Notes for R and RStudio: A few essentials For new users from SIMLESA Conference

Dr Miranda Mortlock
SIMLESA, QAAFI
University of Queensland,
m.mortlock@uq.edu.au

June 22, 2017

Contents

List of Figures

1 Introductory R notes for SIMLESA 2017

These notes are very brief and have been adapted from my workshop which I gave to my tutors at the University of Queensland. Firstly you need to download R and RStudio. from separate websites. (SEE my BeST website *yieldingresults.org* to download both of these). Look in module 0 to see instructions on how to do this and a video. R is the engine or the main program, and R Studio is the 'front end' or user interface that helps you access R with some nice features).

Introduction to RStudio
A few references
Opening a Project in R
Basics in RStudio
R commands and bringing in data
Functions objects
Simple statistics
t tests, plotting
Factors (categories)
Saving your work

References

There are many books on R.
Many are available online through the CRAN site.
There are also many online help documents that become increasingly useful when you gain familiarity with R. I like the following as they have a more statistical approach:

(1) John Maindonald and John Braun. 2007. Data Analysis and Graphics using R. Second Ed. Cambridge University Press.

`http://maths-people.anu.edu.au/~johnm/Personal.html`

(2) Albert, J and M Rizzo. 2012. R by Example. Springer.

(3) Meys, J and de Vries, A. 2001-2. R for Dummies.

(4) Verzani, J. 2001-2. simpleR- using R for Introductory Statistics

Available for download from this site:

http://www.math.csi.cuny.edu/Statistics/R/simpleR/.

http://www.agrocampus-ouest.fr/igagrocampus-ouest.fr
/math/RforStat/

(6) Alistair Sanderson on why he loves R:

http://www.sr.bham.ac.uk/~ajrs/talks/why_I_love_R.pdf

2 RStudio and its display

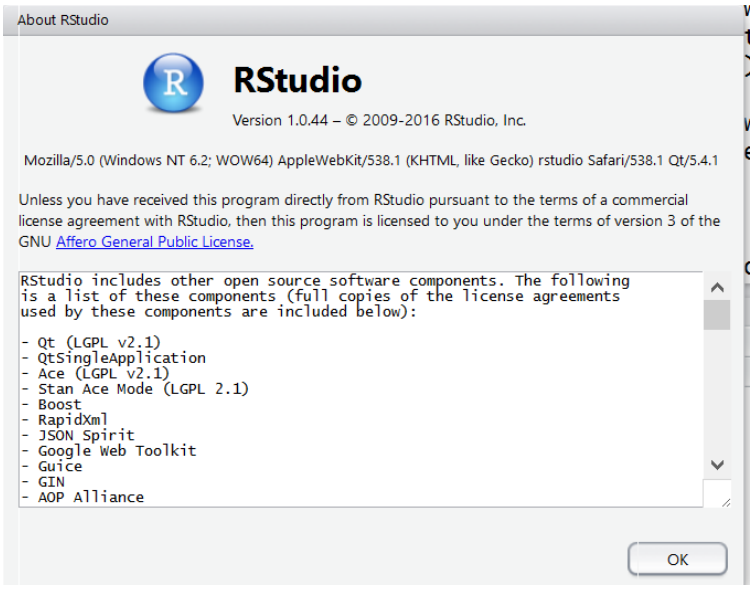

Figure 1: The RStudio Program

Figure 2: The R Studio icon

1. Enter RStudio through the desktop icon
2. It has a four panels in the working area (nb. only 3 when you start)
3. You can check for updates (Top menu- Tools, Check for updates)
4. Get familiar with the way things are stored (.Rdata and .R files)

2.1 The four panels

You will need to get used to the four panels, and how to use them throughout your work. It is useful to always arrange them in the same way for familiarity. You can write code (syntax), and run it, save it and see the results in the

console. You can see what you have done in the History window. Also you can look at the Help, look at the packages installed, look at the working directory and view data sets.

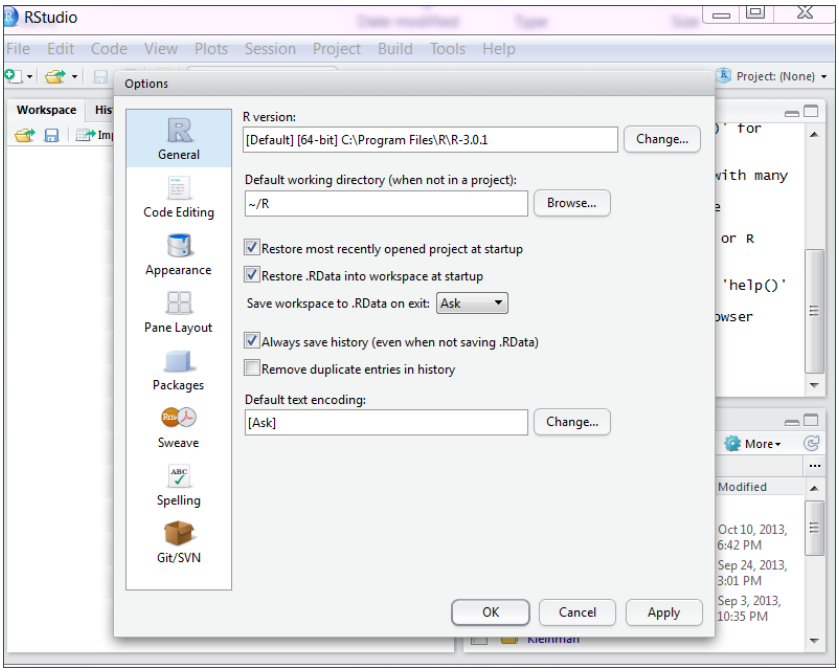

Figure 3: Checking options in RStudio

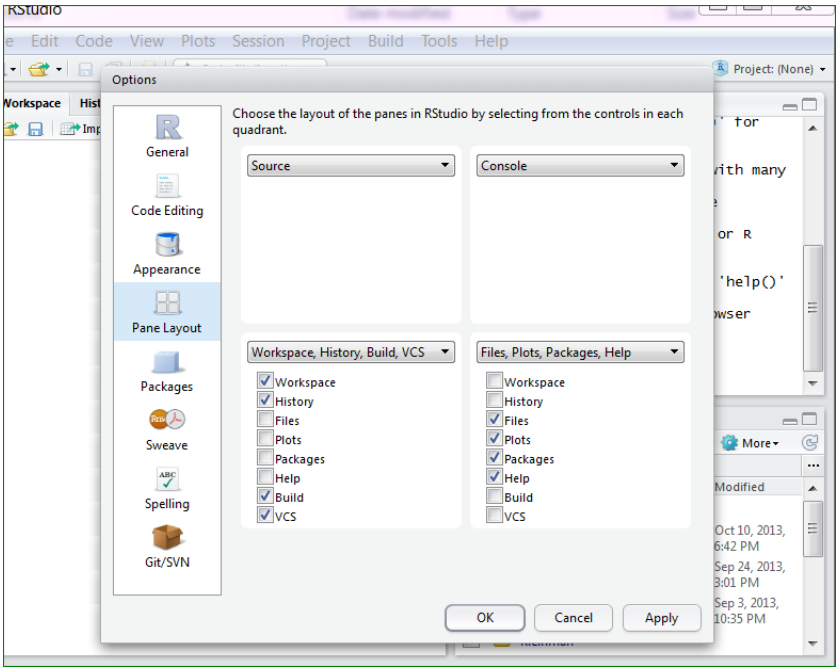

Figure 4: Use options to set display with console on the right

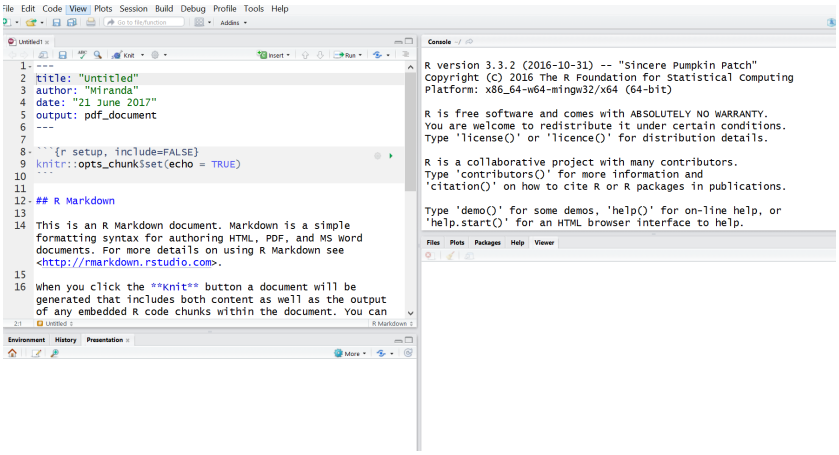

Figure 5: Top left panel for source coding, top right console, bottom left panel for workspace and history and the bottom right panel for directory and help

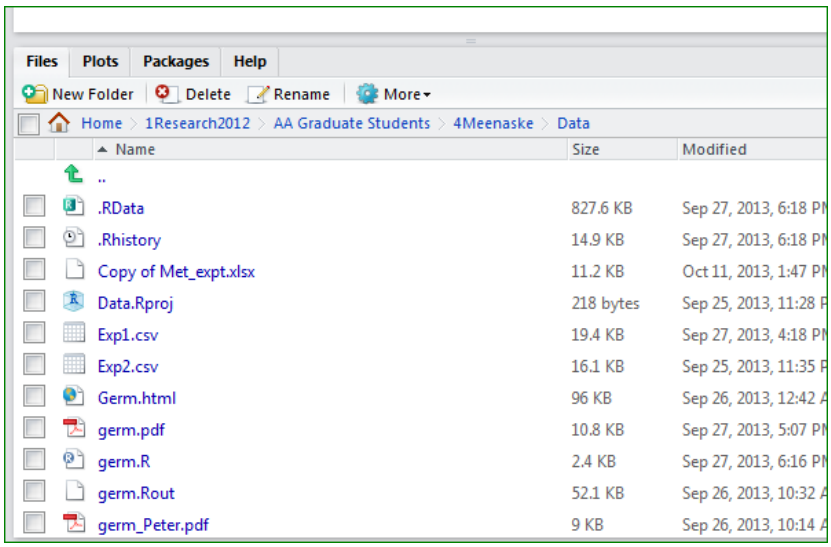

Figure 6: Bottom right panel - showing directory and file types

2.2 R projects and file types

When you start set up an R project.
It is important to get used to the various file types:
The first one you open after creating a project, is to File, New source code as 'filename.R'.

- .RData
- .R
- .Rhistory
- .Rproj

Get familiar with the way data, syntax, and history are stored and how to find them.
The working directory is visible within the RStudio in the files window (bottom right).

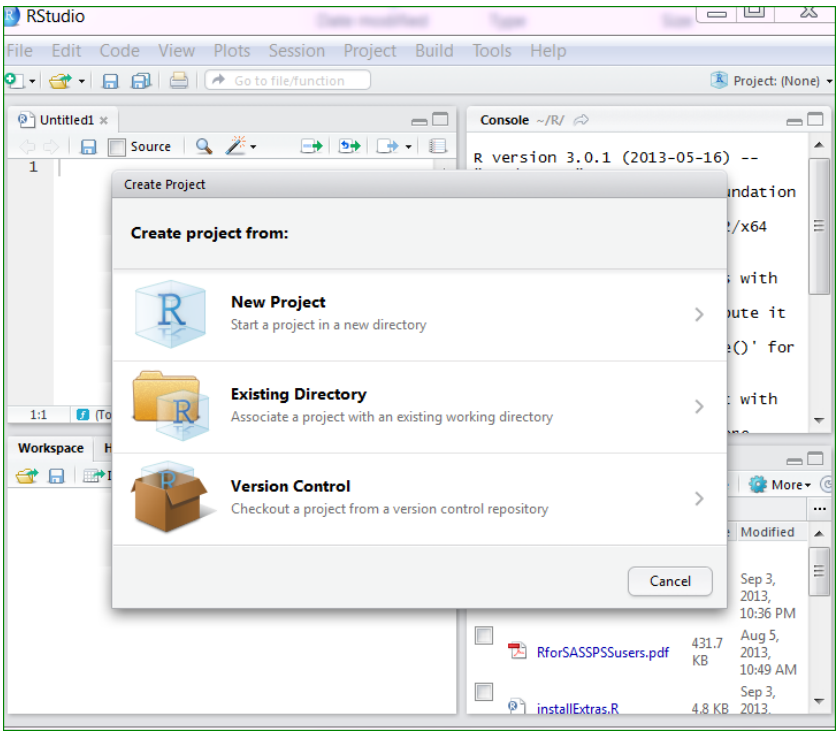

Figure 7: Creating a new project or opening an existing project

2.3 A note on RStudio projects

Work out how to create a Project in the directory of your choice.
Use Project save your work on exit. You can also click on the Project file to reopen and continue the session in the future.

```
See this note for further information on RStudio Projects:

http://www.rstudio.com/ide/docs/using/projects
```

2.4 Blogs on RStudio

Getting help you can also use:

```
http://www.rstudio.com/ide/docs/help_with_r
```

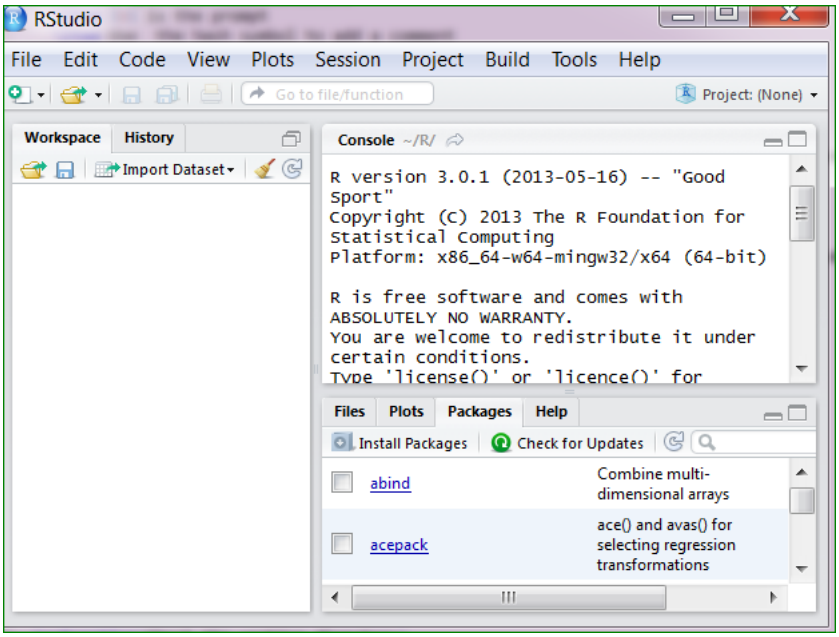

Figure 8: There are only 3 panels when you create a new project

3 The Basics for using RStudio

In the following I have tried to show some code (syntax). Anywhere where a line starts with a hash you can read that line as a comment or additional information. You will also see a hash and a comment after a command in some code.

- R syntax is case sensitive
- > is the prompt (we see this in the Console pane)
- Use the hash symbol to add a comment
- It has a panel of four panes in the working area
- Get familiar with the way things are stored (.Rdata and .R files)
- Use $<-$ or $=$ to allocate or assign a value into an R object
- This symbol is made in R by using the (Alt) key and the dash key (-) giving $<-$
- Use $<-$ to assign what is on right to the left (eg. X $<-$ 10)
- There may be more than one way to do a plot or analysis
- Get familiar with using the help
- Use Control Enter when in the code to run the script.
- When the source file has not been saved its name is in Red.
- Use the Tab key to auto-fill commands (when you need help on a particular command)

3.1 Objects in R

Unlike other software one aspect of R is the naming of objects. An object can be a number, a vector, a matrix , a dataframe, or even the output from a function.
There are also objects associated with results from regression and anova models

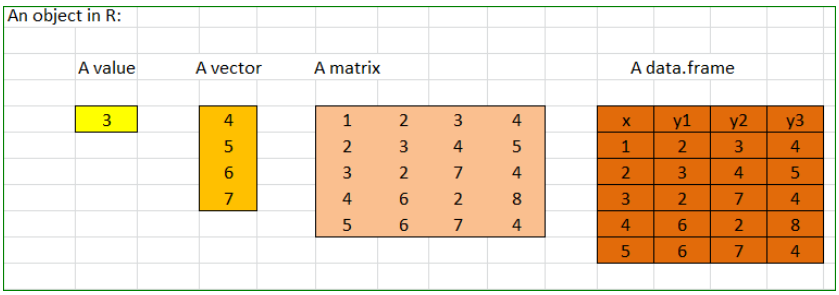

Figure 9: Objects in R

3.2 Set working directory

```
setwd("C:/Users/agmmortl/Documents/R/Kleinman")
```

3.3 Check the working directory

```
getwd()  this is for checking where you are working
```

3.4 Saving a source code file

In R when you open a Project, the first time there will only be three panels on the screen.
What you do is to go to the menu select File, New Source Code and give the file a name. It gets saved as a *.R file this is your source code.

3.5 Bring in a dataset from a .csv file

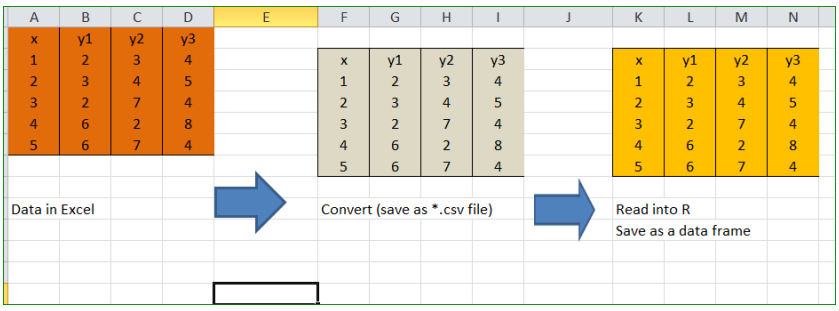

Figure 10: Concept from Excel via a *.csv file and reading into R

```
read.csv
read.table
```

```
## Reading the data from a website
octopus <- read.table("http://www.agrocampus-ouest.fr/math/
RforStat/Octopus.csv",header=T,sep=";")

## Check this out ?
file.info(filename) # Get information about files

## See also this website for importing data
http://www.statmethods.net/input/importingdata.html

Read data from *.csv

# A few options to be aware of:

Robject<- read.csv("filename", header=T, row.names=1,
                                stringsAsFactors=F, )

Robject <- read.table("filename", header, row.names=1,
                      stringsAsFactors=F, sep=,, )
```

3.6 Type data in for small datasets

```
var1 <-     c( 1,3,5,6,7, 8,12,13)
var2 <-      c( 2,4,6,7,12,10,10,12)

# If you type the name in the Consolevthe variable wil be listed

> var1
[1]  1  3  5  6  7  8 12 13
```

3.7 Use lots of comments

- Use hash (#) to make a comment

Use lots of comments throughout your work, as this reminds you of the commands and why you used them. Also it gives a good record of options and you can check your data analysis steps. If you come back to the code in six months you will remember what you did. The use of syntax means

the work is repeatable. In a click and point environment you may forget the sequence and the options that you used.
The command comment adds a comment or label to the dataset.

```
## add a comment here
## use a lot of comments to assist your understanding
## when you go back to the code
# here are some examples. You only need do it once.


comment(dataset) <-  "give it a sensible name"

comment(mydat) <-  "Soil Data from Exp 2"

comment(econdat)  <- " My profits June 2017"
```

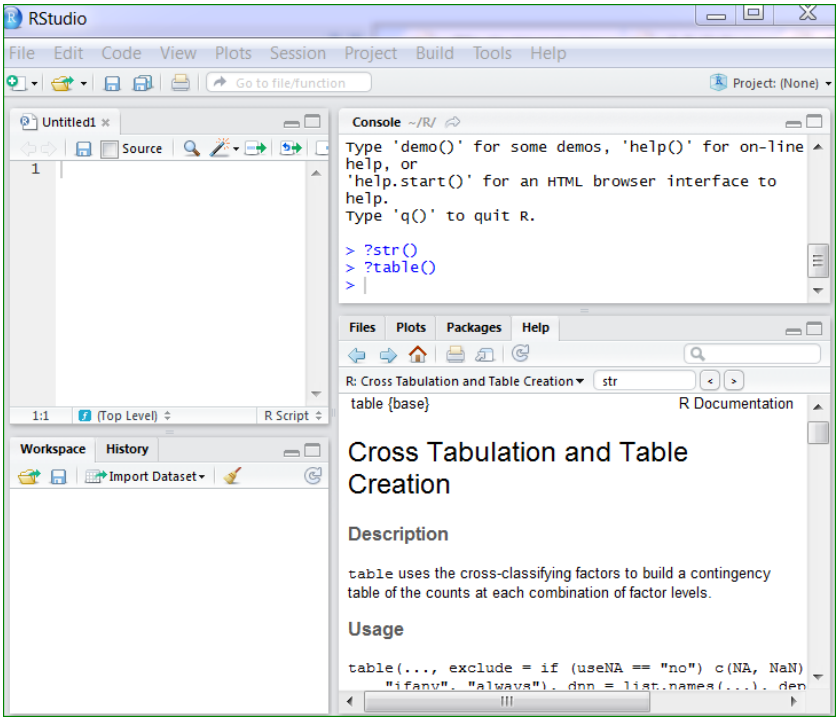

Figure 11: Using ?command() in to access help. It displays help in lower right panel

```
str(dataset[,1:10])   # str gives the structure of the dataset.

dim()   # dimensions of the dataset

head(dataset, n=5)
tail(dataset, n=5)

names(dataset)   # gives variable names
```

str() gives the structure of the dataset. Notice the variables will be integer, numerical or factor.

```
setwd("C:/Users/agmmortl/Documents/R/Kleinman")
```

4 Simple Statistics in RStudio

4.1 Basic functions

looking at some simple functions in data sets:

```
## Column means and roiw means (make sure no missing)
colMeans(Robject,na.rm=T)
rowMeans(Robject,na.rm=T)


#  List of objects in R
ls() in workspace window in Rstudio

## Remove objects in R
rm()  in workspace window in Rstudio
```

4.2 Merging data

Make sure the data is the same width or length.

```
## This is advanced and may not be used often

# Merge 2 table by column (adding columns)
Robject <- cbind(object1, object2)
Row names of the two objects have to be the same


# Merge 2 table by row (adding rows)
Robject <- rbind(object1, object2)
Column names of the two objects have to be the same
```

4.3 Basic statistics

```
mean()
var()
sd()
var()
min()
max()
quantiles()
range()
IQR()  Interquartile range
```

```
# note the use of which to find the values of interest
which(min)
which(max)
```

These will give the actual item in the dataset that is the minimum etc.

4.4 Five number summary

Three ways with slightly different results

```
quantile(x)
fivenum(x)
summary(x)
```

4.5 ECDF and qqplots

```
## These plots are for looking at the normality of the data:

ecdf()         # Empirical Cumulative Distribution Function
qqplot()       # quantile-quantile plot
```

4.6 Z Scores and scaling

These give a z score using scale or a formula expression. Notice here we are assigning the results to an object. This is a simple and it does get a lot more complicated with advanced outputs from statistical modeling.

```
scale (x)
zscoredx <- scale(x)

or

zscoredx <- (x-mean(x))/sd(x))
```

Z Scores and centering, removing the mean. Notice here we are assigning the results to an object.

```
## This  centers the data on zero, rather than the mean:

scale (x)
centre.x <- scale(x, centre=T, scale=F)
```

5 T Tests in R Studio

This does not go into the background on t tests. See the website for that. this is about the R code to do the analysis.

5.1 Data for a t test

We need to make sure we have data in two columns. It is good practice to have a column of the group and a column for the variable.

5.2 A two sample t test

```
t.test(x_1, X_2)
t.test(x ~ y)
t.test\$confit.int
```

5.3 An F test for equal variances

To do a F test which for testing for equal variance between two groups. The dataset is ds, the variable and the groups are specified in the command, along with the level of significance.

```
var.test()

# Testing the equality of variances
var.test(Variable ~ Group, conf.level=.95,data=ds)
```

5.4 Looking at options in the t tests

There are options for getting a test with equal variances, and having a one-sided test.
var.equal = TRUE option to specify equal variances and a pooled variance estimate. You can use the alternative="less" or alternative="greater" option to specify a one tailed test.

```
var.equal = TRUE

alternative="less"
```

```
alternative="greater"
```

5.5 A one sample t test

In this case we are testing where a sample has a population mean of 3. the null hypothesis is Ho: mu=3

```
# one sample t-test

t.test(y,mu=3)
```

6 Simple Plotting in RStudio

6.1 Quick look at the data frame

Check on the variables of interest.

```
str(dataset[,1:10])

head(dataset, n=5)
tail(dataset, n=5)
```

6.2 Simple table of variables

Note the table command creates tables of counts of data.
In this example 'mytable' is a new object name for the table.

```
# use a dataset with two variables
mytable <- table(A,B) # A will be rows, B will be columns

mytable # print table

margin.table(mytable, 1) # A frequencies (summed over B)
margin.table(mytable, 2) # B frequencies (summed over A)
```

6.3 Simple frequency plot

hist(data$var1)

In this case the dataset name and variable are divided by a $. This is how to request a particular variable in a dataset by name.

6.4 Simple boxplot

```
Check from slides
# to get the third column as a boxplot, from the data set x

boxplot(x[,3])




# Comparing the two sub-populations with a boxplot
boxplot(variable ~ Group, ylab="Variable",
        xlab="Group", data=ds)
```

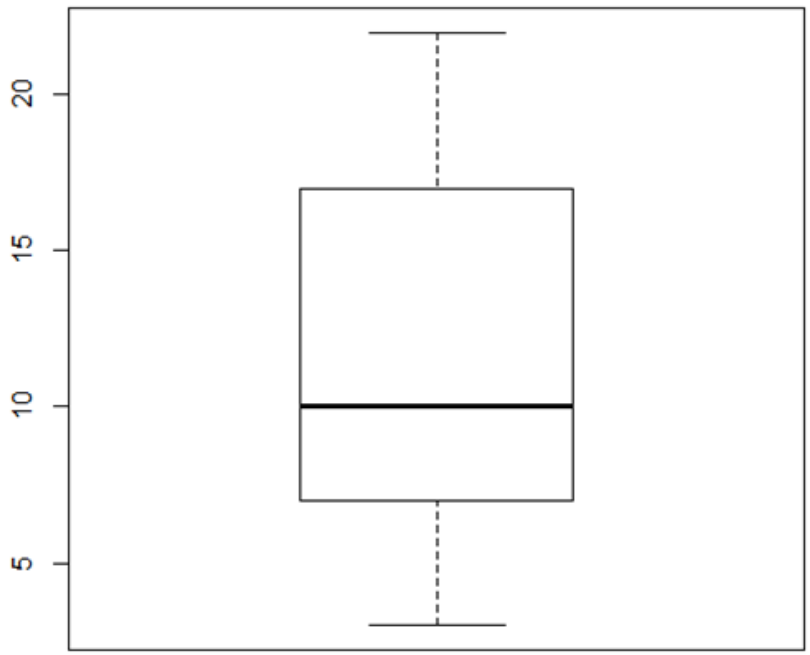

Figure 12: A simple boxplot

6.5 Simple scatterplot

```
# Scatterplots

plot(x,y, )

## other graphics parameters:

col: colour
pch: point type
lty: line type
xlab: label for horizontal axis
ylab: label for vertical axis
main: title of the graph
Other graphic parameters can found in par()
```

6.6 More complex plots

R is a very powerful potting tool. However there are tow or three key packages required to get more complex plots. typically we can use:

```
# a range of basic plots;

http://www.statmethods.net/graphs/index.html

# For complex plots we need to install and then load other packages in R
# Load package ggplot2


library(ggplot2)


# Load package lattice

library(lattice)
```

6.7 Saving your work

The way to save your session will be discussed.

7 Changing a variable to a factor

7.1 Making a Factor

Here is a simple example to show how to make a variable into a factor and how to look at the results.
A factor is a categorical variable that can be used in t test and anova to designate the groups.

```
as.factor()

## put in some data and change to a factor.
## the somersault is numeric and gets changed to a factor

somersault <- c(1:5,5:1)

# print the data
[1] 1 2 3 4 5 5 4 3 2 1

somersault.f <-as.factor(somersault)

## check the levels of the factor
## print the data from somersault.f

[1] 1 2 3 4 5 5 4 3 2 1
Levels: 1 2 3 4 5
```

7.2 Naming variables in R

Notice in this example we have a variable called **somersault**.
This is numeric and when we make it into a factor we give it a new name.

Naming in R is better using another name **somersault.f**

In R the use of a period or full-stop is used in preference to teh underscore.

Another naming convention is to use a mixture of upper and lower case Naming in R is better using another name **UpperCase** and **LowerCase**names.

This makes the reading easy. However remember R is case sensitive.

7.3 Checking a Factor

The use of factors and how they are saved can be further explored using the following:

```
levels(somersault.f)

# prints this result showing the actual levels
[1] "1" "2" "3" "4" "5"

nlevels(somersault.f)

# prints this result showing the number of levels
[1] 5

# then ask for the table of levels

table(somersault.f)

# this gives a small table

somersault.f
1 2 3 4 5
2 2 2 2 2
```

8 Further Resources

8.1 Keep current with versions of R

```
Obtain an up-to-date version from http://cran.r-project.org/.

Learn to use the help screens- by typing ?command() and
checking in the lower right quadrant.

Many commands have a data set that can be used for practice.
```

8.2 Attribution

```
citation()

To cite R in publications use:

R Core Team (2016). R: A language and environment
for statistical computing. R Foundation for
Statistical Computing, Vienna, Austria.
URL https://www.R-project.org/.

A BibTeX entry for LaTeX users is

@Manual{,
title = {R: A Language and Environment for Statistical Computing},
author = {{R Core Team}},
organization = {R Foundation for Statistical Computing},
address = {Vienna, Austria},
year = {2016},
url = {https://www.R-project.org/},
}

We have invested a lot of time and effort in creating R,
please cite it when using it for data analysis.
See also citation("pkgname") for citing R packages.
```

8.3 Plotting

```
## Also see this site
```

```
http://www.statmethods.net/graphs/index.html
```

8.4 YouTube resources

There are a number of YouTube videos that are very helpful.

```
A good start is this one:

http://youtube/MVQYgONzXho

For example this shows how to change options in RStudio.
It is a bit more advanced.

http://www.youtube.com/watch?v=5dNNcC-UBeA
```

8.5 Reading in data with different row lengths

```
## It is very useful when for some reason not all the
## rows have the same number of columns,
## it will produce an error with normal read.

read.csv(tf, fill = TRUE) # 1 column
ncol <- max(count.fields(tf, sep = ","))


read.csv(tf, fill = TRUE, header = FALSE,
         col.names = paste0("V", seq_len(ncol)))
```