Four Alternate Software: Training, Comparison and Review

Written and complied by Stacey Llewellyn, Satomi Okano, Mandy Way and Lachlan Webb

Contents

1.0 Introduction

This document aims to show how basic database practises can be completed in four different statistical software: SPSS, Stata, SAS and R. We are hoping that this will be useful for beginners who want to learn a new statistical software, for those considering trying new statistical software, or for those interested in understanding the differences between these software.

For all the software described, a large variety of analyses can be completed very easily once the basics of how the software works have been mastered. However, each software still relies on the user's knowledge to choose appropriate analyses for the data, so caution should be used when attempting any analyses for this reason. If you do not have sufficient statistical training to check what is required for analyses to be valid, or if in doubt, please contact the statistics department for help.

1.1 Basic outline

Each of the following sections covers a different area: basic data manipulation, presenting and summarising data, and simple statistical tests and models. In each section, you will find the syntax (and for some software, drop down menu options) for each process. Every section is presented for each of the four software in turn. So for example, if you need to summarise and present your data, you can go straight to the section ***Presentation and Summary of Data***. From there, you can choose the software you want to use and refer to the relevant subsection.

If you want to compare different software for what seems easier or most understandable for you, each software subsection in the section presents the same processes for each software. In each ***Notable differences*** subsection, some of the main inconsistencies and differences between the four software are noted, such as default settings, underlying methods, or definitions used. If you are having to replicate someone else's output and using a different software to them, it will be useful to read this section to ensure consistency.

To easily identify and distinguish between the text, syntax and output examples in this report, syntax will be outlined with a blue box, drop down menu with a grey box, and output except tables will be outlined with a black box. The drop down and syntax section will still be labelled to show the difference if this document is printed in black and white.

1.2 Dataset

The data and data dictionary used in this document can be downloaded for free after registration with John Snow Labs: https://www.johnsnowlabs.com/marketplace/dobutamine-stress-echocardiography-data/ You will need to unzip the downloaded csv file.

The dataset is from a study that investigated which measures during a stress echocardiography test were most helpful in predicting a cardiac event in a patient.

Changes to the data:

Rename variables by manually typing the New Variable Names as below and save the file as an excel file (.xlsx) called cardiac data:

Column in csv	Variable Name in csv	New Variable Name	Explanation
A	Subject	id	
N	Age	age	PATIENT'S AGE
O	Gender	gender	PATIENT'S GENDER (male = 0, female =1)
B	Basal_Heart_Rate	bhr	BASAL HEART RATE
C	Basal_Blood_Pressure	basebp	BASAL BLOOD PRESSURE
D	Product_of_Basal_Heart_Rate_Basal_Blood_Pressure	basedp	BASAL DOUBLE PRODUCT (= bhr x basebp)
E	Peak_Heart_Rate	pkhr	PEAK HEART RATE
F	Systolic_Blood_Pressure	sbp	SYSTOLIC BLOOD PRESSURE
G	Product_of_Peak_Heart_Rate_Systolic_Blood_Pressure	dp	DOUBLE PRODUCT (= pkhr x sbp)
T	Is_Positive_Stress_Echocardiogram	posse	STRESS ECHOCARDIOGRAM WAS POSITIVE (0 = yes, 1=no)
U	Is_New_Myocardial_Infarction	newmi	NEW MYOCARDIAL INFARCTION, OR HEART ATTACK (0 = yes, 1=no)
V	Is_Recent_Angioplasty	newptca	RECENT ANGIOPLASTY (0 = yes, 1=no)
W	Is_Recent_Bypass_Surgery	newcabg	RECENT BYPASS SURGERY (0 = yes, 1=no)
X	Is_Death	death	THE PATIENT DIED (0 = yes, 1=no)
AE	Is_Any_Event	anyevent	THIS IS THE OUTCOME VARIABLE. IT IS DEFINED AS "death OR newmi OR newptca OR newcabg". IF ANY OF THESE VARIABLES IS POSITIVE (= 0) THEN "ANY EVENT" IS ALSO POSITIVE (= 0).

Select the column "Is_Resting_Wall_Motion_Abnormality_Seen" and use the find and replace function in Excel and find TRUE and replace with 0 and find FALSE and replace with 1.

Then select all of the data and using the find and replace function in Excel, find TRUE and replace with 1 and find FALSE and replace with 0.

Select the gender variable using the find and replace function in Excel, find male and replace with 0 and find fe0 and replace with 1.

You will also need to create the append 1 and 2 datasets and the merge 1 and 2 datasets in Excel:

Cardiac data_append1: Contains all data for subjects 1 to 200 only

cardiac data_append2: Contains all data for subjects 201 to 558 only

cardiac data_merge1: Contains data for all subjects but only for columns A to S

cardiac data_merge2: Contains data for all subjects but only for columns A, T to AF

The analyses performed in this document are examples only. Choices of analyses were based on data types rather than specific study questions.

1.3 Software

Stata

Stata is an integrated statistical analysis package designed for research professionals.

Stata has five main windows:

1. Command (at the bottom): where you type your command
2. Large window above command: output will be displayed
3. Review (on the left): keep track of the commands used
4. Variables (on the top right): list variables in your dataset
5. Properties (bottom right): displays properties of variables and dataset

You can resize, close or move around the windows as you wish.

* Please note that for Stata versions up to and including 15 you can open only one dataset at a time. You will need to open multiple Stata windows to work with multiple datasets.

Stata is fast and easy to use with a drop-down menu and syntax (command) and it is an excellent software for data manipulation, statistics and visualisation. Stata's syntax is intuitive and easy to learn. Drop-down menu allows beginners to perform analysis and learn the Stata command language (Stata automatically provides the corresponding command in the Review window). You can type, run and save commands in a do-file.

Typical Stata syntax is:

command *varlist* [if *exp*] , *options*

where command denotes Stata command, *varlist* denotes a list of variable names, *exp* denotes an algebraic expression and *options* denote a list of options. Please note that Stata uses a double equal sign (==) to denote equality testing after "if".

Stata's help system provides command information and useful links on syntax rules, drop-down menus, examples and video examples. To get information on a command, type and run "help *command_name*" in the command window or use drop-down menu "Help".

SPSS

SPSS is a user-friendly statistical software that is very easy to learn, with analyses performed via drop down menus, making it perfect for beginners. Both data and variable information are readily accessible in the data editor. Multiple datasets can be opened at one time; however, only one dataset can be "active" at a time. All analysis syntax and results of analysis are available in an "output" file, which can be saved for reference, and also allows further editing. Syntax for analysis performed can also be saved into a syntax file to repeat any analysis in the future via a "paste"

function. Analysis can also be performed solely by syntax, without the use of the drop-down menu but this will not be a focus of this document.

Caution: SPSS does not produce the most aesthetically pleasing graphs (on a scale of Jackson Pollack to da Vinci oil painting, it is a firm kindergarten drawing. Not a bad kindergarten painting. Like, you would be happy to hang it on the fridge when your friends come over and say that your child will be a professional artist one day, but you will throw it out after a couple of weeks).

SAS

SAS is a programming language. Firstly we will establish its main features.

There are three main windows in SAS:

1. Program editor: where you write the command lines/statements
2. Log window: to check if SAS has executed the commands and if there are any errors
3. Results window: displays the results from procedure.

The SAS code (syntax) consists of statements. Statements mostly begin with a keyword, and they ALWAYS end with a SEMICOLON. The semicolon signifies the end of the statement. Examples of keywords are: data, set, run, where, var, proc….

SAS is a programming language which consists of two kinds of SAS statements that you will write in your program: data statements and PROC (short for procedure) statements. Data statements are used to read in the data, inform the computer about the attributes of the dataset and the variables. It is also used to manipulate variables, create new ones and alter datasets (Eg1).

The PROC statements are used to display the data (Eg2), sort the data, perform statistical analyses and produce graphs.

```
Eg1     data fc2; set fc21;
        where age>=40;
        run;
Eg2     proc print data=fc2;
        var id age;
        run;
```

R

R is a statistical programming language, so use of R is coding heavy and relies on functions. The function syntax of R generally has the form:

```
function(input_1 = inp1, input_2 = inp2), or
function(inp1, inp2)
```

The function has a name that is case sensitive, which is followed by round brackets which contain all the inputs. Each input has a name, and can be given a value using the '=' sign. Inputs can also be put in the brackets without naming the input as long as they are in the correct order.

Data sets are stored as data objects in R. Each data object and each variable need to have a name. A variable can be accessed from a data object using a '$' sign (e.g. cardiac$id, variable "id" from

"cardiac" data object). Specific values can be accessed from a variable or a dataset using row and column numbers in side square brackets. When accessing parts of a dataset (e.g. multiple columns of specific rows), specify the row number/s first, then place a comma, and follow by the column number/s. If all rows or all columns are needed, the comma is still used. For example:

cardiac[1,4]	first row, fourth column value
cardiac[3:6,2:4]	rows 3 to 6, columns 2 to 4
cardiac[c(1,3,5),c(2,4)]	rows 1, 3 and 5, columns 2 and 4
cardiac$age[11:20]	age row 11 to 20

Function output, objects and other values are assigned using either '<-' or '='. For example:

```
number <- 2+2

MyAverage <- mean(cardiac$age)
```

This document will use the convention of data$variable when using R. It is possible to use the attach() function which 'attaches' a dataset in R, meaning variables can be used just by naming the variable without specifically mentioning the dataset. The dataset can then be 'detached' using the 'detach()' function. The use of these two functions is not advised if using more than one dataset at a time, or data object, as it can cause unforeseen issues with variables names being the same between datasets and forgetting to detach datasets.

It is advised to use R through the graphical user interface RStudio. Once you have downloaded both R and RStudio, you only need to open and work in RStudio. The main features of RStudio include the window for visualising data and working on script files (you can have many open at once), the console window where all results and messages appear, the environment/history window where you can see all objects in your work environment, and the last window that manages plots, packages and the help function (as well as other things).

See the R Companion booklet provided by the Statistics Unit for more detailed instruction on how to use R.

1.4 Version

- Stata 15.0 (StataCorp. 2017. Stata Statistical Software: Release 15. College Station, TX: StataCorp LLC)
- IBM SPSS version 22 (IBM Corp. Released 2013. IBM SPSS Statistics for Windows, Version 22.0. Armonk, NY: IBM Corp.)
- SAS version 9.4
- R version 3.5.0 ("Joy in Playing") was used for the R sections of this report. RStudio version 1.1.447 was used.

JMP is another software that is available and has only been included for comparison in sections 1.5 through 1.7.

1.5 Software Comparisons

General					
	STATA	**SPSS**	**SAS**	**R**	**JMP**
Price	$$	$$	$$$$	Free	$$$ (free to QIMR)
Mac/Windows	Both	Both	Windows only	Both	Both
Interface	Menus & Syntax	Menus & Syntax	Syntax	Syntax	Menus & Syntax
Learning Curve	Moderate	Gradual	Steep	Steep	Gradual
Data Manipulation	Strong	Moderate	Very Strong	Very Strong	Very Strong
Drop-down menu	✓	✓	X	X	✓

Syntax					
	STATA	**SPSS**	**SAS**	**R**	**JMP**
Case sensitive?	Yes	Yes	No	Yes	Yes (mostly)
Line breaks	*1st line /// 2nd line* or *1st line /* */ 2nd line*		Commands need to be separated by a semi-colon	1st line + 2nd line	Commands need to be separated by a semi-colon
Comments	**comments* or *//comments* or /*comments */	/*	/*comments */	# comments	*//comments* or /*comments */
Logical operators					
Or	|	| or OR	| or OR	|	|
And	&	AND or &	& or and	&	&
Equal	==	= or EQ	= or eq	==	==
Not equal	!= or ~=	NE or <> or ~=	^= or ne	!=	!=
Less than	<	< or LT	< or lt	<	<
Greater than	>	> or GT	>or gt	>	>
Graphic editing and interface	Graph Editor or Syntax	Graph Editor	Syntax	Syntax	Graph Editor or Syntax
Output Display Amount	Very Good	Extensive	Extensive	Minimal *Need to specify what is required	Very Good Output is interactive and can be customised/sections collapsed

Working with Multiple Datasets	X *(≤ Stata 15)* ✓ *(Stata 16/17)*	✓*Single output file	✓	✓	✓

Importing and Exporting					
	STATA	**SPSS**	**SAS**	**R**	**JMP**
Import file types					
Excel (xls, xlsx, csv)	✓	✓	✓	✓	✓
SPSS	✓	✓	✓	✓	✓
STATA	✓*	✓	✓	✓	via CSV
JMP	X	X	✓	✓	✓
SAS	✓	✓	✓	✓	✓
Export file type					
Excel (xls, xlsx, csv)	✓	✓	✓	✓	✓
SPSS	X	✓	✓	✓	X
STATA	✓	✓	✓	✓	X
JMP	X	X	✓	X	✓
SAS	✓	✓	✓	X	✓

*reading a Stata dataset with older version of Stata can be problematic (use saveold command to save in an older format)

Additional Comparisons

- Drop down menus in SAS can be used to import, export and save datasets. These can be saved and used later. It does not provide drop down menus for other procedures like SPSS and STATA.
- SAS, JMP, R and Stata 16 and 17 can work with multiple datasets in one session. Stata 15 and earlier versions have to open a different session for each dataset (or clear a dataset and open another). SPSS can have multiple datasets open, but can only have one of them active at any one time.
- R will autofill dataset and variable names after the first three characters have been typed. Stata 16/17 autofills dataset and variable names and allows you to click on the variable in the dataset and the name can then be copied across to the syntax. If you are using syntax in SPSS and writing it from scratch then there is not autofill option, variable names can be copied from the variable list. JMP autofills commands and column names in JSL (JMP scripting language) if you press control space.
- Zeros before decimal places (i.e. ".0" vs. "0.0") – SPSS does not provide the "0". Stata is inconsistent about this. SAS and R always provide the "0" before the decimal place.

1.6 Online support

Online support is available through a variety of sources. Generally, a simple Google search (or Bing if you want to support the underdog of the search engine world) will direct you to some useful support. However, below we have listed sources of support that are generally reliable for most queries.

Stata

- Statalist: https://www.statalist.org/
- Syntax cheat sheets: https://www.stata.com/bookstore/stata-cheat-sheets/
- UCLA: https: //stats.idre.ucla.edu/stata/
- Princeton University: http://www.princeton.edu/~otorres/Stata/

SPSS

- SPSS Help Topics/Tutorial Tab
- https://statistics.laerd.com/

SAS

- https://www.tutorialspoint.com/sas/index.htm
- https://stats.idre.ucla.edu/sas/

R

- cran.r-project.org
- rstudio.com
- Discussion thread based sites:
    - stackoverflow
    - StackExchange

JMP

- JMP.com

1.7 Conclusions of comparisons

This project was primarily aimed to teach each of the authors the basics from the different software. Each of the authors had experience in one or more of the software, but not all four. By learning each software to an introductory level, we were able to collectively make comparisons over which software is best in different situations.

The authors acknowledge that each software has pros and cons and is fully capable of doing basic analysis. The decision of which software you as the reader should use is likely to be driven by many different factors.

If you plan to only do a small, simple research project and are unlikely to be doing research (particularly analysis) often, then the authors would recommend a software with a smaller learning curve such as SPSS or JMP. This decision would also be influenced by what software is available to you (university, employer etc.). Stata is also a good option here, particularly if Stata is readily available or if your analysis will be more complicated than just basic analysis.

If you plan to be involved with a large amount of research in the long term that will involve many or large complex datasets or complicated analysis, it will be worth your time investing in R or Stata. R is

a free open-source software, so the software can be downloaded on as many machines as you need and can be accessed anywhere. Stata is also a good option but will be dependent on cost and accessibility. SAS could also be considered, but it is expensive, only available on Windows and has a steep learning curve.

2.0 Basic Data manipulation

2.1 Section outline

This section covers basic data manipulation and checking. It will provide instructions for data set appending and merging, excluding cases, creating variables, recoding variables, re-categorizing variables, and changing incorrect values.

2.2 Stata

2.2.1 Open/saving

Opening dataset

<Syntax>:

```
import excel using  "L:\.....\cardiac data.xlsx" , firstrow  sheet("sheetname") clear   /*specify the file location after 'using'; 'firstrow' treat the first row of Excel data as variable names;  'sheet("sheetname")' specifies Excel worksheet to load; 'clear' clears any current dataset */
```

<Drop-down menu>:

File → import → Excel spreadsheet → Browse (*select a excel file that you like to open)* → Select a sheet under "Worksheet:"→ Click "Import first row as variable names" → Click "OK"

Saving data to file in Stata format (.dta)

<Syntax>:

```
save "C:\.....\cardiac data.dta", replace   /*'replace' overwrites any existing dataset with the same name. .dta is default so it does not have to be specified*/
```

<Drop-down menu>:

File → Save as… (Save as type: "Stata Data (*.dta)")

2.2.2 Appending datasets

<Syntax>:

```
*1.  Open one dataset "cardiac data_append1" (called the `master dataset' in Stata)

use  "C:\.... \cardiac data_append1", clear

* 2. Append the dataset "cardiac data_append2" (called the `using dataset' in Stata) to the dataset "cardiac data_append1" (master dataset)

append using  "C:\.... \cardiac data_append2"
```

<Drop-down menu>:

File > Open > select file that you would like to open e.g. "cardiac data_append1" > Open

Data > Combine Datasets > Append datasets > Browse (select dataset that you would like to append "cardiac data_append2" to dataset in memory) > OK

2.2.3 Merging datasets

<Syntax>:

```
*1. Open one dataset "cardiac data_merge1" (called the `master dataset' in Stata)
```

```
use  "C:\.... \cardiac data_merge1", clear

*2. Merge (1:1 match) with the other dataset “cardiac data_ merge2” on a variable “id”

merge 1:1 id using  "C:\.... \cardiac data_merge2"
```

```
Result                           # of obs.
-----------------------------------------
not matched                             0
matched                               558  (_merge==3)
-----------------------------------------
```

2.2.4 Re-categorising variables and labelling variables/variable values

Recoding/grouping age into age group (1 = 40-49, 2 = 50-59, 3 =60-69, 4=70-79, 5=80-100)

<Syntax>:

```
recode age (40/49 = 1) (50/59 =2)(60/69 =3) (70/79 =4) (80/100 = 5), generate(agegroup)
```

<Drop-down menu>:

Data > Create or change data > Other variable-transformation commands > Recode categorical variable > select age under “Variables:” > Enter “(40/49 = 1) (50/59 =2)(60/69 =3) (70/79 =4) (80/100 = 5)” under “Required:” > In Options tab, tick “Generate new variables:” and enter a new *variable name* (agegroup). (default is “Replace existing variables”, which overwrites the existing variable (e.g. age)”) > Click Submit/OK

Labelling a variable “agegroup”

<Syntax>:

```
label variable agegroup "Age group"  /*adding a description "Age group" to a variable agegroup*/
```

<Drop-down menu>:

Data > Variables Manager > Highlight/Click the variable from the list > In Variable properties tab, enter a variable description under “Label:” > Enter

Labelling code values and attaching the code values to a variable “agegroup”

<Syntax>:

```
label define agegroup_vl  1 "40-49"  2 "50-59"  3 "60-69"  4 "70-79"  5 "80+", replace

 /*define a value label called "agegroup_vl" where code 1 is labelled with  "40-49", code 2 is labelled with "50-59", code 3 is labelled with "60-69", code 4 is labelled with "70-79" and code 5 is labelled with  "80+"  */

label values agegroup agegroup_vl   /*attach the value label "agegroup_vl" to a variable "agegroup" */
```

<Drop-down menu>:

Data > Variables Manager > Highlight/Click the variable from the list (e.g. agegroup at the bottom of list) > In Variable properties tab, click “Manage…” under “value label:” > In Manage value labels box, click “Create label” > In Create label box, enter *value label name* (e.g. agegroup_vl) > Enter value/code under “Value:” and a value label under “Label:” (e.g. Value: and Label: 40-49) > Click “Add” > Repeat the steps for the rest of value labels (1: 40-49, 2: 50-59, 3 : 60-69, 4: 70-79, 5: 80+) > Click “OK”> Close Manage value labels box > In Variable properties, select the created value label (e.g. agegroup_vl) under “Value label:“ > Click “Apply”

2.2.5 Recoding existing variables

Recoding a variable "anyevent" to equal '1' when yes and '0' when no (in this dataset '1' means a patient did not suffer a cardiac event and '0' means they did suffer a cardiac event)

<Syntax>:

```
recode anyevent (0=1) (1=0), generate(anyevent1)  / * generate(anyevent1): the recoded variable is generated and named "anyevent1". Repeat for the following variables: posse, newmi, newptca, newcabg, death */
```

<Drop-down menu>:

Data > Create or change data > Other variable-transformation commands > Recode categorical variable > select anyevent under "Variables:" > Enter "(0=1) (1=0)" under "Required:" > In Options tab, tick "Generate new variables:" and enter a new *variable name* (anyevent1). (default is "Replace existing variables", which overwrites the existing variable (e.g. anyevent)") > Click Submit/OK

2.2.6 Creating new variables

Creating a new variable called "dp1" which is a function of 2 existing variables (dp1 = pkhr x sbp)

<Syntax>:

```
generate dp1 = pkhr*sbp
```

<Drop-down menu>:

Data > Create or change data > Create a new variable > Enter "dp1" under Variable name: > Enter "pkhr*sbp" under Specify a value or an expression > Click OK

/*please refer to "help operators" for arithmetic, string, relational, and logical operators*/

2.2.7 Excluding data based on variable categories

Exclude observations with age under 40

<Syntax>:

```
drop if age < 40    /*`if' + expression  after command tells Stata to apply command to the data specified*/
```

<Drop-down menu>:

Data > Create or change data > Drop or keep observations > Click "Drop observations" > Enter "age < 40" under If: (expression) > Click OK

2.2.8 Simple checks for data cleaning

Checking duplicates in all variables or in ID only

<Syntax>:

```
duplicates list    /*default:  list duplicates in terms of all variables*/

duplicates list id  /*list duplicates in terms of a variable "id" only*/
```

<Drop-down menu>:

Data > Data utilities > Report and list duplicated observations > Tick one option under "Report and list duplicated observations > Enter "_all" or select id under "Variables:" to identify duplicates on all or ID.

Checking if a variable "anyevent1" is coded correctly (anyevent1 should be equal to 1 (=Yes) if any of the events (recorded in the variables, death, newmi, newptca and newcabg) occurred

*(1) Create a new variable named “anyevent2” that codes 0=No if none of events (death, newmi, newptca and newcabg) occurred (i.e. all variables coded as No) and codes 1=Yes if ANY of events occurred (i.e. death, newmi, newptca or newcabg is/are coded as Yes)

<Syntax>:

```
gen anyevent2 = 0    /*Step 1: create a variable named “anyevent2” coded as 0 (=No) for all observations*/

replace anyevent2 = 1 if death1 ==1 | newmi1==1 | newptca1==1 | newcabg1==1  /*Step 2: replace the code value 0 with 1 (=Yes) if the following condition applies (death = 1 OR newmi1 =1 OR newptca1=1 OR newcabg1=1)*/

/*Note: use double equal sign ‘==’ after if. Use `|’ for OR and `&’ for AND. Please refer to “help operators” for more details.*/
```

<Drop-down menu>:

Data > Create or change data > Create a new variable > Enter “anyevent2” under Variable name: > Enter“0” under Specify a value or an expression > Click OK

Data > Create or change data > Change contents of variable > Select “anyevent2” under Variable:

> Enter “1” under New contents: (value or expression) > In if/in tab, type “death1 ==1 | newmi1==1 | newptca1==1 | newcabg1==1” > Click OK

Checking if variables anyevent2 and anyevent1 match

<Syntax>:

```
tabulate anyevent2 anyevent1   /*cross table of anyevent2 and anyevent1*/
```

anyevent2	anyevent1 0	1	Total
0	455	0	455
1	1	88	89
Total	456	88	544

/*table shows there is one observation with contradicting values (anyevent2 =1 and anyeven1 = 0) in these variables*/

<Drop-down menu>:

Statistics > Summaries, Tables and Tests > Frequency tables > two-way table with measures of association > Row variable: enter or select anyevent2, Column variable: enter or select anyevent1 > OK

Listing patient ID and other key variables for the observation with contradicting variables

<Syntax>:

```
list id anyevent1 anyevent2 death1 newmi1 newptca1 newcabg1 if anyevent2 != anyevent1  /*`!=’ : not equal */
```

id	anyeve~1	anyeve~2	death1	newmi1	newptca1	newcabg1
214	0	1	0	1	0	0

<Drop-down menu>:

Data > Describe data > List data > Enter variable names that you like to list under "Variables:" (e.g.," id anyevent1 anyevent2 death1 newmi1 newptca1 newcabg1") > In by/if/in tab, enter "anyevent2 != anyevent1" under "If: (expression)"

Correcting data on a variable "anyevent1" for ID 214 (i.e. changing the value of "anyevent1" for ID 214 from 0 to 1)

<Syntax>:

```
replace anyevent1 = 1 if id == 214
```

<Drop-down menu>:

Data > Create or change data > Change contents of variable > Select "anyevent1" under "Variable:" > Enter "1" under New contents: (value or expression) > In if/in tab, enter "id == 214" > Submit/OK

2.3 SPSS

2.3.1 Open/saving

Opening a dataset in an excel document format and saving it as an SPSS dataset

Open the dataset:

File -> Open-> Data -> select "excel" from *Files of Type* drop down menu and *find file (cardiac data)* -> Select *OK*

Save the SPSS dataset:

File -> Save -> (Pick a working data location) File name: "cardiac_workingdata" -> *Save*

Note: SPSS is able to import data from many file types (e.g. txt, csv, stata) by choosing the correct option in the *Files of Type* drop down menu and following the import wizard which follows.

2.3.2 Appending datasets

Appending two datasets together where each dataset has the same list of variables but each include different observations.

Open the first dataset:
File -> Open-> Data -> select "excel" from *Files of Type* drop down menu and *find file (cardiac data_append1)* -> Select *OK*
Sort ascending by ID (not necessary before attempting to append):
Right click on "id" and select "sort ascending"
Go to *variable view* tab and ensure all variables expected are listed
Open the second dataset:
File -> Open-> Data -> select "excel" from *Files of Type* drop down menu and *find file (cardiac data_append2)* -> Select *OK*
Sort ascending by ID (not necessary before attempting to append):
Right click on "id" and select "sort ascending"
To append datasets together:
Select the tab for the first dataset
Data-> Merge files -> add cases

Under *An open dataset* select the second SPSS dataset and select *Continue* and select *OK* on the next page
Go to *variable view* tab and ensure all variables expected from both datasets are now listed.
Save the appended datasets under a new name:
File -> Save As -> File name: "cardiac_post_append" -> Save

2.3.3 Merging datasets

Merging two datasets together which have a matching identifying variable (with the same IDs) but different variables.

Open the first dataset:

File -> Open-> Data -> select "excel" from *Files of Type* drop down menu and *find file (cardiac data_merge1)* -> Select *OK*

Sort ascending by ID (necessary before attempting to merge):

Right click on "id" and select "sort ascending"

Open the second dataset:

File -> Open-> Data -> select "excel" from *Files of Type* drop down menu and *find file (cardiac data_merge2)* -> Select *OK*

Sort ascending by ID (necessary before attempting to merge):

Right click on "id" and select "sort ascending"

To append datasets together:

Select the tab for the first dataset

Data-> Merge files -> add variables

Under "An open dataset" select the second SPSS dataset and select "Continue"

Under *Excluded variables* select "id(+)" and check the *Match cases on key variables* box, followed by *Cases are sorted in order of key variables in both datasets* box and click the arrow to add "id(+)" to the *Key Variables* box- > Select *OK*

Go to *Data view* tab and ensure all observations expected from both datasets are now listed.

Save the appended datasets under a new name:

File -> Save As -> File name: "cardiac_post_merge" -> Save

2.3.4 Re-categorising variables

Recoding the age variable from a continuous format into categories

Transform -> Recode into different variable -> Select "age" and under *Output variable* – Name "Age1", and under *Label* "Age_categorised"

Select *Old and New values*

- ➔ Under *Range*: type "40 through 49" and under *New Value* type "1" and select *Add*
- ➔ Under *Range*: type "50 through 59" and under *New Value* type "2" and select *Add*
- ➔ Under *Range*: type "60 through 69" and under *New Value* type "3" and select *Add*
- ➔ Under *Range*: type "70 through 79" and under *New Value* type "4" and select *Add*
- ➔ Under *Range, value through HIGHEST*: type "80" and under *New Value* type "5" and select *Add*

Under the *Variable View* tab and the "Age1" variable – Select *Values*

- ➔ Under *Value*: type "1" and under *Label* type "40 to 49" and select *Add*
- ➔ Under *Value*: type "2" and under *Label* type "50 to 59" and select *Add*
- ➔ Under *Value*: type "3" and under *Label* type "60 to 69" and select *Add*
- ➔ Under *Value*: type "4" and under *Label* type "70 to 79" and select *Add*
- ➔ Under *Value*: type "5" and under *Label* type "80 +" and select *Add*

To ensure the new variable has been created correctly go to: Analyze -> Descriptive statistics ->Crosstabs and select "age" under the row and "Age1" under the column and Press *OK*

2.3.5 Recoding existing variables

Recoding the variable 'anyevent' to equal '1' when yes and '0' when no (in this dataset '1' mean a patient did not suffer a cardiac event and '0' means they did suffer a cardiac event)

Drop Down Menu:

Transform -> Recode into different variable -> Select anyevent

Name "anyevent1", and under *Label* "anyevent_working"

Select *Old and New values*

- ➔ Under *Value*: type "0" and under *New Value* type "1" and select *Add*
- ➔ Under V*alue*: type "1" and under *New Value* type "0" and select *Add*

Under the *Variable View* tab and the "anyevent1" variable – Select *Values*

- ➔ Under *Value*: type "1" and under *Label* type "Yes" and select *Add*
- ➔ Under *Value*: type "0" and under *Label* type "No" and select *Add*
- ➔ *Check with a crosstabs:*

To ensure the new variable has been created correctly go to: Analyze -> Descriptive statistics -> Crosstabs - > For the row variable select "anyevent" and for the column variable select "anyevent1" - > Select *OK (crosstabs table not shown)*

2.3.6 Creating new variables

Creating a new variable called "dp_check" which is a function of 2 existing variables (dp_check = pkhr x sbp) to confirm previous calculations were correct

Transform-> Compute variable ->

Under *Target Variable* type: "dp_check"

Under *Numeric Expression* type: "pkhr*sbp"

And press OK.

Transform-> Compute variable ->

Under *Target Variable* type: "check2"

Under *Numeric Expression* type: "1"

Select *If* Icon -> check the *Include if case satisfied condition* box and write the following expression into the box: "dp ~= dp_check"

And press *OK*.

To ensure no differences between the two variables go to: Analyze -> Descriptive statistics ->Frequencies and select "check2" and Press OK

check2

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1.00	1	.2	100.0	100.0
Missing	System	543	99.8		
Total		544	100.0		

Note: As one observation in check2 is equal to 1, this indicates one variable was incorrectly calculated previously and requires correction.

2.3.7 Excluding data based on variable categories

Excluding data where individuals are less than 40 years old

Go to Data-> Select Cases-> under "if" type age>=40 and select C*ontinue* followed by *OK*

2.3.8 Simple checks for data cleaning

Checking for duplicate observations (i.e. duplicate ID names)

Data-> Identify Duplicate cases-> Select "ID" variable - > Select *OK*

Indicator of each last matching case as Primary

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Primary Case	544	100.0	100.0	100.0

Checking if a variable "anyevent1" is coded correctly

anyevent1 should be equal to 1 (=Yes) if any of the events (recorded in the variables, death, newmi, newptca and newcabg) occurred

Analyze -> Descriptive statistics - >Crosstabs and select "newmi1" under the row and "anyevent1" under the column and Press *OK*

newmi1 * anyevent1 Crosstabulation

Count

		anyevent1		
		.00	1.00	Total
newmi1	.00	455	61	516
	1.00	1	27	28
Total		456	88	544

There is one observation with contradicting values (anyevent1 =0 and newmi1 = 1)

Finding the patient ID and other key variables for the observation with contradicting variables

Data -> Select Cases - > Select 'If condition satisfied' Press OK, type" "anyevent1=1 & MISSING(anyevent_check)" and select *Continue*.

Analyze -> Descriptive Statistics-> Frequencies-> select variables "id" and select *OK*.

id

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	214	1	100.0	100.0	100.0

Correcting the "anyevent1" observation which should have been equal to 1

Transform-> Compute variable ->

Under *Target Variable* type: “Anyevent1”

Under *Numeric Expression* type: “1”

Select *If* Icon -> check the *Include if case satisfied condition* box and write the following expression into the box: “ID=214”

And press *OK*

To check: Analyze -> Descriptive statistics - >Crosstabs and select “newmi1” under the row and “Anyevent1” under the column and Press OK

newmi1 * anyevent1 Crosstabulation

Count

		anyevent1		
		.00	1.00	Total
newmi1	.00	455	61	516
	1.00	0	28	28
Total		455	89	544

2.4 SAS

2.4.1 Open/saving

Opening a dataset in an excel document format and saving it as an SAS dataset

FC1 is what you are naming the imported dataset and the range is the name of the sheet you are importing.

```
PROC IMPORT OUT= WORK.FC1
            DATAFILE= "L:\....\cardiac da
ta.xlsx"
            DBMS=EXCEL REPLACE;
     RANGE="Sheet1$"; /*this is the sheet name you are importing*/
     GETNAMES=YES;
     MIXED=NO;
     SCANTEXT=YES;
     USEDATE=YES;
     SCANTIME=YES;
RUN;
```

You can also do this by the drop down menu.
File→import data→ choose the data source→ Browse (select where it is coming from)→sheet name you are importing→member(what name will you assign it)→ finish. The commands can be saved and imported.

This will set up a permanent dataset and save the dataset to the selected location.

```
libname fastcar ' L:\....\Software\SAS';
data fastcar.fc6_2;set fc1;run;
```

```
CHECKING LOG

NOTE: WORK.FC1 data set was successfully created.
NOTE: The data set WORK.FC1 has 558 observations and 43 variables.
NOTE: PROCEDURE IMPORT used (Total process time):
      real time           2.77 seconds
      cpu time            0.18 seconds
```

2.4.2 Appending datasets

Appending two datasets together where each dataset has the same list of variables but each include different observations.

Import both sheets first

```
PROC IMPORT OUT= WORK.FC_AP1
            DATAFILE= " L:\....\cardiac data_append1.xlsx"
            DBMS=EXCEL REPLACE;
     RANGE="Sheet1$";
     GETNAMES=YES;
     MIXED=NO;
     SCANTEXT=YES;
     USEDATE=YES;
     SCANTIME=YES;
RUN;

PROC IMPORT OUT= WORK.FC_AP2
            DATAFILE= " L:\....\cardiac data_append2.xlsx"
            DBMS=EXCEL REPLACE;
     RANGE="Sheet1$";
     GETNAMES=YES;
     MIXED=NO;
     SCANTEXT=YES;
     USEDATE=YES;
     SCANTIME=YES;
RUN;
```

The data step creates the appended dataset. The name after the word data is the name of the new dataset. Set is the command that appends them. In a data step, every line must end with a semi colon and the data step is complete after the run statement.

```
data fastcar_appending;
set WORK.FC_AP1 WORK.FC_AP2;
run;

data fastcar_appending_2;
set FC_AP1 FC_AP2;
run;
```

```
CHECKING LOG

NOTE: There were 200 observations read from the data set WORK.FC_AP1.
NOTE: There were 358 observations read from the data set WORK.FC_AP2.
NOTE: The data set WORK.FASTCAR_APPENDING_2 has 558 observations and 43 variables.
```

2.4.3 Merging datasets

Read in both Excel sheets, sort both sheets by the unique variable and then merge by the ID variable. Merge is the command that will merge.

```
PROC IMPORT OUT= WORK.FC_m1
            DATAFILE= " L:\....\cardiac data_merge1.xlsx"
```

```
            DBMS=EXCEL REPLACE;
     RANGE="Sheet1$";
     GETNAMES=YES;
     MIXED=NO;
     SCANTEXT=YES;
     USEDATE=YES;
     SCANTIME=YES;
RUN;


PROC IMPORT OUT= WORK.FC_m2
            DATAFILE= " L:\....\cardiac data_merge2.xlsx"
            DBMS=EXCEL REPLACE;
     RANGE="Sheet1$";
     GETNAMES=YES;
     MIXED=NO;
     SCANTEXT=YES;
     USEDATE=YES;
     SCANTIME=YES;
RUN;
```

```
proc sort data=FC_m1; by id; run;
proc sort data=FC_m2; by id; run;

data fastcar_merging;
merge FC_m1 FC_m2;
by id;
run;
```

```
70
71   data fastcar_merging;
72   merge FC_m1 FC_m2;
73   by id;
74   run;

NOTE: There were 558 observations read from the data set WORK.FC_M1.
NOTE: There were 558 observations read from the data set WORK.FC_M2.
NOTE: The data set WORK.FASTCAR_MERGING has 558 observations and 43 variables
```

2.4.4 Re-categorising variables and labelling

```
data fc2; set fc1;  /*setting age groups*/
if age lt 50 & age ge 40 then age_gp=1;
if age ge 50 & age lt 60 then age_gp=2;
if age ge 60 & age lt 70 then age_gp=3;
if age ge 70 & age lt 80 then age_gp=4;
if age ge 80 then age_gp=5;
label age_gp="age group";****
proc format;
value agp          1='40-49'
                   2='50-59'
                   3='60-69'
                   4='70-79'
                   5='80+';
```

The proc freq statement is a way of checking, via cross tabs. It also assign labels to the variable. Reminder every line ends with a semi colon and it all ends with the run statement.

```
proc freq;
table age_gp*age;
format age_gp agp.;
run;
```

2.4.5 Recoding existing variables

In the same datastep as before add the following after the **

```
if posECG=0 then posECG_r=1;
if posECG=1 then posECG_r=0;
if posSE=0 then posSE_r=1;
if posSE=1 then posSE_r=0;
if newMI=0 then newMI_r=1;
if newMI=1 then newMI_r=0;
if newPTCA=0 then newPTCA_r=1;
if newPTCA=1 then newPTCA_r=0;
if newCABG=0 then newCABG_r=1;
if newCABG=1 then newCABG_r=0;
if death=0 then death_r=1;
if death=1 then death_r=0;
if newCABG=0 then newCABG_r=1;
if newCABG=1 then newCABG_r=0;
if anyevent=0 then anyevent_r=1;
if anyevent=1 then anyevent_r=0;

proc freq data=fc2;
table posECG*posECG_r posSE*posSE_r newMI*newMI_r newPTCA*newPTCA_r
newCABG*newCABG_r death*death_r newCABG*newCABG_r anyevent*anyevent_r
/nocol norow nopercent;;
run;
```

2.4.6 Creating new variables

In the same datastep as before add the following after the **

```
dpcheck=pkhr*sbp;
run;
```

You can use the print statement to check the logic statement where your newly created variable does not equal the variable that exist. If there are any error it will list the id dpcheck and dp.

```
proc print data=fc2;
var id dpcheck dp;
where dpcheck ne dp;
run;
```

Obs	id	dpcheck	dp
314	314	4880	15860

2.4.7 Excluding data based on variable categories

If you are excluding data this has to be done in a new data step, with the where statement. Again it has to end with the run statement

```
data fc2; set fc1;
where age>=40;run;

NOTE: There were 544 observations read from the data set WORK.FC4.
      WHERE age>=40;
NOTE: The data set WORK.FC5 has 544 observations and 60 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

2.4.8 Simple checks for data cleaning

Duplicates can be checked the old fashioned way by a simple frequency table of the id

```
proc freq data=fc1;
table id;
run;
```

OR you can do it the following way, where you can only do it per one variable. This will produce a new dataset called keylist which has counts. Then all you have to do is a logical test to see where the count is greater than one.

```
proc freq data=fc2;
 tables id / noprint out=keylist;
run;

proc print data=keylist;
 where count ge 2;
run;
```

Checking any_event by logic test

```
proc print data=fc2;
var id death_r  newmi_r newptca_r newcabg_r anyevent_r;
where anyevent_r=0 & (death_r=1 | newmi_r=1 | newptca_r=1 | newcabg_r=1);
run;
```

Obs	id	death_r	newMI_r	newPTCA_r	newCABG_r	anyevent_r
214	214	0	1	0	0	0

```
proc print data=fc2;
var id death_r  newmi_r newptca_r newcabg_r anyevent_r;
where anyevent_r=1 & (death_r=0 & newmi_r=0 & newptca_r=0 & newcabg_r=0);
run;
```

There were 0 observations read from the data set WORK.FC3.
WHERE (anyevent_r=1) and (death_r=0) and (newmi_r=0) and (newptca_r=0) and (newcabg_r=0);

If you need to change anything you have to create a new dataset with that change.

```
data fc3; set fc2;
if id=214 then anyevent_r=1;
run;
```

Checking the changes by cross tabulations

```
proc freq data=fc3;
table newMI_r* anyevent_r / norow nocol nopercent;
run;
```

Frequency **Table of newMI_r by anyevent_r**

newMI_r	**anyevent_r**		
	0	**1**	**Total**
0	455	61	516

1	0	28	28
Total	455	89	544

2.5 R

Anytime the function library() is used, it relies on the fact that the function install.package() has already been used to download the package. **To install the pack for the first time, use the function install.package() with the package name in quotation marks, e.g. install.package("xlsx").**

2.5.1 Opening and Saving datasets

Set the working directory using the setwd() function

```
setwd("L:/……../Datasets/Cardiac events data")

library(xlsx)

cardiac <- read.xlsx("cardiac data.xlsx", header = TRUE, sheetIndex = 1)
```

Saving a dataset

```
write.xlsx(cardiac, "cardiac data.xlsx")
```

To read in or save different types of files, other functions are used that have a similar structure. For example to read a .csv file you can use the read.csv() file and write.csv() to save one. If your data is in a simple text file (.txt), you can use read.table() and write.table(). We have used the read.xlsx() as it works on the most recent form of Excel document. Different read functions had different default arguments (e.g. read.table() had default header = FALSE, while read.csv() has default header = TRUE). Different read functions will also read in variables as different types (e.g. integer versus floating point, or character vs factor). The information won't get lost (i.e. the value will stay the same even if it is a different type to what you were expecting), however you should still check variable type after you have read in your data

2.5.2 Appending datasets

Appending datasets. R needs column names to be the same.

```
cardiac_appended <- rbind(cardiac_append1, cardiac_append2)
```

2.5.3 Merging datasets

Merging datasets. If the matching column names are the same in both datasets (as is the case here), the input by = "id" could have been used. Otherwise the columns can be identified by their respective names (eg. by.x = "ID", by.y = "id") or by column numbers.

```
cardiac_merged <- merge(cardiac_merge1, cardiac_merge2, by.x = "id", by.y = "id")
```

2.5.4 Re-categorising variables

Categorising a continuous variable into a categorical variable.

A different approach is required for categorical to categorical. The "right = FALSE" means that the breaks supplied go to the left side of each group, rather than the right (i.e. 50 – 59 rather than 51 – 60).

```
cardiac$age_recoded <- cut(cardiac$age, breaks = c(0,50,60,70,80,100), labels = FALSE, right = FALSE)
```

Labelling categories is shown next. The "labels = FALSE" from the cut function meant that each group was stored as an integer, unique for each category: 1 for the first 0 – 49, 2 for the second 50 – 59. The factor function makes the variable a factor, and the labels function assigns the labels to each level of the factor. It is always good to check that you have re-categorised correctly by comparing the original and new variable.

```
cardiac$age_recoded <- factor(cardiac$age_recoded, labels = c("<50","50-59","60-69","70-79","80+"))
```

2.5.5 Recoding existing variables

Change the coding.

```
library(dplyr)

cardiac$anyevent <- recode_factor(cardiac$anyevent, `0` = "1", `1` = "0")
```

Labelling:

Categorical variables are called factor type variables in R. A factor variable has unique levels (for each category).

```
cardiac$anyevent <- factor(cardiac$anyevent, levels = c(0,1), labels = c("no","yes"))
```

2.5.6 Creating new variables

New variables can be defined for a dataset with $ as long as it is given some values with <-. The variable being created below (double product, a surrogate measure of myocardial oxygen demand and cardiac workload) is peak heart rate multiplied by systolic blood pressure.

```
cardiac$dp_recalc <- cardiac$pkhr * cardiac$sbp
```

2.5.7 Excluding data based on variable categories

Take a subset of the data based on some logical condition. This one keeps only patients with age 40 or above.

```
cardiac <- subset(cardiac,cardiac$age >= 40)
```

2.5.8 Simple checks for data cleaning

Duplicates:

If the below is TRUE there are duplicates. Use the variable that is meant to be the unique identifier.

```
any(duplicated(cardiac$id))
```

The below will tell which IDs are duplicates

```
cardiac$id[duplicated(cardiac$id)]
```

Check logic of combination measures

```
table(cardiac$anyevent, cardiac$newmi)
```

```
      no yes
  no  455   1
  yes  61  27
```

Fixing an issue that was discovered,

```
cardiac$id[cardiac$anyevent=="no" & cardiac$newmi=="yes"]
```

```
## [1] 214
```

```
cardiac$anyevent[cardiac$id=="214"] <- "yes"
```

2.6 Notable differences

Recoding/Re-categorising

- When recoding/re-categorising old variables into new ones, there are some differences between the softwares. If there is an unspecified replacement, SPSS (2.3.4) and R (2.5.4) will give a missing value in the new variable while Stata (2.2.4) will keep the old values in the new variable.
- There is no simple way to categorise a continuous variable into ranges in each software, e.g. 0 to 10 becoming category 1, 11 to 20 becoming category 2 and so on. Stata and R are probably the most simple (2.2.4 and 2.5.4) for syntax; SPSS has the simplest drop down menu method (2.3.4) though Stata is similar; and SAS has a long syntax method to do this (2.4.4).

Merging datasets

- While it is generally recommended to have data sets sorted by the identifying variables before merging (to aid in the checking) SAS specifically requires the data to be sorted by the variable the merge is using to match between data sets. This is not required in the other datasets, or is prompted for in the case of SPSS.
- R alone can merge with matching columns having different names, i.e. the matching columns can be identified by the different names from each dataset or by the column numbers.
- When merging datasets in R or SAS, a name for the new merged dataset can be specified. When merging in SPSS and Stata a “master” dataset is used (the dataset which is open) and additional datasets can be merged with this “master” dataset. A new dataset is not automatically created when merging, and the user will need to manually save the file under a new name or it will override the original master dataset.

3.0 Presentation and Summary of Data

3.1 Section outline

This section will cover the basic ways data can be presented, both with descriptive summary statistics and graphical representations.

3.2 Stata

3.2.1 Frequency table

One-way frequency table for a variable "posse1"

<Syntax>:

```
tabulate posse1

tab posse1, missing   /*`tab': short for tabulate; `missing': include missing value in the table*/
```

```
    posse1 |      Freq.     Percent        Cum.
-----------+-----------------------------------
        No |        409       75.18       75.18
       Yes |        135       24.82      100.00
-----------+-----------------------------------
     Total |        544      100.00
```

<Drop-down menu>:

Statistics > Summaries, tables, and tests > Frequency tables > One-way table > Select variable of interest under "Categorical variable:" >(optional) Tick "Treat missing values like other values"

3.2.2 Summary statistics of continuous variables

Simple and detailed summary statistics of a continuous variable "pkhr" (Peak Heart Rate)

<Syntax>:

```
sum pkhr      /*sum is short for summarize*/
```

```
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
        pkhr |        544    120.2886    22.67627        52        210
```

```
sum pkhr, detail   /*option `detail' produces more detailed summary statistics */
```

```
                          pkhr
-------------------------------------------------------------
      Percentiles      Smallest
 1%           66              52
 5%           81              61
10%           90              61       Obs                 544
25%          106              62       Sum of Wgt.         544

50%          122                       Mean           120.2886
                      Largest          Std. Dev.      22.67627
75%          135             171
90%          147             176       Variance       514.2131
95%          156             182       Skewness      -.0958121
99%          170             210       Kurtosis       3.175542
```

<Drop-down menu>:

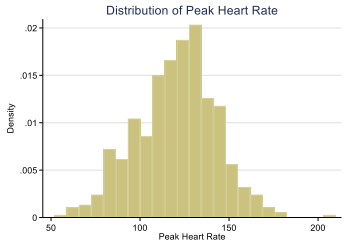
Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Summary statistics > Select variable of interest under “Variables: “> (optional) Tick “Display additional statistics”

3.2.3 Histogram

Histogram of a continuous variable “pkhr” (Peak Heart Rate)

<Syntax>:

```
histogram  pkhr , title("Distribution of Peak Heart Rate")
```



<Drop-down menu>:

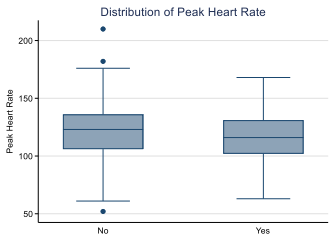
Graphics > Histogram > In Main tab, select a variable (e.g.pkhr) under “Variable:” in Data section > (optional) Select an option for scale under “Y-axis” > In Title tab, enter a title under “Title:” > Click Submit/OK

3.2.4 Box plot

Box plots of a variable “pkhr” (Peak Heart Rate) by “anyevent1” (presence of any event)

<Syntax>:

```
graph box  pkhr, over(anyevent1) title("Distribution of Peak Heart Rate")
```



/* Upper whiskers indicate $y_{[75]} + 1.5 \times (IQR)$ and lower whiskers indicate $y_{[25]} - 1.5 \times (IQR)$*/

<Drop-down menu>:

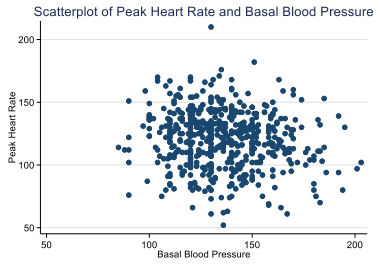
Graphics > Box plot > Select a continuous variable (e.g. bhr, pkhr) under "Variable:" > In Categories tab tick "Group 1" and enter a subgroup variable (e.g. anyevent1) as the "Grouping variable:" > In Title tab, enter a title under "Title:" > Click Submit/OK

3.2.5 Scatter plot

Scatter plot of "pkhr" (peak heart rate) and "basebp" (basal blood pressure)

<Syntax>:

```
scatter pkhr basebp, xtitle("Basal Blood Pressure") ytitle("Basal Heart Rate") title("Scatterplot of Peak Heart Rate and Basal Blood Pressure")   /*`xtitle("  "): x-axis title, `ytitle("  "): y-axis title */
```



<Drop-down menu>:

Graphics > Twoway graph (scatter, line, etc.) > Plots >Click "Create…" > Under "Basic plots: (select type)", select "Scatter" > Under "Plot type: (scatterplot)", select variables for "Y variable:" (e.g. pkhr) and "X variable:" (e.g. basebp) >Click OK/Submit. Enter title using Titles tab.

3.3 SPSS

3.3.1 Frequency table

Descriptive statistics for a categorical variable – posse1

Analyze -> Descriptive statistics - >Frequencies and select ”posse1” and Press *OK*

posse_working

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	.00	409	75.2	75.2	75.2
	1.00	135	24.8	24.8	100.0
	Total	544	100.0	100.0	

3.3.2 Summary statistics of continuous variables

Descriptive statistics for a continuous variable – pkhr

Analyze -> Descriptive statistics - >Explore and select ”pkhr” . For additional statistics go to the “statistics” tab an Press *OK*

Descriptive Statistics

	N	Range	Minimum	Maximum	Mean	Std. Deviation
pkhr	544	158	52	210	120.29	22.676
Valid N (listwise)	544					

Descriptives

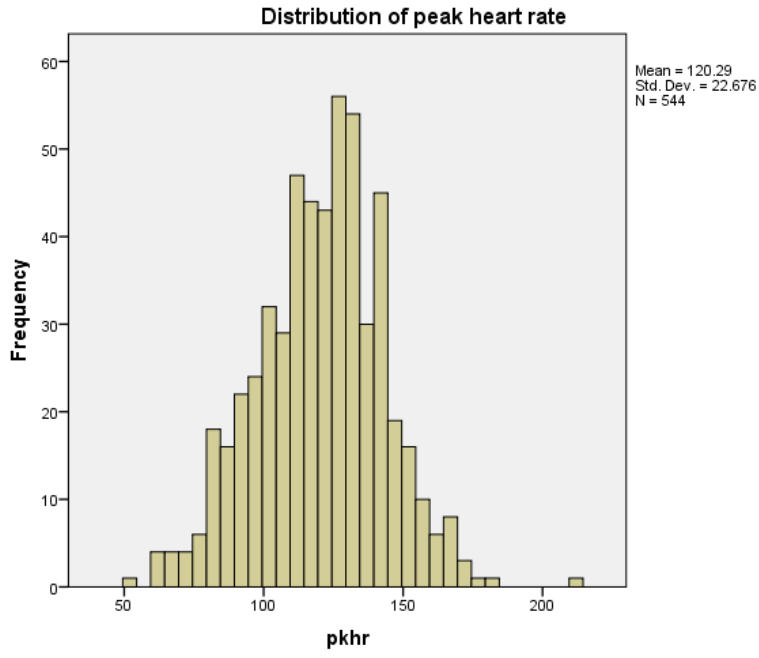
			Statistic	Std. Error
pkhr	Mean		120.29	.972
	95% Confidence Interval for Mean	Lower Bound	118.38	
		Upper Bound	122.20	
	5% Trimmed Mean		120.39	
	Median		122.00	

Variance	514.213	
Std. Deviation	22.676	
Minimum	52	
Maximum	210	
Range	158	
Interquartile Range	29	
Skewness	-.096	.105
Kurtosis	.188	.209

3.3.3 Histogram

Histogram of peak heart rate

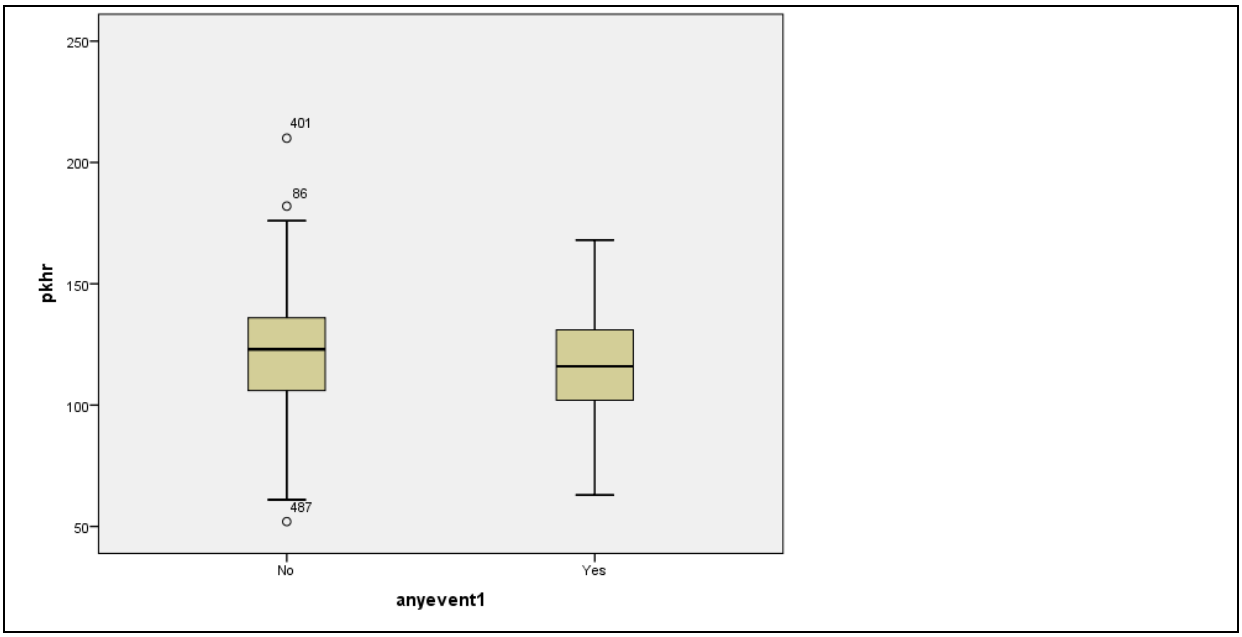
Graphs -> Legacy Dialogs - >Histogram -> Select "pkhr" under variable and 'titles' to add further detail to the graph -> Select *Ok*



3.3.4 Box plot

Pkhr by anyevent

Graphs-> Legacy dialogs -> Boxplot and select *Simple* and "summaries of groups of cases" and select *Define* For *Variable* select "pkhr" and for *Category axis* select "anyevent1" and select *OK*

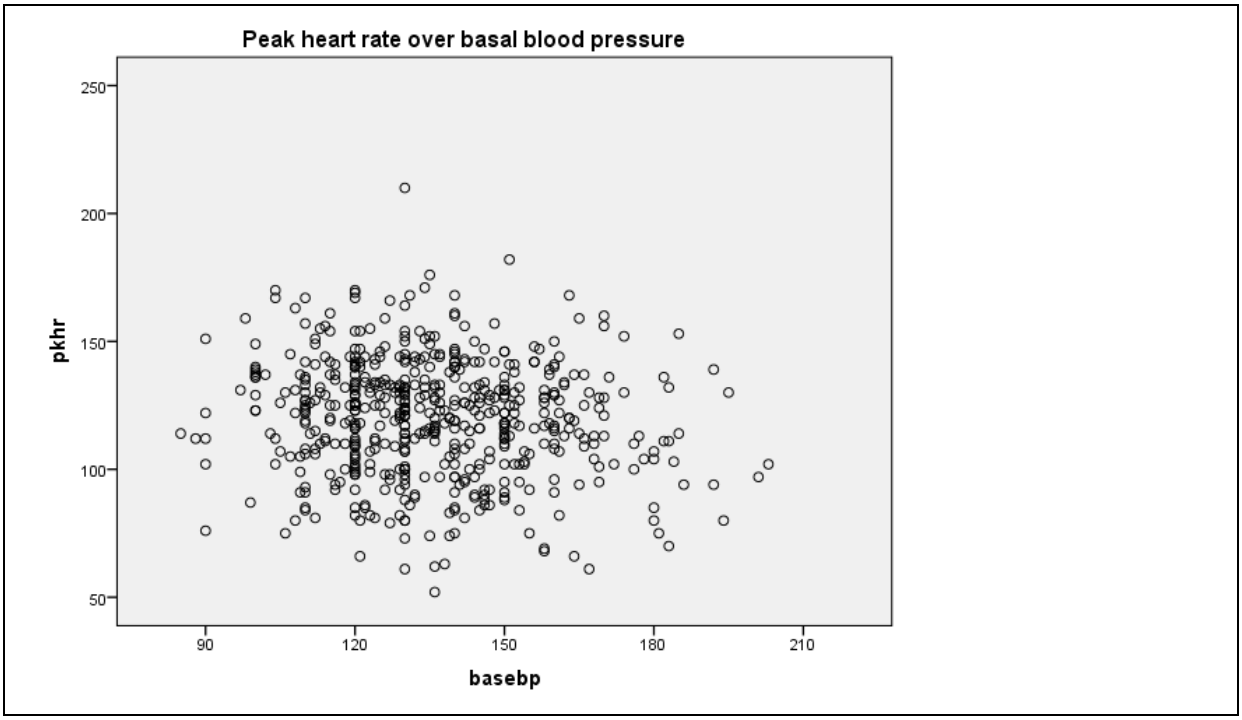


Note: No option is available to add titles in legacy dialogs – Boxplots. Chart builder could alternatively be used, which allows titles.

3.3.5 Scatter plot

Pkhr by basebp

Graphs-> Legacy dialogs -> *Scatter/Dot* and select *Simple* and select Y-axis variable "pkhr" and X-axis variable "basebp". Follow the *Titles*… tab to add titles, subtitles or footnotes. Press *OK*



3.4 SAS

3.4.1 Frequency table

Descriptive statistics for a categorical variable – posse1

This is via the proc freq statement. After table you can list as many variables as you want.

```
proc freq data=fc3;
table posSE_r;
run;
```

The SAS System

The FREQ Procedure

posSE_r	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	409	75.18	409	75.18
1	135	24.82	544	100.00

3.4.2 Summary statistics of continuous variables

Proc means is for continuous variables. The default is to produce mean, STD, min and max. After var you can list as many variables as you want

```
proc means  data=fc3;
var pkhr;
run;
```

The MEANS Procedure

Analysis Variable : pkhr peak heart rate

N	Mean	Std Dev	Minimum	Maximum
544	120.2886029	22.6762664	52.0000000	210.0000000

You can get more information by stating what you want in the above statement as follows

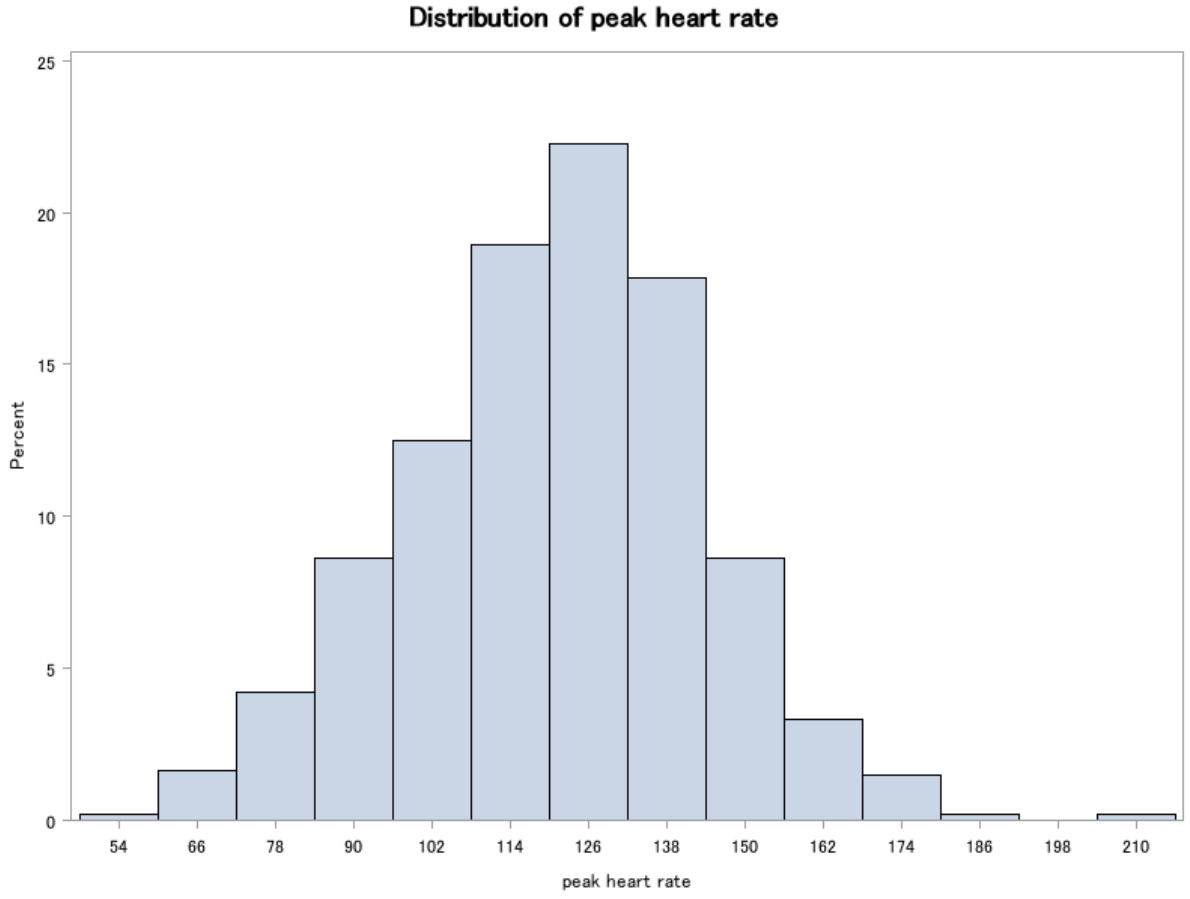
```
proc means n mean stddev min max median Q1 Q3 data=fc3;/*kurt mode and
skew*/
var pkhr;
run;
```

Analysis Variable : pkhr peak heart rate

N	Mean	Std Dev	Minimum	Maximum	Median	Lower Quartile	Upper Quartile
544	120.2886029	22.6762664	52.0000000	210.0000000	122.0000000	106.0000000	135.0000000

3.4.3 Histogram

```
Ods graphics on;
proc univariate data=fc3 noprint; /*This will produce only the histograms.
If you omit noprint it will also give you
basic statistical measures, hence you could get everything without using
proc means above*/
ods graphics off; /*removes unnecessary output*/
var pkhr;
histogram;/*need to add other variables*/
run;
```



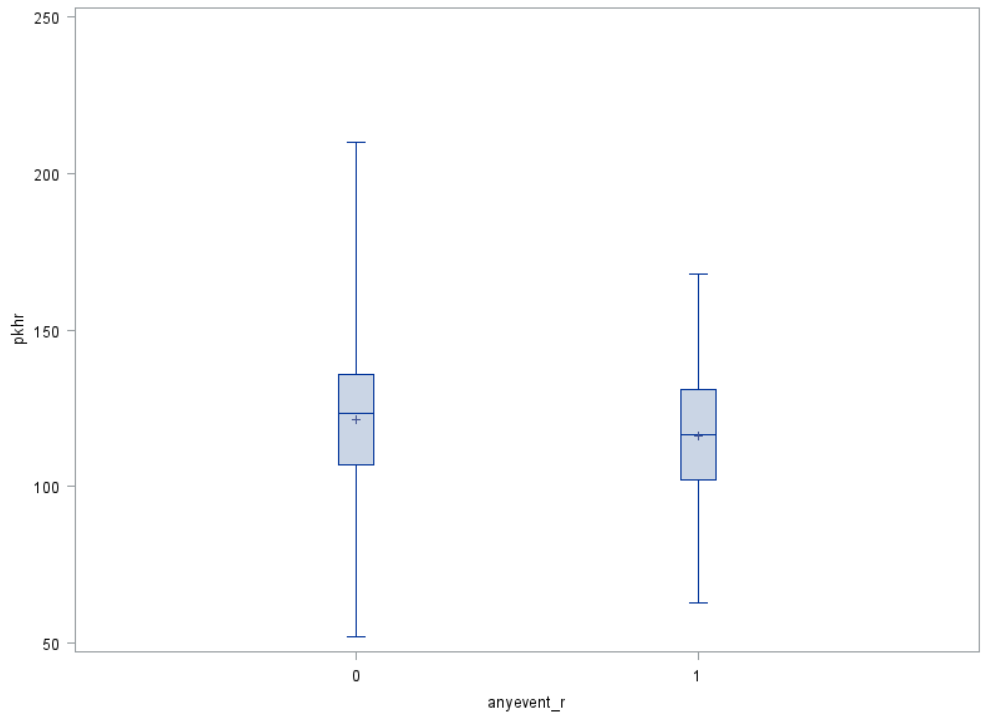
3.4.4 Box plot

You must sort by the categorical variables first. The default goes from the minimum to the maximum. It does not use the correct definition for outliers. The first boxplot uses the incorrect

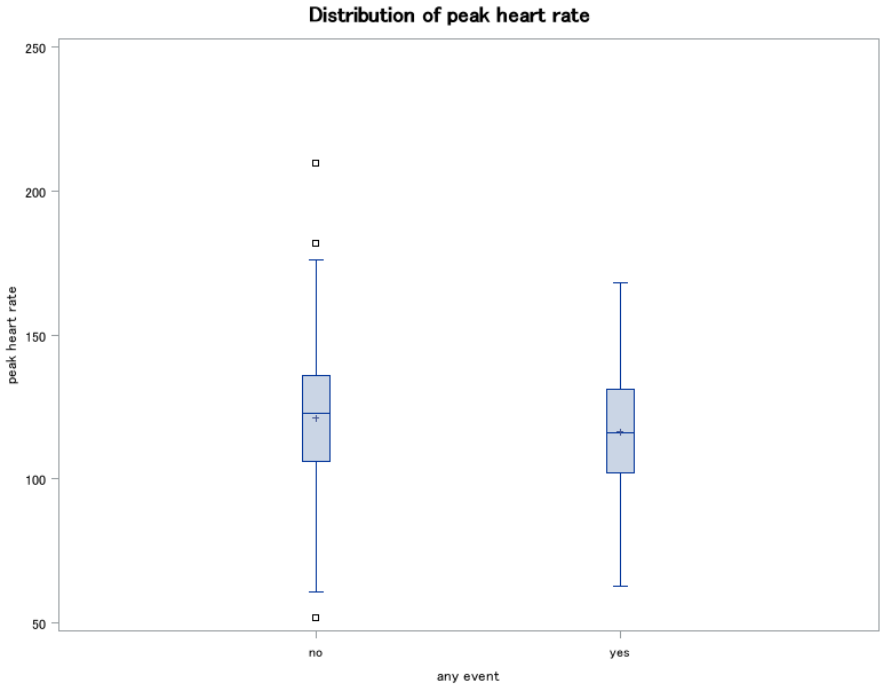
definition. This follows by the boxplot that uses the correct definition. To get this you must add boxstyle=schematic as shown in the example that follows

```
proc sort data=fc3; by anyevent_r; run;

proc boxplot data=fc3;
plot pkhr*anyevent_r; /*by default whiskers represent the minimum and
maximum values */
run;
```



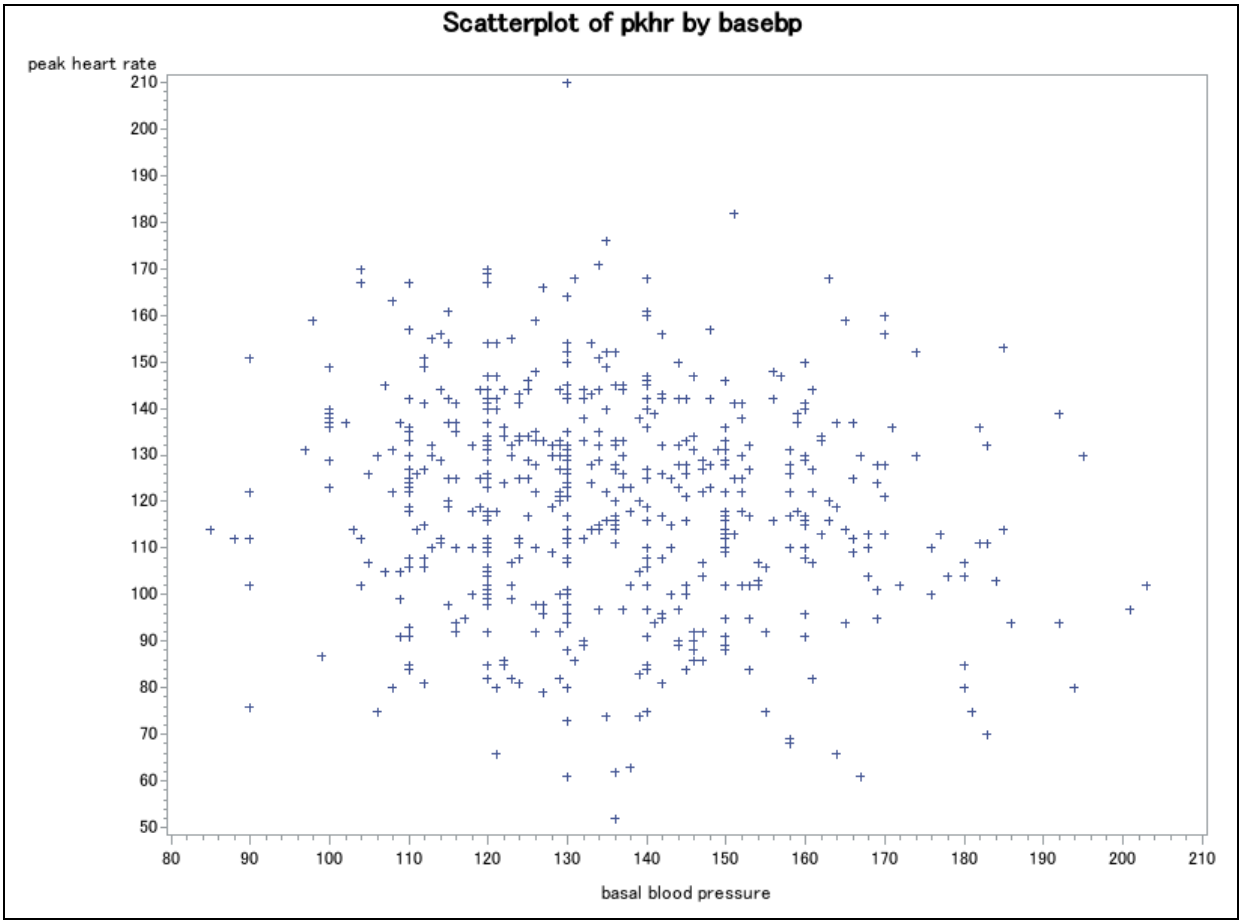

```
proc format;
value ayn 0="no" 1="yes";
proc boxplot data=fc3;
ods graphics off; /*removes unwanted information of graph*/
label anyevent_r="any event";
plot pkhr*anyevent_r/
      boxstyle=schematic; /*boxplot*/
/*inset min mean max stddev;*/
        title "Distribution of peak heart rate";
        format anyevent_r ayn.;
run;
```



3.4.5 Scatter plot

This is via the proc gplot statement.

```
proc gplot data=fc3; /*scatterplot*/
plot pkhr*basebp;
title "Scatterplot of pkhr by basebp";
run;
```


Scatterplot of pkhr by basebp
peak heart rate
210
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
80
90
100
110
120
130
140
150
160
170
180
190
200
210
basal blood pressure

3.5 R

3.5.1 Frequency table

The table() function makes a frequency count table. To get the proportions, use prop.table(). The function prop.table() needs a table object input.

```
table(cardiac$posse)

prop.table(table(cardiac$posse))
```

```
 no yes
409 135
       no       yes
0.7518382 0.2481618
```

3.5.2 Summary statistics of continuous variables

Use the base function summary() for continuous (numeric) variables.

```
summary(cardiac$pkhr)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   52.0   106.0   122.0   120.3   135.0   210.0
```

The function summary() does not give you the standard deviation, so use sd().

```
sd(cardiac$pkhr)
```

```
[1] 22.67627
```

Other ways to summarise continuous data exist.

```
library(Hmisc)

describe(cardiac$ pkhr)
```

```
cardiac$ pkhr

     n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75      .90      .95

   544        0      105        1    120.3     25.5    81.15    90.00   106.00   122.00   135.00   147.00   155.85


lowest :  52  61  62  63  66, highest: 170 171 176 182 210
```

```
library(psych)

describe(cardiac$pkhr)
```

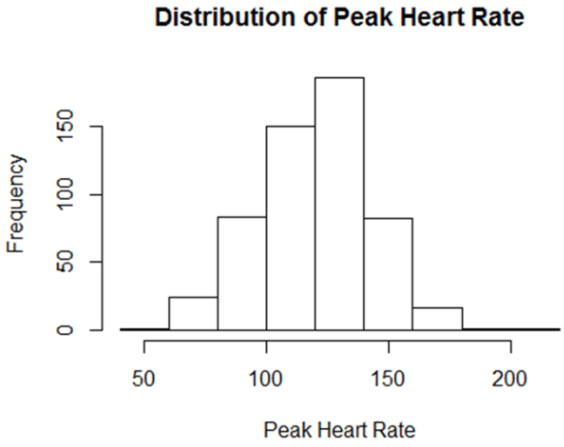
```
   vars     n   mean      sd median trimmed   mad min max range skew kurtosis     se

X1    1 544 120.29 22.68      122    120.64 22.24   52  210    158   -0.1     0.16  0.97
```

3.5.3 Histogram

Use the hist() function. Breaks can be omitted from the inputs and R will decide where the break points will be. The only variable that is definitely necessary is the first one (the variable being plotted).

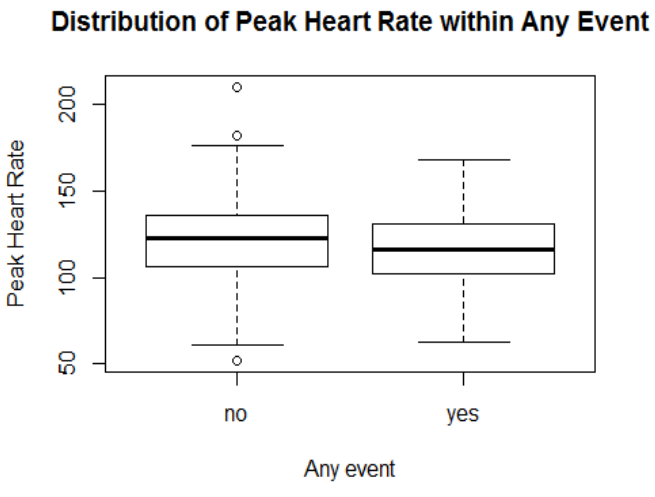
```
hist(cardiac$pkhr, xlab = "Peak Heart Rate", main = "Distribution of Peak Heart Rate", breaks = seq(40,220,20))
```



3.5.4 Box plot

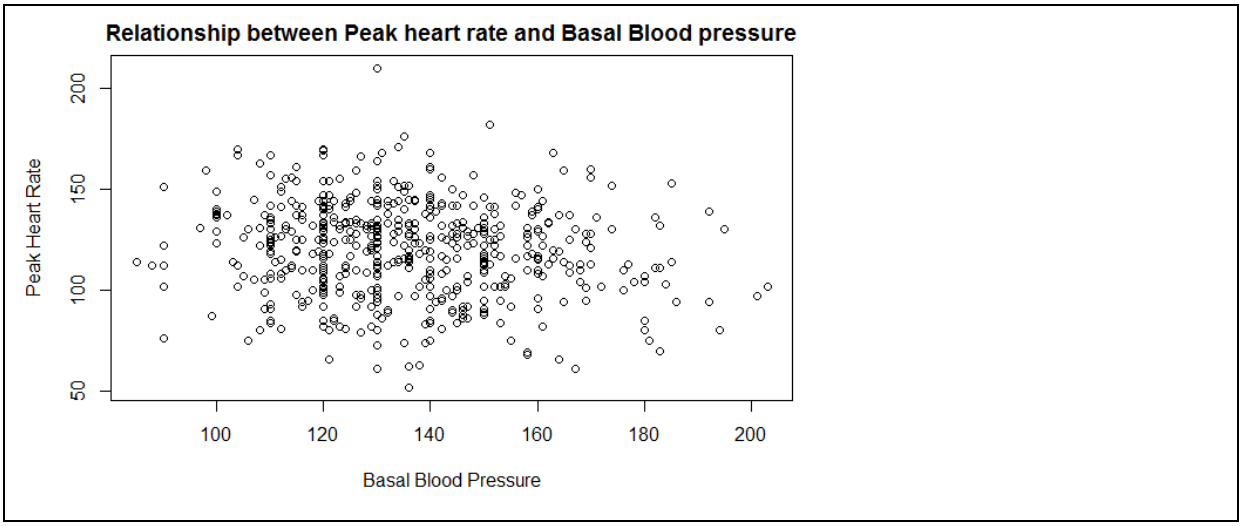
Most plotting and modelling can either be done as 'y ~ x' as an input, or 'x, y' as input in that order.

```
boxplot(cardiac$pkhr ~ cardiac$anyevent, main = "Distribution of Peak Heart Rate within Any Event", xlab = "Any event", ylab = "Peak Heart Rate")
```



3.5.5 Scatter plot

```
plot(cardiac$basebp ~ cardiac$pkhr, main = "Relationship between Peak heart rate and Basal Blood pressure", xlab = "Peak Heart Rate", ylab = "Basal Blood Pressure")
```



3.6 Notable differences

- Box plot: The whiskers (the vertical lines) in Stata, SPSS and R represent one and a half times the interquartile range (1.5 x IQR). In SAS (by default), whiskers extend to the minimum and maximum values.
- A GUI graph editor is available in SPSS, Stata and SAS in addition to syntax for editing graphs after they have been made. A graph editor is not available in R, however there is large potential for altering the graph output through syntax.
- Stata and R have fairly simple syntax for plotting, with the entire plot generally able to be made in one line of code. Other comparisons for graph editing are done in a table in General Comparisons.

4.0 Basic Tests and Models

4.1 Section outline

This section will outline how to analyse different possible associations between variables in the dataset. This includes basic hypothesis testing for different types of variables (i.e. different combinations of categorical and continuous variables), and also basic regression models.

4.2 Stata

4.2.1 Chi-Square Test (including Fishers Exact Test)

Two-way table of posse1 (stress echocardiogram test result) and anyevent1(presence of any event) with Chi-squared test

<Syntax>:

```
tab posse1 anyevent1, row ch exact      /*`row': add row % in table (`column': add column %), `chi2': Chi-squared test, `exact': Fisher's exact test*/
```

```
           |        anyevent1
   posse1  |        No        Yes |     Total
-----------+----------------------+----------
        No |       367         42 |       409
           |     89.73      10.27 |    100.00
-----------+----------------------+----------
       Yes |        88         47 |       135
           |     65.19      34.81 |    100.00
-----------+----------------------+----------
     Total |       455         89 |       544
           |     83.64      16.36 |    100.00

          Pearson chi2(1) =  44.6901   Pr = 0.000
           Fisher's exact =                 0.000
   1-sided Fisher's exact =                 0.000
```

<Drop-down menu>:

Statistics > Summaries, tables, and tests > Frequency tables > Two-way table with measures of association > Main tab, select variables of interest e.g. Row variable: posse1, Column variable: anyevent1. In 'Test statistics' tick 'Pearsons' chi-squared' and 'Fisher's exact test'. In 'cell contents' tick "Within-row relative frequencies" > OK.

4.2.2 Student's t-test

Two sample t-test with equal variances – "pkhr"(peak heart rate) by "anyevent1" (presence of any event)

<Syntax>:

```
ttest pkhr, by(anyevent1)    /*default ; assumption of equal variances*/
```

```
Two-sample t test with equal variances

   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
      No |     455    121.0549    1.070542    22.83544    118.9511    123.1588
     Yes |      89    116.3708    2.283606     21.5435    111.8326     120.909
---------+--------------------------------------------------------------------
combined |     544    120.2886    .9722369    22.67627    118.3788    122.1984
---------+--------------------------------------------------------------------
    diff |            4.684159     2.62299               -.468313     9.83663
------------------------------------------------------------------------------
    diff = mean(No) - mean(Yes)                                   t =   1.7858
Ho: diff = 0                                     degrees of freedom =      542

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.9627         Pr(|T| > |t|) = 0.0747          Pr(T > t) = 0.0373
```

<Drop-down menu>:

Statistics > Summaries, tables, and tests > Classical tests of hypotheses > t test (mean-comparison test) > In Main tab, under “t-tests”, select “Two-sample using groups” > In “Two-sample t test using groups”, select continuous variable of interest under “Variable name:”(e.g. pkhr) and select comparison group variable (e.g. anyevent1) under “Group variable name:” > Click Submit/OK

4.2.3 ANOVA

One-way ANOVA – “pkhr”(peak heart rate) by “agegroup” (Age group)

<Syntax>:

```
anova pkhr agegroup
```

```
                   Number of obs =       544     R-squared     =  0.0071
                   Root MSE      =   22.6798     Adj R-squared = -0.0003

          Source |  Partial SS         df         MS        F     Prob>F
-----------------+----------------------------------------------------------
           Model |   1970.5304          4   492.63259      0.96   0.4303
                 |
        agegroup |   1970.5304          4   492.63259      0.96   0.4303
                 |
        Residual |   277247.16        539   514.37321
-----------------+----------------------------------------------------------
           Total |   279217.69        543   514.21306
```

<Drop-down menu>:

Statistics > Linear models and related > ANOVA/MANOVA > Analysis of variance and covariance > In Main tab, select a variable under “Dependent variable:” (e.g. pkhr) and select a comparison group variable under “Model:” (e.g. agegroup) > Click “Submit”

4.2.4 Mann-Whitney U test (non-parametric)

Mann-Whitney U -test – “pkhr”(peak heart rate) by “anyevent1” (presence of any event)

<Syntax>:

```
ranksum pkhr, by(anyevent1)
```

```
Two-sample Wilcoxon rank-sum (Mann-Whitney) test

   anyevent1 |       obs    rank sum    expected
-------------+--------------------------------
          No |       455    126564.5    123987.5
         Yes |        89     21675.5     24252.5
-------------+--------------------------------
    combined |       544      148240      148240

unadjusted variance  1839147.92
adjustment for ties     -465.90
                     ----------
adjusted variance    1838682.02

Ho: pkhr(anyeve~1==No) = pkhr(anyeve~1==Yes)
             z =   1.900
    Prob > |z| =   0.0574
```

<Drop-down menu>:

Statistics > Nonparametric analysis > Tests of hypotheses > Wilcoxon rank-sum test > In Main tab, select/type a variable under “Variable:” (e.g. pkhr) and select/type a comparison group variable under “Grouping variable:” (e.g. anyevent1) > click Submit/OK

4.2.5 Kruskal-Wallis (non-parametric)

Kruskal-Wallis equality-of-populations rank test – “pkhr”(peak heart rate) by “agegroup” (Age group)

<Syntax>:

```
kwallis pkhr, by(agegroup)
```

```
Kruskal-Wallis equality-of-populations rank test

    agegroup |      Obs |  Rank Sum
-------------+----------+----------
       40-49 |       34 |  10099.50
       50-59 |       84 |  24646.00
       60-69 |      167 |  46527.00
       70-79 |      182 |  47296.00
         80+ |       77 |  19671.50
-------------+----------+----------

chi-squared =     4.646 with 4 d.f.
probability =     0.3256

chi-squared with ties =     4.647 with 4 d.f.
probability =     0.3255
```

<Drop-down menu>:

Statistics > Nonparametric analysis > Tests of hypotheses > Kruskal-Wallis rank test > In Main tab, select/type a variable under “Outcome variable:” (e.g. pkhr) and select/type a comparison group variable under “Variable defining groups:” (e.g. agegroup) > Click Submit/OK

4.2.6 Correlation (Pearson and Spearman)

Pearson correlation coefficient between "pkhr" (peak heart rate) and basebp "basal blood pressure"

<Syntax>:

```
pwcorr pkhr basebp, sig
```

```
             |     pkhr   basebp
-------------+------------------
        pkhr |   1.0000
             |
             |
      basebp |  -0.1191   1.0000
             |   0.0054
```

<Drop-down menu>:

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Pairwise correlations > in Main tab, select/type continuous variables under "Variables: (leave empty for all)" (e.g. pkhr basebp) > Tick "Print significance level for each entry" to display p-value > Click Submit/OK

Spearman correlation coefficient "pkhr" (peak heart rate) and basebp "basal blood pressure"

<Syntax>:

```
spearman pkhr basebp
```

```
  Number of obs =     544
Spearman's rho =     -0.1075

Test of Ho: pkhr and basebp are independent
    Prob > |t| =      0.0121
```

<Drop-down menu>:

Statistics > Nonparametric analysis > Tests of hypotheses > Spearman's rank correlation >> in Main tab, select/type continuous variables under "Variables:" (e.g. pkhr basebp) > Click Submit/OK

4.2.7 Linear regression

Multiple regression – "pkhr" (peak heart rate) as response (dependent) variable and age and gender as predictor(independent) variable

<Syntax>:

```
regress pkhr age i.gender    /*prefix `i. ' is to specify indicators for each level (category) of categorical variables. Reference level = 0 by default*/
```

Source	SS	df	MS	Number of obs	=	544
				F(2, 541)	=	2.34
Model	2391.70888	2	1195.85444	Prob > F	=	0.0976
Residual	276825.98	541	511.693125	R-squared	=	0.0086
				Adj R-squared	=	0.0049
Total	279217.689	543	514.213056	Root MSE	=	22.621

pkhr	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
age	-.1922839	.0893429	-2.15	0.032	-.3677855	-.0167824
1.gender	-.5325655	1.984605	-0.27	0.789	-4.431042	3.365911
_cons	133.7274	6.320909	21.16	0.000	121.3108	146.1439

<Drop-down menu>:

Statistics > Linear models and related > Linear regression >In Main tab, select/type a continuous outcome variable under “Dependent variable:“ (e.g. pkhr) and select/type explanatory variable(s) under “Independent variables:” (age i.gender) > Click Submit/OK

4.2.8 Logistic regression

Multivariable logistic regression -“anyeven1” (Any event) as response(dependent) variable and age and gender as predictor(independent) variables

<Syntax>:

```
logit anyevent1 i.posse1 age i.gender, or      /* `or': display Odds ratio instead of coefficient*/
```

```
Logistic regression                             Number of obs     =        544
                                                LR chi2(3)        =      41.76
                                                Prob > chi2       =     0.0000
Log likelihood = -221.52557                     Pseudo R2         =     0.0861

------------------------------------------------------------------------------
   anyevent1 | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      posse1 |
        Yes |   4.486628   1.099986     6.12   0.000     2.774796    7.254527
         age |     1.0071   .0116958     0.61   0.542      .984436    1.030287
             |
      gender |
      Female |   .7223687    .176082    -1.33   0.182     .4479949    1.164782
       _cons |   .0858996   .0707713    -2.98   0.003     .0170883    .4318003
------------------------------------------------------------------------------
Note: _cons estimates baseline odds.
```

<Drop-down menu>:

Statistics > Binary outcomes > Logistic regression >In Main tab, select/type a binary outcome variable under “Dependent variable:“(e.g. anyevent1) and select/type explanatory variable(s) under “Independent variables:” (i.posse1 age i.gender) > Click Submit/OK

4.3 SPSS

4.3.1 Chi-Square Test (including Fishers Exact Test)

Posse by anyevent

Analysis-> Descriptive Statistics -> Crosstabs -> Chi-square Test - > Select row variable 'posse1' and select column variable 'anyevent1' -> Select *Statistics* tab and check the *Chi-square* box and Select *Continue*-> Press *OK*

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Pearson Chi-Square	44.690[a]	1	.000		
Continuity Correction[b]	42.914	1	.000		
Likelihood Ratio	39.591	1	.000		
Fisher's Exact Test				.000	.000
Linear-by-Linear Association	44.608	1	.000		
N of Valid Cases	544				

Note: SPSS includes fishers exact test in output automatically.

4.3.2 Student's t-test

Pkhr by anyevent

Analysis-> Compare Means -> Independent-Samples T-test-> Select Test variable 'pkhr' and select Grouping variable 'anyevent1' -> Press *OK*

**not all output shown*

Group Statistics

	anyevent1	N	Mean	Std. Deviation	Std. Error Mean
pkhr	.00	455	121.05	22.835	1.071
	1.00	89	116.37	21.543	2.284

Independent Samples Test

	Levene's Test for Equality of Variances		t-test for Equality of Means	
	F	Sig.	t	df

pkhr	Equal variances assumed	.440	.507	1.786	542
	Equal variances not assumed			1.857	129.715

Independent Samples Test

		t-test for Equality of Means			
		Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference
					Lower
pkhr	Equal variances assumed	.075	4.684	2.623	-.468
	Equal variances not assumed	.066	4.684	2.522	-.306

4.3.3 ANOVA

Pkhr by age

Analyze-> Compare Means -> One-Way ANOVA-> Select Test variable 'pkhr' and select *Factor variable* 'Age1' -> Press *OK*

ANOVA

pkhr

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1970.530	4	492.633	.958	.430
Within Groups	277247.159	539	514.373		
Total	279217.689	543			

4.3.4 Mann-Whitney U test (non-parametric)

Pkhr by anyevent

Analysis-> Nonparametric Tests -> Legacy Dialogs->2 Independent Samples -> Select Test variable 'pkhr' and select Grouping variable 'Anyevent1' and check the box for *Man Whitney U*-under *Test Type*-> Press *OK*

Ranks

	anyevent1	N	Mean Rank	Sum of Ranks
pkhr	No	455	278.16	126564.50
	Yes	89	243.54	21675.50
	Total	544		

Test Statistics[a]

	pkhr
Mann-Whitney U	17670.500
Wilcoxon W	21675.500
Z	-1.900
Asymp. Sig. (2-tailed)	.057

4.3.5 Kruskal-Wallis (non-parametric)

Pkhr by Age (categorised)

Analysis-> Nonparametric Tests -> Legacy Dialogs-> K Independent Samples -> Select Test variable 'pkhr' and select Grouping variable 'Anyevent1' and check the box for *Kruskal-Wallis H*-under *Test Type*-> Press *OK*

Ranks

	age_catagorised	N	Mean Rank
pkhr	40 to 49	34	297.04
	50 to 59	84	293.40
	60 to 69	167	278.60
	70 to 79	182	259.87
	80+	77	255.47
	Total	544	

Test Statistics[a,b]

	pkhr
Chi-Square	4.647
df	4
Asymp. Sig.	.326

a. Kruskal Wallis Test

b. Grouping Variable: age_catagorised

4.3.6 Correlation (Pearson and Spearman)

Pkhr by basebp

Pearson

Analyse -> Correlation -> Bivariate - > Select Variables – "pkhr" "basebp" and ensure "Pearson" box is checked under *Correlation Coefficients* - > Select *OK*

Correlations

		pkhr	basebp
pkhr	Pearson Correlation	1	-.119**
	Sig. (2-tailed)		.005
	N	544	544
basebp	Pearson Correlation	-.119**	1
	Sig. (2-tailed)	.005	
	N	544	544

**. Correlation is significant at the 0.01 level (2-tailed).

Spearman

Analyse -> Correlation -> Bivariate - > Select Variables – "pkhr" "basebp" and ensure "Spearman" box is checked under *Correlation Coefficients* - > Select *OK*

Correlations

			pkhr	basebp
Spearman's rho	pkhr	Correlation Coefficient	1.000	-.107*

	Sig. (2-tailed)	.	.012
	N	544	544
basebp	Correlation Coefficient	-.107*	1.000
	Sig. (2-tailed)	.012	.
	N	544	544

*. Correlation is significant at the 0.05 level (2-tailed).

4.3.7 Linear regression

Pkhr by continuous age and gender

Analysis-> Regression -> Linear -> Select "pkhr" for *Dependent* and "age" and "gender" for *Independent(s)* -> Press *OK*

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.093[a]	.009	.005	22.621

a. Predictors: (Constant), gender, age

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	2391.709	2	1195.854	2.337	.098[b]
	Residual	276825.980	541	511.693		
	Total	279217.689	543			

a. Dependent Variable: pkhr

b. Predictors: (Constant), gender, age

Coefficients[a]

Model	Unstandardized Coefficients	Standardized Coefficients	t	Sig.	95.0% Confidence Interval for B

		B	Std. Error	Beta			Lower Bound	Upper Bound
1	(Constant)	133.727	6.321		21.156	.000	121.311	146.144
	age	-.192	.089	-.092	-2.152	.032	-.368	-.017
	gender	-.533	1.985	-.011	-.268	.789	-4.431	3.366

a. Dependent Variable: pkhr

4.3.8 Logistic regression

Any event by posse and continuous age and gender

Analysis-> Regression ->Binary logistcr -> Select "Anyevent1 " for *Dependent* and "age" and "gender" and "posse1" for *Covariate(s)* ->Select Categorical and add both "gender" and "posse1" to the Categorical Covariates section and change the Contrast for both to "last" -> Press Continue-> Press *OK*

**Not all output shown*

Dependent Variable Encoding

Original Value	Internal Value
.00	0
1.00	1

Categorical Variables Codings

		Frequency	Parameter coding (1)
gender	Male	215	.000
	Female	329	1.000
posse_working	.00	409	.000
	1.00	135	1.000

Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	443.051[a]	.074	.125

Variables in the Equation

		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1[a]	posse1(1)	1.501	.245	37.487	1	.000	4.487
	age	.007	.012	.371	1	.542	1.007
	gender(1)	-.325	.244	1.780	1	.182	.722
	Constant	-2.455	.824	8.876	1	.003	.086

Coefficient p-value Odds ratio

Variables in the Equation

		95% C.I.for EXP(B)	
		Lower	Upper
Step 1[a]	posse1(1)	2.775	7.255
	age	.984	1.030
	gender(1)	.448	1.165
	Constant		

a. Variable(s) entered on step 1: posse1, age, gender.

4.4 SAS

4.4.1 Chi-Square Test (including Fishers Exact Test)

This is via the proc freq statement, add chisq or fisher after a /, as below.

```
proc freq data=fc3;
tables  posSE_r*anyevent_r  /chisq fisher;
run;
```

The FREQ Procedure

Table of posSE_r by anyevent_r

Frequency / Percent / Row Pct / Col Pct	posSE_r	anyevent_r 0	anyevent_r 1	Total
	0	367	42	409
		67.46	7.72	75.18
		89.73	10.27	
		80.66	47.19	
	1	88	47	135
		16.18	8.64	24.82
		65.19	34.81	
		19.34	52.81	
	Total	455	89	544
		83.64	16.36	100.00

Statistics for Table of posSE_r by anyevent_r

Statistic	DF	Value	Prob
Chi-Square	1	44.6901	<.0001
Likelihood Ratio Chi-Square	1	39.5907	<.0001
Continuity Adj. Chi-Square	1	42.9143	<.0001
Mantel-Haenszel Chi-Square	1	44.6079	<.0001
Phi Coefficient		0.2866	
Contingency Coefficient		0.2755	
Cramer's V		0.2866	

Fisher's Exact Test

Cell (1,1) Frequency (F)	367
Left-sided Pr <= F	1.0000
Right-sided Pr >= F	<.0001
Table Probability (P)	<.0001
Two-sided Pr <= P	<.0001

Sample Size = 544

4.4.2 Student's t-test

For a t-test you must have the class statement, for the categorical variables. You can only have one variable as the class. In the var statement you can add as many variables as you want. I have added a statement at the start (ods graphics off) which will exclude diagnostic graphs. If you do not add this then you will get a surplus of output.

```
ods grahics off;

proc ttest data=fc3; /*student t-test*/
  class anyevent_r;
  var pkhr;
run;
```

Scatterplot of pkhr by basebp

The TTEST Procedure

Variable: pkhr (peak heart rate)

anyevent_r	N	Mean	Std Dev	Std Err	Minimum	Maximum
0	455	121.1	22.8354	1.0705	52.0000	210.0
1	89	116.4	21.5435	2.2836	63.0000	168.0
Diff (1-2)		4.6842	22.6307	2.6230		

anyevent_r	Method	Mean	95% CL Mean		Std Dev	95% CL Std Dev	
0		121.1	119.0	123.2	22.8354	21.4419	24.4242
1		116.4	111.8	120.9	21.5435	18.7770	25.2737
Diff (1-2)	Pooled	4.6842	-0.4683	9.8366	22.6307	21.3600	24.0633
Diff (1-2)	Satterthwaite	4.6842	-0.3056	9.6739			

Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	542	1.79	0.0747
Satterthwaite	Unequal	129.72	1.86	0.0655

Equality of Variances

Method	Num DF	Den DF	F Value	Pr > F
Folded F	454	88	1.12	0.5093

4.4.3 ANOVA.

This is via the proc GLM statement. This includes the model statement.

```
proc glm data=fc2; /*one Way ANOVA*/
  class age_gp;
  model pkhr = age_gp;
  means age_gp;
run;
```

The GLM Procedure

Dependent Variable: pkhr peak heart rate

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	4	1970.5304	492.6326	0.96	0.4303
Error	539	277247.1590	514.3732		
Corrected Total	543	279217.6893			

R-Square	Coeff Var	Root MSE	pkhr Mean
0.007057	18.85449	22.67980	120.2886

Source	DF	Type I SS	Mean Square	F Value	Pr > F
age_gp	4	1970.530371	492.632593	0.96	0.4303

Source	DF	Type III SS	Mean Square	F Value	Pr > F
age_gp	4	1970.530371	492.632593	0.96	0.4303

The GLM Procedure

Level of age_gp	N	pkhr	
		Mean	Std Dev
1	34	123.411765	26.1570057
2	84	122.547619	22.2945262
3	167	121.491018	24.0503659
4	182	118.208791	21.5381027
5	77	118.753247	20.9737655

4.4.4 Mann-Whitney U test (non-parametric).

The proc npar1way is the statement for the Mann-Whitney procedure. It also gives you Kruskal-Wallis.

```
proc npar1way data=fc3 wilcoxon; /*Mann whitney non parameteric test*/
  class anyevent_r;
  var pkhr;
run;
```

The NPAR1WAY Procedure

Wilcoxon Scores (Rank Sums) for Variable pkhr
Classified by Variable anyevent_r

anyevent_r	N	Sum of Scores	Expected Under H0	Std Dev Under H0	Mean Score
0	455	126564.50	123987.50	1355.98010	278.163736
1	89	21675.50	24252.50	1355.98010	243.544944

Average scores were used for ties.

Wilcoxon Two-Sample Test

Statistic	21675.5000

Normal Approximation

Z	-1.9001
One-Sided Pr < Z	0.0287
Two-Sided Pr > |Z|	0.0574

t Approximation

One-Sided Pr < Z	0.0290
Two-Sided Pr > |Z|	0.0579

Z includes a continuity correction of 0.5.

Kruskal-Wallis Test

Chi-Square	3.6118
DF	1
Pr > Chi-Square	0.0574

4.4.5 Kruskal-Wallis (non-parametric)

The Kruskal-Wallis as mentioned above comes from the npar1way statement.

```
proc npar1way data=fc2 wilcoxon;/*non parametric Kruskal Wallis*/
class age_gp;
 var pkhr;
run;
```

The NPAR1WAY Procedure

Wilcoxon Scores (Rank Sums) for Variable pkhr
Classified by Variable age_gp

age_gp	N	Sum of Scores	Expected Under H0	Std Dev Under H0	Mean Score
5	77	19671.50	20982.50	1277.78097	255.474026
2	84	24646.00	22890.00	1324.55858	293.404762
4	182	47296.00	49595.00	1729.58772	259.868132
3	167	46527.00	45507.50	1690.75828	278.604790
1	34	10099.50	9265.00	887.31364	297.044118

Average scores were used for ties.

Kruskal-Wallis Test

Chi-Square	4.6467
DF	4
Pr > Chi-Square	0.3255

4.4.6 Correlation (Pearson and Spearman)

This is via the proc corr statement. Adding Pearson and spearman will give you both.

```
proc corr data=fc3 Pearson Spearman;  /*correlation*/
var pkhr basebp;
run;
```

The CORR Procedure

2 Variables: pkhr basebp

Simple Statistics

Variable	N	Mean	Std Dev	Median	Minimum	Maximum	Label
pkhr	544	120.28860	22.67627	122.00000	52.00000	210.00000	peak heart rate
basebp	544	135.63051	20.72590	133.50000	85.00000	203.00000	basal blood pressure

Pearson Correlation Coefficients, N = 544
Prob > |r| under H0: Rho=0

	pkhr	basebp
pkhr	1.00000	-0.11905
peak heart rate		0.0054
basebp	-0.11905	1.00000
basal blood pressure	0.0054	

Spearman Correlation Coefficients, N = 544
Prob > |r| under H0: Rho=0

	pkhr	basebp
pkhr	1.00000	-0.10748
peak heart rate		0.0121
basebp	-0.10748	1.00000
basal blood pressure	0.0121	

4.4.7 Linear regression

This is via the proc reg statement. You need the model statement . After the equal sign add all the variables you want. Adding clb will give you 95% confidence limits.

```
proc reg data=fc3 ;
model pkhr=age gender/clb;
run;
```

The REG Procedure
Model: MODEL1
Dependent Variable: pkhr peak heart rate

Number of Observations Read 544

Number of Observations Used 544

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	2391.70888	1195.85444	2.34	0.0976
Error	541	276826	511.69312		
Corrected Total	543	279218			

Root MSE	22.62063	R-Square	0.0086
Dependent Mean	120.28860	Adj R-Sq	0.0049
Coeff Var	18.80530		

Parameter Estimates

Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	95% Confidence Limits	
Intercept	Intercept	1	133.72735	6.32091	21.16	<.0001	121.31082	146.14388
age	age	1	-0.19228	0.08934	-2.15	0.0318	-0.36779	-0.01678
gender	gender	1	-0.53257	1.98461	-0.27	0.7885	-4.43104	3.36591

4.4.8 Logistic regression

This is via the proc logistic statement. Any categorical variables are placed in the class statement. If you have coded your outcome to be 1s and 0s with 1 the outcome of interest, you must use descending to model 1 as the outcome. The default is that the reference is the highest number for the categorical variables. If you want to change this you can use ref=first after the categorical variable. Expb will give you the odds ratios.

```
proc logistic data=fc3 descending; /*descending models the 1's*/
class anyevent_r posSE_r gender; /*the default is the highest number*/
model anyevent_r= posSE_r gender age /  expb;
run;
```

This will produce similar outputs to STATA, SPSS and R, where the reference is the lowest category (eg. 0)

```
proc logistic data=fc3 descending; /*descending models the 1's*/
class anyevent_r posSE_r(ref=first) gender(ref=first); /*the default is the
highest number*/
model anyevent_r= posSE_r gender age /  expb;
run;
```

Model Information

Data Set	WORK.FC6
Response Variable	anyevent_r
Number of Response Levels	2
Model	binary logit
Optimization Technique	Fisher's scoring

Number of Observations Read	544
Number of Observations Used	544

Response Profile

Ordered Value	anyevent_r	Total Frequency
1	1	89
2	0	455

Probability modeled is anyevent_r=1.

Class Level Information

Class	Value	Design Variables
posSE_r	0	-1
	1	1
gender	0	-1
	1	1

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	486.809	451.051
SC	491.108	468.247
-2 Log L	484.809	443.051

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	41.7577	3	<.0001
Score	46.6877	3	<.0001
Wald	41.6470	3	<.0001

Type 3 Analysis of Effects

Effect	DF	Wald Chi-Square	Pr > ChiSq
posSE_r	1	37.4885	<.0001
gender	1	1.7799	0.1822
age	1	0.3711	0.5424

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq	Exp(Est)
Intercept	1	-1.8666	0.8138	5.2607	0.0218	0.155
posSE_r 1	1	0.7506	0.1226	37.4885	<.0001	2.118
gender 1	1	-0.1626	0.1219	1.7799	0.1822	0.850
age	1	0.00707	0.0116	0.3711	0.5424	1.007

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
posSE_r 1 vs 0	4.487	2.775	7.255
gender 1 vs 0	0.722	0.448	1.165
age	1.007	0.984	1.030

Association of Predicted Probabilities and Observed Responses

Percent Concordant	70.0	Somers' D	0.405
Percent Discordant	29.4	Gamma	0.408
Percent Tied	0.6	Tau-a	0.111
Pairs	40495	c	0.703

4.5 R

4.5.1 Chi-Square Test (including Fishers Exact Test)

The chisq.test() function can be done on two variables or on a table object. Neither the chisq.test() or the fisher.test() functions produce the crosstabs so they have to be produced separately.

To make a cross table use the table() function. This output can also be assigned to an object. The first variable input is the rows, and the second variable input is the columns. Other functions can be used to make tables. See prop.table().

```
table(cardiac$anyevent, cardiac$posse)
```

```
      no yes
  no  367  88
  yes  42  47
```

To have no correction in the chi square test, include 'correct = FALSE'. Default is TRUE.

```
chisq.test(cardiac$anyevent, cardiac$posse, correct = FALSE)
```

```
      Pearson's Chi-squared test

data:  cardiac$anyevent and cardiac$posse
X-squared = 44.69, df = 1, p-value = 2.308e-11
```

Fisher's Exact test uses fisher.test(). This needs to be done on a table object, either defined beforehand or made inside the function.

```
fisher.test(table(cardiac$anyevent, cardiac$posse))
```

```
      Fisher's Exact Test for Count Data

data:  table(cardiac$anyevent, cardiac$posse)
p-value = 3.455e-10
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 2.811904 7.725425
sample estimates:
```

```
odds ratio
  4.650214
```

4.5.2 Student's t-test

The variance can be assumed equal (conventional) or not. The default is FALSE (variance not equal).

```
t.test(cardiac$pkhr ~ cardiac$anyevent, var.equal = TRUE)
```

```
      Two Sample t-test

data:  cardiac$pkhr by cardiac$anyevent
t = 1.7858, df = 542, p-value = 0.07469
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.468313  9.836630
sample estimates:
 mean in group no mean in group yes
        121.0549          116.3708
```

4.5.3 ANOVA

```
summary(aov(pkhr ~ anyevent, data = cardiac))
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
age_recoded   4   1971   492.6   0.958   0.43
Residuals   539 277247   514.4
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

4.5.4 Mann-Whitney U test (non-parametric)

The 'wilcox.test' is used for the Mann-Whitney test. Include 'correct = FALSE' to perform tests without continuity correction.

```
wilcox.test(pkhr~anyevent, data = cardiac, correct = FALSE)
```

```
        Wilcoxon rank sum test

data:  pkhr by anyevent
W = 22824, p-value = 0.05737
alternative hypothesis: true location shift is not equal to 0
```

4.5.5 Kruskal-Wallis (non-parametric)

The Kruskal-Wallis test is performed with the kruskal.test() function.

```
kruskal.test(pkhr ~ age_recoded, data = cardiac)
```

```
        Kruskal-Wallis rank sum test

data:  pkhr by age_recoded
Kruskal-Wallis chi-squared = 4.6467, df = 4, p-value = 0.3255
```

4.5.6 Correlation (Pearson and Spearman)

Pearson correlation is default in the function cor().

```
cor(cardiac$pkhr,cardiac$basebp)
```

```
-0.1190545
```

To get the p value of correlation use the cor.test() function.

```
cor.test(cardiac$pkhr,cardiac$basebp)
```

```
        Pearson's product-moment correlation

data:  cardiac$pkhr and cardiac$basebp
t = -2.7915, df = 542, p-value = 0.005431
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.20110822 -0.03534163
sample estimates:
       cor
-0.1190545
```

Spearman correlation for non-normal data. The default method is "Pearson", so to get the Spearman you have to specify the method.

```
cor(cardiac$pkhr,cardiac$basebp, method = "spearman")
```

```
-0.1074786
```

```
cor.test(cardiac$pkhr,cardiac$basebp, method = "spearman")
```

```
Cannot compute exact p-value with ties
        Spearman's rank correlation rho

data:  cardiac$pkhr and cardiac$basebp
S = 29715000, p-value = 0.01213
alternative hypothesis: true rho is not equal to 0
sample estimates:
       rho
-0.1074786
```

4.5.7 Linear regression

A Linear model with an outcome of peak heart rate being and explanatory of age. The output of 'lm()' is made to be an object called 'lin.mode.2'.

```
lin.model.2 <- lm(pkhr ~ age + gender, data = cardiac)

summary(lin.model.2)

confint(lin.model.2)
```

```
Call:
lm(formula = pkhr ~ age + gender, data = cardiac)

Residuals:
   Min     1Q Median     3Q    Max
-68.65 -14.77   1.56  14.39  89.88

Coefficients:
```

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  133.72735    6.32091  21.156   <2e-16 ***
age           -0.19228    0.08934  -2.152   0.0318 *
genderFemale  -0.53257    1.98461  -0.268   0.7885
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 22.62 on 541 degrees of freedom
Multiple R-squared:  0.008566, Adjusted R-squared:  0.004901
F-statistic: 2.337 on 2 and 541 DF,  p-value: 0.09759
> confint(lin.model.2)
                  2.5 %       97.5 %
(Intercept)  121.3108195 146.14388240
age           -0.3677855  -0.01678237
genderFemale  -4.4310419   3.36591094
```

4.5.8 Logistic regression

Logistic regression is a general linear model, so it uses the glm() function.

```
log.model.2 <- glm(anyevent ~ posse + age + gender, family = "binomial", data = cardiac)

summary(log.model.2)
```

```
Call:
glm(formula = anyevent ~ posse + age + gender, family = "binomial",
    data = cardiac)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0519  -0.5170  -0.4509  -0.4214   2.2467

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.454576   0.823829  -2.979  0.00289 **
posseyes      1.501101   0.245160   6.123 9.19e-10 ***
age           0.007075   0.011613   0.609  0.54234
genderFemale -0.325220   0.243745  -1.334  0.18212
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 484.81  on 543  degrees of freedom
Residual deviance: 443.05  on 540  degrees of freedom
AIC: 451.05

Number of Fisher Scoring iterations: 4
```

Odds Ratio (OR) as the effect size. The OR is the exponential of the coefficient estimate.

```
exp(coefficients(log.model.2))
```

```
(Intercept)     posseyes          age genderFemale
  0.08589958   4.48662773   1.00710038   0.72236874
```

Getting the 95% Confidence Interval.

```
exp(confint(log.model.2))
```

```
                   2.5 %    97.5 %
(Intercept)  0.0164453 0.4185306
posseyes     2.7785318 7.2796961
age          0.9845664 1.0305300
genderFemale 0.4480448 1.1677029
```

4.6 Notable differences

- R doesn't provide large amounts of output for tests. For example, a cross table is not produced when the chi squared test function is called, just the test result.
- SAS uses the highest category as the reference in regression (e.g. category 1 is the reference and category 0 has the coefficient). In contrast R, Stata, and SPSS all use the lowest category as the reference
- SPSS refers to the p-value as "Sig", while other software directly calls it a "P-value", though R will sometimes refer to it as Pr(>|z|).