



Faculty of Natural and Agricultural Sciences

Fakulteit Natuur- en Landbouwetenskappe

Mathematical Sciences

Department of Statistics

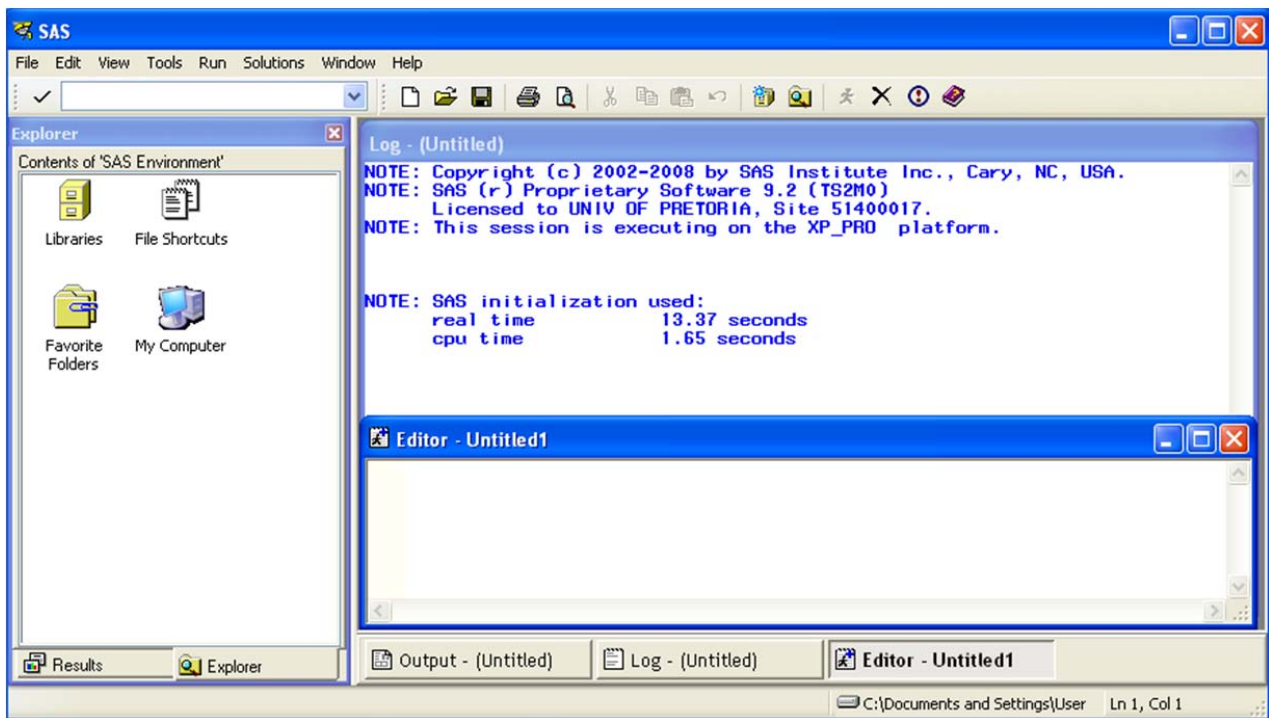
Introductory SAS course
WST 795 / STK 795

Table of Contents



1. HOW TO GET STARTED	3
2. DATA.....	4
3. SAS HELP	5
4. HOW TO GET YOUR DATA IN SAS	6
4.1 ENTER THE DATA DIRECTLY INTO SAS	6
4.2 IMPORT THE DATA FROM EXCEL	9
4.3 USE THE INFILE STATEMENT.....	14
5. CREATING A NEW LIBRARY IN SAS	15
6. PROC FORMAT and the FORMAT STATEMENT	16
7. CREATING NEW VARIABLES	18
8. PROC PRINT.....	20
9. PROC UNIVARIATE.....	21
10. PROC MEANS	22
11. PROC FREQ	24
12. PROC GPLOT	26
13. PROC CORR	28
14. PROC REG	28
15. PROC TTEST	31
16. PROC GLM	32
17. Introduction to PROC IML (Interactive Matrix Language).....	35
SUMMARY OF SAS PROCEDURES	46
ASSIGNMENT.....	47
REFERENCES.....	48

1. HOW TO GET STARTED

When you open SAS, this is the screen that you will see.



SAS consists of three windows:

- | | |
|---------------|--|
| Editor window | <ul style="list-style-type: none">• type a new program or open an existing program• edit a program• run a program • Take note: if you want to save a program this window must be active when you click on save. |
| Log window | <ul style="list-style-type: none">• shows the statements submitted• contains error messages in red and warning messages in green• Hint: Always clear the log window before running a new program  |
| Output window | <ul style="list-style-type: none">• If this window is not visible (version 9.3.) go to tools->options ->preferences->results->create listing (create HTML creates HTML output)• shows the results of the program submitted• Take note: if you want to save the output this window must be active when you click on save. |

SAS programs consist of two types of steps:

- | | |
|------------|---|
| DATA steps | <ul style="list-style-type: none">• put data in a form that the SAS program can use it• give names to the variables• create new variables, etc. |
| PROC steps | <ul style="list-style-type: none">• use procedures to analyse the data |

2. DATA

Consider the data collected for the 2007 springbok rugby world cup squad.


Name	Surname	Province	Birthdate	Birthplace	Position	Height	Weight
Bakkies	Botha	B	1979/09/22	Newcastle	1	2.01	118
BJ	Botha	S	1980/01/04	Durban	2	1.82	116
Gary	Botha	B	1981/10/12	Pretoria	3	1.81	108
Schalk	Burger	W	1983/04/13	Port_Elizabeth	4	1.93	110
Jean	de_Villiers	W	1981/02/24	Paarl	5	1.91	100
Bismarck	du_Plessis	S	1984/05/22	Bethlehem	3	1.89	112
Jannie	du_Plessis	C	1982/11/16	Bethlehem	2	1.88	116
Fourie	du_Preez	B	1982/03/24	Pretoria	6	1.82	89
Os	du_Randt	C	1972/09/08	Elliot	2	1.9	125
Jaque	Fourie	L	1983/03/04	Johannesburg	5	1.89	95
Bryan	Habana	B	1983/06/12	Johannesburg	8	1.79	94
Butch	James	S	1979/01/08	Johannesburg	7	1.86	97
Enrico	Januarie	W	1982/02/01	Hopefield	6	1.68	82
Victor	Matfield	B	1977/05/11	Polokwane	1	2	108
Percy	Montgomery	S	1974/03/15	Namibia	9	1.83	88
Johann	Muller	S	1980/01/06	Mossel_Bay	1	2	114
Akona	Ndugane	B	1981/02/20	Umtata	8	1.82	92
Wynand	Olivier	B	1983/06/11	Welkom	5	1.86	93
Ruan	Pienaar	S	1984/03/10	Bloemfontein	6	1.87	90
JP	Pietersen	S	1986/07/12	Cape_Town	8	1.9	95
Andre	Pretorius	L	1978/12/29	Johannesburg	7	1.76	90
Danie	Rossouw	B	1978/06/05	Sabie	1	1.96	117
Bob	Skinstad	S	1976/07/03	Zimbabwe	4	1.94	108
John	Smit	S	1978/04/03	Polokwane	3	1.88	117
Juan	Smith	C	1981/07/30	Bloemfontein	4	1.96	107
Gurthro	Steenkamp	B	1981/06/12	Paarl	2	1.86	118
Francois	Steyn	S	1987/05/14	Aliwal_North	5	1.91	100
Albert	van_den_Berg	S	1974/01/26	Hoopstad	1	2.01	107
Wikus	van_Heerden	B	1979/02/25	Johannesburg	4	1.93	108
CJ	van_der_Linde	C	1980/08/27	Welkom	2	1.89	125
Ashwin	Willemse	L	1981/09/08	Swellendam	8	1.87	91

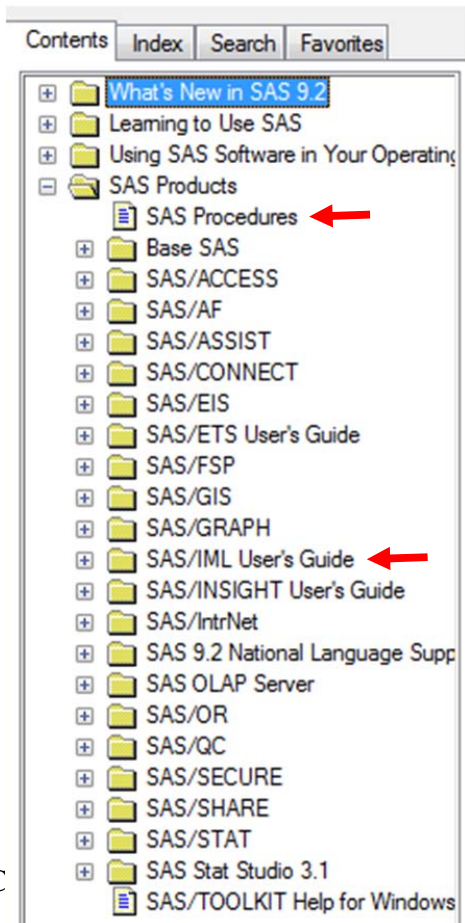
Codes for the categorical variables

Lock	1
Prop	2
Hooker	3
Flanker	4
Centre	5
Scrumhalf	6
Flyhalf	7
Wing	8
Fullback	9

Blue Bulls	B
Cheetahs	C
Lions	L
Sharks	S
Western Province	W

3. SAS HELP

This is the **most important tool** to ensure success using SAS. SAS Help can be opened by clicking on the  button.



The search and index functions are not very user-friendly, rather use the **contents** tab.

Expand the SAS Products folder, inside this folder you will find a list of all the SAS procedures with syntax, options and examples for each as well as, amongst others, a very useful guide to using PROC IML with syntax and examples.

4. HOW TO GET YOUR DATA IN SAS

4.1 ENTER THE DATA DIRECTLY INTO SAS

Consider the following dataset of the province, height and weight of the first 5 players:

Province	Height	Weight
Bulls	2.01	118
Sharks	1.82	116
Bulls	1.81	108
WP	1.93	110
WP	1.91	100

The basic syntax:

```
data data_set_name;  
input variable(s);  
datalines;  
(enter the data)  
;  
run;
```

Let us look at this in more detail using an example.

```
data rugby; ①  
input province$ height weight; ②  
datalines; ③  
bulls      2.01      118 ④  
sharks     1.82      116  
bulls      1.81      108  
wp         1.93      110  
wp         1.91      100  
; ⑤  
run;  
  
proc print data=rugby; ⑥  
run;
```

- ① The `data` statement names the dataset. In other words we want to create a dataset with the name `rugby`.
The name must start with a letter or an underscore character (`_`) and can then include letters, underscores or digits. Using blanks or characters like a comma are not allowed.

- ② The `input` statement defines the names of the variables in the dataset. The variable names must conform to the same rules as the dataset name. The dollar sign (\$) after province indicates that it is an alphanumeric variable. The @@ symbol can be used to denote that more than one player's information are entered into a line, for example

```
input province$ height weight @@;
datalines;
Bulls      2.01      118      Sharks      1.82      116
Bulls      1.81      108      WP           1.93      110
WP         1.91      100
;
```

If you do not type @@, you must type the data for each player in a new line.

- ③ `datalines` indicates that the data follows.
`cards` or `lines` are alternatives for `datalines`.
- ④ Type in the data.
 SAS uses blank spaces to distinguish between data values.
 Use a decimal point for decimal digits.
 A missing value or non-response is indicated by a decimal point (.). For example, suppose we did not know the height of the first player,
`datalines;`
 Bulls . 118
- ⑤ Type a semicolon at the end of all the data on a new line.
- ⑥ This is just to check the dataset. At a later stage we will look at `proc print` in more detail.



What if the data is without spaces between the observations, for example

```
B2.01118
S1.82116
B1.81108
W1.93110
W1.91100
```

We need to adjust our `input` statement. Instead of just mentioning what to call the variables, we need tell SAS where to find them. For example

```
data rugby;
input province$ 1 height 2 - 4 weight 5-7;
datalines;
B2.01118
S1.82116
```

```

B1.81108
W1.93110
W1.91100
;
run;

proc print data=rugby;
run;

```



How to handle a date in SAS.

In the input statement you have to specify the format of your date.

The generic format of a date is ddmmyyxx or yymmddxx where

x - specifies the separator to be used (D – dash, S – slash, P – period etc)

w - the length of the date

If x is omitted then the date will be formatted using the input format.

In this example we have year/month/day and it is a total of 10 characters long.

You need to put a colon (:) between the variable name and the informat.

When you print the data you will notice that SAS converted the date to a number.

Use the format statement in the print procedure to print the date in a date format. You may use a different format here, for example ddmmyy10. will first give the day, then month and lastly the year.

```

data rugby;
input province$ date : yymmdd10. height weight;
datalines;
B 1979/09/22 2.01 118
S 1980/01/04 1.82 116
B 1981/10/12 1.81 108
W 1983/04/13 1.93 110
W 1981/02/24 1.91 100
;
run;

proc print data=rugby;
run;

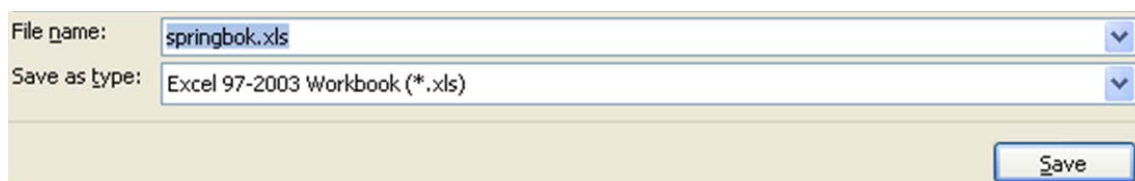
proc print data=rugby;
format date ddmmyy10.;
run;

```

Note: You can separate the year, month, and day values by blanks or by special characters like (/).

4.2 IMPORT THE DATA FROM EXCEL

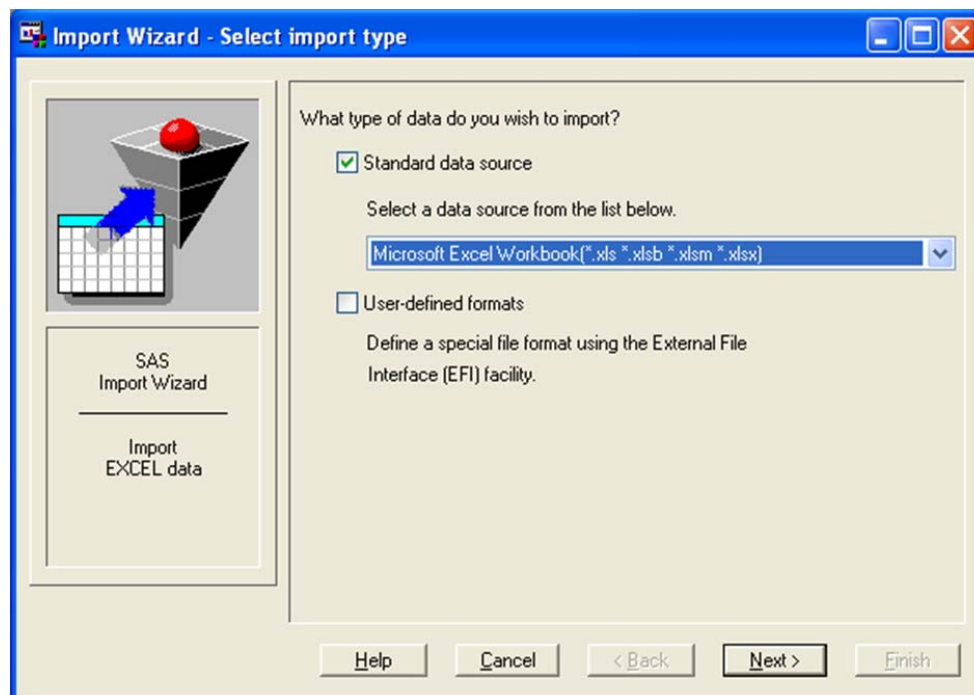
Step 1: EXCEL: Save the file as an *.xls, *.csv or *.xlsx file and close the file.



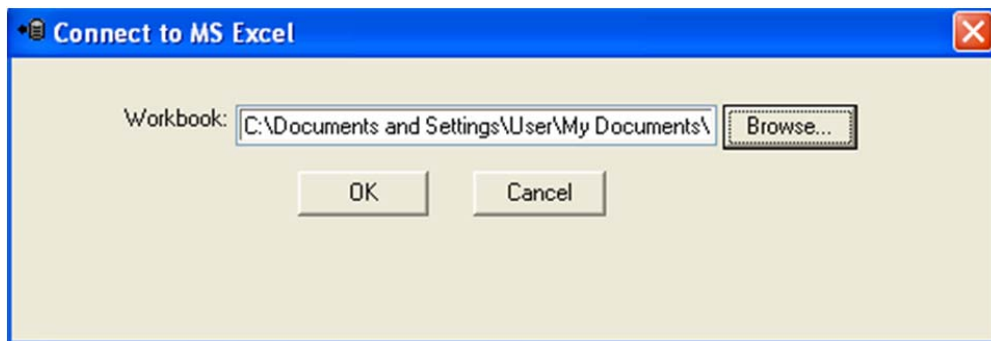
Step 2: SAS: Go to **File – Import Data...**



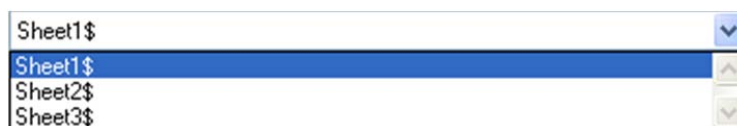
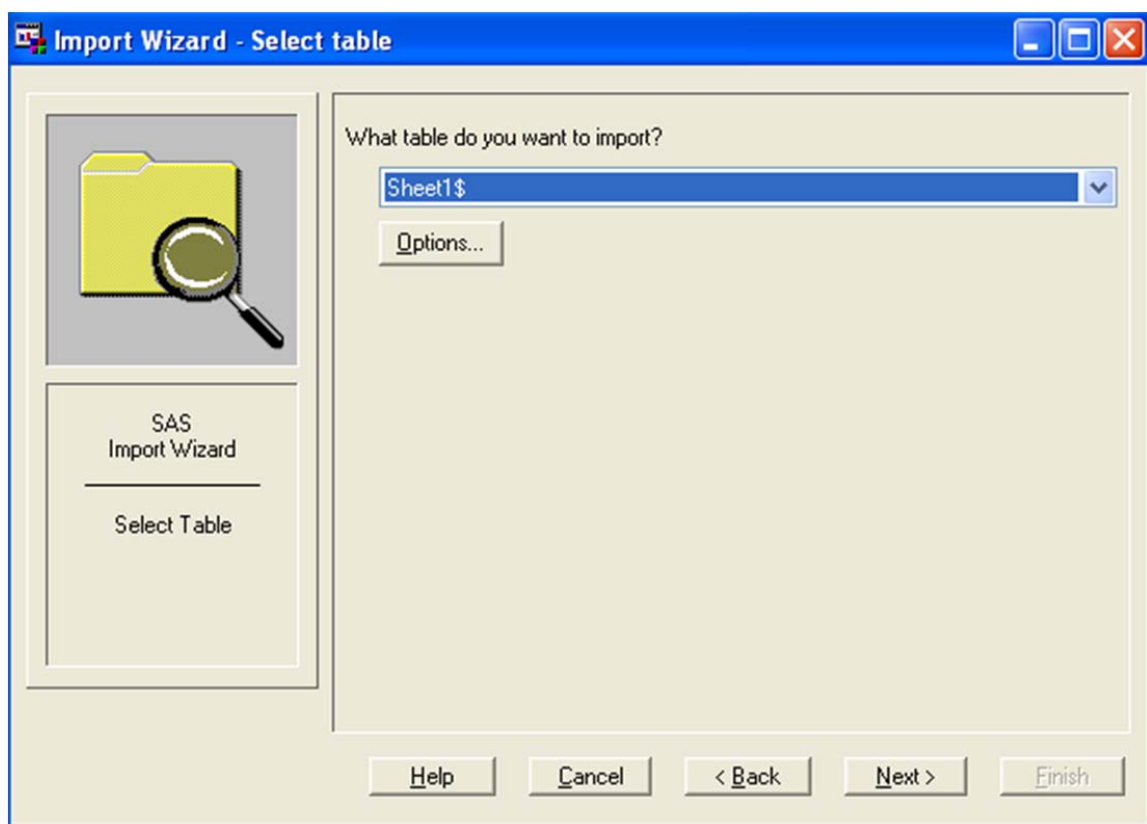
Step 3: Select the type of file format (click on the dropdown arrow to change the filetype) that you want to import and click **Next**.



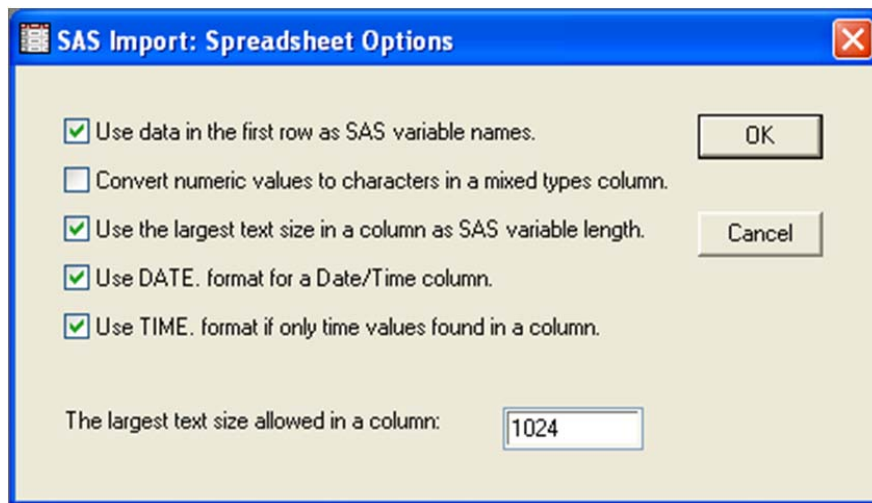
Step 4: Click on **Browse**. Select the exact location of the Excel file (springbok.xls) and click **OK**.



Step 5: Select the specific worksheet in the springbok.xls file.

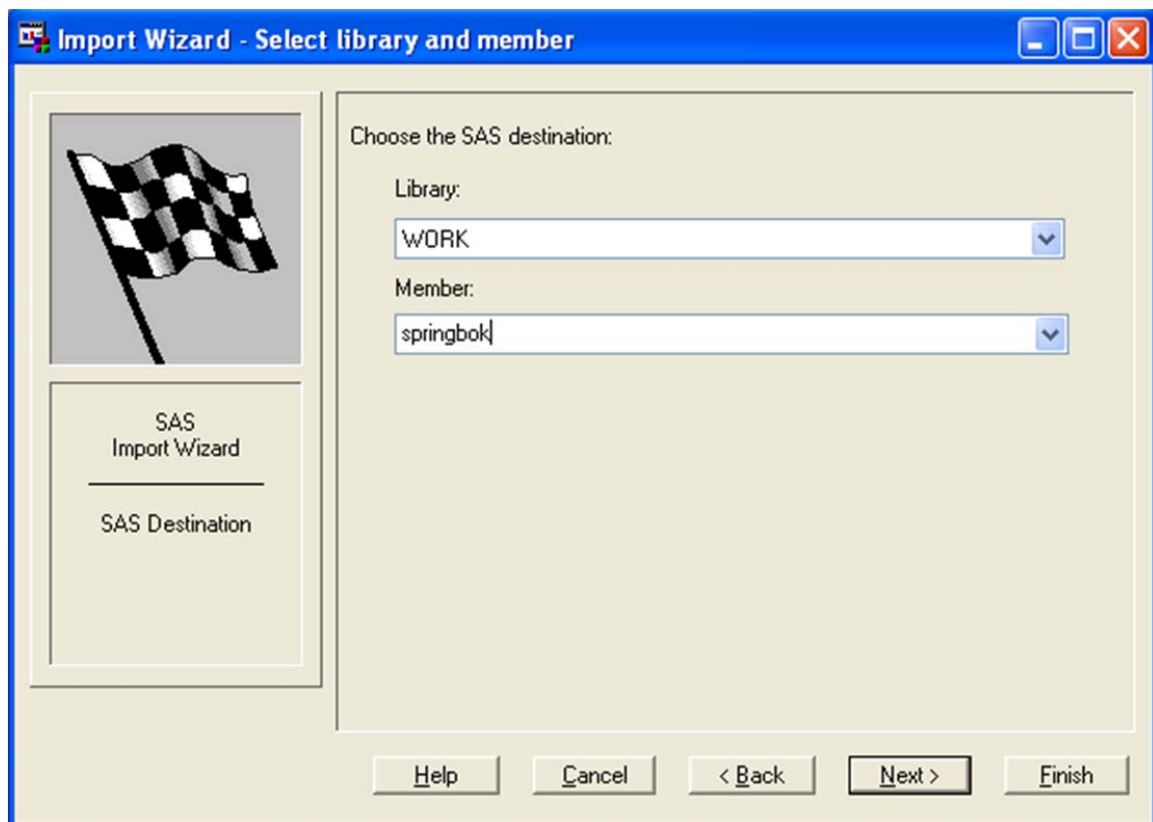


Step 6: Click **Options** and tick the box that the variable names appear in the first row of your worksheet if this is the case. Click **OK** and **Next**.

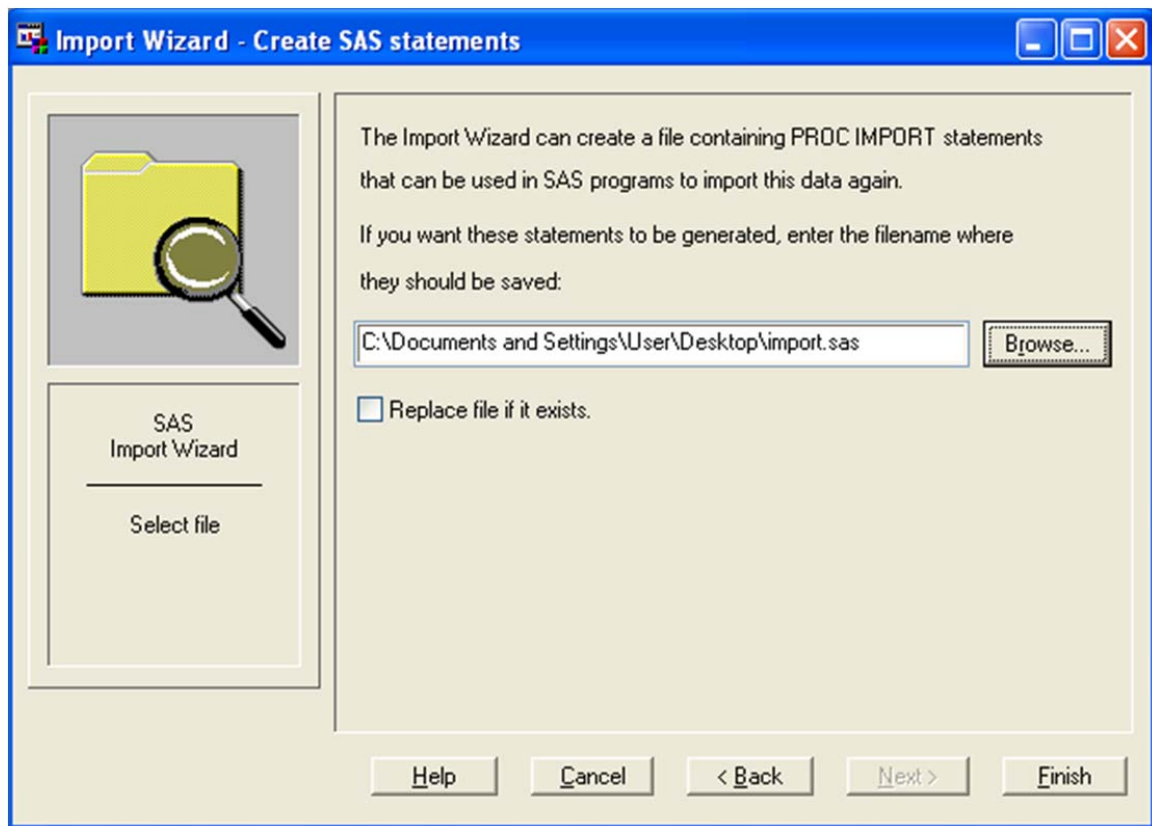


Step 7: Select the SAS destination. Select the library and in the **Member** box, type a name for the imported dataset. It does not have to be the same name as the Excel file. Click **Next** or **Finish**.

Note: The WORK library is a temporary library where data are stored. The contents are lost when SAS is closed.



Step 8: If you want to create a file containing PROC IMPORT statements you can specify where it must be saved. You can use these statements then in your program so that every time you open SAS and run the program it will import the data automatically. Click **Finish**.



Step 9: Check the message in the LOG window to determine whether the dataset was successfully created.

```
Log - (Untitled)
NOTE: Copyright (c) 2002-2008 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.2 (TS1M0)
      Licensed to UNIV OF PRETORIA, Site 0051400017.
NOTE: This session is executing on the XP_PRO platform.

NOTE: SAS initialization used:
      real time      10.59 seconds
      cpu time       1.07 seconds

NOTE: WORK.SPRINGBOK data set was successfully created.
```

The data step in the SAS program will be as follows:

```
data rugby; (give the dataset a name, you may give the same name as the Excel file)
set springbok; (this is the member name used in step 7)
```

Step 10: Check the data to make sure that the data was imported correctly.

Print the dataset

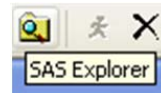
```
proc print data = rugby;  
run;
```

Check the contents and properties

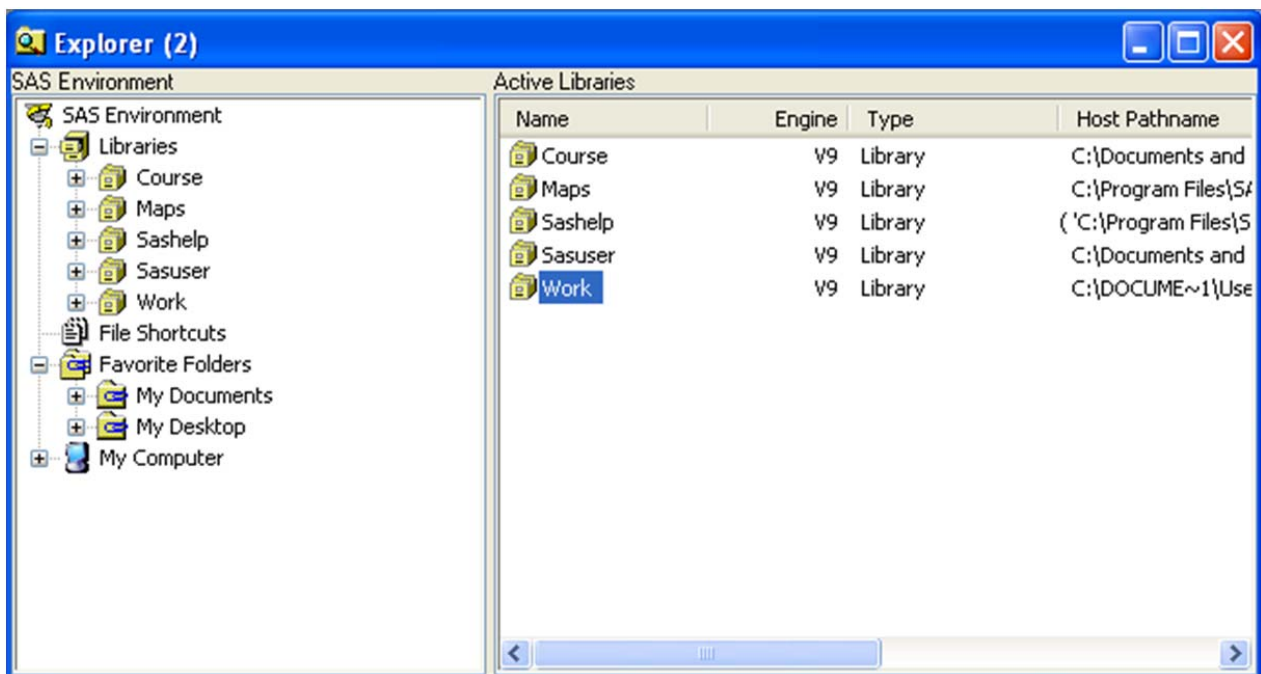
```
proc contents data = rugby;  
run;
```

Look in the SAS explorer

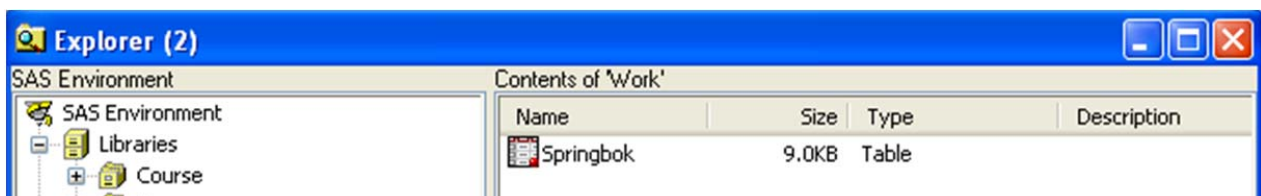
- a. Click on the SAS Explorer Icon



- b. Double click on the Work library



- c. Double click on Springbok.



- d. Check the dataset.

VIEWTABLE: Work.Springbok								
	Name	Surname	Province	Birthdate	Birthplace	Position	Height	Weight
1	Bakkies	Botha	B	22SEP1979	Newcastle	1	2.01	118
2	BJ	Botha	S	04JAN1980	Durban	2	1.82	116
3	Gary	Botha	B	12OCT1981	Pretoria	3	1.81	108
4	Schalk	Burger	W	13APR1983	Port_Elizabeth	4	1.93	110
5	Jean	de_Villiers	W	24FEB1981	Paarl	5	1.91	100
6	Bismarck	du_Plessis	S	22MAY1984	Bethlehem	3	1.89	112
7	Jannie	du_Plessis	C	16NOV1982	Bethlehem	2	1.88	116
8	Fourie	du_Preez	B	24MAR1982	Pretoria	6	1.82	89
9	Os	du_Randt	C	08SEP1972	Elliot	2	1.9	125

4.3 USE THE INFILE STATEMENT

This statement can be used when you have a dataset that is stored in a file outside of the SAS program (external file).

Consider the following example:

```
data rugby; ①
infile "c:\springbok.txt"; ②
input name$ surname$ province$ date : yymmdd10. place$ position
height weight; ③
run;

proc print data=rugby;
run;
```

- ① Give the dataset a name.
- ② Specify the exact location of the external file.
- ③ Specify the variables that are read in and give these variables names.

Exercise



Get the dataset on the 2007 springbok rugby world squad in SAS using three methods:

1. Enter directly into SAS. (You do not have to type it in, you can just copy from Excel and paste it in the SAS editor.)
2. Import the data from Excel.
3. Use the Infile statement.

5. CREATING A NEW LIBRARY IN SAS

Datasets are by default stored in the WORK library of SAS and they are temporary SAS datasets. On exiting the SAS program all the data in the WORK library is deleted. Other libraries can be created and used to store datasets permanently. The data in these libraries will not be lost when the program is terminated.

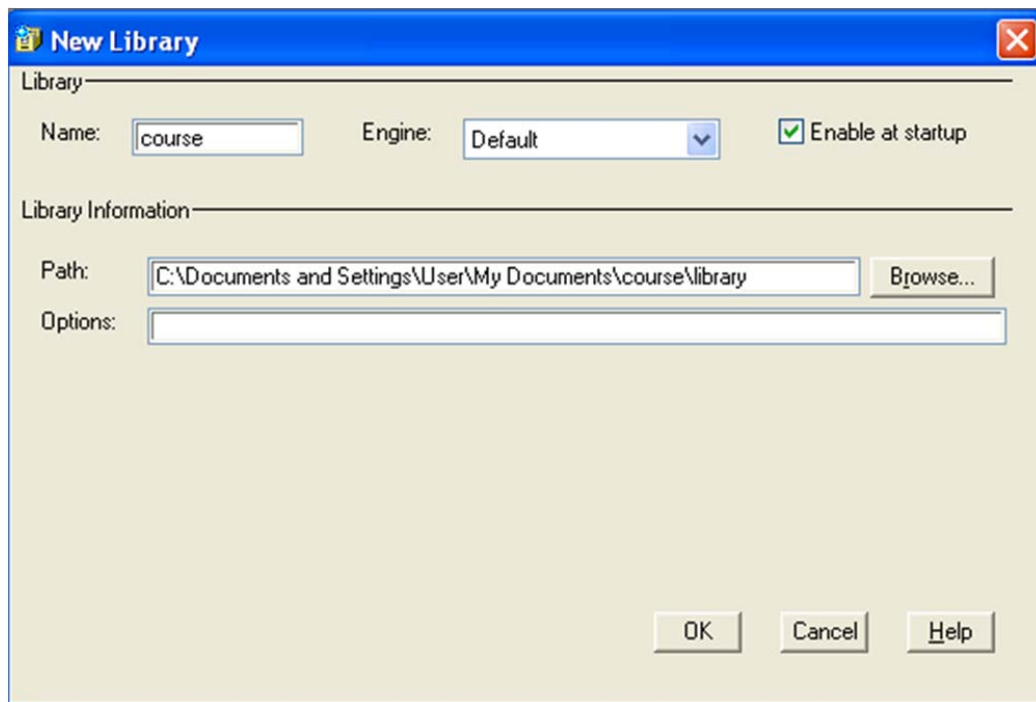
It is good practice to create libraries for every different task or assignment to minimize the time and avoid confusing of datasets.

Step 1: Create a directory, for example: C:\ My Documents\course\library. This will be used to store the data in. The name of this directory need not be the same as the name of the library.

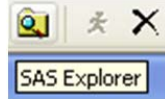
Step 2: Click on the New Library icon.



Step 3: Give a name to your library.
Tick the box with "Enable at startup".
Specify the path of the directory created in step 1.
Click **OK**.



Step 4: Click on the SAS Explorer icon and verify that your new library is there.



How to get your data into this new library when you

- **enter the data directly into SAS / use the infile Statement**

```
data course.rugby;
```

This statement creates a dataset named rugby in the course library.

- **import the data form Excel**

Select your new library in step 7 of the import procedure.

```
data rugby;  
set course.springbok;
```

This statement reads the data from the dataset springbok in the permanent library course and stores it in a temporary dataset rugby in the work library.

If you want the rugby dataset to be stored in the course library you need to prefix the dataset name by library name, for example `data course.rugby;`

6. PROC FORMAT and the FORMAT STATEMENT

`proc format` and the `format` statement is used to improve the readability of the output by giving descriptions to the codes of the categorical variables. This is achieved in two steps. The first step is to define the descriptions of the codes; this is done with `proc format` and it is positioned at the top of the SAS program, before the data step. The second step is to associate a defined format with one or more of the variables; this is done using the `format` statement.

The basic syntax needed to create a format that gives a description to use to print variable values, for example to print “sharks” instead of “s” in the sas output, is given below:

```
proc format <option(s)>;  
value <$>name <format-option(s)> value-range-set(s);
```

This is just a part of the syntax of `proc format` that we will use to describe the codes of the categorical variable. Everything between the brackets <> are optional. For this procedure you will have to write `proc format;` after that you will write `value`. The compulsory bit for the `value` statement is to give it a name and `value-range-set`. The syntax will be explained in more detail using an example.


```

proc format;
value ① position ②      1='Lock'           ③
                        2='Prop'
                        3='Hooker'
                        4='Flanker'
                        5='Centre'
                        6='Scrumhalf'
                        7='Flyhalf'
                        8='Wing'
                        9='Fullback'; ④

value $team ⑤          'B'='Blue Bulls'     ③
                        'C'='Cheetahs'
                        'L'='Lions'
                        'S'='Sharks'
                        'W'='Western Province';

/*****format statement in data step*****/
data rugby;
set springbok;
format province $team. position position. birthdate ddmmyy10.; ⑥

/*****format statement a proc step*****/
proc print data=rugby;
format province $team. position position. birthdate ddmmyy10.;
run;

```

- ① value gives more descriptive names to the codes of the variables.
- ② position is the name of the first format that we are creating.
This name can be the same as that of the variable or something different. This name cannot be the same as a format supplied by SAS. The name must be a valid SAS name. It is the best to give a name that is easy to associate with the variable that the format is intended for.
- ③ The value-range-set specifies one or more variable values and a description (character string) or an existing format.
The character codes on the left-hand side of = sign must be in single or double quotation marks.
- ④ The semicolon is only at the end of each value statement.
- ⑤ \$team is the name of the second format that we are creating.
For character codes the format name must start with a \$ sign.

After descriptions of the codes in `proc format`, it must still be allocated to the variables by using a `format` statement. A `format` statement starts with the word `format` followed by the name of the variable(s) followed by the format name to use for the preceding variables. There is a full stop after each format name. The `format` statement can appear in either the data step or in a `proc` step. If we place the `format` statement in the data step, SAS will use these formats in all `proc` steps that follow. If we place the `format` statement in the `proc` step, SAS will use these formats only for the specific procedure.

The syntax:

```
format variable-1 <. . . variable-n> format of variable-1. <. . . variable-n>;
```

A `format` statement is used to activate the formats.

- ⑥ Name one or more variables for SAS to associate with a format and specify the format name that must be used, followed by a full stop.
Take note that you can also use existing formats for example `ddmmyy10.`
Take note that groups of variables can share a format. In this case you will list all the variables, followed by the format name and full stop.

7. CREATING NEW VARIABLES

Sometimes it will be necessary to create new variables. Creating new variables forms part of the data step. In the following example, we will create two new variables. We will determine the body mass index for each player which is calculated using the formula $bmi = \frac{weight(kg)}{height^2(m^2)}$ and group the players according to their position into “forwards” and “backs” (this variable will be called `group`).

Forwards	Backs
Lock	Centre
Prop	Scrumhalf
Hooker	Flyhalf
Flanker	Wing
	Fullback

If you entered the data directly into SAS or if you used the `infile` statement (like in the example below) the new variables are created below the input statement. This make sense because you first tell SAS which variables you have (in the input statement) and then you use some of these variables to create new ones. The statements used to create the new variables are given in bold.

```

data rugby;
infile "c:\springbok.txt";
input name$ surname$ province$ date : yymmdd10. place$ position
height weight;
bmi=weight/height**2;
if position <= 4 then group = "forward";
else group = "backs";
run;

```

If you imported the data, your program will look as follows,

```

data rugby;
set springbok;
format province $team. position position. birthdate ddmmyy10.;
bmi=weight/height**2;
if position <= 4 then group = "forward";
else group = "backs";
run;

```

Always remember to print the data afterwards to make sure your new variables are there.

```

proc print data=rugby;
run;

```

Exercise



Determine the age of the players at the start of the rugby world cup competition on 7 September 2007. One way this can be established is with the `yrdif` function in SAS.

```
age=yrdif(date, '07sep2007'd, 'act/act');
```

This function returns the difference in years between two dates. The first argument is the starting date, for this we will use the birthdates of the players which is the variable `date`. The second argument is the end date, for this we will use the start of the world cup competition. The third argument has to do with the way SAS calculates the difference and there are different ways to choose from. Look in the SAS help function if you want more information.

Based on age create a new variable “agegroup”. If a person is 26 years old or younger you classify them as “young” otherwise they are considered “old”

8. PROC PRINT

This procedure is used to print the observations in a dataset. You can either use all the variables or just select some. It is always a good idea to print your dataset before you start with statistical procedures to make sure the data is read the way you intended.

The syntax:

```
proc print <option(s)>;  
by <DESCENDING> variable-1 <...<DESCENDING> variable-n><NOTSORTED>;  
id variable(s) <option>;  
sum variable(s) <option>;  
var variable(s) <option>;
```

Let us consider an example to illustrate the most popular statements:

```
proc print data=rugby;    ①  
var surname province height weight;    ②  
run;
```

① `proc print` is used to print the observations in the dataset.

The `data=rugby` option specifies the dataset to print.

If you omit the `data=` option, the last dataset used will be printed. It is a good habit to specify with each procedure the dataset that must be used.

② List the variables you want to print in the order that it must be printed.

If you omit this statement all the variables in the dataset will be printed in the order they were read.

Exercises



Exercise 1

Determine what the `by`, `id` and `sum` statements are used for. Take note that for the `sum` statement you can use the keyword `_numeric_` to refer to all numerical variables instead of listing them individually.

Exercise 2

Print the surname, position height and weight of the players in the rugby dataset separately for the different provinces. Replace the observation number next to each player with the players' name. Also print the total height and weight of all the players. Also remember to include the format for position and province in your program.

[Hint: you need to sort the data before you can use a `by` statement. Type "sort procedure" in SAS help and look for the syntax to find out how to sort.]

9. PROC UNIVARIATE

PROC UNIVARIATE gives a thorough set of descriptive statistics for continuous variables. A histogram can also be produced as well as PP plots. Parametric and nonparametric model estimation can also be done.

The basic syntax:

```
proc univariate <options>;  
var variable(s);  
run;
```

Let us consider an example:

```
proc univariate data=rugby normal; ①  
var height weight; ②  
run;
```

- ① The `data=rugby` option specifies the dataset to be used for this procedure. If you do not specify a dataset the last dataset will be used to calculate the descriptive statistics.

The option `normal` requests tests for normality of the variables.

- ② In the `var` statement you list the continuous variables for which the descriptive statistics must be calculated.

If this statement is omitted, descriptive statistics will be calculated for all numerical variables whether they are continuous or categorical. For example it will also calculate descriptive statistics for the date and position even though this does not make sense.

To construct a histogram for the variable height the following code is used:

```
proc univariate data=rugby normal;  
var height weight;  
histogram height; ③  
run;
```

- ③ Different options can be added for example to fit a normal curve on the histogram to assess the possibility of a normal distribution / `normal`, the midpoints can be specified / `midpoints = 5.6 5.8 6.0 6.2 6.4 etc.`

Go to SAS Help/SAS Products/SAS Procedures/Univariate/Histogram for other parametric and nonparametric estimation options.

Exercise



Exercise 1

Generate descriptive statistics for the weight of the rugby players. Include in your program

- an option that will generate a stem-and-leaf plot, a box plot and a normal probability plot.
- a statement that specifies that the name of the player must be next to the smallest and largest values to make it easy to identify.
- a statement that produces a histogram with a superimposed normal plot.

Exercise 2

Generate descriptive statistics for the weight of the rugby players for the two groups of players (backs and forwards) separately.

10. PROC MEANS

PROC MEANS can be used to determine some basic descriptive statistics of the continuous variables, for example the mean, standard deviation, etc. It is less powerful than PROC UNIVARIATE.

The basic syntax:

```
proc means <option(s)> <statistical keyword(s)>; ①  
var variable(s); ②  
run;
```

- ① If you do not specify any options or keywords `proc means` will calculate descriptive statistics for the last dataset. By default it will give the sample size, the mean, the standard deviation and the minimum and maximum values.

Examples of <options>:

specify the input dataset, `data=`

specify the number of decimal places to be displayed in the output, `maxdec=`

Example: `proc means data=bok maxdec=3;`

You may specify which statistics to compute and display in the output by using <statistical keywords>, for example:

sample size, `n`

average, `mean`

sum of observations, `sum`

Example: `proc means n mean sum;`

- ② This statement specifies continuous variables for which descriptive statistics must be calculated. If the statement is omitted, descriptive statistics will be calculated for all numeric variables, including the categorical variables.

Use the dataset “rugby” and create a new dataset “bok” which only contains the variables: name, province, position, height and weight.

Type the following SAS program and see how the output differs.

```
data bok;  
set rugby;  
keep name province position height weight;  
run;
```

```
proc means data=bok;  
var height weight;  
run;
```

```
proc means n mean sum maxdec=3;  
var height weight;  
run;
```

```
proc means data=bok;  
run;
```

Take note: When you do not specify the variables `proc means` also calculates descriptive statistics for position which is a categorical variable and therefore nonsensical.

Exercises



Exercise 1

Write a SAS program and use `PROC MEANS` to determine the sample size, the mean, the variance and the median for the weight of the 2007 world cup springbok squad.

Exercise 2

Redo exercise 1 but this time include a statement in order to get the summary statistics for the different provincial teams.

[Hint: You need to first sort the data according to province and then use `PROC MEANS`. Use SAS help.]



The following can help you a lot when you write a long program or when you write the program now and will look at it again in a month..

If you write a long program and only want to run a certain part, you can select just the part that you want to run and click on the run icon.

Another way to achieve this is to block out a certain part of the program. This is achieved by putting a `/*` before the part you want to block and then putting a `*/` at the end of the part you want to block out. When you do this these statements in SAS will become green, for example

```
/*  
proc means data=bok;  
var height weight;  
run;  
*/
```

This is also useful for writing comments in the program, for example

```
/* This program creates a new dataset */  
data bok;  
set rugby;  
keep name province position height weight;  
run;
```

If you want it to stand out you can even put it in a block,

```
/* -----  
| This program creates a new dataset |  
-----*/
```

Another way of making a comment is to start with a star and end with a semi-colon,

```
*****  
* This program creates a new dataset *  
*****;
```

11. PROC FREQ

This procedure is used to construct frequency tables for categorical variables and to do statistical analysis on such tables.

The basic syntax:

```
proc freq <options>;  
tables requests </options>;  
run;
```

Let us consider an example:

```
proc freq data=rugby; ①  
tables province province*group; ②  
run;
```


- ① If you omit the `data=` option, the last dataset is used to construct the frequency tables.
- ② After the `tables` statement you list your *requests*, in other words you list the variables that you want frequency tables of.

If you list a single variable a one-way frequency table is constructed.

If you list two variables with a asterisk (*) in between a two-way frequency table is constructed for the variables province (rows) by group (columns).

If the `tables` statement is omitted, one-way frequency tables will be constructed for all the variables, including the continuous variables.

It is possible to request more than one table in a single table statement, but it is also possible to have more than one table statement if you want to specify different options for the different tables.

There are various options available for the `tables` statement, for example `chisq` will compute the chi-square statistic for testing independence between two variables. You can also manipulate the contents of the frequency table by including, for example, percentages or omitting them. Take a look in the SAS help function to familiarise yourself with this.

Exercise



Exercise 1

Construct a one-way frequency table for the variable province. Include an option to omit the cumulative frequencies and percentages.

Exercise 2

Construct a two-way frequency table of agegroup (old vs young) by group (backs vs forwards). Include an option to calculate the chi-square test statistic to test for independence between the two variables. Each cell of your frequency table should include the following information:

- the cell contribution to the chi-square statistic
- the expected cell frequency under independence

Exercise 3



Sometimes you won't have the raw data, but instead it will already be summarised in a table, for example:

	Group	
Agegroup	Backs	Forwards
Old	5	13
Young	8	4

The following program can be used to enter this data into SAS:

```
data a;
input agegroup$ group$ count;
cards;
old backs 5
old forwards 13
young backs 8
young forwards 4
;
run;

proc print data=a;
run;
```

Now, construct a two way frequency table of agegroup by group and include an option to calculate the chi-square statistic.

[Hint: Look in the SAS help function for the appropriate statement in PROC FREQ.]

12. PROC GLOT

This procedure is used to plot data. PROC PLOT can also be used but the graphics are not that nice, it gives a text plot, therefore we will focus on PROC GLOT.

The following syntax will give you a basic plot of the variables height against weight.

```
proc gplot data=rugby;
plot height*weight;
run;
```

There are various options available to make the graph look nice. We will consider some of these options next, using an example.

```
options reset = all; ①
title1 'Scatter plot of height against weight'; ②
symbol1 i=r value=dot c=red line=2; ③
```

```
axis1 label=('Height in m'); ④  
axis2 label=('Weight in kg');
```

```
proc gplot data=rugby;  
plot height*weight ⑤ / vaxis=axis1 haxis=axis2; ⑥  
run;
```

- ① This statement specifies values for graphical options. It controls characteristics of the graph such as the size, font, etc. This statement can be anywhere in your program before the `gplot` procedure. The graphics options remain in effect until you use the `reset=` option or when you terminate SAS.

The `reset=` option delete all graphical options specified before.

- ② This statement specifies a title for the graph. It can also be used in other procedures, which will result in a title for the output. You can look in SAS help for various options to control the appearance of the title.
- ③ The `symbol` statement defines the appearance of the symbol used to plot the data. If you plot more than one graph on the same set of axis you can specify more than one symbol statement, the next one you will call `symbol2`.

The `i=r` is an interpolation option. This will fit a straight line to the data. Other interpolation options include `i=join`, this will join consecutive observations and is used for example with a time plot. Look in the SAS help for other options.

`value=dot` or `v=dot` specifies the plot symbol. Other symbols include square, circle, plus and many more.

`c=red` or `colour=red` specifies a colour for the whole definition, including symbol and the line.

`line=2` specifies the line type to be used. Option 1 will be a straight line and 2 – 46 will be different dashed lines.

- ④ The `axis` statement controls the location and the appearance of the axes in plot. Using this statement you can change the scale, the appearance, tickmarks, etc. of the axis.

The `label=` option modifies the axis label.

- ⑤ This will plot the variable before the asterisk on the y-axis and the variable after the asterisk on the x-axis. You can
You can request more than one plot in a single `plot` statement or you can have multiple `plot` statements in a single `proc gplot`.
- ⑥ The `vaxis=` and `haxis=` options assigns a definition to the axes.

Another useful option is `overlay` which places all the plots on the same set of axes.

13. PROC CORR

This procedure can be used to calculate the correlation or partial correlation between variables.

The basic syntax:

```
proc corr <options>; ①  
var variable(s); ②  
partial variable(s); ③  
run;
```

- ① Calculates the correlation coefficient and simple statistics for all the variables listed in the `var` statement.

Some of the options for this procedure are:

- `pearson` which calculates the Pearson correlation coefficient. If you do not specify any options this is the default.
- `spearman` which calculates the Spearman correlation coefficient which is based on the ranks (non-parametric)

- ② List the variables that you want to determine the correlation between.

- ③ Include this statement when you want to determine the partial correlation between the variables listed in the `var` statement, removing the effect of the variables listed in the `partial` statement.

Exercise



Compute the Spearman correlation coefficient between height and weight for the two groups (forwards and backs) separately.

14. PROC REG

`PROC REG` can be used to perform regression analysis. Regression analysis can also be performed using `PROC GLM`, but `PROC REG` has more options available that are useful in regression analysis.

The basic syntax:

```
proc reg <options>;  
id variable(s);  
model dependents=<regressors> </options>;  
output <out=sas-data-set> <keyword=names> <...keyword=names>;  
run;
```

Consider the following example,

```
proc reg data=rugby; ①  
id height; ②  
model weight=height / p; ③  
output out=regression p=yhat; ④  
run;  
  
proc print data=regression; ⑤  
run;
```

- ① There are various dataset options and display options available. For example, the
- `outtest=` option creates a dataset with the parameter estimates and model fit summary statistics
 - `corr` displays the correlation matrix of the variables listed in the `model` statement
 - `alpha=` sets the significance level for confidence intervals
- ② When the model statement options include `CLI`, `CLM`, `P`, `R`, etc. then the variable listed in the `id` statement are displayed next to each observation. This is useful to identify the observations. If this statement is omitted, the observation number will appear next to each observation.
- ③ The model statement defines the theoretical relationship between the dependent and explanatory variables. It is possible to have more than one model statement.

After the `model` statement you write the dependent variable on the left side of the `=` sign, followed by the explanatory variable(s) on the right. If you do not specify explanatory variables, only an intercept term is fitted.

The option `p` will compute predicted values and include, in the output, the observed, predicted and residual values for each observation.

The options for the `model` statement include options for model selection, fit statistics, dataset, regression calculation, details on the estimates, predicted values and residuals as well as display options. Refer to SAS help for more details.

- ④ This statement creates a new dataset containing information generated by the procedure.

The `out=` option specifies a name for the new dataset. In this example the name is "regression". You can choose your own name.

You need to specify at least one *keyword=names*. This option specifies the statistics to include in the output dataset and new names for the variables. Look in SAS help for the list of keywords.

In this example `predicted=` or `p=` stands for predicted values and after the `=` sign we gave a new name 'yhat' that will be in the output dataset.

- ⑤ The new dataset contains all the variables in the original dataset as well as the predicted value which we named yhat.

You are now able to use this dataset in other procedures.

Also look in the SAS explorer at your new data set.

Exercise



Exercise 1

Fit a regression model without an intercept term where weight is your dependent variable and height the explanatory variable. Include the following in your program:

- an option to compute 90% confidence limits for the individual predicted values
- create an output dataset that contains the predicted values and the residuals

Use the output dataset and

- test the residuals for normality
- plot the original data as well as the predicted value of weight against height on the same axes.

15. PROC TTEST

This procedure is used to perform a t-test based on the assumption of underlying normality. For a one sample t-test and a paired t-test `proc univariate` or `proc means` can also be used.

The basic syntax:

```
proc ttest <options>; ①  
class variable; ②  
paired variable(s); ③  
var variable(s) <options>; ④  
run;
```

- ① One of the options is to specify the value under the null hypothesis by using `h0=value` where you replace `value` with some number.
If you do not specify this value it will test the hypothesis against zero.

- ② The `class` statement specifies the name of the classification variable in the case of two independent sample t-test. The classification variable must have only two levels.

For a one sample and paired t-test this statement must be omitted.

- ③ The `paired` statement specifies the variables to be compared in a paired t-test.

The variables in this statement are separated by an asterisk (*) that compares each variable on the left with each variable on the right.

The `class` and `var` statements cannot be used with the `paired` statement.

- ④ The `var` statement specifies the variable(s) to use for the comparison of means.

If this statement is omitted, all the numerical variables are included in the analysis, excluding the ones specified in the `class` statement.

Exercise



Exercise 1

Use `proc ttest` to test the hypothesis that the height of rugby players is 1.8 metres.



This one sample t-test can also be performed using `proc univariate` or `proc means`. The important thing to remember is that these procedures will test the hypothesis against zero, therefore in order to test whether the height is 1.8 we need to create a new variable.

Create a new variable called `testheight` which will be the difference between the height of a player and 1.8.

Type the following program and compare the answers to what you obtained using `proc test`.

```
proc means data=rugby t prt;  
var testheight;  
run;
```

```
proc univariate data=rugby;  
var testheight;  
run;
```

Take note, for `proc means` we needed to add the options `t prt` while for `proc univariate` the t-test is part of the standard output.

Take note, for a paired t-test you will also create a new variable that is the difference between the two measures and then you test whether this difference is zero.

Exercise 2

Use `proc ttest` to test the hypothesis that the height of forward and the backs are the same.

16. PROC GLM

In an analysis of variance (ANOVA) we test whether the means of two or more groups are equal. For any ANOVA problem one can use either `proc glm` or `proc anova`. The general form for these two procedures is the same. This course will focus on `proc glm`. The reason is twofold, firstly `proc glm` can be used for an unbalanced design and secondly it is possible to create dummy variables for your categorical variable(s) and use them in your model statement.

Have a look at the syntax of `proc glm` in SAS help. This procedure can be used for a wide variety of things and instead of going into that we will rather consider `proc glm` using an example of a **one-way ANOVA**, completely randomized design where we want to compare the weight of the players for the different provinces.

The following program uses the categorical explanatory variables as defined in the dataset and uses a `class` statement to specify that these variables are categorical.


```
proc glm data=rugby;
class province; ①
model weight = province; ②
means province / scheffe lines cldiff; ③
run;
```

- ① The class statement is used to specify the classification variables. Here we will list our categorical explanatory variables.

These variables can have numeric or character values.

If you use a `class` statement it must appear before the `model` statement.

If this statement is omitted, `proc glm` will assume the explanatory variables listed in the `model` statement are numerical and therefore will see it as a regression model. Remember we mentioned before when we looked at `proc reg` that `proc glm` can be used to do regression

- ② In the model statement you specify the dependent variable = explanatory variables.
- ③ The `means` statement requests multiple comparisons. You only look at this part of the output if your ANOVA suggested that the means differ.

The `scheffe` option requests Scheffe's method of multiple comparisons. Look in SAS help for other methods of comparison.

The `lines` option lists the means in descending order and indicates groups that do not differ significantly from each other with the same letter of the alphabet.

The `cldiff` option request confidence intervals for the pairwise groups.



For a **two way ANOVA** you can list more than one categorical explanatory variable in the model statement as well as an interaction term. The output will include separate hypothesis for the main effects and the interaction effect.

The categorical variables can be transformed to continuous variables by making use of dummy variables. The 5 levels of province can be presented using 4 dummy variables, for example

	x_1^A	x_2^A	x_3^A	x_4^A
Blue Bulls	1	0	0	0
Cheetahs	0	1	0	0
Lions	0	0	1	0
Sharks	0	0	0	1
Western Province	-1	-1	-1	-1

This one-way ANOVA problem can then be written as a general linear model:

$$E(y) = \beta_0 + \beta_1^A x_1^A + \beta_2^A x_2^A + \beta_3^A x_3^A + \beta_4^A x_4^A$$

The following program will create a new dataset called “a” by using the “rugby” dataset and then create the dummy variables. The dummy variables will then be used in the `model` statement and the `class` statement will be omitted.

```
data a;
set rugby;
x1a=0;
x2a=0;
x3a=0;
x4a=0;
if province='B' then x1a=1;
if province='C' then x2a=1;
if province='L' then x3a=1;
if province='S' then x4a=1;
if province='W' then do;
    x1a=-1;
    x2a=-1;
    x3a=-1;
    x4a=-1;
end;

proc glm data=a;
model weight = x1a x2a x3a x4a;
run;
```

Compare the output of the two programs.

The first program will yield an overall test of significance as well as a significance test for each effect. For the one-way ANOVA these two tests are the same.

The second program gives an overall test for significance as well as the coefficients for the general linear model. For a randomized block or factorial design you can include a `contrast` statement to test the significance of the individual effects.

For our example include the following statement after the `model` statement and look at the output:

```
contrast 'pp' x1a 1, x2a 1, x3a 1, x4a 1;
```

This tests the hypothesis that $\beta_1^A = \beta_2^A = \beta_3^A = \beta_4^A = 0$ and in the output the contrast is named 'pp'. For the one-way ANOVA this test is the same as the overall test for significance because there is only one effect in the model.

17. Introduction to PROC IML (Interactive Matrix Language)

Let us assume that we have 3 matrices,

$$\text{Matrix } A = \begin{pmatrix} 3 & 1 & 1 & 0 \\ 4 & 2 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 14 \\ 10 \\ 15 \\ 0 \end{pmatrix} \text{ and } C = (1 \quad 1 \quad 2 \quad 3),$$

Syntax

Always start the program by writing “`proc iml`” on top, and finish the program off with a “`quit`” statement.

```
proc iml;
(statements)
quit;
```

How to enter a matrix

It is important to notice the format of writing down the matrices!

- We must always give our matrix a name (for e.g. **A**) then set it equal = and write out the rows and columns
- Start the matrix by typing {
- Write the first row and remember to finish off with a comma ,
This comma indicates that row 1 is finished and all numbers written after the comma indicate the numbers for row 2 and so on.
- After the whole matrix has been typed in, close the matrix with } and ;

How to print a matrix

Use the print statement and specify the names of the matrices you want to print next to it. If you separate the names using a space it will be printed next to each other, with a comma underneath each other and with a slash (/) it will print it on different pages.

Example 1

Define the matrices A, B and C and print them.

```
proc iml;  
a = {3 1 1 0,4 2 0 1,1 0 1 0,1 0 0 1};  
b = {14,10,15,0};  
c = {1 1 2 3};  
print a b c;  
quit;
```

Now we are ready to start various operations with the matrices that we have defined above.

Consider the following matrices $A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$, $B = \begin{pmatrix} 5 & 2 \\ 1 & 4 \end{pmatrix}$

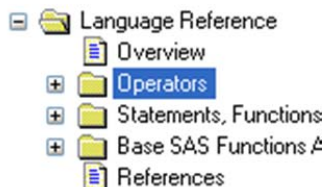
Operators

Operator	Symbol	Function	Example
Addition	+	Adds corresponding elements of matrices	A+B
Subtraction	-	Subtracts corresponding matrix elements	A-B
Matrix multiplication	*	Performs matrix multiplication	A*B
Matrix power operator	**	Raises a matrix to a power (matrix multiplied with itself 'power' times)	A**2
Division	/	Divides each element of first matrix by the corresponding element of the second matrix / divides each element of a matrix by a scalar	A/B A/2
Elementwise multiplication	#	Multiplies the corresponding elements of matrices / multiplies each element with a scalar	A#B 5#B
Elementwise power operator	##	Raises each element of the matrix to the specified power	A##2
Horizontal concatenation		Joins two matrices horizontally	A B
Vertical concatenation	//	Joins two matrices vertically	A//B
Transpose	` (below Esc)	Transpose a matrix (exchange the rows and columns of a matrix)	B`

For more information and examples of these operators access the HELP function, select the "Contents" tab and click "SAS Products"



In the drop down list click on  and then click on “Language Reference” and select “Operators”



Example 2

Define the matrices A, B and write an IML program where you use the different operators. Study the output to make sure you understand each operator.

```
proc iml;
a={1 2,2 3};
b={5 2,1 4};
add=a+b;
sub=a-b;
mult=a*b;
mpower=a**2;
div1=a/b;
div2=a/2;
emult1=a#b;
emult2=5#b;
epower=a##2;
hcon=a||b;
vcon=a//b;
trans=b`;
print a b, add sub, mult mpower, div1 div2, emult1 emult2, epower,
hcon vcon, trans;
quit;
```

Functions

Here are some useful functions. For more functions and information click on “Overview” where they are grouped according to functionality or click on “Statements, Functions, and Subroutines” which provides an alphabetical list.



Function	Description
DET(<i>matrix</i>)	Computes the determinant of square <i>matrix</i>
DIAG(<i>argument</i>)	If <i>argument</i> is a square matrix it creates a matrix with diagonal elements equal to the corresponding diagonal elements. If <i>argument</i> is a vector it creates a matrix with diagonal elements that are the values in the vector.
I(<i>dimension</i>)	It creates an identity matrix of size <i>dimension</i>
INV(<i>matrix</i>)	It computes the inverse of a square nonsingular <i>matrix</i> .
J(<i>nrow,ncol,val</i>)	Creates a matrix of identical values with <i>nrow</i> rows, <i>ncol</i> columns filled up with <i>val</i> .
MAX(<i>matrix</i>)	Returns the largest value in the <i>matrix</i> .
MIN(<i>matrix</i>)	Returns the smallest value in the <i>matrix</i> .
NCOL(<i>matrix</i>)	Finds the number of columns of <i>matrix</i> .
NROW(<i>matrix</i>)	Finds the number of rows of <i>matrix</i> .
SQRT(<i>matrix</i>)	Calculates the square root of each element of <i>matrix</i> .
SUM(<i>matrix</i>)	Adds up all the elements of <i>matrix</i> .
VECDIAG(<i>matrix</i>)	Creates a column vector of the main diagonal elements of a square <i>matrix</i> .

Subsetting matrices

Suppose you want to create a new matrix or vector from an existing matrix.

Consider the following matrix $C = \begin{pmatrix} -1 & 2 & 0 \\ 2 & 4 & 5 \\ 3 & -2 & 1 \end{pmatrix}$

We give our new matrix a name (e.g. C1). We specify the existing matrix, and in squared brackets we specify the specific rows and columns. The format in the square brackets is as follows [begin row : end row , begin column : end column]. Here are some examples:

c1=c[2:3,2:3]; (a 2x2 matrix with the last two rows and last two columns)
c2=c[,1]; (a column vector with all the rows of the first column)
c3=c[2,]; (a row vector with all the columns of the second row)

Therefore $c1 = \begin{pmatrix} 4 & 5 \\ -2 & 1 \end{pmatrix}$, $c2 = \begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix}$ and $c3 = (2 \quad 4 \quad 5)$.

You can also use sub setting to add the elements of the rows, columns or matrix for example:

c4=c[+,]; (add up the elements of the rows)
c5=c[,+]; (add up the elements of the columns)

$c6=c[+]$; (add up all the elements in the matrix)

Therefore $c4 = (4 \ 4 \ 6)$, $c5 = \begin{pmatrix} 1 \\ 11 \\ 2 \end{pmatrix}$ and $c6 = 14$.

How to create a matrix from a SAS dataset: READ Statement

SAS Code	Explanation	SAS Output
<pre>data a; input x y; cards; 1 12 3 15 6 19 ; proc print data = a; run;</pre>		<pre>Obs X Y 1 1 12 2 3 15 3 6 19</pre>
<pre>proc iml; use a; read all into zzz; print zzz ; run;</pre>	<pre>use: opens a SAS dataset for reading read: reads observations from a dataset all: all observations zzz: name of matrix</pre>	<pre>ZZZ 1 12 3 15 6 19</pre>

How to create a SAS dataset from a matrix:

SAS Code	Explanation	SAS Output
<pre>proc iml; b= {5 10 , 6 8, 7 4}; print b ; cn = {'x' 'y'}; create myb from b[colname=cn]; append from b; run;</pre>	<p>You create a dataset called "myb" from matrix "b" and give the columns names by specifying them in "cn".</p> <p>Note "b", "cn" and "myb" are names that you can choose yourself.</p>	<pre>B 5 10 6 8 7 4</pre>
<pre>proc print data = myb; run;</pre>		<pre>Obs x y 1 5 10 2 6 8 3 7 4</pre>

Exercises



Exercise 1: Calculate descriptive statistics

Use the “rugby” dataset and create a column vector called ‘x’ of the weight of the players.

[Hint: Look at the “read statement” in SAS help to find a way to select a specific variable to read into the vector.]

Use `proc iml` to calculate the following descriptive statistics:

[Hint: Try to write the program in a general form so that if you use another dataset you will also be able to get all these statistics without changing the program.]

Statistic	Instruction / hints
mean	$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ <p>To calculate the mean you need the sample size as well as the sum of the observations / elements of the matrix.</p> <p>The sample size will be the number of rows in the matrix. Look in SAS help to find a function that will determine this.</p> <p>Use three different methods to determine the sum of the observations:</p> <ol style="list-style-type: none"> 1. sum function 2. subsetting 3. matrix multiplication (How will you add up the elements of a column vector using matrix multiplication?)
variance	$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$ <p>Always think what you have and what you need and keep in mind the dimensions of the matrices you are working with.</p> <p>You have: $x: n \times 1$ with the observed values of weight.</p> <p>You need to subtract the mean value from each of the observations. The mean value is a scalar i.e <i>mean</i>: 1×1. Calculate $x - \text{mean}$ and see if you get the correct answer. Even though you get the correct answer you need to be careful, if you did this manually you need matrices of the same dimensions in order to subtract them from one another. Rather get in the habit to also do it in SAS.</p> <p>Therefore, first create an $n \times 1$ matrix with the mean value using the J function.</p>

	Calculate $\sum_{i=1}^n (x_i - \bar{x})^2$ using two methods: 1. square each observation using the appropriate operator and then determine the sum 2. matrix multiplication
standard deviation	
range	
median	Arrange the data in ascending order (smallest to largest value). [Look at <code>sort</code> call in SAS help] For an odd number of observations, the median is the middle value. For an even number of observations, the median is the average of the two middle values. Determine the position of the median. $\lceil \frac{n+1}{2} \rceil$ Determine the median.

Verify all your answers using `PROC MEANS`.

Construct a table to summarise all the calculated statistics.

First initialize your table. You want a table with 1 row and 6 columns and you can fill it up with any number, because you will overwrite it. Use the `J` function.

Example: `table=J(rows, columns, elements)`

Now you can change each of the elements of your table by assigning a value to it, for example the element in the first row and first column must be equal to `n` (sample size).

Example: `table[1,1]=n;`

If you want a certain element to be blank you can assign a `."` to it.

Example: `table[3,5]=.;`

To give your columns and rows headings you can use the following.

```
cn={"n" "mean" "variance" "standard deviation" "range" "median"};
rn={"weight"};
print table[colname=cn rowname=rn];
```

Your final answer should be similar to this:

	table					
	n	mean	variance	standard deviation	range	median
weight	31	104.19355	143.82796	11.992829	43	107

Exercise 2: Calculate descriptive statistics

Repeat Exercise 1 but this time for height and weight simultaneously i.e. for each of the descriptive statistics your answer should be a vector. Try to use as few steps as possible. You can only use one method to calculate the mean and the sum of squares.

[Hint: For the variance, it is easier to first determine the covariance matrix and then the diagonal elements will give the variances.]

Exercise 3: Correlation coefficient

Use `proc iml` to calculate the correlation coefficient between the height (x) and weight (y) of the rugby players.

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

[Hint: First determine the matrix of variances and covariances.]

Exercise 4: Regression

Regression can also be done using matrices.

Consider a multiple regression model:

$$Y_1 = \beta_0 + \beta_1 X_{11} + \beta_2 X_{21} + \beta_3 X_{31} + u_1$$

$$Y_2 = \beta_0 + \beta_1 X_{12} + \beta_2 X_{22} + \beta_3 X_{32} + u_2$$

...

$$Y_n = \beta_0 + \beta_1 X_{1n} + \beta_2 X_{2n} + \beta_3 X_{3n} + u_n$$

This can be written in matrix form:

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix} = \begin{pmatrix} 1 & X_{11} & X_{21} & X_{31} \\ 1 & X_{12} & X_{22} & X_{32} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & X_{1n} & X_{2n} & X_{3n} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} + \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$$

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{u}$$

We are not going into the derivation of the following results, but we will need them in our `proc iml` program.

Parameter estimates: $\hat{\mathbf{b}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$

Variance covariance matrix of the parameter estimates: $\mathbf{var} - \mathbf{cov}(\hat{\mathbf{b}}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}$

An unbiased estimator for σ^2 : $\hat{\sigma}^2 = \frac{\hat{\mathbf{u}}' \hat{\mathbf{u}}}{n-k}$ where $\hat{\mathbf{u}} = \mathbf{y} - \mathbf{X}\hat{\mathbf{b}}$, n is the sample size and k is the number of explanatory variables including the intercept (i.e. the number of columns in \mathbf{X})

The coefficient of determination: $R^2 = \frac{\hat{\mathbf{b}}' \mathbf{X}' \mathbf{y} - n \bar{y}^2}{\mathbf{y}' \mathbf{y} - n \bar{y}^2}$

The overall test for significance: $F = \frac{R^2 / (k-1)}{(1-R^2) / (n-k)}$

Use `proc iml` to estimate the following regression model.

$$Y = \beta_0 + \beta_1 X + u$$

where Y =weight and X =height of the rugby players.

Calculate:

- (a) The estimated regression model.
- (b) $\hat{\sigma}^2$
- (c) R^2
- (d) F test for overall significance including the p-value

PROBF Function

Syntax: `probf(f, ndf, ddf)`

`f` is a numeric random variable that is greater than or equal to zero

`ndf` is a numeric numerator degrees of freedom parameter

`ddf` is a numeric denominator degrees of freedom parameter.

The `probf` function returns the probability that an observation from an F distribution, with numerator degrees of freedom `ndf`, denominator degrees of freedom `ddf` is less than or equal to `f`.

- (e) t test for individual parameter estimates including the p-value

PROBT Function

Syntax: `probt(t, df)`

`t` is a numeric random variable.

`df` is a numeric degrees of freedom parameter

The `probt` function returns the probability that an observation from a Student's t distribution, with degrees of freedom `df`, is less than or equal to `t`.

Exercise 5: Regression with a dummy variable

Use `proc iml` to estimate the following regression model.

$$Y = \beta_0 + \beta_1 X + \beta_2 D + u$$

where Y =weight, X =height and D is a dummy variable indicating whether they are “forwards” or “backs”.

Create a dummy variable with $D = \begin{matrix} 0 & \text{Forwards} \\ 1 & \text{Backs} \end{matrix}$.

(You can create this in the data step before you start with `proc iml`.)

Calculate:

- (a) The estimated regression model.
- (b) Use an F test to test the hypothesis:

$$H_0: \beta_1 = 0$$

$$H_A: \beta_1 \neq 0$$

Include the p-value.

Hint: You can use restricted least squares to test this hypothesis using the following formula

$$F = \frac{(R_{UR}^2 - R_R^2)/m}{(1 - R_{UR}^2)/(n - k)}$$

For restricted least squares you have an unrestricted model which includes all the variables in the regression model. For the restricted model you apply your restriction, i.e null hypothesis. In this example we will have:

Unrestricted model (UR): $Y = \beta_0 + \beta_1 X + \beta_2 D + u$

Restricted model (R): $Y = \beta_0 + \beta_2 D + u$

Number of restrictions: m (This is basically the number of things that you are setting to zero)

Number of observations: n

Number of parameters in the unrestricted regression: k



This restricted least square method can also be used to test specific effects in the ANOVA model.

Exercise 6: Generating data using a do loop

Use `proc iml` to

- (a) Generate a matrix $X: n \times k$ with $n = 10$ and $k = 2$. The first column of X is a column of ones, and the second column is a variable from a UNIF(100,150) distribution.

Create the matrix X with two columns. The first column must be filled with ones, while the second column can be filled with anything because we will overwrite it using the do loop.

You want to change the elements of the second column of matrix X . For each of the rows in the second column you want to generate an observation from the uniform distribution. You

will do it for the first row, then the second row and so on up to the last row. For this you need the do loop.

Start and end off your do loop as follows:

```
do i=1 to n;
```

```
end;
```

Inside the do loop you will specify that you want to change row *i* of column 2 and assign to it an observation generated from the uniform distribution. This can be achieved by:

```
x[i,2]=ranuni(0);
```

The `ranuni` function generates a number from the uniform distribution on the interval (0,1). You want a number between 100 and 150, so you will have to change it a bit.

(b) Generate a set of theoretical error terms ($u: n \times 1$) from a $N(0,4)$ distribution.

(c) Calculate the following function: $Y_i = 20 + 2X_i + u_i$

You can (b) and (c) in the same do loop that you started with in (a).

(d) Estimate the regression model $Y = \beta_0 + \beta_1 X + u$

(e) Include another do loop to repeat this whole process 5 times.

(f) Finally, change your program in such a way that you summarise the parameter estimates of the 5 repetitions in one matrix. The first column must contain the intercept and the second column the slope.



Take note: You can also generate data in the SAS data step. Enter the following program and look at the output

```
data a;  
n=10;  
do i=1 to n;  
x=ranuni(0);  
output;  
end;  
  
proc print data=a;  
run;
```

The position of the output statement is very important. If you put the output statement outside the do loop you will only get one value for *x* because it will write over the value the whole time.

(g) Repeat (a) to (d) using the data step and `proc reg`. Compare your answers.

SUMMARY OF SAS PROCEDURES

What you want to do	Procedure to use
Calculate simple descriptive statistics (Mean, median, variance, percentiles)	PROC MEANS, PROC UNIVARIATE
Construct a frequency distribution for categorical variables	PROC FREQ
Construct a histogram	PROC UNIVARIATE
Construct a PP Plot / QQ Plot	PROC UNIVARIATE
Do parametric (Normal, Beta, Gamma etc.) and nonparametric (Kernel etc.) density estimation	PROC UNIVARIATE
Test for normality	PROC UNIVARIATE
Tests for location (Students' t, Sign, Signed Rank)	PROC UNIVARIATE
Tests for location when normality is assumed (One sample, independent samples, paired sample)	PROC TTEST
Test for equality of variances when normality is assumed	PROC TTEST
Test hypotheses regarding the linear relationship between two variables	PROC CORR
Fit a regression model between variables	PROC REG, PROC GLM
One-way ANOVA	PROC GLM, PROC ANOVA
Two-way ANOVA	PROC GLM, PROC ANOVA

NB. A **nonparametric alternative** to all the hypotheses tests mentioned in the table can be obtained in PROC NPAR1WAY.

PROC NPAR1WAY performs tests for location and scale differences based on the following scores of a response variable: Wilcoxon, median, Van der Waerden (normal), Savage, Siegel-Tukey, Ansari-Bradley, Klotz, Mood, and Conover. Additionally, PROC NPAR1WAY provides tests that use the raw input data as scores. When the data are classified into two samples, tests are based on simple linear rank statistics. When the data are classified into more than two samples, tests are based on one-way ANOVA statistics. Both asymptotic and exact *p*-values are available for these tests. PROC NPAR1WAY also provides Hodges-Lehmann estimation, including exact confidence limits for the location shift.

ASSIGNMENT

This assignment is due on 6 February 12h00 on ClickUp. NO EXCEPTIONS will be made.

Import the dataset IQ.xlsx into SAS that is available on ClickUp in the SAS folder. This dataset contains some information of 10 participants in a study to establish if there is a relationship between IQ and physical characteristics.

Answer the following questions on the ClickUp test "Assignment" in the SAS folder.

1. Calculate the mean IQ amongst the group of participants.
2. Calculate the variance of the variable age.
3. Test at a 5% level of significance if the variable IQ are normally distributed (Also construct a histogram with a normal curve overlayed for visual confirmation).
4. Test at a 5% level of significance if the average of the variable IQ differs significantly from 125. Which procedure did you use?
5. Construct a new variable "Heighttype" classifying participants into short and tall, where short is a height of 1.7m or less and tall is a height of more than 1.7m. Construct a frequency distribution of the variable Heighttype and give the frequency for the category "Short".
6. Test at a 1% level of significance if the average Eye width differs significantly between the Short participants and the Tall participants (Use 5).
7. Test if there is a significant linear relationship between Age and Eye width. Give the p-value associated with this test.
8. According to 5, is it statistically plausible at a 1% level of significance to say that the older you get, the wider eyes you will have. What about logically?
9. Test at a 5% level of significance if the variable gender has any effect on the IQ. Give the test statistic value calculated to perform this test.

Construct the following matrices in SAS and use PROC IML to answer the following questions:

$$A = (\text{Age}_{10 \times 1} \quad \text{IQ}_{10 \times 1}), B = \begin{pmatrix} \text{Eye width}_{1 \times 10} \\ \text{Height}_{1 \times 10} \end{pmatrix}$$

10. What are the dimensions of $A \cdot B$?
11. What are the dimensions of $A \# B$?
12. Give the 2nd row total of A.
13. Concatenate A and B` horizontally and calculate the largest entry in the 8th row.
14. Estimate β in the regression model $IQ_{1 \times 10} = (\alpha \quad \beta)B_{2 \times 10} + \varepsilon$

REFERENCES

Cody, R.P. & Smith, J.K. 2006. *Applied Statistics and the SAS Programming Language*, 5th ed.

Elliot, R.J. 2000. *Learning SAS in the Computer Lab*, 2nd ed.

SAS Help function.

Course notes of undergraduate subjects presented by the Department of Statistics.

Adamski, K. 2014. *Introductory SAS course notes*.