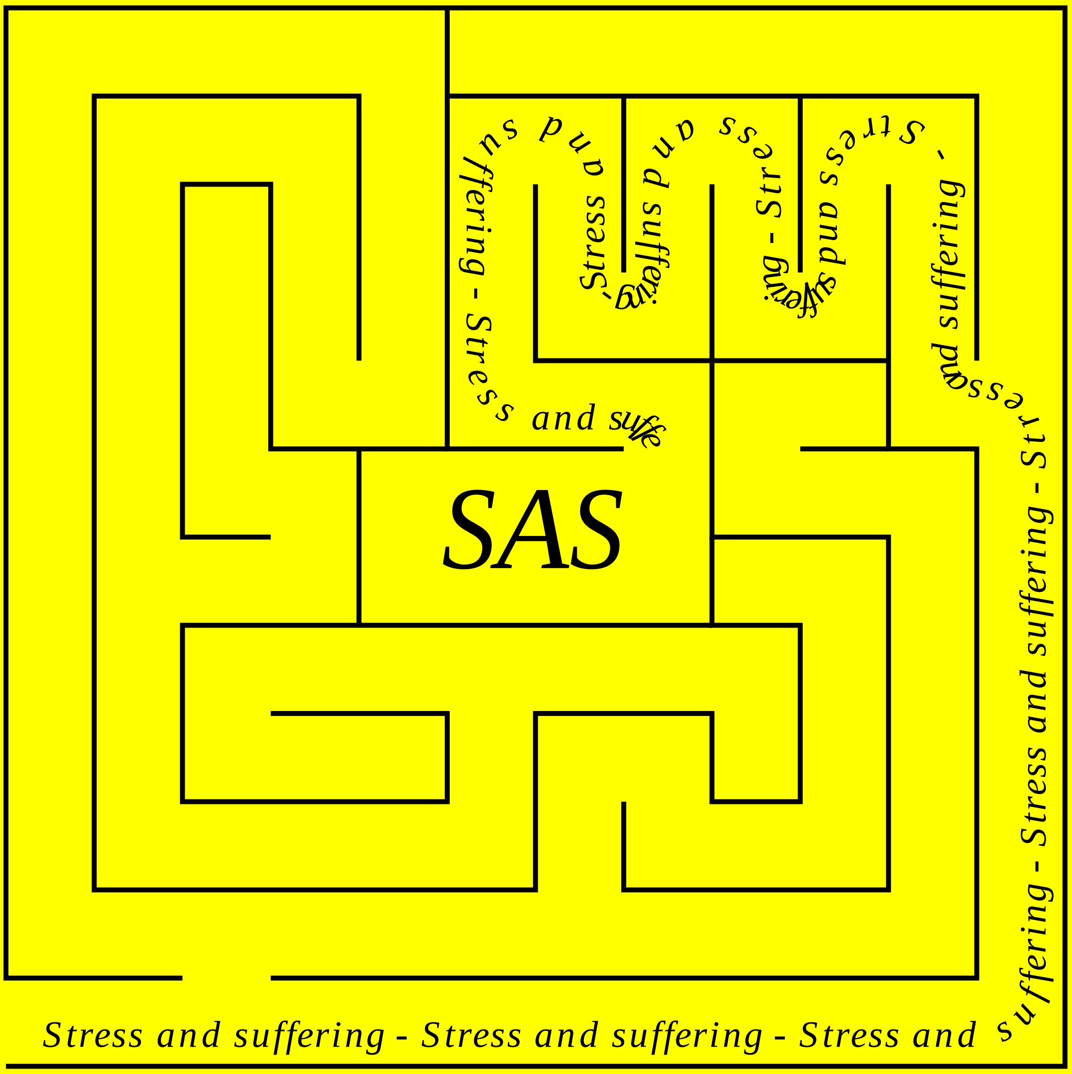
Introductory Guide to Unix-SAS

Department of Mathematics and Statistics


SAS
Stress and suffering - Stress and suffering - Stress and suffering - Stress and suffering - Stress and suffering - Stress and suffering - Stress and suffering - Stress and suffering - Stress and suffering - Stress and suffe

Written by Julian Visch

When SAS first appeared on the scene it was an acronym for the Statistical Analysis System. Since that time SAS has become an integrated system of software products and wants to lose the astigmatism of being considered just a statistical analysis system. The name SAS is no longer considered an acronym, but a name, and as such does NOT stand for anything.

Contents

1 The Basics

1.1 Introduction

When programming in any package the 8 crucial things you must learn when you start are

- How to start the package.
- How to exit the package.
- How to create and edit programs for running within the package.
- How to run the aforementioned programs.
- How to load data into the package.
- How to produce output within the package.
- How to produce printed versions of the aforementioned output.
- How to transfer output to a file.

1.2 How to Create Your Own Programs

As the program editor in SAS is not easy to work with you are advised to open up a new file using your favourite editor (e.g xedit hobbes1). For reference I have titled the initial SAS file hobbes1, your title may differ.

1.2.1 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 Starting SAS

To get onto SAS type "SAS &" (without the quotes) in the directory in which you intend to write your macros/programs.
e.g If my file hobbes1 was stored in the directory /users/math/staff/hobbes/sas then I would run SAS from the same directory.

Three windows will now appear with a fourth being automatically iconised. The windows are

- SAS: OUTPUT
  This window displays all the output generated from your programs, hence the name.
- SAS: LOG
  This window displays any error messages that occur because of bugs in your programs.
- SAS: PROGRAM EDITOR
  This window is where you retrieve, write, modify or submit(run) your programs.
- SAS: Session
  The xsassm client provides a convenient way of iconising or mapping the windows associated with its SAS client session, or interrupting and terminating its associated SAS client session via the UNIX signal mechanism.

Note: it is important to place the SAS: PROGRAM EDITOR near the top of the screen because later when you try and open a file a dialog box will appear which is positioned according to where the SAS: PROGRAM EDITOR is.

1.4 Exiting SAS

Either click “Terminate” on the SAS: SESSION window or select exit from the file menu.

1.5 Creating a Simple Program

All SAS programs are made up of one or more procedures, your first program will consist of just one simple procedure.
Firstly we will enter the following into our file “hobbes1”

```
                            ⎧tells SAS to create a temporary
data ageclub;             ← ⎩data set called ageclub.
  input StudNo YearBorn;  ← specifies variable names.
  cards;                 ← specifies that the data will follow.
123 88
101 89                    ← the data.
432 90
run;                      ← specifies the end of the procedure.
```

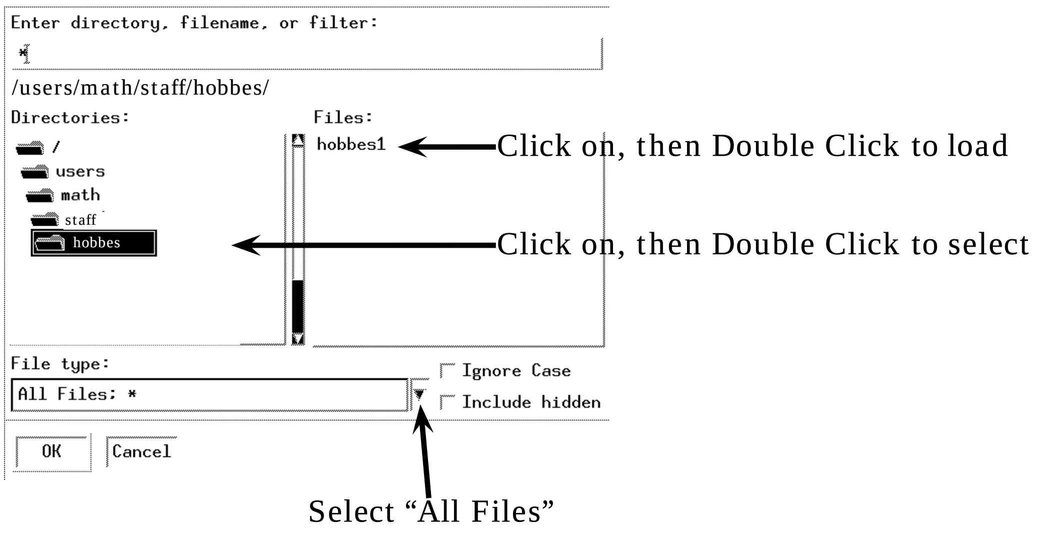
Note: It is important to note that tabs are not the same as spaces, therefore make sure no tabs exist in your data, as otherwise you may encounter problems when you run your sas programs.

1.5.1 Rules for SAS Statements

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

1.6 Running a Program

Go to SAS: PROGRAM EDITOR and select from the file menu "Open" followed by "Read file..."


Enter directory, filename, or filter:
*
/users/math/staff/hobbes/
Directories:
Files:
/
users
math
staff
hobbes
hobbes1
Click on, then Double Click to load
Click on, then Double Click to select
File type:
All Files: *
Ignore Case
Include hidden
OK
Cancel
Select "All Files"

You will now see the program appear on the SAS: PROGRAM EDITOR window, select from the Local menu "Submit". This will cause SAS to run your program. Once you have submitted the program you will note that in the SAS: LOG window something similar to that below will appear.

```
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 SAS: 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.

1.6.1 Short Cuts

For the cases where you have only a portion of a program to run you can skip the "loading in stage" by firstly selecting with the mouse your entire program and then selecting from the "local" menu "submit clipboard". Try selecting hobbes1 and then "submit clipboard".

To rerun entire programs quickly write in the program editor

```
%inc 'hobbes1';
run;
```

and then select "submit". Then to repeat it just select "recall text" followed by "submit".

1.7 Producing Output

To see some output you need to add the print procedure to your program.

```
proc print data=ageclub;
```

$\leftarrow$ Specifies which data set to print. If you omit the data=ageclub, the print procedure processes the most recently created SAS data set in the program.

Any additional options would be inserted here

```
run;
```

Now re-run your program (see Subsection 1.6.1).
You should now get the following in your output window.

```
      The SAS System                          1
                   09:38 Monday, June 10, 1996

     OBS     STUDNO     YEARBORN

      1        123         88
      2        101         89
      3        432         90
```

Date of submission

Page number

1.7.1 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.

1.7.2 Increasing the Default Page Size

Click on the Globals menu and select Global options followed by SAS options. A new dialogue box will now appear. Using the scroll bar to scroll down find PAGESIZE and change it the size you require (somewhere between 15 and 32767 lines).

1.8 Loading Data from a File

You have created data within a SAS file, but on occasion you will need to load data from a separate file. To do this we include an "infile" statement e.g infile 'hobbes2';

Currently hobbes2 does not exist, therefore you will need to create it by entering into a new xedit window the data

123 88
101 89
432 90

and then saving it as hobbes2 in the same directory as hobbes1. hobbes1 then needs to be changed to

```
data ageclub;
  infile 'hobbes2';
  input StudNo YearBorn;
run;

proc print data=ageclub;
run;
```

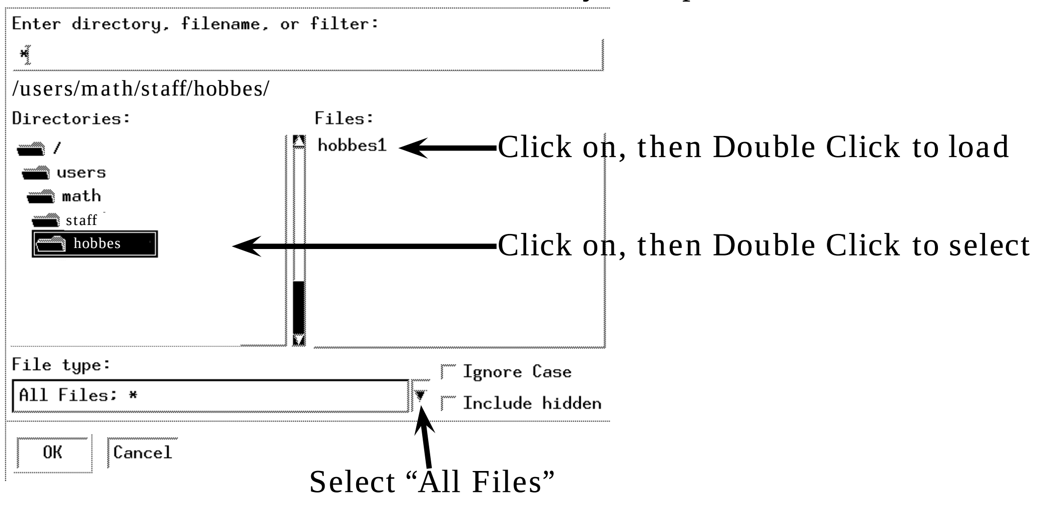
You can now re-run hobbes1 and you should get the same answer.

1.9 Printing your Output

You can print your output from the SAS: OUTPUT window by selecting "Print" from the File menu followed by the option "Print file...". Specify the name you wish to call it under `''Enter destination file name:''`. Click on OK. Ignore `''Enter form name:''` as the default is what you want.

1.10 Transferring Output to a File

You can save your output from the SAS: OUTPUT window into a file by selecting “Save as” from the File menu followed by the option “Write to file...”.



```
Enter directory, filename, or filter:
*
/users/math/staff/hobbes/
Directories:          Files:
/                     hobbes1
 users
  math
   staff
    hobbes
File type:
All Files: *
Ignore Case
Include hidden
OK   Cancel
```

Click on, then Double Click to load
Click on, then Double Click to select
Select “All Files”

1.11 Storing Permanent SAS Data Files

Should you wish to store the data permanently into a SAS data file you must first create a subdirectory in which to store them.
e.g `mkdir susie`
Then you will need to modify hobbes1 to

```
libname calvin 'susie'; ←
```

The libname statement informs SAS that you want the reference “calvin” to refer to the subdirectory called susie.

```
data calvin.ageclub; ←
  infile 'hobbes2';
  input StudNo YearBorn;
run;
```

The data statement informs SAS that you wish to store a permanent data set called ageclub into the subdirectory that corresponds with the libname calvin, in this case the subdirectory susie.

```
proc print data=calvin.ageclub;  ←

run;
```

When using permanently stored data files you must also include the libname

Note 1: Later when you do your SAS project, do not create a permanent data set, because having the whole class create the same permanent data set is very wasteful of computer memory. Instead you will have to modify the infile command to that specified by the lecturer.
Note 2: Should you not finish this booklet in one session and wish to carry on at a later date, you will need to re-include the libname statement.

1.12 Creating Subsets of Permanent Data Files

As some data sets are quite large you may which 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 calvin.ageclub which contained only students after and including row 2 of the data, create the program hobbes4

  ```
  data calvin.ageclub2;
     set calvin.ageclub(firstobs=2);
  run;

  proc print data=calvin.ageclub2;
  run;
  ```

- Obs=n
  Specifies the last observation to be read from the SAS data set you specify in the "set" statement.
  e.g. To create a subset of calvin.ageclub which contained only students up to and including row 2 of the data, modify hobbes4 to

  ```
  data calvin.ageclub2;
     set calvin.ageclub(obs=2);
  run;

  proc print data=calvin.ageclub2;
  run;
  ```

- Keep
  Specifies the variables to be included.
  e.g. To only include the variable StudNo, modify hobbes4 to

  ```
  data calvin.ageclub2;
    set calvin.ageclub;
    keep StudNo;
  run;

  proc print data=calvin.ageclub2;
  run;
  ```

- Drop
  Specifies the variables to be excluded.
  e.g. To remove all the variables except StudNo, modify hobbes4 to

  ```
  data calvin.ageclub2;
    set calvin.ageclub;
    drop YearBorn;
  run;

  proc print data=calvin.ageclub2;
  run;
  ```

2 Ways for Inputting Data

2.1 Inserting Character Variables

There are two types of variables “numerical” and “character”, where
numerical is what you have been using so far, and character is 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.
e.g to include the student names in the data set modify the input
statement in hobbes1 to

```
input StudNo YearBorn Student $;
```

and modify hobbes2 to

123 88 Hobbes
101 89 Calvin
432 90 Susie

When you re-run your program you should now see the students names also appearing in the output.

2.2 List Input

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

2.2.1 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.

2.3 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.

2.3.1 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.

2.3.2 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 reread, 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.

$$\left.\begin{array}{l}\texttt{12388Hobbes}\\ \texttt{10189Calvin}\\ \texttt{43290Susie}\end{array}\right\}\leftarrow \text{temporary modify hobbes2 to.}$$

Note: Make sure there is a blank space after Susie, 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
```

For the next subsection re-insert the spaces into hobbes2.

2.4 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 ommision 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 Hobbes 123 88
101 89 Calvin 101 89
432 90 Susie 432 90
```

You can see that the columns method would fail after Susie because the “4” or the “432” would fall in the wrong column in which case you would have to use the list input.

2.5 Data Files with Different Formats

For brevity I have not shown you every possible format for data set which SAS can handle but should you wish to explore more refer to one of the SAS manual which are available at the computer services library.

2.6 Impossible Data Files

Occasionally you may get an impossible data file such as

```
12388Hobbes12388
10189Calvin10189
43290Susie43290
```

In such cases you will have to edit the data file first.

2.7 Creating Dependent Variables

Sometimes you will wish to create variables which depend on 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. You can do this by inserting behind the input statement

```
libname calvin 'susie';
data calvin.ageclub;
  infile 'hobbes2';
  input StudNo YearBorn Student $;
  StudAge=97-YearBorn;
run;
proc print data=calvin.ageclub;
run;
```

For reference the addition operators are as follows

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

2.7.1 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. Therefore to create a new variable called "old" add the following after where you had defined age

```
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

2.8 Use of “If” Statements

Suppose you wished to divide the ages further into the categories young, middle aged and old. This can be done via an “if” statement as follows

```
 libname calvin ’susie’;
 data calvin.ageclub;
   infile ’hobbes2’;
   input StudNo YearBorn Student $;
   StudAge=97-YearBorn;
   if StudAge>=9 then AgeCat=’Senior’;
   else if StudAge=8 then AgeCat=’Middle Aged’;
   else AgeCat=’Child’;
 run;
 proc print data=calvin.ageclub;
 run;
```

Note: The “if” does not require an “end” statement.
Submitting the above program gives you the output

```
             Age of Club Members              11
                      09:08 Tuesday, October 1, 1996

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, to modify the previous one, you can insert a length statement as follows.

```
 StudAge=97-YearBorn;
 length AgeCat $ 11;
 if StudAge>=9 then AgeCat=’Old’;
```

You also use “and” and “or” statements. For instance the “and” statement works as follows

```
 if StudAge>7 & StudAge<9 then MidAged='Yes';
     else MidAged='No ';
```

or similarly

```
 if 7<StudAge<9 then MidAged='Yes';
     else MidAged='No ';
```

Similarly the “or” statement works as follows

```
 if StudAge<=7 | StudAge>=9 then MidAged='No ';
     else MidAged='Yes';
```

2.9 Deleting Observations

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

```
  if AgeCat='Old' then delete;
```

3 Odds and Ends

For the odds and ends section, first copy the data set hobbes3 from the directory specified by the lecturer. Then run the following in SAS

```
 libname calvin 'susie';
 data calvin.ageclub;
   infile 'hobbes3';
   input StudNo YearBorn Student $ 8-25 Sex $
   Sport $ 29-40 WheatBix MuesliBr Stat321;
 run;
 proc print data=calvin.ageclub;
 run;
```

Note: There are two pages of output, to see the first page use the scroll bar.

3.1 Defining Keys

On occasion you will find that you are constantly repeating the same series of commands. After a while this becomes very tedious. But you can define a key combination to relieve this tedium.

For instance it would be nice to be able to assign a key binding for recalling text, for submitting text, or even better doing both at once. To do this first select the “keys” option from the help menu. A dialog window will appear informing you of various keys, and key combinations, some of which already being giving definitions.

Note: The definitions linked with the keys F1-F12 will not work as they have already been assigned for system use, e.g. F1 iconises the window. Shift-F1 to Shift-F12 can be defined but be careful when defining shift-F10 because should you forget the shift component you will destroy your window. The KP keys all work except the "KP_," which doesn't exist, but for some of them such as KP_0 (the Ins key) you must also use the shift. All the control(Ctrl) keys also work.

Now suppose you wished to define CTRL A to be the key which will recall the last program and then submit it. To do this click next to Ctrl A and type

```
 pgm; recall; submit
```

Note: pgm is short for program manager(editor).
Now select save from the File menu to store the key binding.

3.2 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 */
```

3.3 Sorting your Data

SAS's Sort procedure has the form of the addition to hobbes1 below

```
 proc sort data=calvin.ageclub out=sortclub;
    by YearBorn;
 run;
 proc print calvin.sortclub; /* Prints sorted data */
 run;
```

Note: If you are not using permanent data sets, you may omit references to data sets.

3.3.1 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 wished to sort also by the variables Sex and Wheatbix then you would modify as follows.

```
 proc sort data=calvin.ageclub out=sortclub;
    by Sex Wheatbix;
 run;
 proc print;
 run;
```

Note: As sortclub was the last data set created, “proc print”, selects it as the default.

3.3.2 Sorting in Descending Order

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

```
proc sort data=calvin.ageclub out=sortclub;
   by descending YearBorn;
run;
proc print;
run;
```

3.3.3 Removing Duplicates

The “Noduplicates” option in the sort procedure deletes any observation that duplicates the values of all variables of another observation in the data set. For instance Lucy van Pelt appears twice in the data set, so to remove the duplication write the following

```
proc sort data=calvin.ageclub out=calvin.ageclub
   noduplicates;
   by YearBorn;
run;
proc print;
run;
```

3.4 Using the Inbuilt Help Facility

In general the inbuilt help facility is useless or frustrating. But with items like the sort procedure their help is handy as a reminder of the syntax and options. To find help on “sort”, you need to do the following.

- Select “Extended Help” from the “Help” menu. (A new window will appear).
- Click on “Data Management”.
- Click on “Sort”.
- Click on “Introduction”.
- After reading the introduction, click on “GoBack”.
- Click on “Syntax”.

You can click on any bold text and you will get a brief explanation of that topic, use the “GoBack” command to go back to the previous screen. Once you feel that you have explored the “sort” procedure sufficiently click on GoBack repeatedly until you have exited help. Do not just click on “exit help”, unless you wish your next help query to start where you left off.

3.5 Using the Means Procedure

Instead of having a series of procedures for calculating minimums, maximums etc, they have all been combined into the proc means procedure where you can select which statistical information you require from the available options. For a list of these options and the syntax, use the extended help as follows

- Click on “Modeling and Analysis Tools”.
- Click on “Data Analysis”.
- Click on “Means”.
- Click on topics of interest. e.g Click on Syntax, then options.
- Once finished click on GoBack until you exit help.

3.6 Calculating Correlations and Covariances

These can be calculated by using the correlation procedure (the covariance portion is one of the suboptions). Help on its use can be found with the online help under Modeling & Analysis Tools → Data Analysis → corr.
Note: unlike previous procedures this procedure does send output to the SAS: OUTPUT window, and therefore an additional print procedure is not needed.
e.g.

```
proc means data=calvin.ageclub n min max;
  class Sex;
run;
```

4 SAS Graphics

SAS has two styles of graphics, low level graphics consisting of ASCII characters, and high level graphics.

4.1 Low Level Graphics (Scatter Plots)

An example of a simple plot of YearBorn vs WheatBix is

```
proc plot data=calvin.ageclub;
   plot WheatBix*YearBorn;
run;
```

Where WheatBix 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.

4.1.1 Adding Title and Labels to a Plot

```
proc plot data=calvin.ageclub;
   plot WheatBix*YearBorn;
   title 'Plot of WheatBix versus Year of Birth';
   label WheatBix='WheatBix Eating'
         YearBorn='Year of Birth';
run;
```

4.1.2 Modifying Tickmarks

```
proc plot data=calvin.ageclub;
   plot WheatBix*YearBorn / vaxis=0 to 10 by 2;
run;
```

This will modify the vertical axis to start at 0 and then going up in steps of 2 up to 10. You can similarly modify the horizontal axis using “haxis”.

4.1.3 Specifying Plotting Symbols

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

```
proc plot data=calvin.ageclub;
   plot WheatBix*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.

4.1.4 Creating Multiple Plots

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

```
proc plot data=calvin.ageclub;
   plot WheatBix*YearBorn='*' MuesliBr*YearBorn='o';
run;
```

4.1.5 Creating Multiple Plots on the Same Page

To help make it easier to compare graphs you can do multiple plots on the same “page” (by page I refer to a SAS page) via “vpercent” and “hpercent”. Using the vpercent you have the following

```
proc plot data=calvin.ageclub vpercent=50;
   plot WheatBix*YearBorn='*' MuesliBr*YearBorn='o';
run;
```

and using hpercent you have

```
proc plot data=calvin.ageclub hpercent=50;
   plot WheatBix*YearBorn='*' MuesliBr*YearBorn='o';
run;
```

Note: The percent component refers to the percentage of the page to be used for each plot.

4.1.6 Creating Multiple Plots on the Same Axes

```
proc plot data=calvin.ageclub;
   plot WheatBix*YearBorn='*'
        MuesliBr*YearBorn='o' / overlay;
run;
```

4.2 Low Level Graphics (Bar Plots)

With SAS you can produce vertical bar charts (histograms) or horizontal bar charts as shown below.

```
proc chart data=calvin.ageclub;
   vbar WheatBix;
run;
```

When you use the vbar statement without any options, proc chart automatically

- Labels the axes.
- Selects midpoints and ranges.
- Places tick marks on the vertical axis based on the ranges it selects.

To produce a horizontal bar chart use hbar instead of vbar. You should notice the following statistics also appearing,

- Freq (frequency)
  Is the number of observations in a given range.
- Cum. Freq (cumulative frequency)
  Is the number of observations in all ranges up to and including a given range. The cumulative frequency for the last range is equal to the number of observations in the data set.
- Percent
  Is the percentage of the observations in the given range.

- Cum. Percent (Cumulative percent)
  Is the percentage of observations in all ranges up to and including a given range. The cumulative percentage for the last range is always 100.

To eliminate these statistics modify your SAS program to

```
 proc chart data=calvin.ageclub;
    hbar WheatBix / nostat;
run;
```

4.2.1 Specifying Midpoint Values

You can specify midpoints as follows

```
proc chart data=calvin.ageclub;
   vbar Stat321 / midpoints=10 to 90 by 20;
run;
```

The corresponding ranges are

0 to 19
20 to 39
40 to 59
60 to 79
80 to 99

4.2.2 Specifying Number of Midpoints

You can specify the number of midpoints by using the levels statement as follows

```
proc chart data=calvin.ageclub;
   vbar Stat321 / levels=5;
run;
```

4.2.3 Charting every Value

Should you wish to have a bar of every value, one can use the discrete option as follows

```
proc chart data=calvin.ageclub;
   vbar WheatBix / discrete;
run;
```

4.2.4 Creating Subgroups within a Range

To split WheatBix consumption within the bars by gender you can use the following SAS statements.

```
proc chart data=calvin.ageclub;
   vbar WheatBix / subgroup=sex;
run;
```

4.2.5 Grouping Bars

To split WheatBix consumption into separate bar charts according gender you can use the following SAS statements.

```
proc chart data=calvin.ageclub;
   vbar Stat321 / group=sex;
run;
```

5 High Level Graphics

5.1 Scatter Plots

A simple example of a scatter plot is

```
proc gplot data=calvin.ageclub;
     plot WheatBix*YearBorn;
run;
```

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

5.1.1 Adding a Title

You can add a title using the command “title”
e.g. `title ’Plot of WheatBix versus Year of Birth’;`
Note: To view subsequent graphs press the PgDn key, and to view previous graphs press the PgUp key.
The title command has several options which are as follows

- Position
  ```
  title move=(-1cm,-1cm)
       ’Plot of WheatBix versus Year of Birth’;
  ```
- Size
  ```
  title height=0.5cm
       ’Plot of WheatBix versus Year of Birth’;
  ```
- Font
  ```
  title font=brush
  ’Plot of WheatBix versus Year of Birth’;
  ```

- Mix Fonts
  ```
  title font=brush ’Plot of ’ font=cscript ’
  ’WheatBix ’ font=brush ’versus ’
  font=cscript ’Year of Birth’;
  ```
- Multiple titles
  ```
  title font=brush
  ’Plot of WheatBix versus Year of Birth’;
  title2 font=cscript ’For the Club’;
  ```

Note: one can have up to ten titles, numbered 1-10, e.g title5.
For more details of titles and other graphic options consult the online help.

5.2 Bar Plots

Firstly start off looking at a simple histogram

```
proc gchart data=calvin.ageclub;
   vbar sport;
   title font=brush ’Histogram of Sports’;
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 subcommand to define another pattern style, an example showing its use will be given later. See the “extended help” for exact details.

5.2.1 Bar Plots with Subgroups

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

```
proc gchart data=calvin.ageclub;
   vbar sport / subgroup=sex;
   title font=brush ’Histogram of Sports by Sex’;
run;
```

You will notice that SAS automatically assigns a pattern for each gender and then includes a legend (key), informing you of which pattern it has assigned to which gender. You can similarly define your own as shown in the following example

```
proc gchart data=calvin.ageclub;
   vbar sport / subgroup=sex;
   title font=brush ’Histogram of Sports by Sex’;
   pattern1 value=L1;
   pattern2 value=X1;
run;
```

5.3 Pie Charts

Here is a simple example of a pie chart

```
proc gchart data=calvin.ageclub;
   pie sport
    / NOHEADING value=none
    slice=arrow PERCENT=none;
   title font=brush ’List of Sports’;
   pattern1 value=p3n0;
   pattern2 value=p3n45;
   pattern3 value=p3n90;
   pattern4 value=p3n135;
   pattern5 value=p3x0;
run;
```

5.4 Graphic Options

Whenever you produce a graph in which you define some graphic options, these options will also exist for future graphs within your session, unless modified. In some situations you may wish to know what graphic options you have already defined, to do this, you will use the goptions procedure as follows

```
proc goptions;
run;
```

Examining the SAS: LOG window will give you an alphabetic list of options with a description and present values.
Note: there is also a goptions statement which you can use to define graphic options. For instance to include the graphics options for the previous pie chart you would proceed as follows.

```
goptions;
 title font=brush ’List of Sports’;
pattern1 value=p3x0;
pattern2 value=p3n90;
pattern3 value=p3n135;
pattern4 value=p3n45;
pattern5 value=p3n0;
/* Note: goptions does not require a run statement.*/
proc gchart data=calvin.ageclub;
   pie sport
    / NOHEADING value=none
    slice=arrow PERCENT=none;
run;
```

5.5 Printing Graphs

To print your graphs on the laser printer, you proceed with

```
filename gsasfile ’snoopy.ps’;
goptions device=ps300 gaccess=gsasfile;
proc gchart data=calvin.ageclub;
   pie sport;
run;
```

The filename statement has the syntax

```
filename fileref ’external-file’;
```

You replace fileref with a SAS name of no more than eight letters, in this case it must be called “gsasfile”. You replace external-file with the name you wish to call your graph. Your filename can be of any length but you are advised to include a .ps extension so that you can easily identify it as a postscript file in the future.
Note: Once you have defined a SAS name to be a file, you cannot redefine the name as something else until you have quit out of the prior name.
For example

```
quit;
filename gsasfile ’kermit.ps’;
goptions device=ps300 gaccess=gsasfile;
proc gchart data=calvin.ageclub;
   pie sport;
run;
```

Where the quit statement at the beginning cancels out the earlier reference to snoopy.ps. You will have noticed that your graphs are not appearing on the screen but are being sent only to a file. The next subsection will inform you of how to view your graphs on the screen again.

5.6 Graphics to file $\rightarrow$ graphics to screen

To return to having graphics on your screen specify the device to be empty.
i.e.

```
goptions device=’’;
proc gchart data=calvin.ageclub;
   pie sport;
run;
```

6 Exercises

1. Create a simple program which creates a temporary data set called butterfly with the numerical variables “acre” and “transline”. Insert into your program the first three rows of data for acre and transline as shown in the appendix. Submit your program in SAS, then before you hand it in examine the log window to ensure that your program contains no errors. For future exercises you may be required to use this program, hence you are advised to store this program in your directory.
2. Modify the program from ex 1 to include a print procedure, submit your program and save the output to a file.
3. Create a permanent data set of the data set using all 26 values of the data using all of the variables as shown in the appendix. To save time, you may wish to copy the file from the “/users/math/underg/sas” directory into your own area.
4. Create a new program designed to obtain a subset of the data set created in exercise 3 such that it only contains the last ten observations.
5. Add to your program from exercise 3 the string variable “ageclass”.
6. Create a subset of the data so that only data for ageclasses being young are selected.
7. Write a sas formula for converting the area in acres to meters and save it as the new variable msq.
8. Create a new string variable called size, such that
   (a) If acre < 25, size is small
   (b) If $25 \leq$ acre < 50, size is medium
   (c) If $50 \leq$ acre < 100, size is large
   (d) If acre ≥ 100, size is xtlarge

   Once finished compare with classsize and make sure that they are the same.
9. Remove all data that has classsize as extra large.
10. Add a comment to your program
11. Sort your data according to acre in reverse order. How have the classsizes been grouped?
12. Using the help facility find out how to use anova and impliment with the model kbb=acre.
13. Plot acre by kbb using a “o” as the plotting symbol
14. Create a vbar chart of kbb with midpoints ranging from 0 to 50 in steps of 10 by the variable ageclass.
15. Create a pie chart of ageclass.

7 Appendix

1995 Summer survey for Karner blue butterflies, Wisconsin, USA

26 sites were surveyed, each where a different size (acres), and surveyed using different length of transect (tranline). The sites were all in forest plantations, where either young, medium or old aged pine trees were growing. Sites are also classified as being small, medium, large or extra large depending on their acreage.

sitenum	acre	tranline	ageclass	sizeclass	kbb
1	39	15459	young	m	13
2	99	17277	young	l	44
3	18	4806	medium	s	3
4	19	3623	medium	s	2
5	14	2447	young	s	24
6	52	11977	medium	l	8
7	52	11977	medium	l	16
8	16	3828	medium	s	10
9	42	9394	old	m	4
10	25	6042	medium	m	2
11	41	8111	medium	m	2
12	52	13671	young	l	6
13	52.5	13671	young	l	9
14	15	4321	young	s	1
15	366	60697	old	xl	5
16	80	22949	young	l	4
17	80	22949	young	l	21
18	250	50667	young	xl	18
19	172	46238	medium	xl	2
20	172	42970	medium	xl	4
21	38	21067	medium	m	33
22	74	18470	medium	l	76
23	40	8355	medium	m	9
24	113	30885	young	xl	16
25	185	49450	young	xl	4
26	199	44280	young	xl	8

Index