be reliably sexed, so the missing value code, a period, is entered in place of the sex code. The third variable contains shell lengths, the fourth variable contains head widths and the last variable contains body weights.

One or more columns of blanks separate variables. SAS will assign variable names to each field when the data are read.

In SAS terminology, each variable is said to occupy a field and the above data are said to be in fixed field format because for any one variable, there is a fixed field width. It is not necessary in SAS to arrange data in fixed field format, but it enhances readability and ease of editing.

Table 1-1. Measurements of the pig-nosed turtle, Carettochelys insculpta*.*

Tag Number	Sex	Carapace Length (cm)	Head Width (cm)	Weight (Kg)
10	MALE	41.0	7.15	7.60
11	FEMALE	46.4	8.18	11.00
2	.	24.3	4.42	1.65
15	.	28.7	4.89	2.18
16	.	32.0	5.37	3.00
3	FEMALE	42.8	7.32	8.60
4	MALE	40.0	6.60	6.50
5	FEMALE	45.0	8.05	10.90
12	FEMALE	44.0	7.55	8.90
13	.	28.0	4.85	1.97
6	FEMALE	40.0	6.53	6.20
8	.	32.0	5.35	2.90
9	MALE	35.0	5.74	3.90
17	FEMALE	35.1	6.04	4.50
19	MALE	42.3	6.77	7.80
22	FEMALE	48.1	8.55	12.80
105	MALE	44.0	7.10	9.00
14	MALE	43.0	6.60	7.20
7	FEMALE	48.0	8.67	13.50
1	.	29.2	5.10	2.38
104	MALE	44.0	7.35	9.00

Data usually are kept in a separate file on disk. There they can be readily accessed by a variety of computer packages, including SAS. SAS recognises the conventional drive:\path\filename.ext conventions of the Windows environment, though it uses a different convention to refer to its own workfiles (Refer to the section on Permanent SAS Data Sets later in this Workbook).

SAS can access data directly from Microsoft Excel spreadsheets, which is convenient for those familiar with that software.

The DATA step

Having created the file containing the raw data, you must instruct SAS to accept the data as a prelude to analysis. This is done by means of a programming step called a **DATA step**.

In a nutshell, the DATA step reads the raw data and transfers a copy to a temporary SAS workfile. In the process, SAS names are given to each of the variables and SAS can be instructed to create new variables, delete unwanted variables or observations, assign labels to values of a variable, etc.

Simple SAS programs contain only a single DATA step. Consider the following:

```
DATA TURTLE;
     INFILE "C:\MY DOCUMENTS\CARETTO.DAT";
     INPUT IDNO SEX $ LENGTH HDWIDTH WEIGHT;
     LGLENGTH=LOG10(LENGTH);
     LGWEIGHT=LOG10(WEIGHT);
RUN;
```

Note that the DATA step comprises a sequence of instructions or statements, each terminated with a semi-colon, so that lengthy statements may extend over many lines. Failure to include the semi-colon at the end of a statement is the most common error made. The indenting is to enhance readability, and is optional.

When this DATA step is submitted for execution, SAS goes to the source of the data nominated in the **INFILE statement** (ie file CARETTO.DAT in the directory C:\My Documents\) and reads one line of data.

Information is extracted from the line and stored in variables IDNO, SEX, LENGTH, HDWIDTH and WEIGHT as nominated in the **INPUT statement**. The names for these variables will be held with the data in a temporary SAS workfile.

Two new variables, LGLENGTH and LGWEIGHT, are created in **assignment statements** by taking the logarithm to base 10 of LENGTH and WEIGHT respectively. Note that these two new variables will be created in the temporary SAS workfile and not in the raw data file CARETTO.DAT.

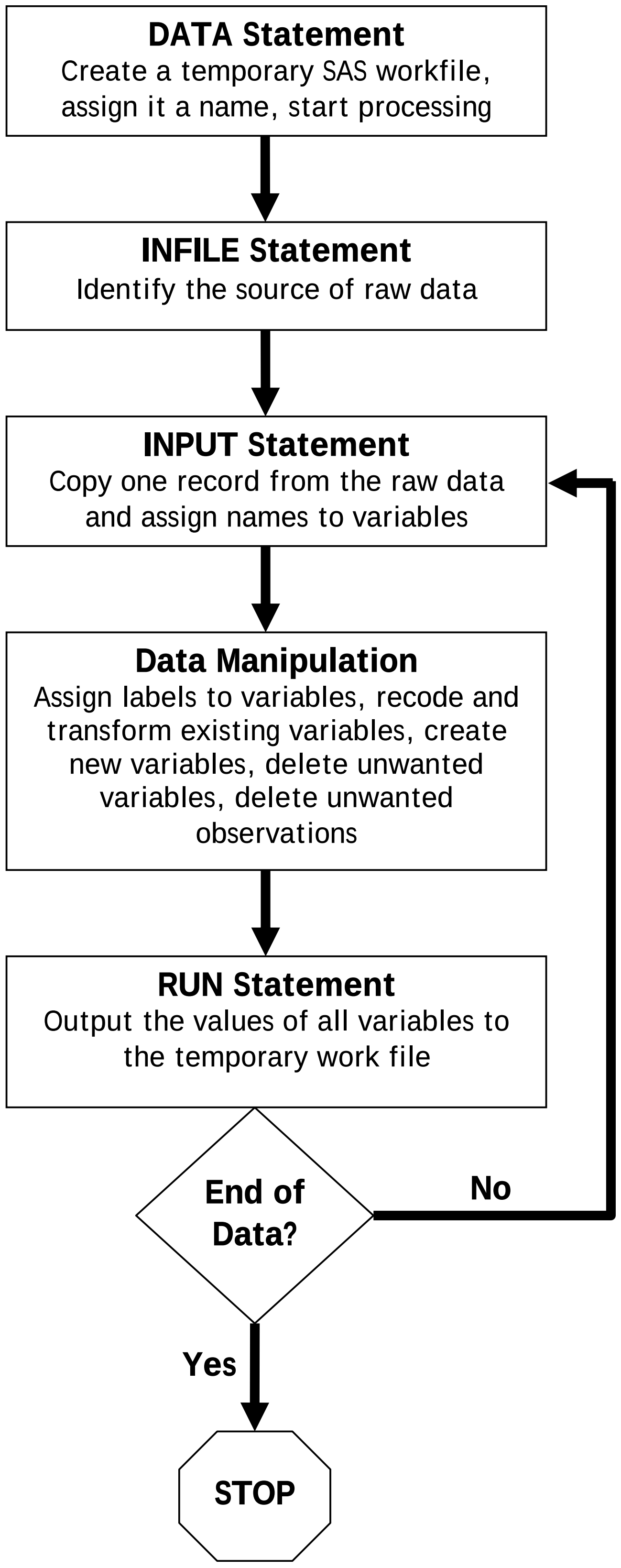
It is not until the RUN statement is encountered that SAS actually writes the data to the temporary SAS workfile, returns to the beginning of the DATA step and processes the next line of data. The entire procedure is repeated until all lines of data have been processed (Figure 1-3).

Concept

The DATA step should be viewed as a program loop, beginning with the DATA statement and ending with the RUN statement, and executing all statements once for each line of data in the raw data file.

In the absence of instructions to the contrary, all of the statements in a DATA step are executed once before data are written to the temporary SAS workfile. Appreciation of this point will save considerable confusion later.

Figure 1-3. Workflow for a simple DATA step. The analysis is not executed until the RUN statement is encountered.


DATA Statement
Create a temporary SAS workfile, assign it a name, start processing
INFILE Statement
Identify the source of raw data
INPUT Statement
Copy one record from the raw data and assign names to variables
Data Manipulation
Assign labels to variables, recode and transform existing variables, create new variables, delete unwanted variables, delete unwanted observations
RUN Statement
Output the values of all variables to the temporary work file
End of Data?
No
Yes
STOP

The PROC step

Once the data have been read from the raw data file into the SAS workfile, analysis can begin in earnest. Statistical analysis is performed by means of a programming code called a PROC step. Typically, a SAS program will comprise a single DATA step followed by one or more PROC steps.

A PROC step takes the data in the temporary SAS workfile (not the raw data), processes it and produces results in the form of output on the screen or printer (Figure 1-4).

The following is an example of a SAS procedure:

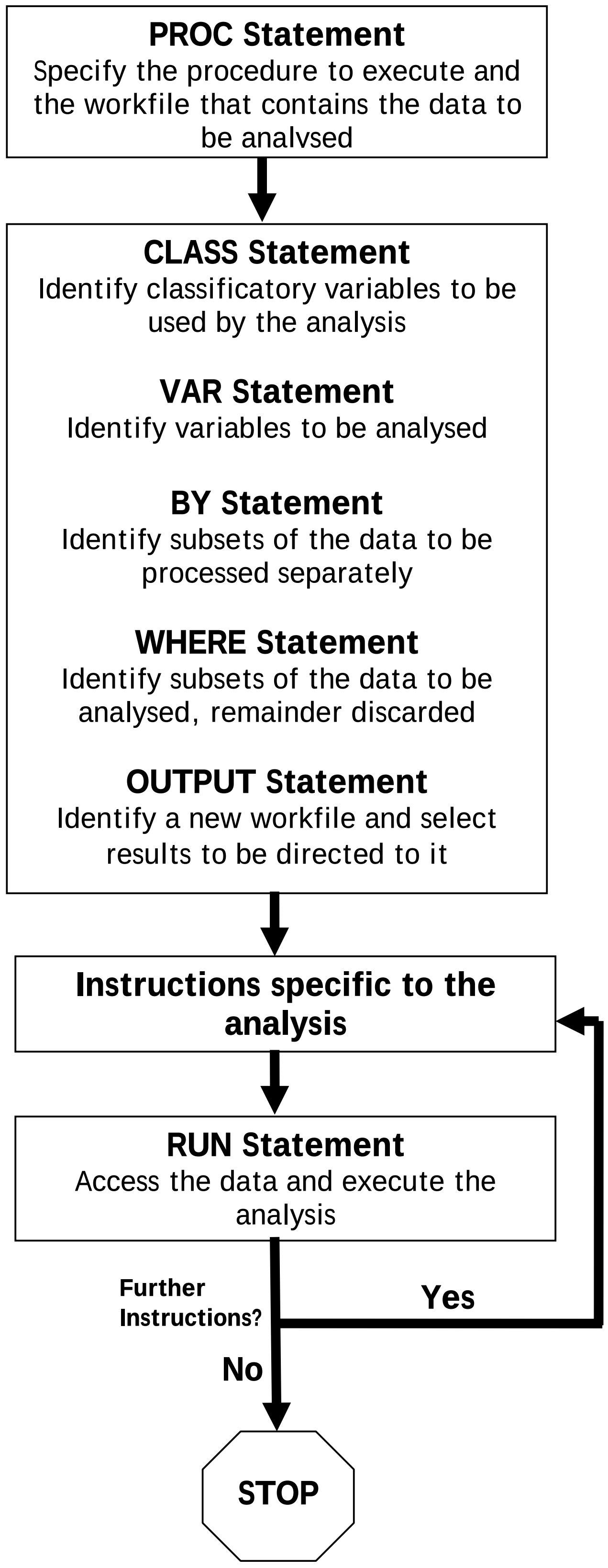
```
PROC MEANS DATA=TURTLE;
     VAR LENGTH HDWIDTH WEIGHT;
     BY SEX;
RUN;
```

This will produce means and other summary statistics for the variables LENGTH, HDWIDTH and WEIGHT, calculated separately for each value of SEX. The results will appear in the OUTPUT window.

Concept

PROC steps do not access the raw data directly, but rather the temporary SAS work file. The raw data file is only accessed in the initial DATA step, not by subsequent analysis. Hence, changes that are made to the data as they are processed do not alter the raw data in any way.

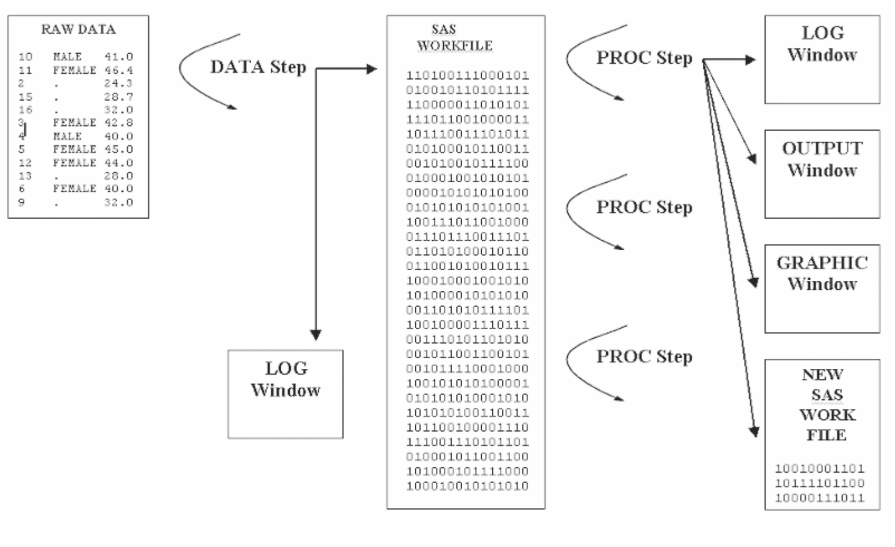
Figure 1-4. Workflow for a simple PROC step. The analysis is not executed until the RUN statement is encountered.


PROC Statement
Specify the procedure to execute and the workfile that contains the data to be analysed
CLASS Statement
Identify classificatory variables to be used by the analysis
VAR Statement
Identify variables to be analysed
BY Statement
Identify subsets of the data to be processed separately
WHERE Statement
Identify subsets of the data to be analysed, remainder discarded
OUTPUT Statement
Identify a new workfile and select results to be directed to it
Instructions specific to the analysis
RUN Statement
Access the data and execute the analysis
Further Instructions?
Yes
No
STOP

Where have we come?

In summary, SAS first reads data into a temporary workfile according to the instructions you give it in a DATA step, then analyses the data in this temporary file according to the instructions you give it in one or more PROC steps. With few exceptions, transformations, creation of new variables, and general manipulations of the data must occur in the DATA step prior to analysis, so some forethought is required.

A diagrammatic representation of the flow of information during the execution of a simple DATA step and a simple PROC step is presented in Figure 1-5.



```
10    MALE   41.0
11    FEMALE 46.4
2     .      24.3
15    .      28.7
16    .      32.0
3     FEMALE 42.8
4     MALE   40.0
5     FEMALE 45.0
12    FEMALE 44.0
13    .      28.0
6     FEMALE 40.0
9     .      32.0
```

Figure 1-5. Information flow during the execution of a simple DATA step followed by one or more PROC steps. The DATA step copies data from the raw data file, manipulates it, and transfers it to a SAS workfile, analysing the DATA therein to produce textual and graphic output, and selected results stored in a new workfile, if required. The progress of statements contributing to each step are logged in the LOG window.

Key take-home points are:

- SAS is a programming language.
- DATA steps are segments of programming code for reading and manipulating data.
- DATA steps do not alter the raw data, but rather copy the data to a SAS working file along with whatever manipulations you request.
- DATA steps are loops, executing once for each line of data.
- DATA steps transfer data to the SAS workfile on encountering the RUN statement or DATALINES statement (or when explicitly requested to do so with an OUTPUT statement).
- PROC steps are segments of programming code for analysing your data.
- A simple SAS program ypitically comprises one DATA step and several PROC steps.

Lesson 2: Step-through Examples

Getting Started

If SAS is properly installed, you should then be able to run it by double-clicking on the relevant icon on the desktop or in an Applications Group window. Consult the computer systems officer at your institution for details of how to invoke SAS.

Download the data for the course and place the files in the directory C:\My Documents\, if you have not done so already

Double-click on the SAS icon.

Note

If you have placed the data elsewhere on your hard disk, you will need to replace C:\My Documents\ with the appropriate drive and sub-directory specifications (e.g. C:\Temp\).

Throughout this module, your action is required only when you encounter instructions inside an Activity Box like this one.

SAS windows

Once SAS has commenced running, the program will display five windows (see Figure 1-1):

- the OUTPUT window,
- the LOG window,
- the EDITOR window,
- the EXPLORER window and
- the RESULTS window.

Both the LOG and OUTPUT windows are minimised, and the RESULTS window is hidden behind the EXPLORER window for the time being.

The EDITOR window is used to create SAS programs and for entering data. Several EDITOR windows can be open at once.

When you run a SAS program, the results of the analysis appear in the OUTPUT window and a report of the progress of the program, including error messages, appears in the LOG window.

The RESULTS window provides a history of the overall analysis. The EXPLORER window enables you to view and edit SAS workfiles.

You can move between the windows using the window bar at the bottom of the SAS display –- RESULTS, EXPLORER, OUTPUT, LOG and EDITOR. Each window has a number of useful icons on the tool bar at the top of the screen (Figure 1-1).

Move to the EDITOR window. Familiarise yourself with the functions of the icons on the tool bar by moving to each one in turn with the mouse – this will display the function of the icon.

To demonstrate how this system of five windows operates, a SAS program to perform a simple task has been provided. You must first read the demonstration program into the EDITOR, so that you may peruse it and ultimately submit it for execution.

Locate and select the file C:\My Documents\DEM01.SAS

Note that several lines of code, a SAS program, have appeared in the EDITOR. Do not worry about what the lines mean at this stage.

A SAS program must be submitted before it is executed.

Submit the demo program for execution.

Note that the LOG is displayed with a description of the progress on the analysis. If there are no serious errors identified in the LOG (and there should be none), move to the OUTPUT window to view the results of the analysis. In this case, you should see a histogram, a statistical summary and a listing of the data.

You can navigate around the OUTPUT window using the standard keyboard arrow keys, or by using the RESULTS NAVIGATOR.

Move to the OUTPUT window. Use the RESULTS NAVIGATOR to view the various elements of the output.

Clear the contents of the OUTPUT and LOG windows.

Creating a data file

You can create data files on your data disk before running SAS by using your favourite editor, spreadsheet or word processor (in text only mode) if you wish. Alternatively, data can be entered via the EDITOR window.

Recall that the data in Table 1-2 are from a study of the pig-nosed turtle, *Carettochelys insculpta*, in Kakadu National Park (Georges and Kennett, 1989).

Table 1-2. Measurements of the pig-nosed turtle, Carettochelys insculpta, *from Pul Pul billabong in Kakadu National Park.*

Tag Number	Sex	Carapace Length (cm)	Head Width (cm)	Weight (Kg)
10	MALE	41.0	7.15	7.60
11	FEMALE	46.4	8.18	11.00
2	.	24.3	4.42	1.65
15	.	28.7	4.89	2.18
16	.	32.0	5.37	3.00
3	FEMALE	42.8	7.32	8.60
4	MALE	40.0	6.60	6.50
5	FEMALE	45.0	8.05	10.90
12	FEMALE	44.0	7.55	8.90
13	.	28.0	4.85	1.97
6	FEMALE	40.0	6.53	6.20
8	.	32.0	5.35	2.90
9	MALE	35.0	5.74	3.90
17	FEMALE	35.1	6.04	4.50
19	MALE	42.3	6.77	7.80
22	FEMALE	48.1	8.55	12.80
105	MALE	44.0	7.10	9.00
14	MALE	43.0	6.60	7.20
7	FEMALE	48.0	8.67	13.50
1	.	29.2	5.10	2.38
104	MALE	44.0	7.35	9.00

The first variable contains the numbers printed on the tags issued to each turtle. The second variable contains a code for the sex of each turtle, MALE for males, FEMALE for females. Juveniles cannot be reliably sexed, so the missing value code, a period, is entered in place of the sex code. SAS recognises the period as a missing value. The third variable contains shell lengths, the fourth variable contains head widths and the last variable contains body weights.

Clear the contents of the EDITOR. Type in the data of Table 1-2. Use SPACES, not TABS to align data into columns.

The data are lined up neatly in columns for ease of reading and editing. SAS requires only that the numbers and character strings be separated from each other by one or more blanks. Character strings cannot exceed eight characters in total.

Once you have checked your typing and corrected any errors, you can save the data in a disk file in the directory C:\My Documents, or wherever you have chosen to store your files.

Make sure that you are in the EDITOR window, then save the contents to the file C:\My Documents\MYDATA.DAT

Note

You should follow the convention of using .DAT as the extension to all files containing raw data.

Clear the contents of the EDITOR.

Creating a program

The usual way to create programs is in the EDITOR window.

Move to the EDITOR window and type in the following sample SAS program.

```
DATA MYDATA;
     INFILE "C:\MY DOCUMENTS\MYDATA.DAT";
     INPUT IDNO SEX $ LENGTH HDWIDTH WEIGHT;
     LGLENGTH=LOG10(LENGTH);
     LGWEIGHT=LOG10(WEIGHT);
RUN;
PROC MEANS;
     VAR LENGTH HDWIDTH WEIGHT;
RUN;
```

Note

Be very careful to terminate each statement with a semi-colon. Failure to do this will lead to confusion. It is the most common mistake made in SAS programming.

Be very careful to balance the quotes around the filename C:\MY DOCUMENTS\MYDATA.DAT. Omission of one of the quotes will lead to an obscure error message that will persist until the second quote is submitted.

Again, you don't need to worry at all about what the program means at this stage.

You can use your favourite editor, spreadsheet or word processor to create a SAS program, provided that you can save it in text mode. You can then open the file into the EDITOR window.

It is prudent to save a copy of your program on your data disk before submitting the program to SAS for execution.

Make sure that you are in the EDITOR window, then save the contents to the file C:\MY DOCUMENTS\MYPROG.SAS

Note

You should follow the convention of using .SAS as the extension to all disk files containing SAS programs. It will save a great deal of confusion.

Executing a program

A SAS program must be submitted for execution. After execution, it will remain in the EDITOR window for editing and re-submission, if things go awry.

Submit the demo program for execution.

Quite a number of actions are taken when you submit a program for execution. First, a log of the progress of the program will appear in the LOG window. You should take note of any errors, shown in red text, as these indicate a fatal problem with your program. Warnings should also be heeded, as they indicate that the syntax is correct, but the analysis itself may have problems.

Second, a library called WORK will contain a SAS workfile named, in this case, MYDATA. This is the name that you provided in the data step. Under SAS filename conventions, you can refer to this workfile explicitly as WORK.MYDATA, but this is seldom necessary.

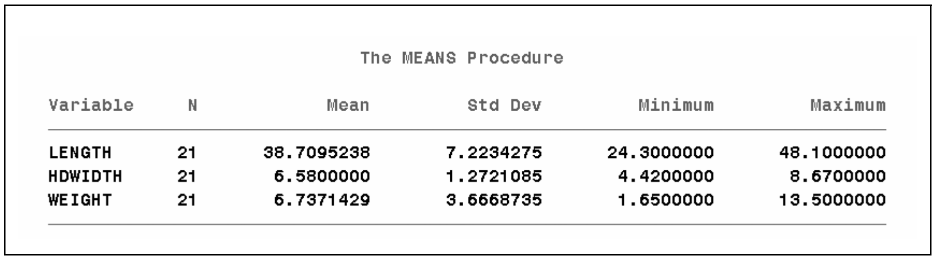
You can peruse the data at this point to see if it has been read as intended.

Use the EXPLORER window to locate the SAS workfile WORK.MYDATA and examine its contents.

Note that the variables created in the DATA step, LGLENGTH and LGWEIGHT, appear in the workfile along with those you read from the raw data file.

Third, the output of the analysis, if any, appears in the OUTPUT window (Box 1-1).

Close the window containing the data table, move to the OUTPUT window and peruse the output.


Box 1-1. Sample output from a SAS program as it would appear in the OUTPUT window

```
                            The MEANS Procedure

Variable     N            Mean         Std Dev         Minimum         Maximum
------------------------------------------------------------------------------
LENGTH      21      38.7095238       7.2234275      24.3000000      48.1000000
HDWIDTH     21       6.5800000       1.2721085       4.4200000       8.6700000
WEIGHT      21       6.7371429       3.6668735       1.6500000      13.5000000
------------------------------------------------------------------------------
```

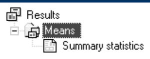
You can save the output to disk for later incorporation into word processing documents or for printing.

Make sure that you are in the OUTPUT window, then save the contents to the file C:\MY DOCUMENTS\MYPROG.LST

Note

You should follow the convention of using .LST as the extension to all disk files containing SAS output. It will save a great deal of confusion.

Finally, an entry pointing to the output of the analysis appears in the RESULTS NAVIGATOR. This will assist you in moving back to the results of the analysis later.


Results
Means
Summary statistics

Move to the RESULTS NAVIGATOR window and locate the entry pointing to the current analysis.

When things go wrong

There is a dreaded shadow that hangs over all who engage in computing — SYNTAX. If you don't get it right, the program will not work. Finding out why it will not work is not always easy but SAS assists by colour coding its syntax in the EDITOR window. Many simple mistakes will be immediately evident.

In the example you have just run, there should have been no errors. Had there been, you would expand the LOG window and peruse its contents to gain clues as to what went wrong. Once the problem is identified, you can edit the program and re-submit it to SAS. Let's try running a program in which there is a deliberate error.

Move to the EDITOR window.

Move the cursor to the semi-colon on the PROC MEANS line and delete the semi-colon. A subtle colour change indicates the error.

Re-submit the flawed program.

The program will no longer work so nothing should appear in the OUTPUT window. The contents of the LOG window relating to the MEANS procedure will read as follows:

```
56   PROC MEANS
57       VAR LENGTH HDWIDTH WEIGHT;
             ------ -------

             22     202

ERROR 22-322: Syntax error, expecting one of the following:
;, ALPHA, CHARTYPE, CLASSDATA, CLM, COMPLETETYPES, CSS, CV,
DATA, DESCEND, DESCENDING, DESCENDTYPES, EXCLNPWGT,
EXCLNPWGTS, EXCLUSIVE, FW, IDMIN, KURTOSIS, LCLM, MAX,
MAXDEC, MEAN, MEDIAN, MIN, MISSING, N, NDEC, NMISS, NONOBS,
NOPRINT, NOTHREADS, NOTRAP, NWAY, ORDER, P1, P10, P25, P5,
P50, P75, P90, P95, P99, PCTLDEF, PRINT, PRINTALL,
PRINTALLTYPES, PRINTIDS, PRINTIDVARS, PROBT, Q1, Q3,
QMARKERS, QMETHOD, QNTLDEF, QRANGE, RANGE, SKEWNESS,
STDDEV, STDERR, SUM, SUMSIZE, SUMWGT, T, THREADS, UCLM,
USS, VAR, VARDEF.

ERROR 202-322: The option or parameter is not recognized
and will be ignored.

58   RUN;

NOTE: The SAS System stopped processing this step because
of errors.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.06 seconds
      cpu time            0.00 seconds
58 PROC MEANS
```

Note

The SAS System stopped processing this step because of errors.

In this case, the omitted semi-colon led SAS to take the first statement in the program to be:

```
PROC MEANS VAR LENGTH HDWIDTH WEIGHT;
```

and realised the error when it encountered the words LENGTH and HDWIDTH. It underlined the offending words and printed out the diagnostics to the LOG window.

Go to the EDITOR window again and correct the error.

Re-submit the program once more to see that it's working.

Clear the contents of the EDITOR.

Writing SAS Programs

You now know how to enter, edit and save data, to enter a SAS program, to submit it for execution, and to modify and re-submit SAS programs if they do not initially work. Your next step is to learn how to write sensible SAS programs.

The DATA step

SAS programs consist of at least two steps—a DATA step followed by one or more PROC steps. In the DATA step, you specify where the raw data resides, the variable names to be assigned to each field in the data file, and you perform any manipulations or transformations of the data before analyses. Consider the DATA step you used in the section Program Creation and Execution:

```
DATA MYDATA;
     INFILE "C:\MY DOCUMENTS\MYDATA.DAT";
     INPUT IDNO SEX $ LENGTH HDWIDTH WEIGHT;
     LGLENGTH=LOG10(LENGTH);
     LGWEIGHT=LOG10(WEIGHT);
RUN;
```

Note

Note that all statements within the data step must end with a semi-colon. This permits statements to extend over more than one line, terminated only when SAS encounters a semi-colon.

This particular DATA step will read five fields of raw data from the disk file MYDATA.DAT in directory C:\MY DOCUMENTS\ into a temporary SAS workfile called WORK.MYDATA.

The first field of the data will be called IDNO (remember, it contains the turtle tag numbers), the second field will be called SEX, the third LENGTH and so on. The dollar sign after the variable name SEX indicates to SAS that SEX is a discrete variable containing characters, in this case the words MALE or FEMALE. Variable names cannot exceed eight characters in length.

SAS will perform all subsequent analyses on the data contained in the temporary SAS file WORK.MYDATA, until another SAS file is referred to in a second DATA step or explicitly in a PROC step.

Lines 4 and 5 contain assignment statements, which create new variables called LGLENGTH and LGWEIGHT. The column LGLENGTH will contain the length values logged to base 10.

The last line in the DATA step is a RUN statement. SAS does not actually execute any of the statements in the DATA step until it encounters the RUN statement.

Remember that you saved the above DATA step to disk earlier in this tutorial. Read it back into SAS, remove the statements relating to the MEANS procedure, and submit it once more, then peruse the log window in case the program was adulterated when you were exploring what happened when a semicolon was omitted.

Move to the EDITOR window and read in the file
C:\MY DOCUMENTS\MYPROG.SAS

Remove the PROC MEANS step, then submit the program for execution.

Assignment statements

Assignment statements allow you to use any arithmetic expression to create contents for an existing or new variable. For example, we might want to convert our ablsolute measure of ground cover in 2 x 2 m plots to a percentage ground cover:

```
PCOVER = COVER*100/4;
```

or to convert a variable measured in metres to cm

```
LENGTH = LENGTH/100;
```

Yield of wood from a tree might be calculated as a function of wood density, basal diameter and tree height:

```
YIELD=0.85*DENSITY*3.1416*DIAMETER*HEIGHT;
```

Any number of such assignment statements can be included in a DATA step. Some useful functions that can be included in assignment statements are:

ABS (exp)	Take the absolute value of an expression (exp)
ARSIN (exp)	Take the arcsin of an expression (answer in radians)
COS (exp)	Take the cosine of an expression
EXP (exp)	Raises e to the power of an expression
LOG (exp)	Takes the natural log of an expression
LOG10 (exp)	Takes the log to base 10 of an expression
SIN (exp)	Takes the sine of an expression
SQRT (exp)	Takes the square root of an expression
TAN (exp)	Takes the tangent of an expression

A full list of SAS functions can be obtained using the SAS help facility (click on the HELP button and peruse the bookmarks). To exit from a help screen, double-click on the box in the top left hand corner of the screen.

For example, transforming counts of macroinvertebrates using a standard square root transformation is effected by

```
COUNT = SQRT(COUNT+0.5);
```

Any arithmetic expression using + (addition), – (subtraction), * (multiplication), / (division) or **(exponentiation) can be used in an assignment statement.

If fish size changes with time exponentially in accordance with the Von Bertalanffy growth equation

$$L = 3.00 - 2.5e^{0.138t}$$

we can easily code this in SAS as

```
LENGTH = 3.00 - 2.5*EXP(0.138*TIME);
```

More complicated equations are possible by splitting them over several lines. For example, the Sharpe-DeMichele growth model relating embryonic growth with temperature is given by

$$\frac{ds}{dt} = \frac{RHO_{25}\frac{T}{298.15}\exp\left[\frac{H_A}{r}\left(\frac{1}{298.15}-\frac{1}{T}\right)\right]}{1+\exp\left[\frac{H_L}{r}\left(\frac{1}{T_L}-\frac{1}{T}\right)\right]+\exp\left[\frac{H_H}{r}\left(\frac{1}{T_H}-\frac{1}{T}\right)\right]}$$

and can be coded in SAS as follows, using the temporary intermediary variables C1, C2 and C3 which are subsequently discarded with a DROP statement. Temperature is in degrees Kelvin, and *r* in the equation above is the Gas Constant 1.987.

```
DEGK=TEMP+273.15;
C1=EXP((1/298.15-1/DEGK)*HA/1.987);
C2=EXP((1/TL-1/DEGK)*HL/1.987);
C3=EXP((1/TH-1/DEGK)*HH/1.987);
RATE=(RHO25*DEGK*C1/298.15)/(1+C2+C3);
DROP C1 C2 C3;
```

PROC steps

SAS is interactive, so the DATA step can be written and submitted to SAS separately from subsequent PROC steps. Indeed, each of the PROC steps should be submitted individually to enable flexibility in decisions on the direction that the analysis takes.

In a PROC step, you specify the statistical analyses to be performed on the data read in with the DATA step. A typical SAS program consists of a single DATA step followed by one or more PROC steps. It is important to appreciate that you do not need to read in the data each time you perform an analysis. This is a common mistake. Only read in the data once for a particular session even if you plan to execute several PROC steps.

A PROC step consists of a PROC statement, various sub-statements and a RUN statement. For example,

```
PROC REG DATA=MYDATA;
     MODEL WEIGHT=HDWIDTH;
RUN;
```

will perform a simple linear regression of body weight (WEIGHT) on head width (HDWIDTH).

There are many procedures available in SAS and the following exercises are designed as a painless introduction to some of them. As you have executed a DATA step to read in the data for the pig-nosed turtle, the following exercises analyse these data.

Where have we come?

The objective of the step-through exercises we have just done was to reinforce the key concepts introduced in lesson 1. In particular, you can now appreciate more that

- SAS is a programming language.
- DATA steps are segments of programming code for reading and manipulating data, including transforming, recoding and creating new variables using assignment statements.
- PROC steps are segments of programming code for analysing your data.
- A simple SAS program ypitically comprises one DATA step and several PROC steps.

You have also learnt the value of the SAS Windows.

- The EDITOR Window can be used for creating data sets and for creating your SAS programs.
- SAS programs need to be submitted for execturion.
- The results of the analysis appear in the OUTPUT Window.
- When things go wrong, the first port of call is the LOG Window.

Now let’s move on to the Lesson 3 to try a variety of PROC steps.

Lesson 3: Sample Statistical Analyses

Getting Started

If you are not continuing directly from the previous lesson, you will need to start SAS again by double-clicking on the relevant icon on the desktop or in an Applications Group window.

Double-click on the SAS icon.

If you are not continuing directlry from the previous lesson, you will need also to refamiliarise SAS with the dataset. If you have the relevant DATA step saved, read it into the EDITOR Window. Otherwise, you will need to type it in again.

```
DATA MYDATA;
     INFILE "C:\MY DOCUMENTS\CARETTO.DAT";
     INPUT IDNO SEX $ LENGTH HDWIDTH WEIGHT;
     LGLENGTH=LOG10(LENGTH);
     LGWEIGHT=LOG10(WEIGHT);
RUN;
```

Submit the above program for execution.

Listing data

Having read the data into SAS and performed some transformations, you might wish to list it to the screen to see that it was read in correctly. Use the procedure PRINT.

Move to the EDITOR window and enter the following SAS program immediately after your DATA step.

```
PROC PRINT DATA=MYDATA;
     VAR LENGTH HDWIDTH WEIGHT LGLENGTH
     LGWEIGHT SEX;
RUN;
```

Check the syntax, correct any errors, highlight the PROC PRINT step with the mouse and submit the program for execution.

By highlighting the PROC PRINT step before submission, only that step is executed. This avoids the problem of repeated submission of all the code that went before, in this case the DATA step.

The output is shown in Box 1-2. PROC PRINT is a quick and easy way to verify that your data has been read by SAS as intended.

Box 1-2. Output from PROC PRINT, a procedure used to display the contents of a SAS datafile in the OUTPUT window.

```
 Obs    LENGTH    HDWIDTH    WEIGHT    LGLENGTH    LGWEIGHT    SEX

   1     41.0      7.15       7.60     1.61278     0.88081    MALE
   2     46.4      8.18      11.00     1.66652     1.04139    FEMALE
   3     24.3      4.42       1.65     1.38561     0.21748
   4     28.7      4.89       2.18     1.45788     0.33846
   5     32.0      5.37       3.00     1.50515     0.47712
   6     42.8      7.32       8.60     1.63144     0.93450    FEMALE
   7     40.0      6.60       6.50     1.60206     0.81291    MALE
   8     45.0      8.05      10.90     1.65321     1.03743    FEMALE
   9     44.0      7.55       8.90     1.64345     0.94939    FEMALE
  10     28.0      4.85       1.97     1.44716     0.29447
  11     40.0      6.53       6.20     1.60206     0.79239    FEMALE
  12     32.0      5.35       2.90     1.50515     0.46240
  13     35.0      5.74       3.90     1.54407     0.59106    MALE
  14     35.1      6.04       4.50     1.54531     0.65321    FEMALE
  15     42.3      6.77       7.80     1.62634     0.89209    MALE
  16     48.1      8.55      12.80     1.68215     1.10721    FEMALE
  17     44.0      7.10       9.00     1.64345     0.95424    MALE
  18     43.0      6.60       7.20     1.63347     0.85733    MALE
  19     48.0      8.67      13.50     1.68124     1.13033    FEMALE
  20     29.2      5.10       2.38     1.46538     0.37658
  21     44.0      7.35       9.00     1.64345     0.95424    MALE
```

Descriptive statistics

Descriptive statistics are a useful place to start an analysis. They can be obtained by using the MEANS procedure as follows:

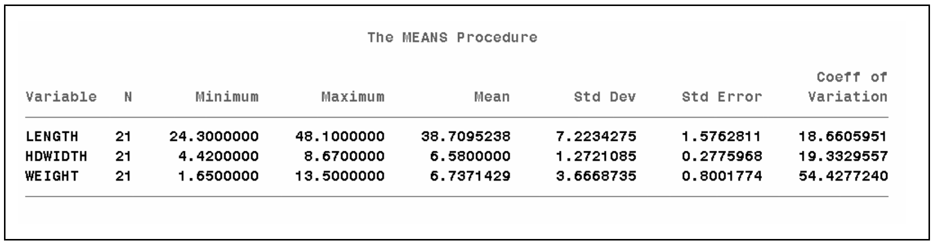
```
PROC MEANS DATA=MYDATA N MIN MAX
     MEAN STD STDERR CV;
     VAR LENGTH HDWIDTH WEIGHT;
RUN;
```

Enter the PROC MEANS step above immediately after the PROC PRINT step, highlight it with the mouse and submit it for execution.

Remember, you don't want to run the DATA step and PROC PRINT step again. Highlighting the code you wish to execute avoids this problem.

The PROC MEANS step will calculate means, standard deviations, standard errors, sample sizes, minimum and maximum values, and coefficients of variation for each of the variables specified in the VAR statement. The output is shown in Box 1-3.

Box 1-3. Sample output from PROC MEANS.



```
                                   The MEANS Procedure

                                                                                 Coeff of
Variable   N       Minimum       Maximum          Mean       Std Dev     Std Error     Variation
-----------------------------------------------------------------------------------------------
LENGTH    21    24.3000000    48.1000000    38.7095238     7.2234275     1.5762811    18.6605951
HDWIDTH   21     4.4200000     8.6700000     6.5800000     1.2721085     0.2775968    19.3329557
WEIGHT    21     1.6500000    13.5000000     6.7371429     3.6668735     0.8001774    54.4277240
-----------------------------------------------------------------------------------------------
```

Alternatively, you might want descriptive statistics calculated separately for each sex class, in which case the following steps would be needed:

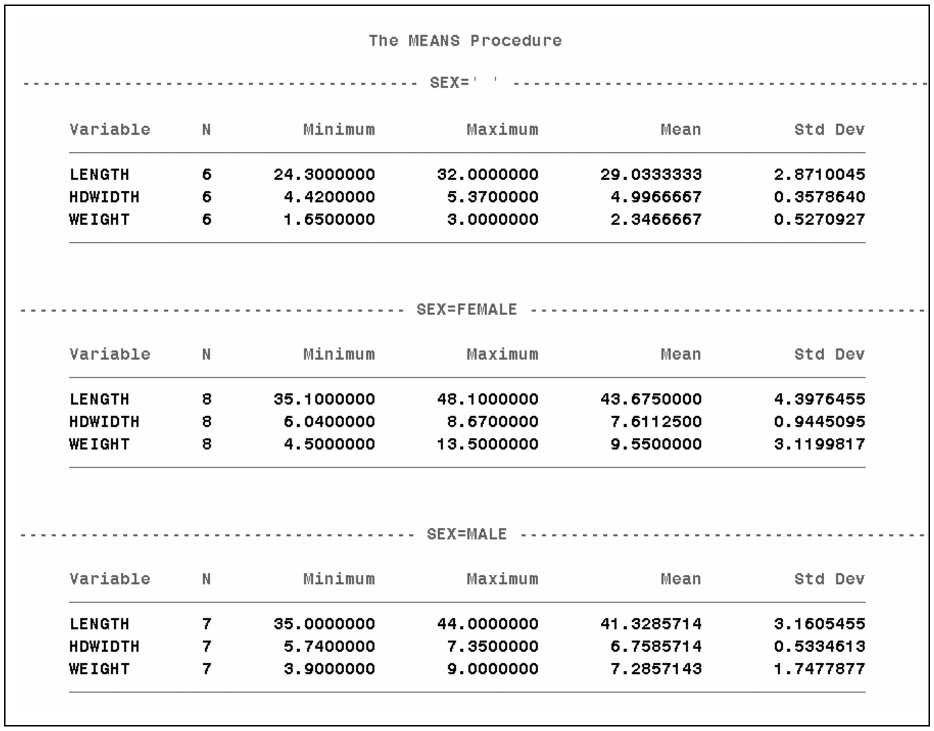
```
PROC SORT DATA=MYDATA;
     BY SEX;
PROC MEANS N MIN MAX MEAN STD;
     VAR LENGTH HDWIDTH WEIGHT;
     BY SEX;
RUN;
```

You need to sort the data on SEX using PROC SORT before analysing it with PROC MEANS applied separately to each SEX class with the BY statement.

Submit the above program for execution.

The output is shown in Box 1-4. The first set of results is for the animals of unspecified sex. The second and third sets are for females and males, respectively.

Box 1-4. Sample output from PROC MEANS with a BY statement. The statistics are presented separately for each value of the variable SEX.



```
                                  The MEANS Procedure

-------------------------------------- SEX=' ' ---------------------------------------

     Variable     N         Minimum         Maximum            Mean         Std Dev
     ------------------------------------------------------------------------------
     LENGTH       6      24.3000000      32.0000000      29.0333333       2.8710045
     HDWIDTH      6       4.4200000       5.3700000       4.9966667       0.3578640
     WEIGHT       6       1.6500000       3.0000000       2.3466667       0.5270927
     ------------------------------------------------------------------------------


------------------------------------- SEX=FEMALE --------------------------------------

     Variable     N         Minimum         Maximum            Mean         Std Dev
     ------------------------------------------------------------------------------
     LENGTH       8      35.1000000      48.1000000      43.6750000       4.3976455
     HDWIDTH      8       6.0400000       8.6700000       7.6112500       0.9445095
     WEIGHT       8       4.5000000      13.5000000       9.5500000       3.1199817
     ------------------------------------------------------------------------------


-------------------------------------- SEX=MALE ---------------------------------------

     Variable     N         Minimum         Maximum            Mean         Std Dev
     ------------------------------------------------------------------------------
     LENGTH       7      35.0000000      44.0000000      41.3285714       3.1605455
     HDWIDTH      7       5.7400000       7.3500000       6.7585714       0.5334613
     WEIGHT       7       3.9000000       9.0000000       7.2857143       1.7477877
     ------------------------------------------------------------------------------
```

More detailed statistical summaries can be provided by PROC UNIVARIATE, which yields modes, medians, percentiles and diagnostic statistics such as tests of normality in addition to what is provided by PROC MEANS.

```
PROC UNIVARIATE DATA=MYDATA;
 VAR LENGTH HDWIDTH WEIGHT;
RUN;
```

Submit the above program for execution.

The output is too voluminous to reproduce here.

Histograms, barcharts

Size distributions are an important biological characteristic of populations of animals with indeterminate growth, such as turtles. To obtain a size distribution for the Kakadu population of pig-nosed turtles, the following step is appropriate:

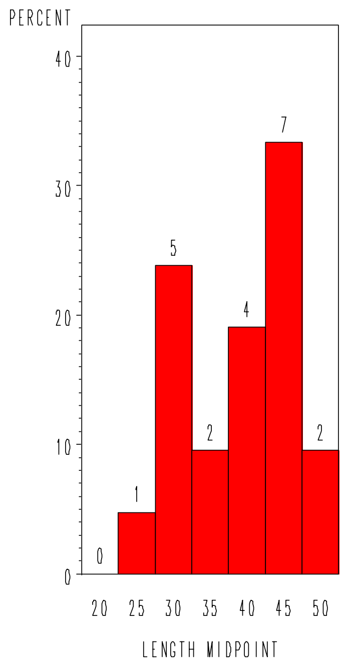
```
PROC GCHART DATA=MYDATA;
     VBAR LENGTH / TYPE=PCT SPACE=0
     MIDPOINTS = 20.0 TO 50.0 BY 5.0 FREQ;
RUN;
```

This step will produce a histogram of the variable LENGTH based on the percentage (TYPE=PCT) of individuals falling in each of the intervals specified by the MIDPOINTS option. Each interval will be 5.0 cm wide. The raw counts of turtles in each interval will be given at the head of each column (FREQ option).

Submit the above program for execution.

The output appears in Figure 1-6. Note that your graph may differ from that shown owing to differences in screen attributes.

Figure 1.6. Size distribution for the population of pig-nosed turtles from Kakadu National Park. Length is in cm. The graph was produced with PROC GCHART.



Histograms are fine for continuous data, but for discrete data it is customary to construct barcharts (the columns of a barchart are separated by a space, Figure 1-7). In the turtle data set the variable SEX is discrete, and the following program is appropriate:

```
PROC GCHART DATA=MYDATA;
     VBAR SEX / DISCRETE TYPE=PCT SPACE=10;
RUN;
```

Submit the above program for execution.

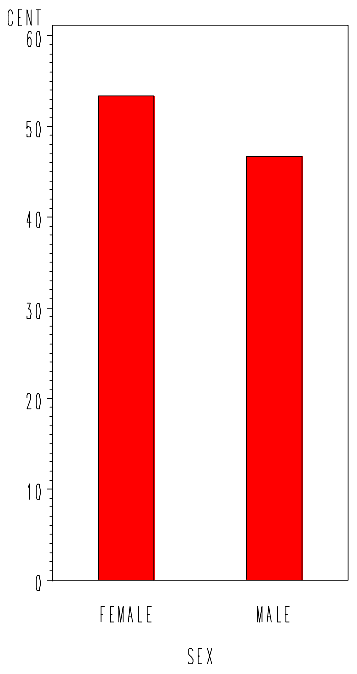


Figure 1.7. A barchart showing the relative frequency of males and females in a population of pig-nosed turtles from Kakadu National Park. The graph was produced using PROC GCHART.

T-tests

To perform a student's T-test with SAS, the data must be in the form of a measurement variable occupying one data column and a breakdown variable occupying another. The breakdown variable must have only two values, not counting missing values.

In the example at hand, we might choose to compare the body weights of males (SEX coded as MALE) with those of females (SEX coded as FEMALE). Because juveniles were coded with the missing value code '.' the TTEST procedure will ignore them. Here is the appropriate program:

```
PROC TTEST DATA=MYDATA;
     CLASS SEX;
     VAR LENGTH;
RUN;
```

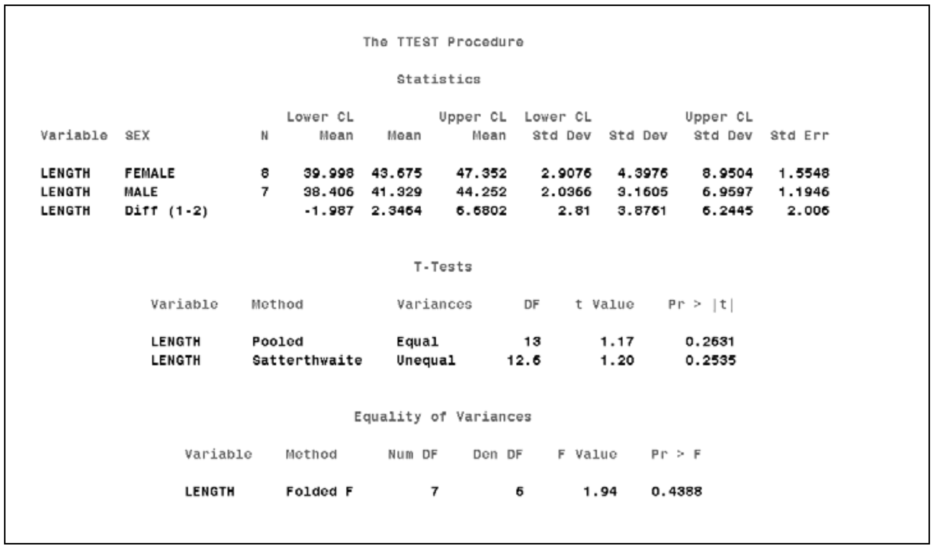
Submit the above program for execution.

The output is shown in Box 1-5. Those familiar with the T-test should note that the bottom line is a two-tailed F-test of homogeneity of variances. In this case, it fails to provide evidence of unequal population variances (probability of obtaining the observed F, or one greater, by chance alone is 0.4388).

Details of two T-tests are given, one for when the population variances can be considered equal (student's t = 1.17 with 13 df, non-significant) and one for when the population variances cannot be considered equal (Satterthwaite's t = 1.20 with 12.6 df, non-significant). In this case, on the basis of the F-test, we would choose student's t.

Performing paired T-tests in SAS is not straightforward, and given that the turtle data set does not provide the opportunity to perform a paired T-test, it is left for Chapter 3: T-tests and Related Comparisons.

Box 1-5. Comparison of mean carapace lengths of male and female pig-nosed turtles from Kakadu National Park. Output is produced using PROC TTEST.



```
                                 The TTEST Procedure

                                      Statistics

                                   Lower CL          Upper CL  Lower CL          Upper CL
Variable  SEX                 N        Mean    Mean      Mean   Std Dev  Std Dev   Std Dev  Std Err

LENGTH    FEMALE              8      39.998  43.675    47.352    2.9076   4.3976    8.9504   1.5548
LENGTH    MALE                7      38.406  41.329    44.252    2.0366   3.1605    6.9597   1.1946
LENGTH    Diff (1-2)                 -1.987  2.3464    6.6802      2.81   3.8761    6.2445    2.006

                                        T-Tests

              Variable    Method           Variances      DF    t Value    Pr > |t|

              LENGTH      Pooled           Equal          13       1.17      0.2631
              LENGTH      Satterthwaite    Unequal      12.6       1.20      0.2535

                                 Equality of Variances

                  Variable    Method      Num DF    Den DF    F Value    Pr > F

                  LENGTH      Folded F         7         6       1.94    0.4388
```

Scatterplots

Moving on to the bi-variate procedures, scatterplots are an important prelude to both correlation and regression analyses. Before performing a linear regression or correlation analysis, it is important to be sure that the relationship between the two variables under consideration is roughly linear. Consider the relationship between LENGTH and WEIGHT:

```
GOPTIONS RESET=ALL;
SYMBOL1 C=RED V=DOT I=NONE;
AXIS1 LENGTH=50 PCT ORDER=20.0 TO 50.0 BY 10.0;
AXIS2 LENGTH=50 PCT ORDER=0.0 TO 15.0 BY 5.0;
PROC GPLOT DATA=MYDATA;
     PLOT WEIGHT*LENGTH /
           HAXIS=AXIS1
           VAXIS=AXIS2;
RUN;
```

Note

The PLOT statement extends over three lines, so there is no semi-colon at the end of the first and second lines.

This step will produce the plot shown in Figure 1-8, with WEIGHT on the vertical axis and LENGTH on the horizontal axis.

Submit the above program for execution.

As expected, the relationship is not linear, weight being more of a function of body volume than of body shell length. It is for this reason that the transformations of length and weight were included in the initial DATA step.

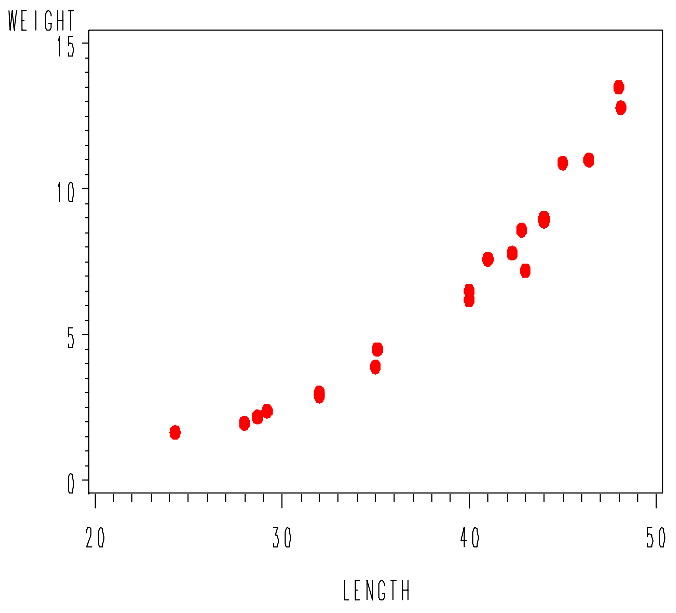


Figure 1-8. Relationship between body weight and carapace length for the pig-nosed turtle from Kakadu National Park. Length is in cm and weight is in kg. The scatterplot was produced with PROC GPLOT.

The extent to which the transformations linearise the relationship between body weight and length can be judged from the following analysis:

```
GOPTIONS RESET=ALL;
SYMBOL1 C=RED V=DOT I=NONE;
AXIS1 LENGTH=50 PCT ORDER=1.3 TO 1.7 BY 0.2;
AXIS2 LENGTH=50 PCT ORDER=0.0 TO 1.2 BY 0.4;
PROC GPLOT DATA=MYDATA;
    PLOT LGWEIGHT*LGLENGTH /
          HAXIS=AXIS1
          VAXIS=AXIS2;
RUN;
```

Submit the above program for execution.

The output is shown in Figure 1-9. With the possible exception of the left-most point, the relationship between logged shell length and logged body weight appears linear.

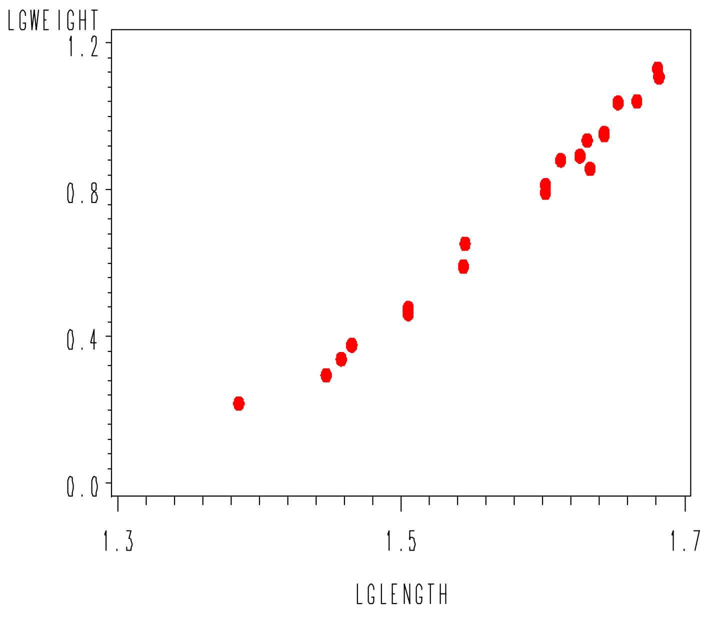


Figure 1-9. Log-linear relationship between body weight and carapace length for the pig-nosed turtle from Kakadu National Park. Length is in cm and weight is in kg. Both variables have been transformed by logs to base 10. The scatterplot was produced with PROC GPLOT.

Correlations

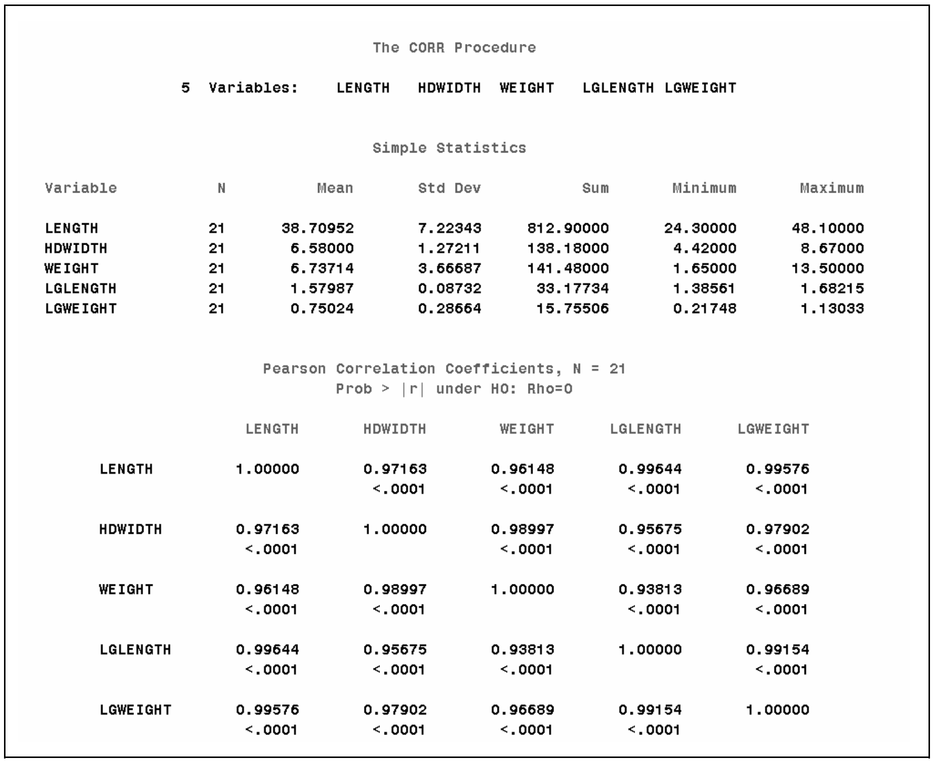
The next step might be to calculate a correlation matrix for all the measurement variables, in particular the transformed variables:

```
PROC CORR DATA=MYDATA;
   VAR LENGTH HDWIDTH WEIGHT
   LGLENGTH LGWEIGHT;
RUN;
```

Submit the above program for execution.

This step yields the output shown in Box 1-6.

Box 1-6. A correlation matrix for measurements taken from the pig-nosed turtle in Kakadu National Park. The output was produced with PROC CORR.



```
                              The CORR Procedure

           5  Variables:    LENGTH   HDWIDTH  WEIGHT   LGLENGTH LGWEIGHT


                              Simple Statistics

Variable           N        Mean     Std Dev          Sum      Minimum      Maximum

LENGTH            21    38.70952     7.22343    812.90000     24.30000     48.10000
HDWIDTH           21     6.58000     1.27211    138.18000      4.42000      8.67000
WEIGHT            21     6.73714     3.66687    141.48000      1.65000     13.50000
LGLENGTH          21     1.57987     0.08732     33.17734      1.38561      1.68215
LGWEIGHT          21     0.75024     0.28664     15.75506      0.21748      1.13033


                   Pearson Correlation Coefficients, N = 21
                           Prob > |r| under H0: Rho=0

                 LENGTH       HDWIDTH        WEIGHT      LGLENGTH      LGWEIGHT

   LENGTH       1.00000       0.97163       0.96148       0.99644       0.99576
                               <.0001        <.0001        <.0001        <.0001

   HDWIDTH      0.97163       1.00000       0.98997       0.95675       0.97902
                 <.0001                      <.0001        <.0001        <.0001

   WEIGHT       0.96148       0.98997       1.00000       0.93813       0.96689
                 <.0001        <.0001                      <.0001        <.0001

   LGLENGTH     0.99644       0.95675       0.93813       1.00000       0.99154
                 <.0001        <.0001        <.0001                      <.0001

   LGWEIGHT     0.99576       0.97902       0.96689       0.99154       1.00000
                 <.0001        <.0001        <.0001        <.0001
```

SAS provides some basic statistics on all variables listed in the VAR statement and a Pearson correlation matrix for all pair-wise combinations of those variables. The figure below each correlation coefficient is the probability of obtaining the observed coefficient, or one further from zero, by chance alone. Clearly the coefficients are all significant, as we would expect for measurements of objects with a well defined shape (the turtles).

Simple linear regression

As Aboriginal residents of Kakadu National Park regularly eat turtles, one can often obtain shells of the species that are the remains of a meal. In order to estimate the weight of a turtle from its shell length, a predictive regression of weight on shell length is required.

Because of curvilinearity in the relationship between the two untransformed variables, a linear regression of LGWEIGHT on LGLENGTH is appropriate:

```
PROC REG DATA=MYDATA;
 MODEL LGWEIGHT=LGLENGTH;
RUN;
```

Submit the above program for execution.

The output is shown in Box 1-7.

Box 1-7. Regression of log weight against log carapace length for the pig-nosed turtle from Kakadu National Park. Output was produced using PROC REG.

```
                             The REG Procedure
                               Model: MODEL1
                       Dependent Variable: LGWEIGHT

                           Analysis of Variance

                                     Sum of           Mean
Source                   DF         Squares         Square    F Value    Pr > F

Model                     1         1.61560        1.61560    1108.78    <.0001
Error                    19         0.02768        0.00146
Corrected Total          20         1.64328


             Root MSE              0.03817    R-Square     0.9832
             Dependent Mean        0.75024    Adj R-Sq     0.9823
             Coeff Var             5.08796


                           Parameter Estimates

                         Parameter       Standard
     Variable      DF     Estimate          Error    t Value    Pr > |t|

     Intercept      1     -4.39213        0.15466     -28.40      <.0001
     LGLENGTH       1      3.25492        0.09775      33.30      <.0001
```

For simple linear regression, the last two lines of the output suffice for a quick interpretation of the analysis. The predictive relationship is

```
LGWEIGHT = 3.2549*LGLENGTH – 4.39
```

The regression coefficient (slope) is significant ($t = 33.298$, $p < 0.0001$).

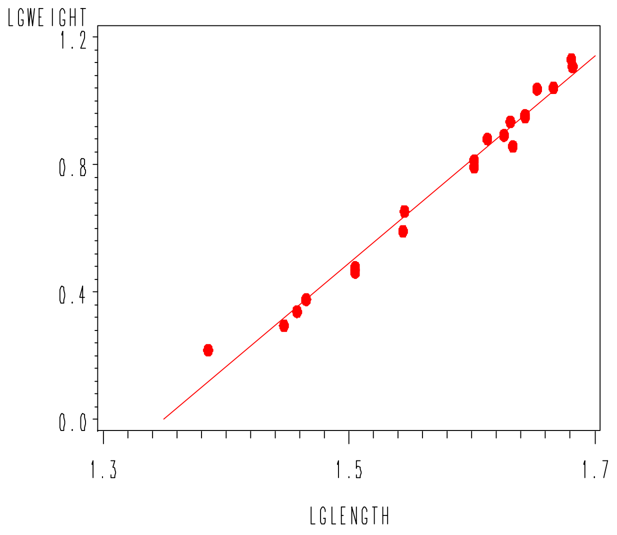
A graph of the relationship can be produced using I=RL as an option on the SYMBOL statement (I for Interpolate, RL for Regression Linear)(Figure 1-7).

```
GOPTIONS RESET=ALL;
SYMBOL1 C=RED V=DOT I=RL;
AXIS1 LENGTH=50 PCT ORDER=1.3 TO 1.7 BY 0.2;
AXIS2 LENGTH=50 PCT ORDER=0.0 TO 1.2 BY 0.4;
PROC GPLOT DATA=MYDATA;
 PLOT LGWEIGHT*LGLENGTH /
       HAXIS=AXIS1
       VAXIS=AXIS2;
RUN;
```

Submit the above program for execution.

This step yields the output shown in Figure 1-10.

Figure 1-10. Log-linear relationship between body weight and carapace length for the pig-nosed turtle from Kakadu National Park. Both variables have been transformed by logs to base 10. The scatterplot and least-squares line was produced with PROC GPLOT with the RL option.



Finishing up

The preceding analyses of the turtle data set have exposed you to SAS procedures for some commonly used basic statistical techniques. Before you instruct the computer to exit from SAS, you may wish to produce a printout of your program.

Tidy up the program listing in the EDITOR window by ensuring there are no elements remaining of the program that did not work.

Print the contents, and then save the program to disk.

Before you leave SAS, it's worthwhile to try the online help facility. Detailed help is available on a wide range of SAS options. Try obtaining help on topics that strike your interest.

Click on the HELP icon and peruse the help files.

The basic introduction to SAS is now complete, so exit from the SAS environment:

Exit from SAS by choosing File_Exit from the Menu Bar.

Lesson 4: More on SAS Operations

Now that you have completed the step-by-step demonstration of the operation of SAS, it is appropriate to learn something more of how SAS processes instructions and handles data.

The material that follows is meant to be read only. It is not hands-on.

Sources of data

Data in a separate file

A common approach to managing data is to keep it on disk in a file separate from the program that analyses it. The advantage of this is that the data can be accessed readily by a variety of programs other than SAS — word processors, spreadsheets, databases, editors and the like.

Suppose that the following data in Table 1-3 are stored as the file CAFISH.DAT in C:\MY DOCUMENTS\.

Table 1-3 Data from the recreational fish catch in South California counties during 1998. Source: Marine Recreational Fisheries Statistics Survey (MRFSS) database.

SPECIES	COUNTY	MODE	COUNT	TRIPS	WEIGHT
ALBACORE	SD	6	11817	57812	60978.5
ALBACORE	SD	6	20663	59151	133409.0
ALBACORE	VT	6	14114	5709	137985.3
ALBACORE	LA	7	3846	61860	.
ALBACORE	SD	7	5598	63012	29834.8
ALBACORE	SD	7	12368	75473	81842.9
BANK_ROCKFISH	LA	6	147	30183	95.2
BANK_ROCKFISH	SD	6	147	53626	75.1
BANK_ROCKFISH	SD	6	783	15312	432.3
BANK_ROCKFISH	SB	6	951	3647	302.9
BANK_ROCKFISH	VT	6	5191	22388	1774.8
BANK_ROCKFISH	VT	6	293	10549	79.1
BANK_ROCKFISH	SD	7	3706	67402	.
BARRED_BASS	LA	2	0	12452	0.0

The data are from the recreational fish catch in South California counties during 1998. The data are lined up to form columns of values (referred to as variables). The first variable contains the name of the species caught. The second variable contains a code for the county in which it was caught, the third variable contains a code for the mode of capture, the fourth contains the count of fish caught, the fifth variable contains the number of trips recorded as associated with the catch, and the sixth variable contains the estimated weight of the fish caught (kg).

A program to read data stored in the disk file CCAFISH.DAT might look like this:

```
DATA FISH;
 INFILE "C:\MY DOCUMENTS\CAFISH.DAT";
  LENGTH SPECIES $25;
  INPUT SPECIES $ COUNTY $ MODE COUNT TRIPS WT;
   F_PER_T = COUNT/TRIPS;
RUN;
```

The LENGTH statement specifies that the variable SPECIES contains text that is greater in length than the default 8 characters. The fourth line of the above code creates a new variable, F_PER_T, to be calculated as the count of fish per trip, a useful statistic.

Data embedded in your program

If you prefer, the raw data can be included within the SAS program using the DATALINES statement as follows:

```
DATA FISH;
 INFILE "C:\MY DOCUMENTS\CAFISH.DAT";
  LENGTH SPECIES $25;
  INPUT SPECIES $ COUNTY $ MODE COUNT TRIPS WT;
   F_PER_T = COUNT/TRIPS;
DATALINES;
ALBACORE            LA   6    1576    49477   15284.4
ALBACORE            SD   6    11817   57812   60978.5
ALBACORE            SD   6    20663   59151   133409.0
ALBACORE            VT   6    14114   5709    137985.3
ALBACORE            LA   7    3846    61860   0.0
ALBACORE            SD   7    5598    63012   29834.8
ALBACORE            SD   7    12368   75473   81842.9
BANK_ROCKFISH       LA   6    147     30183   95.2
BANK_ROCKFISH       SD   6    147     53626   75.1
BANK_ROCKFISH       SD   6    783     15312   432.3
BANK_ROCKFISH       SB   6    951     3647    302.9
BANK_ROCKFISH       VT   6    5191    22388   1774.8
BANK_ROCKFISH       VT   6    293     10549   79.1
BANK_ROCKFISH       SD   7    3706    67402   0.0
BARRED_SANDBASS     LA   2    0       12452   0.0
;
```

A semi-colon must occupy the final line of the DATA step.

The data is still stored by SAS in a temporary workfile called WORK.FISH.

Reading data from Excel spreadsheets

Many researchers use Microsoft Excel as a data management tool, because of the great convenience of working with data in spreadsheets. Reading data directly into SAS from and Excel spreadsheet is therefore highly desirable.

There are many ways of doing this, but using the dynamic data exchange facility of Windows is one of the most satisfactory.

```
FILENAME XLFILE DDE 'EXCEL|C:\My
 Documents\[CAFISH.XLS]SHEET1!R1C1:R15C6';
DATA FISH;
 INFILE XLFILE;
 LENGTH SPECIES $25;
 INPUT SPECIES $ COUNTY $ MODE COUNT TRIPS WT;
RUN;
```

Note that you need to specify the data range in RowColumn format, that is, R1C1:R15C6 specifies that the data lie in the block defined by Row 1 Column 1 to Row 15 Column 6.

Note also that the Excel application and Excel spreadsheet must be open at the time the SAS code is executed.

This creates a SAS workfile called WORK.FISH that can be used in subsequent analysis.

The big advantage of this approach is that as you make changes to the data in the Excel spreadsheet, the changes are automatically incorporated into the SAS analysis, **provided of course that you re-run the DATA step**.

This brings to you the power and ease of use of Excel for manipulating your data and undertaking exploratory analyses before drawing upon the grunt of SAS.

Arranging your data

There are three ways in SAS of specifying how the data are to be read in the DATA step.

List directed input

List directed input is the simplest form and has been introduced in the example above and the practical exercises. With list directed input, you simply list the names of the variables in the order that they appear on each data line in the data file (or datablock if you prefer the DATALINES option). For example:

```
INPUT SPECIES $ COUNTY $ MODE COUNT TRIPS WT;
```

Note that names of variables that are expected to hold character data (such as W or R) must be followed with a $ sign. With the single $ sign, you are restricted to character strings of eight or fewer characters (unless modified with a LENGTH statement) and must have no embedded blanks. This may be too restrictive, and there are several options for handling strings of more than eight characters. Consider the following lines of data:

```
ALBACORE           LA   6    1576    49477   15284.4
ALBACORE           SD   6    11817   57812   60978.5
ALBACORE           SD   6    20663   59151   133409.0
ALBACORE           VT   6    14114   5709    137985.3
ALBACORE           LA   7    3846    61860   0.0
ALBACORE           SD   7    5598    63012   29834.8
ALBACORE           SD   7    12368   75473   81842.9
BANK_ROCKFISH      LA   6    147     30183   95.2
BANK_ROCKFISH      SD   6    147     53626   75.1
BANK_ROCKFISH      SD   6    783     15312   432.3
BANK_ROCKFISH      SB   6    951     3647    302.9
BANK_ROCKFISH      VT   6    5191    22388   1774.8
BANK_ROCKFISH      VT   6    293     10549   79.1
BANK_ROCKFISH      SD   7    3706    67402   0.0
BARRED SANDBASS    LA   2    0       12452   0.0
```

The input statement

```
INPUT SPECIES $ COUNTY $ MODE COUNT TRIPS WT;
```

will read the truncated value of BANK_ROC (eight characters) into variable SPECIES. Even worse, when SAS comes to the data for BARRED SANDBASS (embedded blank), it will read BARRED into the variable SPECIES, SANDBASS into the variable COUNTY and the program will "fall over" when it tries to read LA into a numeric variable MODE (character values are unacceptable in a numeric field).

To overcome this you can specify the number of characters to read into a character variable as follows:

```
INPUT SPECIES $16. COUNTY $ MODE COUNT TRIPS WT;
```

This method will also cope with embedded blanks (eg BARRED SANDBASS), but it requires that each value in the field SPECIES has a constant width, in this case 16 characters.

With list directed input, a period must be used to indicate missing data so that the sequence of values is maintained. For example:

```
ALBACORE          LA   6    1576    49477   15284.4
ALBACORE          SD   6    .       57812   60978.5
ALBACORE          SD   6    20663   59151   133409.0
ALBACORE          VT   6    14114   .        137985.3
ALBACORE          LA   7    3846    61860   0.0
```

It is possible to arrange data such that there is more than one observation per line as follows:

```
ALBACORE  LA  6   1576  49477   15284.4  BANK_ROCKFISH  LA  6   147  30183    95.2
ALBACORE  SD  6      .  57812   60978.5  BANK_ROCKFISH  SD  6   147  53626    75.1
ALBACORE  SD  6  20663  59151  133409.0  BANK_ROCKFISH  SD  6   783  15312   432.3
ALBACORE  VT  6  14114      .  137985.3  BANK_ROCKFISH  SB  6   951   3647   302.9
ALBACORE  LA  7   3846  61860       0.0  BANK_ROCKFISH  VT  6  5191  22388  1774.8
```

The data must then be read with the @@ option. The @@ signifies to SAS that the variables are to be read repeatedly until no more data remain on the line. For example:

```
INPUT SPECIES $16. COUNTY $ MODE COUNT TRIPS WT @@;
```

Column directed input

Column directed input is a second method for specifying how data are to be read. This method requires that you follow each variable name with the range of columns its values occupy in the data file. For example:

```
INPUT SPECIES $ 1-16 COUNTY $ 18-19 MODE 22-23
      COUNT 25-29 TRIPS 31-35 WT 37-45;
```

The advantages of column directed input are that you can easily skip unwanted variables, and blank fields for numeric variables are considered to be missing values.

Format directed input

Format directed input provides greater flexibility for data input, but the syntax is beyond this introductory guide. If the greater sophistication is required, refer to the SAS Language Guides for details.

Adding comments to your programs

Comment lines can be added by beginning the text with an asterisk and ending the comment with a semi-colon. For example:

```
* CALCULATE SUMMARY STATISTICS;
```

Alternatively, large blocks of text or code can be converted to comment by preceding it with /* and terminating it with */. This is useful for temporarily removing segments of your program from execution, while debugging the remainder. For example,

```
DATA FISH;
 INFILE "C:\MY DOCUMENTS\CAFISH.DAT";
 LENGTH SPECIES $25;
 INPUT SPECIES $ COUNTY $ MODE COUNT TRIPS WT;
 F_PER_T = COUNT/TRIPS;
RUN;
/*
PROC MEANS DATA=FISH;
 VAR WT;
RUN;
*/
PROC CORR DATA=FISH;
 VAR WT F_PER_T;
RUN;
```

will skip the PROC MEANS step, treating it as a comment.

Adding a descriptive title

A descriptive title can be added to SAS output using the TITLE statement. For example, the statement

```
TITLE "Analysis of Recreational Fish Catch Data";
```

will result in all subsequent pages of output bearing that title. The title will persist until another TITLE statement is given or until terminated by:

```
TITLE;
```

TITLE statements can appear anywhere in a SAS program.

Labelling variables

With a limitation of eight characters for variable names, it is often necessary to use rather cryptic names only to discover difficulties when perusing the results of an analysis some months later.

Does the variable name LGWT imply that weight was log-transformed to base e or to base 10?

Was the variable CL measured in mm or cm, and what does CL stand for anyway?

The prudent SAS programmer will use LABEL statements in the DATA step to enhance readability and understanding of the results of subsequent analyses. For example, consider the following code:

```
DATA FISH;
 INFILE "C:\MY DOCUMENTS\CAFISH.DAT";
 LENGTH SPECIES $25;
 INPUT SPECIES $ COUNTY $ MODE COUNT TRIPS WT;
 F_PER_T = COUNT/TRIPS;
 LABEL COUNT = "Total Fish Count"
       TRIPS = "Reported number of fishing trips"
       WT = "Live Body Wt (kg)"
       F_PER_T = "Fish per trip";
RUN;
```

Each time statistics are produced for one of the labelled variables in a PROC step, the descriptive label will accompany the variable name.

Labels can be up to 40 characters long, with blanks counting as characters.

Note that several variables can be given labels in a single LABEL statement and that labels can be assigned to variables created in an assignment statement (eg F_PER_T).

Labelling values of a variable

The confusion that may arise with eight character variable names may also arise through abbreviations used for values of a variable. In the example at hand, you have used LA, SD, VT for the counties. In another study of an animal species, you may choose to use SEX = 0 for missing data, 1 for unsexed juveniles, 2 for juvenile males, 3 for juvenile females, 4 for mature males and 5 for mature females, but will you remember these codes when perusing output at a later date? If not, then descriptive labels can be assigned to the variables COUNTY and SEX using the FORMAT procedure. For example, consider the following code:

```
PROC FORMAT;
 VALUE $COUNTY
       "LA" = "LOS ANGELES"
       "SB" = "SANTA BARBARA"
       "OR" = "ORANGE"
       "SD" = "SAN DIEGO"
       "VT" = "VENTURA";
 VALUE MODE
       1     = "PIER"
       2     = "SHORE"
       3     = "CHARTER BOAT"
       4     = "PRIVATE BOAT";
RUN;
```

This procedure has produced two sets of value labels, called formats in the SAS manuals. They can subsequently be referred to as:

$COUNTY. and MODE.

Note

The period following a SAS format is mandatory. The $ preceding the format containing county names indicates that the values being assigned labels are characters.

To use the formats for enhancing output, a FORMAT statement can be included in procedures. For example:

```
PROC MEANS DATA=FISH;
 VAR WT;
 BY COUNTY MODE;
 FORMAT COUNTY $COUNTY. MODE MODE.;
RUN;
```

Alternatively, labels can be assigned to values of specific variables in the DATA step for use in all subsequent procedures:

```
PROC FORMAT;
 VALUE $COUNTY
       "LA" = "LOS ANGELES"
       "SB" = "SANTA BARBARA"
       "OR" = "ORANGE"
       "SD" = "SAN DIEGO"
       "VT" = "VENTURA";
 VALUE MODE
       1     = "PIER"
       2     = "SHORE"
       3     = "CHARTER BOAT"
       4     = "PRIVATE BOAT";
RUN;
DATA FISH;
 INFILE "C:\MY DOCUMENTS\CAFISH.DAT";
 LENGTH SPECIES $25;
 INPUT SPECIES $ COUNTY $ MODE COUNT TRIPS WT;
 F_PER_T = COUNT/TRIPS;
 FORMAT COUNTY $COUNTY. MODE MODE.;
RUN;
```

The FORMAT line is then unnecessary in any subsequent PROC steps.

Permanent SAS data sets

What is a SAS data set?

So far the only SAS data sets you have encountered have been temporary, such as WORK.FISH, where SAS stores data temporarily in preparation for analysis. Permanent SAS data sets are useful if you wish to repeatedly access the one data set for analyses of various kinds in sessions spread over days, weeks, months or years.

Permanent SAS data sets contain not only the data, but all variable labels, derived variables, transformed or recoded variables, value labels, etc, so that these do not have to be specified when the data are accessed in future.

The first part of the name of a SAS data set is a variable that contains the disk and sub-directory which contains the file, the second part is the DOS filename. Hence, the SAS data set FLOPPY.FISH would refer to the file FISH.sas7bdat on the disk and path stored in the variable FLOPPY.
SAS data set files have the extension SSD or similar.

Creating a SAS data set

To create a permanent SAS data set, you need to first define a 'path' telling SAS where to store the data set on disk. This is done with the LIBNAME statement.

```
LIBNAME FLOPPY "C:\MY DOCUMENTS\";
```

or

```
LIBNAME HARDISK "C:\DATA\";
```

The path name can be any name up to eight characters, so you are not restricted to FLOPPY or HARDISK. Any eight character name will do, but avoid using the word WORK.

The next step is to give the data set a double name in the first line of the DATA step. The first part of the name is the LIBNAME and the second part is a label up to eight characters that enables you to identify the data set. For example:

```
LIBNAME FLOPPY "C:\MY DOCUMENTS\";
DATA FLOPPY.FISH;
 INFILE "C:\MY DOCUMENTS\CAFISH.DAT";
 LENGTH SPECIES $25;
 INPUT SPECIES $ COUNTY $ MODE COUNT TRIPS WT;
 F_PER_T = COUNT/TRIPS;
RUN;
```

The program will create a permanent data set called FISH.sas7bdat in directory C:\MY DOCUMENTS\ containing the raw data and their variable names and labels and the new variable F_PER_T.

Using a SAS data set

Reading in and analysing the data at a later date is now a simple process. No DATA step is required.

```
LIBNAME FLOPPY "C:\MY DOCUMENTS\";
PROC MEANS DATA=FLOPPY.FISH;
 VAR WT;
RUN;
```

Should you forget details of the contents of a SAS data set, you can view its contents using the EXPLORER window.

SAS is capable of very sophisticated manipulations of data in the DATA step prior to analysis. You have had some exposure to this already in the form of assignment statements using both standard arithmetic operations and specialised functions. It is also possible to selectively delete unwanted records or variables to create a subset of the master data set, to selectively apply assignment statements in order to recode data, to sort data, to join or merge data sets and to do a suite of complex data manipulations that would be expected of any high level programming language.

The following examples demonstrate some of these capabilities, and can be equally applied to permanent SAS data sets, to the temporary SAS workfile, or at the time of reading the raw data in the initial DATA step.

Copying SAS data sets

The SAS data set FLOPPY.FISH can be copied into a new SAS data set called FLOPPY.CALIFORNIA using the following code:

```
LIBNAME FLOPPY "C:\MY DOCUMENTS\";
DATA FLOPPY.CALIFORNIA;
 SET FLOPPY.FISH;
RUN;
```

Here the SET statement means essentially "read in the data from". A similar program can be used to convert the SAS workfile WORK.FISH into a permanent data set:

```
LIBNAME FLOPPY "C:\MY DOCUMENTS\";
DATA FLOPPY.CALIFORNIA;
 SET FISH;
RUN;
```

It is also possible to copy a SAS dataset onto itself, adding modifications on the way. This is quite useful if you do not wish to go back to the initial DATA step to make the modifications.

```
LIBNAME FLOPPY "C:\MY DOCUMENTS\";
DATA FLOPPY.FISH;
 SET FLOPPY.FISH;
 LOGWT = LOG10(WT + 1);
RUN;
```

The above code adds a new variable to the SAS dataset FLOPPY.FISH.

Selectively deleting variables

It is unlikely that you will want to copy files with no modification. A more common requirement is to create a subset of the original SAS data set. For example, the master data set FLOPPY.FISH may contain the variables SPECIES, COUNTY, MODE, WT, COUNT, TRIPS and F_PER_T, but once the variable F_PER_T has been calculated, subsequent analyses may not require the variables COUNT and TRIP. The unwanted variables may be discarded with the DROP statement or the required variables may be retained with the KEEP statement:

```
DATA FLOPPY.CALIFORNIA;
 SET FLOPPY.FISH;
 DROP COUNT TRIP;
RUN;
```

or equivalently:

```
DATA FLOPPY.CALIFORNIA;
 SET FLOPPY.FISH;
 KEEP SPECIES COUNTY MODE WT F_PER_T;
RUN;
```

To permanently drop the unwanted variables from the master data set FLOPPY.FISH, that data set should be nominated in both the DATA statement and the SET statement:

```
DATA FLOPPY.FISH;
 SET FLOPPY.FISH;
 KEEP SPECIES COUNTY MODE WT F_PER_T;
RUN;
```

The DROP and KEEP statements can also be used in the original DATA step. Once you have created the new variable F_PER_T to represent relative catch, there may be no need to retain the original fish and trip counts and they can be discarded before data is output to the SAS workfile WORK.FISH.

```
DATA FISH;
 INFILE "C:\MY DOCUMENTS\CAFISH.DAT";
 INPUT SPECIES $ COUNTY $ MODE COUNT TRIPS WT;
 F_PER_T = COUNT*100/TRIPS;
 DROP COUNT TRIP;
RUN;
```

Selectively deleting observations

Some analyses may require data excluding some observations. The IF-THEN statement can be used to delete observations for Los Angeles (for example):

```
LIBNAME FLOPPY "C:\MY DOCUMENTS\";
DATA FLOPPY.CALIFORNIA;
 SET FLOPPY.FISH;
 IF (COUNTY = "LOS ANGELES") THEN DELETE;
RUN;
```

The new SAS data set FLOPPY.CALIFORNIA will contain data excluding Los Angeles.

The unwanted observations can be permanently removed from the master data set FLOPPY.FISH by nominating FLOPPY.FISH in both the DATA statement and the SET statement.

IF-THEN statements can also be used in the original DATA step to eliminate unwanted observations when reading in the raw data.

As an alternative to selectively deleting observations with the DELETE statement, one can selectively retain observations with the OUTPUT statement.

```
LIBNAME FLOPPY "C:\MY DOCUMENTS\";
DATA FLOPPY.CALIFORNIA;
 SET FLOPPY.FISH;
 IF (COUNTY="ORANGE" OR COUNTY="SAN DIEGO")
 THEN OUTPUT;
RUN;
```

The OUTPUT statement is used to direct SAS explicitly to add an observation to the SAS data set.

If you do not include an OUTPUT statement in a DATA step, SAS automatically writes the current observation to the new data set when it reaches the RUN statement or the end of the DATA step. When an OUTPUT statement does appear in the DATA step, SAS no longer automatically outputs the observation at the end of the DATA step. Instead, the observation is added to the data set only when the OUTPUT statement is executed.

Earlier, I outlined that a feature of SAS was that all data manipulations were to be undertaken in the DATA step, prior to analysis with a series of PROC steps. There is an exception to this. It is possible to place a condition on the observations analyzed by a PROC step by using a WHERE statement. For example,

```
PROC MEANS DATA=MYDATA;
 VAR LENGTH;
 WHERE COUNTY="LOS ANGELES" AND MODE=1;
RUN;
```

This procedure will calculate statistics only for fish caught from piers in Los Angeles. The WHERE statement is a useful addition to the SAS commands.

Recoding variables

Values of a variable can be selectively recoded using IF-THEN statements:

```
LIBNAME FLOPPY "C:\MY DOCUMENTS\";
DATA FLOPPY.FISH;
 SET FLOPPY.FISH;
 IF (WT < 1000)
       THEN SIZE = "SMALL";
 IF (WT >= 1000)
       THEN SIZE = "LARGE";
RUN;
```

This DATA step creates a new variable SIZE that contains either LARGE or SMALL and adds it to the master file FLOPPY.FISH.

Where have we come?

Having completed this module, you have the basic knowledge and skills to undertake simple statistical analyses in SAS. In particular, you will appreciate that:

- SAS is a programming language that uses two primary constructs –- the DATA step and the PROC step. A simple SAS program ypitically comprises one DATA step and several PROC steps.
- SAS has a windows interface including an EDITOR window for creating programs, OUTPUT and GRAPH windows for receiving results, a LOG window for monitoring progress of the analysis and identifying errors when they occur, an EXPLORER window for perusing SAS datasets, and a NAVIGATOR window for navigating among output of the various analyses.
- Data can be accessed from separate raw data file, from within the SAS program itself, or from and Excel spreadsheet. DATA steps do not modify the raw data, but rather make a copy for subsequent manipulation and analysis. DATA steps are loops that execute once for each line of data.
- Once the data are read into SAS, there are a very large number of PROC steps for analysing those data.
- There are many statements for improving the appearance of your output (for adding titles, descriptive labels to variables, descriptive labels to values of a variable), for copying, renaming and modifying SAS datasets (through creating new variables, deleting variables and observations, recoding variables) and for permanently storing this information with the data as a permanent SAS dataset.

It needs to be said that this module has introduced only a very small part of the capability of SAS, and has been designed to be a minimal introduction, to get you up to speed as quickly as possible.

Your capacity for using SAS will grow with use, and you need to keep a good notebook to record new procedures and to share these with your colleagues.

You should also refer to the SAS Applications Guide and in particular, the chapters on Recoding Variables (Chapter 2), Reshaping your Data (Chapter 3) and Merging Data Sets (Chapter 4).

References

Georges, A & Kennett, R (1989). Dry-season distribution and ecology of *Carettochelys insculpta* (Chelonia: Carettochelydidae) in Kakadu National Park, northern Australia. *Australian Wildlife Research* 16:323-335.