

# Introductory Guide to SAS for Windows



Department of Mathematics and Statistics  
University of Canterbury, New Zealand

*Written by Julian Visch and Carl Scarrott*

This introduction to the software **SAS** is in no way endorsed by **SAS**.

## **In the beginning...**

**SAS** originally stood for for “statistical analysis software”. However, since it’s release **SAS** has become an integrated system of software products and services for ‘Business Intelligence’. **SAS** is now the company name, so is no longer considered an acronym and as such doesn’t stand for anything.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	What is SAS and Why are we using it? . . . . .	1
1.2	Where can I access SAS? . . . . .	1
1.3	SAS Programming . . . . .	1
1.3.1	Starting and Exiting SAS . . . . .	2
1.3.2	How to Create, Edit and Save Your Own Programs . . . . .	2
1.3.3	Running a Program . . . . .	4
1.3.4	Displaying Output . . . . .	6
1.3.5	Outputting/Printing Results . . . . .	7
1.3.6	Loading Data into SAS . . . . .	7
1.4	Where to get help with SAS? . . . . .	7
1.4.1	How to use the SAS? Help . . . . .	8
1.4.2	Other Sources of SAS Help . . . . .	8
1.4.3	Common Mistakes When Using SAS . . . . .	8
<b>2</b>	<b>Data Processing</b>	<b>9</b>
2.0.4	Creating Subsets of Data Files . . . . .	9
2.1	Inputting Data . . . . .	10
2.1.1	Reading in Character Variables . . . . .	10
2.1.2	List Input . . . . .	10
2.1.3	Column Input . . . . .	11
2.1.4	Combining List and Column Inputs . . . . .	11
2.1.5	Data Files with Different Formats . . . . .	12
2.1.6	Impossible Data Files . . . . .	12
2.1.7	Creating Dependent Variables . . . . .	12
2.1.8	Use of ‘IF...THEN...ELSE’ Statements . . . . .	14
2.1.9	Deleting Observations . . . . .	15
2.2	Including Comments Within Your SAS Program . . . . .	15
2.3	Sorting your Data . . . . .	15

<b>3</b>	<b>SAS Procedures</b>	<b>17</b>
3.1	Basic Statistics . . . . .	17
3.1.1	Using the MEANS Procedure . . . . .	17
3.1.2	Calculating Correlations and Covariances . . . . .	18
3.1.3	Univariate Statistics . . . . .	19
3.2	SAS Graphics . . . . .	19
3.2.1	Low Level Graphics (Scatter Plots) . . . . .	19
3.2.2	High Level Graphics (Scatter Plots) . . . . .	20
3.2.3	Pie Charts . . . . .	22
3.2.4	Exporting graphs . . . . .	22

# Chapter 1

## Introduction

### 1.1 What is SAS and Why are we using it?

In this guide, **SAS** stands for a statistical software package that allows the users to manipulate and perform statistical analyses of data. This software package is used in many companies and is pretty much a standard in many industries. Users come from a wide range of disciplines (not just statistics!), including medical, biological, actuarial and social sciences. Knowing the **SAS** programming language will not only help you with this course and future research. Many key employers of graduates also desire experience with **SAS**.

The purpose of this introductory guide is to familiarize you with carrying out some basic tasks in **SAS**. Some familiarity with using Windows is assumed. These notes complement the hints and instructions provided with the assignments.

### 1.2 Where can I access SAS?

**SAS 9.1** is installed on available from all the PC's in the Mathematics and Statistics Department labs in the basement of the Erskine building. All the software facilities on these machines is available off campus (i.e. from the Halls of Residence or any PC with high speed internet access - broadband) using the Windows "Remote Desktop" software, see the "Computer Documentation" on the dept website for more details. **SAS** is also installed on all the PC's in the labs run by ICTS.

In the dept labs, **SAS** is on the Windows Desktop. From other machines it will often be found under Start > Programs menu.

### 1.3 SAS Programming

When programming in any package the crucial things you must learn are

- How to start the package.
- How to exit the package.
- How to create, edit and save programs for running within the package.

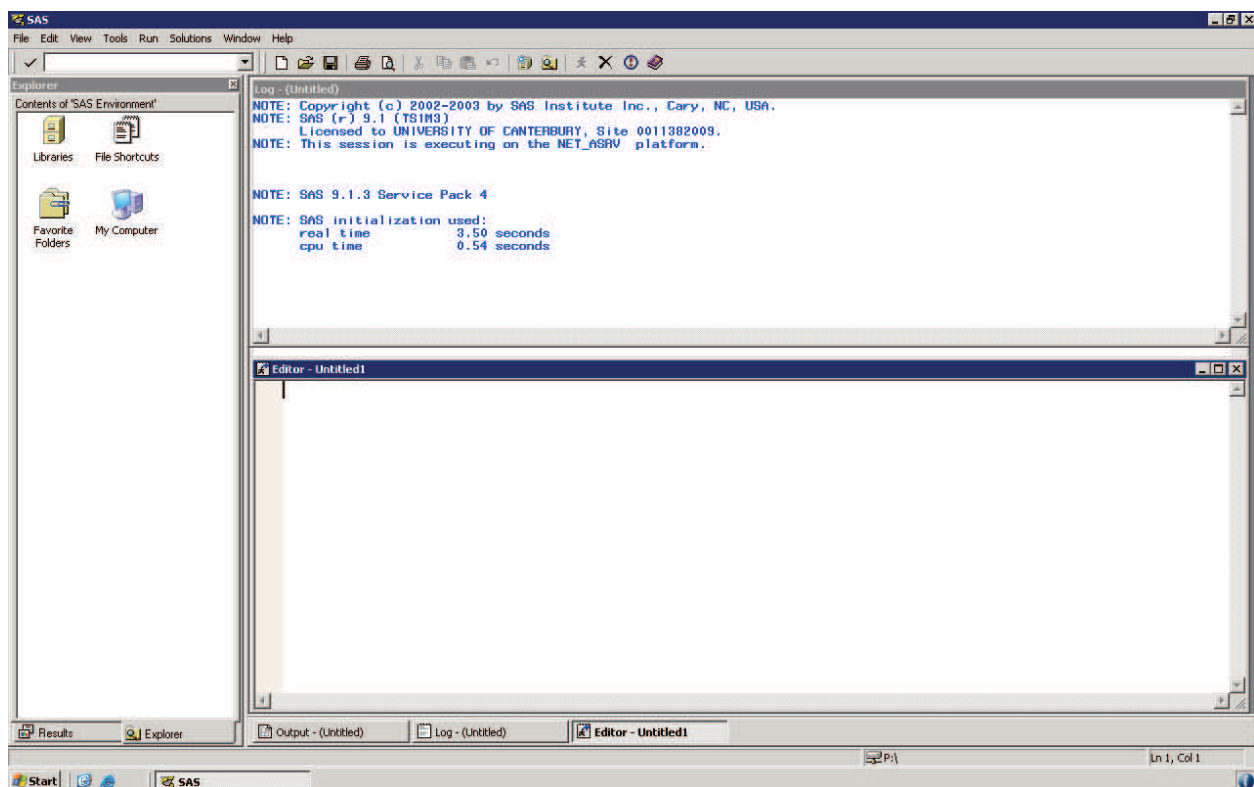
- How to run programs (e.g. do statistical analysis).
- How to display the output from the program.
- How to save/print the output to another file.
- How to load external data into the package.

### 1.3.1 Starting and Exiting SAS

When you first open SAS three windows will appear within the main SAS program window:

- **SAS: OUTPUT**  
This window displays all the output generated from your programs.
- **SAS: LOG**  
This window displays any error messages that occur because of bugs in your program.
- **SAS: EDITOR**  
This window is where you open, write/edit, submit (run) and save programs.

To quit SAS select Exit from the File menu.



### 1.3.2 How to Create, Edit and Save Your Own Programs

To create your own programs use SAS's "EDITOR". You can save files from the "EDITOR" using Save option from the File menu.

## Creating a Simple Program

All SAS programs are made up of one or more procedures, made up of a sequence of statements. The following is an example of a simple procedure:

---

DATA ageclub;	← create a dataset called <b>ageclub</b>
INPUT StudNo YearBorn;	← specifies variables ( <b>StudNo</b> and <b>YearBorn</b> ) within dataset
CARDS;	← specifies that the data will follow
123 88	
101 89	← the data
432 90	
RUN;	← specifies the end of the procedure.

---

### Notes:

1. Tabs are not the same as spaces in SAS. Make sure no tabs exist in your data, as otherwise you may encounter problems when you run your SAS programs.
2. SAS ignores the case of letters used. In this guide we tend to use capital letters for SAS statements and lower case for data/variable names.

Type the above into the **PROGRAM EDITOR** and save the program in a file with the name “example.sas”. Notice that SAS program files have the extension ‘.sas’, so SAS recognizes them.

## Rules for SAS Statements

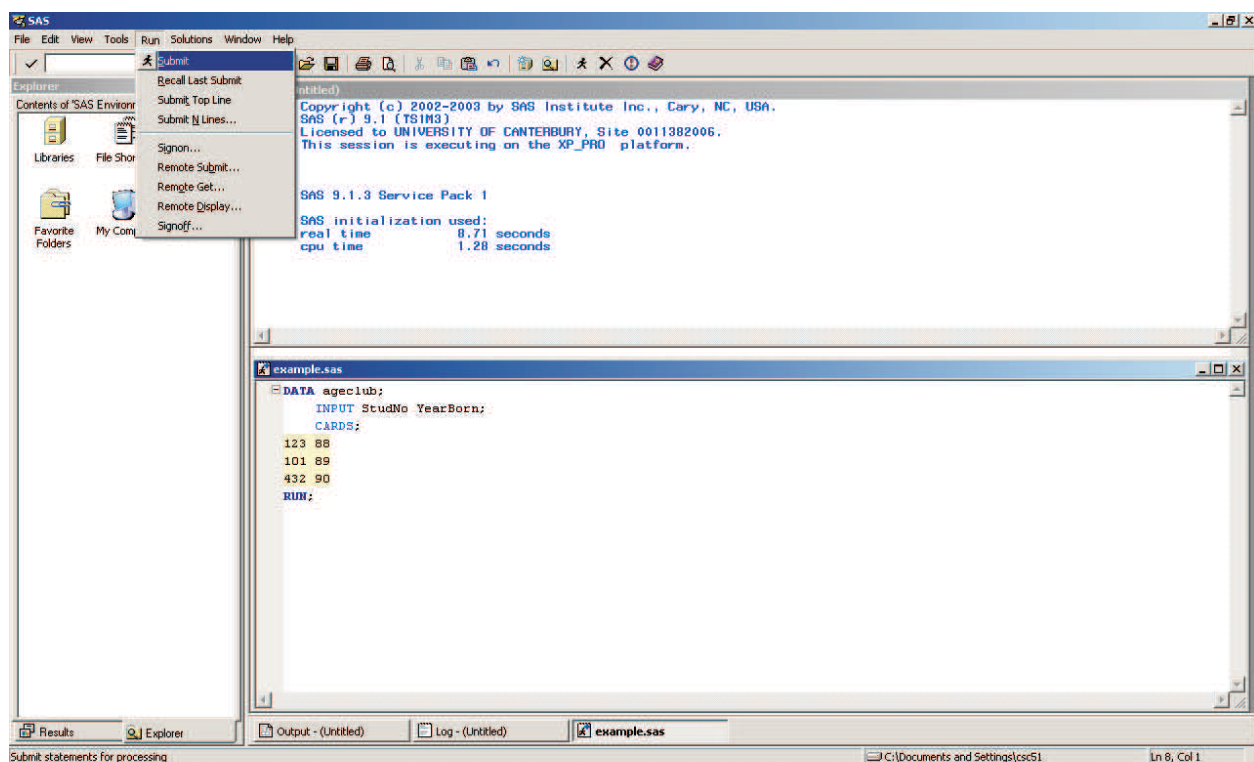
- SAS statements end with a semi colon (;).
- SAS statements can be entered in lowercase, uppercase, or a mixture.
- Any number of SAS statements can appear on a single line.
- A SAS statement can be continued over multiple lines, as long as no word is split.
- Words in SAS statements are separated by blanks or by special characters such as the equal sign.

## Rules for SAS Names

- A SAS name can contain from one to eight characters.
- The first character must be a letter or an underscore “\_”.
- Subsequent characters must be letters, numbers, or underscores.
- Blanks cannot appear in SAS names.

### 1.3.3 Running a Program

If the program file is not yet open, then use the Open option in the File menu. There are many ways to submit (run) a program. To run the whole program, in the EDITOR select ‘Submit’ from the ‘Run’ menu, or click on the “running man” icon. SAS will run your program.



Once you have submitted the program the LOG window will say something like:

```
1    DATA ageclub;  
2        INPUT StudNo YearBorn;  
3        CARDS;
```

NOTE: The data set WORK.AGECLUB has 3 observations  
and 2 variables.

NOTE: DATA statement used:

real time	0.69 seconds
cpu time	0.21 seconds

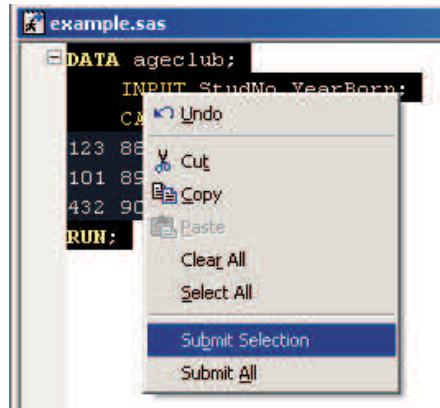
```
7    RUN;
```

The LOG window shows how SAS went step-by=step through your program adding appropriate comments as it goes through the steps. Should you have made an error in writing your program then some comment to that effect will also appear here.



## Shortcuts

When you only want to run a portion of a program, highlight with the mouse the portion of your program you wish to submit, right-click with mouse (over highlighted program) and select 'Submit Selection'. Try selecting all of `example.sas`, right click and 'Submit Selection'.



### 1.3.4 Displaying Output

Some procedures automatically output the main results. However, to see additional results you may need to add the PRINT procedure to your program. For example:

```
PROC PRINT DATA=ageclub; ← { Specifies which dataset to print. If you
                             omit the DATA=ageclub, the most recently
                             created SAS dataset is printed
Additional options inserted here
RUN;
```

Add this code to 'example.sas', then re-run your program (see Subsection 1.3.3). You should now get the following in the OUTPUT window:

The SAS System 09:38 Monday, July 10, 2007 1

OBS	STUDNO	YEARBORN
1	123	88
2	101	89
3	432	90

Date of submission \_\_\_\_\_

Page number \_\_\_\_\_

## Output (PRINT) Options

Try out each of the PRINT procedure's two options

- Specifying which variables to print  
`VAR YearBorn;`
- Specifying a title  
`TITLE 'Age of Club Members';`

**Note:** Once you insert a title it will also appear in subsequent outputs.

### 1.3.5 Outputting/Printing Results

You can print your output from any window, e.g. the OUTPUT window, by selecting 'Print' from the 'File' menu. Outputs, log and programs can be saved using 'File' followed by 'Save' option. Exporting graphs is covered in Section 3.2.4.

### 1.3.6 Loading Data into SAS

This guide outlines three options for reading datasets into SAS, but others exist. Suppose you have saved the 3 data items in a text file 'P:\ageclub.txt' (with no header line).

1. Use the **CARD** statement:  
See example in Section 1.3.3 above. Note: it is important that the last semi-colon goes on the line after all of the data, otherwise the last observation will be deleted! There is a similar **DATALINES** statement, see the SAS help for details.

2. Using the **INFILE** statement:  
Text files can be read into SAS using the **INFILE** statement. For example, a file in your P: drive:

```
DATA ageclub;  
    INFILE P:\ageclub.txt;  
    INPUT StudNo YearBorn;  
RUN;
```

3. Using the Import Data Wizard:  
To import the data, go to 'File' and 'Import Data'. A wizard window will ask about the format of the file to import. Click on the pull-down menu and select your file type (i.e. Excel, Lotus, text, etc.). Then, click **Next** which will take you to a window that asks where the file is saved, e.g., P:\ageclub.txt. Click **Next** again. You will then be asked for the name of your dataset. Click **Next** one more time, then click on the **Finish** button.

## 1.4 Where to get help with SAS?

The SAS help menu is helpful. There are two ways of getting SAS help. One is to go to the help menu and then SAS system help. Then go to the Index tab and type in the name of the procedure. SAS will give you the syntax of the procedure as well as some examples.

### 1.4.1 How to use the SAS? Help

The SAS help system is very useful if you want to self-learn about SAS procedures. Every procedure is listed, it's syntax (way to write program), including discussion of all statement options. At the end of each procedure, there is usual an example of implementation as well. For example, to find help on the 'SORT', you need to do the following:

- Select SAS Help and Documentation' from the 'Help' menu.
- Select 'Index'
- Type "Sort".
- Select the "Procedure under Windows".
- Use 'Previous' and 'Next' options to navigate the help.

The current SAS Help system is very similar to most Windows programs, with Contents, Index, Search and Favourites options. You can click on any underlined text and you will get a brief explanation of that topic, use the "Back" command to go back to the previous screen. If you have a specific question use the Search tab and type in the keyword.

### 1.4.2 Other Sources of SAS Help

There is a lot of information about using SAS on the web. If you ever want examples on how to use a procedure, e.g. the SORT procedure, then try a google search with terms like "PROC SORT" and SAS. In particular, the SAS website and those based at universities are generally the most useful.

### 1.4.3 Common Mistakes When Using SAS

To successfully run any program, you need to ensure you have:

1. a semi-colon at the end of every statement (except CARD lines used for data input);
2. an input (e.g. INFILE or CARDS) data statement that reads in your dataset (unless you used the Import Data wizard). **Always check the LOG window for error messages;**
3. a DATA statement which names the dataset inside of SAS. **Always use the PROC PRINT procedure to check the data has been read in correctly;**
4. an INPUT statement which names the variables in your dataset;
5. at least one space between each word or statement;
6. a RUN statement.

A semi-colon tells SAS that an operation, procedure, or statement is finished, so that it can go onto run the next. A RUN statement tells SAS to process the previous bit of the program you wrote. If there is no RUN statement, SAS will not process anything. Lack of semi-colons and RUN statements are the two most common mistakes in a SAS programs.

# Chapter 2

## Data Processing

### 2.0.4 Creating Subsets of Data Files

Some datasets are quite large you may wish to take subsets of them rather than loading the full data set using the set command. The set command has four possible options

- **FIRSTOBS=n** - Specifies the first observation to be read from the **SAS** data set you specify in the 'SET' statement. e.g. To create a subset of **ageclub** which contained only students after and including row 2 of the data:

```
DATA ageclub2;  
    SET ageclub(FIRSTOBS=2);  
RUN;
```

```
PROC PRINT DATA=ageclub2;  
RUN;
```

- **OBS=n** - Specifies the last observation to be read from the **SAS** dataset you specify in the 'SET' statement. e.g. To create a subset of **ageclub** which contained only students up to and including row 2 of the data:

```
DATA ageclub2;  
    SET ageclub(obs=2);  
RUN;
```

```
PROC PRINT DATA=ageclub2;  
RUN;
```

- **KEEP** - Specifies the variables to be included. e.g. To only include the variable **StudNo**:

```
DATA ageclub2;  
    SET ageclub;  
    KEEP StudNo;  
RUN;
```

```
PROC PRINT DATA=ageclub2;  
RUN;
```

- DROP - Specifies the variables to be excluded. e.g. To remove all the variables except StudNo:  

```
DATA ageclub2;
    SET ageclub;
    DROP YearBorn;
RUN;

PROC PRINT DATA=ageclub2;
RUN;
```

## 2.1 Inputting Data

### 2.1.1 Reading in Character Variables

There are two types of variables ‘numerical’ and ‘character’, where character is non-numeric variables such as names, colours, countries, etc. To inform SAS that you want your variable to be a character variable you must insert a \$ sign. For example, to include the student names in the dataset above modify the program:

```
DATA ageclub;
    INPUT StudNo YearBorn Student $;
    CARDS;
123 88 Bob
101 89 Belle
432 90 Tiye
RUN; PROC PRINT DATA=ageclub;
RUN;
```

When you run this program the students names also appear in the output.

### 2.1.2 List Input

List input uses the simplest form of the INPUT statement but places restrictions on data.

#### Rules for List Input Statements

- Fields (variables) must be separated by at least one blank.
- Each must be specified in order.
- Missing values must be represented by a place holder such as a period, because a blank field will cause the matching of variable names to get out of sync.
- Character values can’t contain embedded blanks. Instead the characters must all be joined.
- The default length of character variables is 8, any character variables exceeding 8 will be truncated.
- Data must be in standard character or numeric format.

### 2.1.3 Column Input

With column input, data values occupy the same fields within each record. When using column input, you list in the `INPUT` statement the variable names and identify the location of the corresponding data fields in the data lines by specifying the column positions. You can use column input when your raw data is in fixed columns and in standard character or numerical format.

#### Reading Embedded Blanks and Longer Variables

The advantage column input has over list input is that you can read embedded blanks and have longer variables. This is due to

- Column input uses the columns specified to determine the length of column character variables.
- Column input, unlike list input, reads data until it reaches the last specified column, not until it reaches a blank space.

#### Picking and Choosing Variables

Column input also allows fields to be skipped or to be read in any order. You can also cause only part of a value to be read or re-read, for instance you may wish to only have the first letter of the students name, then all you have to do is select the column with that first letter. This is especially important when the data is in the form shown below, because none of the fields are separated by spaces.

```
12388Bob
10189Belle
43290Tiye
```

**Note:** There has to be a blank spaces after ‘Bob’ and ‘Tiye’, so that they are all of the same length.

e.g. To select the data in reverse order, modify the input statement to:

```
INPUT Student $ 6-11 YearBorn 4-5 StudNo 1-3
```

### 2.1.4 Combining List and Column Inputs

Now that we have included the students names we may wish to exclude the student number, this we can do by modifying the input line to

```
INPUT YearBorn 5-6 Student $;
```

Where 5-6 refers to the columns where your first desired variable is. The omission of any further column references informs **SAS** that each subsequent variable in the input statement will correspond to the subsequent variable in the data file.

Suppose you just wanted the student number and their name but not their age then you would need to modify the input statement as follows

```
INPUT StudNo Student $ 8-13;
```

Where 8-13 refers to the columns where the student names are. The dollar sign can be on either side of the column reference.

**Note:** this assumes that all the data is properly set up in columns, because if for example you had the following

```
123 88 Bob 123 88
101 89 Belle 101 89
432 90 Tiye 432 90
```

You can see that the columns method would fail at 'Belle' because the "le" would fall in the wrong column. In this case, you would have to use the list input.

### 2.1.5 Data Files with Different Formats

For brevity, this guide does not show you every possible format which SAS can handle, but you will just need to explore as required. Refer to the SAS help, or the manuals which are available in the library.

### 2.1.6 Impossible Data Files

Occasionally you may get an data file such as

```
12388Bob12388
10189Belle10189
43290Tiye43290
```

This type of formatting is always going to be difficult to read into any package, and you will often have to edit the datafile first.

### 2.1.7 Creating Dependent Variables

Sometimes you'll need to create variables from existing ones. For example, you may wish to know the actual ages of the students according to the current year minus their date of birth. Firstly, we save the 3 datalines above into the file 'P:\ageclub.txt'. Then

```
DATA ageclub;
  INFILE 'P:\ageclub.txt';
  INPUT StudNo YearBorn Student $;
  StudAge=107-YearBorn;      ← 107 to represent 2007
RUN;
PROC PRINT DATA=ageclub;
RUN;
```

For reference, the basic math operators are:

- addition +
- subtraction -
- multiplication \*
- division /
- power \*\*

## Logic Statements

Suppose you wished to divide the club into those older than 8 and those below, you can do this via a logic statement. To create a new variable called 'old' add the following after where you had defined `StudAge`:

```
Old=StudAge>=9;
```

When you re-run your program you will obtain the following output.

OBS	STUDNO	YEARBORN	STUDENT	STUDAGE	OLD
1	123	88	Hobbes	9	1
2	101	89	Calvin	8	0
3	432	90	Susie	7	0

There are also the "&" and "|" logic statements which refer to "and" and "or" respectively. For example, if you define middle aged students to be between 7 and 9 then you could write it as follows.

```
MidAge=StudAge>7 & StudAge<9;
```

For reference here is the full set of relation operators

Symbol	Mnemonic Operator	Meaning
=	EQ	equal to
~=	NE	not equal to
>	GT	greater than
<	LT	less than
>=	GE	greater than or equal to
<=	LE	less than or equal to



## 2.1.8 Use of 'IF...THEN...ELSE' Statements

Suppose you wished to divide the ages further into the categories young, middle aged and old. This can be done via an 'IF...THEN...ELSE' statement:

```
DATA ageclub;
  INFILE 'P:\example.txt';
  INPUT StudNo YearBorn Student $;
  StudAge=107-YearBorn;
  IF StudAge>=19 THEN AgeCat='Senior';
  ELSE IF StudAge=18 THEN AgeCat='Middle Aged';
  ELSE AgeCat='Child';
RUN;
PROC PRINT DATA=ageclub;
RUN;
```

**Note:** The 'IF' does not require an 'END' statement, unlike many other programming languages.

Submitting the above program gives:

```
Age of Club Members          11
          09:08 Monday, July 10, 2007
```

OBS	STUDNO	YEARBORN	STUDENT	STUDAGE	AGECAT
1	123	88	Hobbes	9	Senior
2	101	89	Calvin	8	Middle
3	432	90	Susie	7	Child

Notice how AgeCat got truncated to the length of its first item. You can specify the length you require via a 'LENGTH' statement, e.g.:

```
StudAge=107-YearBorn;
LENGTH AgeCat $ 11;
IF StudAge>=19 THEN AgeCat='Old';
```

You also use 'AND' and 'OR' statements. For instance, the 'AND' statement works as follows

```
IF StudAge>17 & StudAge<19 THEN MidAged='Yes';
  ELSE MidAged='No ';
```

or similarly

```
IF 17<StudAge<19 THEN MidAged='Yes';
  ELSE MidAged='No ';
```

Similarly the 'OR' statement works as follows

```
IF StudAge<=17 | StudAge>=19 THEN MidAged='No ';
```

```
  ELSE MidAged='Yes';
```

### 2.1.9 Deleting Observations

Suppose we wished to delete from the data all students who were too old, you can do this via the DELETE command as follows.

```
IF AgeCat='Old' THEN DELETE;
```

## 2.2 Including Comments Within Your SAS Program

To include comments in your SAS programs all you need to do is insert a /\* at the start of your comment and a \*/ at the end of your comment. e.g.

```
/* This is a sample comment */
```

## 2.3 Sorting your Data

SAS's SORT procedure has the form:

```
PROC SORT DATA=ageclub OUT=sortclub;  
    BY YearBorn;  
RUN;  
PROC PRINT DATA=sortclub; /* Prints sorted data */  
RUN;
```

### Sorting by more than one Variable

You can sort by more than one variable by including the additional variables in the BY statement. For instance, if you wish to sort also by the variables StudNo and StudAge then you would modify as follows.

```
PROC SORT DATA=ageclub OUT=sortclub;  
    BY StudNo StudAge;  
RUN;  
PROC PRINT;  
RUN;
```

**Note:** As sortclub was the last data set created, "PROC PRINT", selects it as the default.

### Sorting in Descending Order

To sort a variable in descending order you only need to modify the program as follows

```
PROC SORT DATA=ageclub OUT=sortclub;  
    BY DESCENDING YearBorn;  
RUN;  
PROC PRINT;  
RUN;
```

## Removing Duplicates

The ‘NODUPPLICATES’ option in the sort procedure deletes any observation that duplicates the values of all variables of another observation in the dataset:

```
PROC SORT DATA=ageclub OUT=sortclub;  
    NODUPPLICATES;  
    BY DESCENDING YearBorn;  
RUN;  
PROC PRINT;  
RUN;
```

# Chapter 3

## SAS Procedures

SAS programs are composed of a sequence of (statistical) procedures. Some of the most commonly used procedures for statistical analysis are explained in detail below. Many SAS procedures are now accessible by the menu options. These are usually fine for one-off small analyses. However, for larger analyses (which may be repeated later), it is **good practice** to write a program to do the analysis, which can be saved in a `.sas` file for later re-use.

### 3.1 Basic Statistics

#### 3.1.1 Using the MEANS Procedure

Instead of having a series of procedures for calculating basic summary statistics (e.g. minimum, maximum) of numerical variables, they have all been combined into the **MEANS** procedure. It also calculates confidence limits for the mean and identifies extreme values. You can select which statistical information required from the available options, see the SAS help for the syntax and a list of options.

Here's an example for a mean and its confidence interval calculation:

```
PROC MEANS DATA=ageclub ALPHA=0.05 CLM MEAN MEDIAN N MIN MAX;  
RUN;
```

The mean, median, sample size, minimum, maximum, and 95% confidence intervals will be computed for variables age and weight. The **ALPHA** option specifies the confidence level for the confidence limit, **CLM** asks SAS to calculate the confidence interval of the mean. These summary statistics for all numerical variables in the dataset will be calculated (i.e. **StudNo** and **YearBorn**).

If you only want to calculate the mean some of the variables in your dataset, use the **VAR** option and list the variables after the keyword **VAR**. If you want the means of the variables by group, use the **BY** option. For example:

```
DATA ageclub;  
  INFILE 'P:\example.txt';  
  INPUT StudNo YearBorn Student $;  
  StudAge=107-YearBorn;
```

```

    IF StudAge>=19 THEN AgeCat='Senior';
    ELSE IF StudAge=18 THEN AgeCat='Middle Aged';
    ELSE AgeCat='Child';
RUN;

PROC SORT DATA=ageclub;
    BY AgeCat;
RUN;

PROC MEANS DATA=ageclub MEAN;
    VAR StudAge;
    BY AgeCat;
RUN;

```

Notice that the data must be sorted by the group variable **AgeCat**, before using the **MEANS** procedure. **SAS** will compute the mean age for each value of **AgeCat**, i.e. Senior, Middle Aged and Child.

### 3.1.2 Calculating Correlations and Covariances

These can be calculated by using the **CORR** procedure (the covariance is an option). For example:

```

PROC CORR DATA=ageclub;
    VAR StudNo YearBorn;
RUN;

```

The Pearson correlation coefficient and its *p*-value are computed by default and displayed in a correlation matrix:

Pearson Correlation Coefficients, N = 3		
Prob >  r  under H0: Rho=0		
	StudNo	YearBorn
StudNo	1.00000	0.83478 0.3712
YearBorn	0.83478 0.3712	1.00000

The correlation coefficient between **StudNo** and **YearBorn** is 0.83478 and 0.3712 is the *p*-value for testing the null hypothesis that the coefficient is zero. In this case, the *p*-value is greater than 0.05, so there is insufficient evidence to reject the null hypothesis of zero correlation.

### 3.1.3 Univariate Statistics

PROC UNIVARIATE produces an elementary statistical analysis. It outputs the basic summary statistics and has optional statements to generate QQ-plots and histograms. For example:

```
PROC UNIVARIATE DATA=ageclub;  
    VAR YearBorn;  
    QQPLOT;  
    HISTOGRAM;  
RUN;
```

If the VAR statement was not specified SAS would perform the analysis on all the numeric variables in the dataset.

## 3.2 SAS Graphics

SAS has two styles of graphics, low level graphics consisting of text (ASCII) characters, and high level graphics. Limited details are provided on the low level graphics as they are slowly(!) becoming defunct.

### 3.2.1 Low Level Graphics (Scatter Plots)

The PLOT procedure produce low-level graphics, which hark back to the days of line printers (how many of you can remember those?). An example of a simple plot of **StudNo** and **YearBorn** is

```
PROC PLOT DATA=ageclub;  
    PLOT StudNo*YearBorn;  
RUN;
```

Where **StudNo** is along the vertical axis and **YearBorn** is along the horizontal. By default, the PLOT procedure

- Prints the name of the vertical variable next to the vertical axis and the name of the horizontal variable beneath the horizontal axis.
- Chooses ranges for both axes and automatically places tick marks at reasonably spaced intervals.
- Uses the letter A as the plotting symbol to indicate one observation; the letter B if two observations coincide; and so on.
- Prints a legend naming the variables in the plot and explaining the plotting symbols.

## Adding Title and Labels to a Plot

Here is an example of adding axis labels and titles to a plot:

```
PROC PLOT DATA=ageclub;
  PLOT StudNo*YearBorn;
  TITLE 'Plot of Student Number against versus Year of Birth';
  LABEL StudNo='Student Number'
        YearBorn='Year of Birth';
RUN;
```

## Specifying Plotting Symbols

Should you wish to specify the “asterisk (\*)” as the plotting symbol, you simply do the following

```
PROC PLOT DATA=ageclub;
  PLOT StudNo*YearBorn='*';
RUN;
```

**Note:** When you specify a plotting symbol, PROC PLOT uses that symbol for all points on the plot regardless of how many observations may coincide. If observations do coincide, a message appears at the bottom of the plot telling you how many observations are hidden.

## Creating Multiple Plots

Rather than do separate plots for different variables, they can be combined as follows

```
PROC PLOT DATA=ageclub;
  PLOT StudNot*YearBorn='*' StudNo*StudAge='o';
RUN;
```

The plots can be overlaid on the same axis using the option OVERLAY:

```
PROC PLOT DATA=ageclub;
  PLOT StudNot*YearBorn='*' StudNo*StudAge='o' / OVERLAY;
RUN;
```

### 3.2.2 High Level Graphics (Scatter Plots)

A simple example of a scatter plot is

```
PROC GPLOT DATA=ageclub;
  PLOT StudNo*YearBorn;
RUN;
```

Notice that the only difference between this and the low level graphics is the GPLOT statement.

## Adding Title(s)

You can add a title using the command ‘title’ as with the PLOT procedure above. Sub-titles can also be added:

```

PROC GPLOT DATA=ageclub;
  PLOT StudNo*YearBorn;
  TITLE 'Plot of Student Number against versus Year of Birth';
  TITLE2 'Example scatter plot using GPLOT';
  LABEL StudNo='Student Number'
        YearBorn='Year of Birth';
RUN;

```

**Note:** you can have up to ten titles, numbered 1-10, e.g title5.

For more details of titles and other graphic options consult the SAS help.

## Bar Plots

Firstly start off looking at a simple histogram

```

DATA ageclub2;
  INPUT StudNo YearBorn Student $;
  StudAge=107-YearBorn;
  IF StudAge>=19 THEN AgeCat='Senior';
  ELSE IF StudAge=18 THEN AgeCat='Middle Aged';
  ELSE AgeCat='Child';
  CARDS;
123 88 Bob
101 89 Belle
432 90 Tiye
524 89 Sidney
RUN;
PROC GCHART DATA=ageclub2;
  VBAR YearBorn;
  TITLE 'Histogram of Year of Birth';
RUN;

```

You will notice that each of the bars are solid blocks. While this looks nice on the screen, when printing it is very wasteful of printer ink, not to mention slower. One can use the **PATTERN** option to define another style. See the SAS help for details.

## Bar Plots with Subgroups

Supposing you want to produce a bar chart with subgroups, you could do that as follows

```

PROC GCHART DATA=ageclub2;
  VBAR YearBorn / SUBGROUP=AgeCat;
  TITLE 'Histogram of Year of Birth by Age category';
RUN;

```

You will notice that SAS automatically assigns a pattern for each subgroup and then includes a legend (key).



### 3.2.3 Pie Charts

Here is a simple example of a pie chart

```
PROC GCHART DATA=ageclub2;  
    PIE StudAge / NOHEADING VALUE=none SLICE=arrow PERCENT=none;  
    TITLE 'Pie chart of Ages';  
RUN;
```

### 3.2.4 Exporting graphs

Often you will need to export SAS graphics other documents, e.g. Word or Powerpoint. Graphs export best from PROC Gplot, as the low-level graphs from PROC PLOT often don't look good in other document formats. Low level graphs can simply be copied and pasted to other programs as they are just plain text.

SAS provides many different formats in which to save graphs and many options for how to export them, so only a selection is given below.

#### Exporting to Word/Powerpoint

The following outline methods for exporting to Word or Powerpoint, but will also work for other programs.

- From GRAPHICS window showing the results of PROC Gplot, find the graph you wish to export and left-click on it, go to the 'Edit' menu and click 'Copy'. In Word/Powerpoint, go to the 'Edit' menu, and click on "Paste Special". Use the option "Picture" to paste the graph. Depending on your Microsoft Office setup you may be able to simply use "Paste" instead.
- After selecting the graph you wish to export as before, go to the 'File' menu, and click "Export as Image". You can choose a variety of different formats in which to save the graph. After you choose the format, go to the Word/Powerpoint document, go to the 'Insert' menu, click on "Picture from file", find the file containing the graph you want to insert and click "OK".

# Index

\$, 10  
SAS Names, 4  
Bar Plots, High Level, 21  
Cards, 3  
Character Variables, 10  
Column Input, 11  
Comments, 15  
Correlation, 18  
Covariance, 18  
Dataset, 3  
Deleting Observations, 15  
display, 6  
Duplicates, 16  
Editor, 2  
exit, 2  
Formats, 12  
Gcharts, 21  
Graph, Title, 20  
Graphics, High Level, Scatter Plots, 20  
Graphics, Low Level, 19  
Graphs, Label, 20  
Graphs, Title, 20  
Help, 8  
If, 14  
Input, 3  
Inputting Data, 10  
Label, 20  
List Input, 10, 11  
Loading Data, 7  
Log, 2, 5  
Logic Statements, 13  
MEANS Procedure, 17  
Multiple Plots, 20  
Open, 4  
Output, 2, 6  
Pie Charts, 22  
Plotting Symbols, 20  
PRINT, 6  
Printer, 7  
Printing, 7  
Quit, 2  
Relation Operators, 13  
Running, 4  
Scatter Plots, Low Level, 19  
Screen Output, 7  
Selecting, 9, 11  
Shortcuts, 6  
Sort, Descending, 15  
Sort, Multivariables, 15  
Sorting, 15  
Start, 2  
Statements, Rules, 4  
Submit, 4  
Subsets, 9  
Tab, 3  
Title, 20  
Variables, Dependent, 12