

R workshop series: Introduction to R basics, data structures and graphics

Dr Cameron Hurst
cphurst@gmail.com

DAMASAC, CEU and ACRO, Khon Kaen University

7th August 2556



What we will cover....

- 1 About R
- 2 R basics
- 3 Data Structures and I/O
- 4 Basic summary statistics and plots
- 5 Getting help

Introduction

This (and the other R sessions) will be structured....:

- 45-60 minute presentation
- 90 minutes of exercises (where you will practice coding and running analysis in R)
- As we don't have time for hands-on session for every session, I will theme topics together into a R hands-on session

Introduction

What we cover in this session

- 1 About R
- 2 R basics
- 3 Data Structures and I/O
- 4 Basic summary statistics and plots
- 5 Getting help

Conventions

The conventions I will use:

Note:.....

Things to note will occur in a green box

Pitfalls:.....

Common mistakes and things to watch out for will occur in a red box

R syntax:....

All R syntax will be in purple boxes and be in `courier` font.

"Learning" syntax

Before we move onto R, I would just like to make one comment:

You are not expected to memorize R syntax!!!!

You are just expected to remember which session we covered a particular topic, and refer back to those notes (or other resources...e.g. R help files).

Hint: Programming in R

USING (not memorizing) syntax is the best way to learn it

About R.....

- R is freely available open-source software created under the GNU agreement
- It brings together the basic functionality of the creators of R (core packages) and work from others such as you and I (contributed packages)
- In this respect R is at the cutting edge in a way that the expensive competitors (i.e. SAS, Stata and SPSS) can't be

What can R do?

R can do pretty much all statistical analysis:

The basic tests (**Later today**):

- Pearson's Correlation Coefficient
- Independent and paired t-test
- χ^2 test of independence
- etc etc....

These can be a little cumbersome. Where it's real strength is in the intermediate to advanced methods (i.e. statistical modelling) .

Intermediate and advanced methods

The linear models

- General linear models (ANOVA, linear regression) (**Later...**)
- Generalized linear models (Logistic regression, Poisson Regression etc) (**Later....**)
- Cox proportional hazard regression (Survival analysis)
- Mixed models (Linear mixed models, Generalized linear mixed models and GEEs)

The linear models are R's real strength because of the approach R uses makes it very easy (if you can specify a Linear regression you can specify anything).

Intermediate and advanced methods

Other methods

- Generalized additive models (good for Environmental epidemiology)
- Multivariate methods (Principal component analysis, Partial least squares, clustering and many more)
- Time series analysis methods
- Analyses for particular study types (e.g. Genetic/genomic data, environmental epidemiology etc)

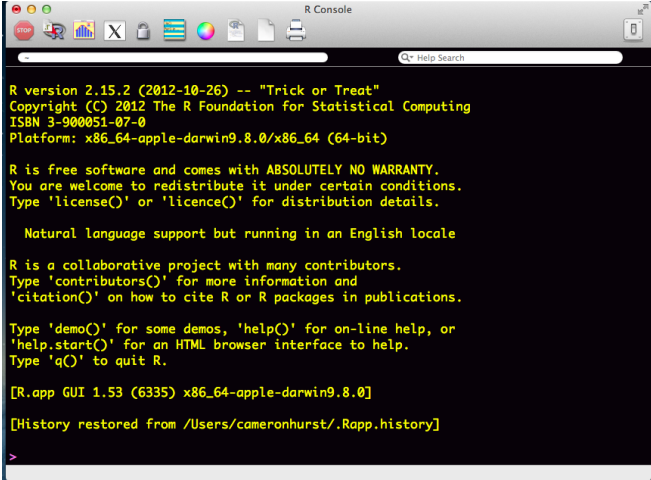
Graphics

- R also produces excellent publication quality graphics
- It provides a wide range of graphs; and
- (once your R coding is good enough) all statistical method and graphs can be easily extended or enhanced

Where can I get R?

- R can be downloaded (for free) from `http://cran.r-project.org/`
- This web site provides both the base (core) and contributed (written by others) packages.
- It is available on Windows, Linux/Unix and Mac platforms

R:Screenshot



The screenshot shows the R Console window on a Mac. The title bar reads "R Console". The menu bar includes "STOP", "R", "Help Search", and a magnifying glass icon. The main content area has a black background with yellow text. The text displays the R version (2.15.2), copyright information (© 2012 The R Foundation for Statistical Computing), ISBN (3-900051-07-0), and platform (x86_64-apple-darwin9.8.0/x86_64 (64-bit)). It also includes a disclaimer about the warranty and a list of useful commands like 'license()', 'demo()', 'help()', and 'q()'. At the bottom, it shows the R.app GUI version and the restored history file path.

```
R version 2.15.2 (2012-10-26) -- "Trick or Treat"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.53 (6335) x86_64-apple-darwin9.8.0]

[History restored from /Users/cameronhurst/.Rapp.history]

>
```

Our first R session

Let's start an R session....

- As you can see you are presented with a blank screen.
- We can use R interactively
- I can type something in:

```
> x <- 1.3  
> y <- 2.5  
> x+y
```

```
[1] 3.8
```

Interactive or script file?

Using R interactively (typing in individual lines) is generally not the best way to use it.

As you get better you will want to start collecting your code into syntax files (".R" extension).

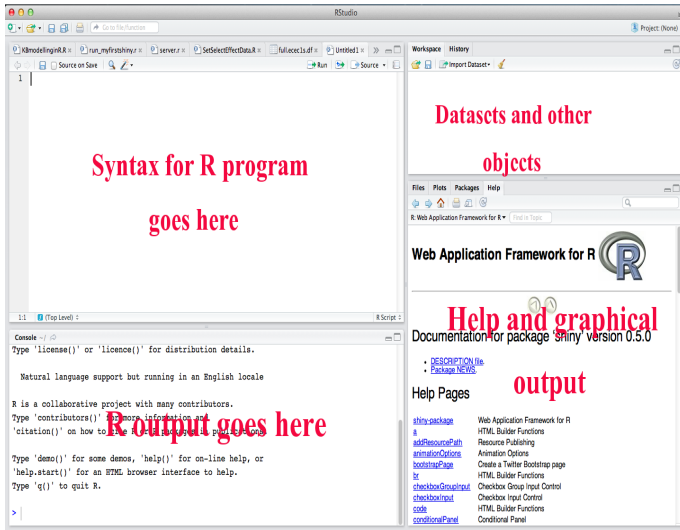
R comes with a (VERY) basic script file editor, but there are a few much better ones (freely) available. The ones I have used are:

- tinn-R (version 1.17.2.4 later versions are unstable)
- notepad++
- R-studio (preferred)

Hint:

R studio can be downloaded from www.Rstudio.com

R studio:Screenshot



Objects

R works on **objects**. These include:

- single number or character
- vector or 1-D array of numbers, logical values, or factors etc
- matrices (tables of values of a **single** type)
- **Most important:** Data frames (table allowing variables of different types) ~ SAS, Stata or SPSS dataset
- lists (data frames are a special case of a list)
- functions and user defined objects
- analyses

Hint:

It is a good idea to include type in object name e.g.
`mydataframe.df` and `mymatrix.mat`

Naming objects

Object names tend to use letters, numbers and periods
Values (or functions) are assigned to an object using the `<-` operator. For example: `my.val <- 7` stores the value of 7 into the object `my.val`

Aside:

The character `=` can also be used to assign values to an object, but `<-` is much more widely used.

Warning/Pitfall:

- ▶ Avoid underscores in names. While technically legal, it can cause problems
- ▶ Note that the operator for "equals to" is `==`, not `=`

Some basic data structures

```
> A.val<-7  
> A.vec<-c(1:6)  
> A.mat<-matrix(A.vec, nrow=2, byrow=TRUE)  
> Alogical.vec<-A.vec <4
```

To find out what is in each object, just type name at prompt

```
> A.val
[1] 7
> A.vec
[1] 1 2 3 4 5 6
> A.mat
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> A2.vec
[1] TRUE TRUE TRUE FALSE FALSE FALSE
```

Some examples of objects

A very useful function for you to know is the `class()` function. This tells us what type of object we are dealing with. For example:

R syntax: Determining object 'type'

```
class(A.mat)
```

will return

```
[1] "matrix"
```

Data input and data frames

- The most commonly used data structure you will use in R is the **data frame**
- The data frame represents a matrix-like object where the columns represent variables and the rows observations
- Unlike matrices however, data frames are allowed to have columns of different data types (e.g. columns 1 might be numerical, and column 2 a factor etc.)
- Rarely will we manually enter the data directly into R, it is much more convenient to read it in using a data file

Data input and data frames

Motivating example

Consider a dataset containing 200 patients on which the following variables are measured:

- 1 id (Patient ID)
- 2 age
- 3 chol (Cholesterol level)
- 4 sbp (Systolic blood pressure)
- 5 dbp (diastolic blood pressure)
- 6 ses (Socio-economic status <- education and income)
- 7 bmi (body mass index)

These data are currently stored in an excel comma delimited (csv) file. Note: A few more composite variables have been provided for your convience.

Cholesterol data in csv file

Motivating example

Cholestrol.csv - Microsoft Excel

Home Insert Page Layout Formulas Data Review View

Clipboard Font Alignment Number Styles Cells Editing

K10 fx

	A	B	C	D	E	F	G
1	id	age	sbp	dbp	chol	ses	bmi
2	1	42	110	65	291	2	25.0324
3	2	53	130	72	278	1	24.46602
4	3	53	120	90	342	4	31.60108
5	4	48	120	80	239	4	30.81169
6	5	53	118	74	243	3	21.45665

Cholestrol

Ready 100%

Cholesterol data in csv file (Windows OS)

Motivating example

To read in the csv file and dump it to a dataframe:

R syntax: Reading in data

```
mychol.df<-read.csv("E:/Introduction to R/Basics/Chol.csv")
```

Warning/Pitfall: Common error

Note the use of the forward slash "/" rather than the traditional backslash, "\". Note that a double backslash "\\" works, too

Alternatively we can set up a working directory (much easier):

R syntax: Reading in data

```
setwd("c:/Introduction to R/Basics")  
mychol.df<-read.csv("Chol.csv")
```

Mac users: Cholesterol data in csv file

Motivating example

Mac users: remember to use the mac directory structure: E.g.

R syntax: Reading in data for Mac users

```
setwd("/Users/cameronhurst/Documents/")  
mychol.df<-read.csv("Chol.csv")
```

Documenting your code: The importance of comments

If you want to leave yourself "notes" about your syntax use #

R syntax: Reading in data (with comments)

```
# I can write anything I want here, blah blah  
# Set directory for reading and saving files  
setwd("c:/Introduction to R/Basics")  
  
# Read in a comma delimited text file  
mychol.df<-read.csv("Chol.csv")
```

Hint: Documenting your code

Documenting your code has two MAJOR advantages:

- ❶ You (or other people) can understand what you have done
- ❷ If you do it well, it is a 'code' for your methods section

Reading in data from other stats packages

We can also read data in SPSS (.sav), Stata (.dta) and other packages using the **foreign** R library.

E.G. To read in the same data above from a **Stata** data file:

R syntax: Reading in data from other stats packages

```
library(foreign)
setwd("E:/Introduction to R/Basics")
mychol.dta.df<-read.dta("Chol.dta")
```

Warning/Pitfall:

Avoid using this approach. OK for simple datasets, but tricky for other types of data (e.g. data formatted for survival analysis). Best to export data from your stats package (or using excel) into a comma delimited (.csv) format

Aside: Libraries

- Most of the analysis you will perform when you start with R will use methods available in the libraries that come with the base distribution of R (core libraries)
- For example, most statistical analysis will be from the libraries `stats` or `MASS`, and most graphics will employ the `graphics` library
- These standard automatically libraries load into to memory when we start up R

Aside: Libraries

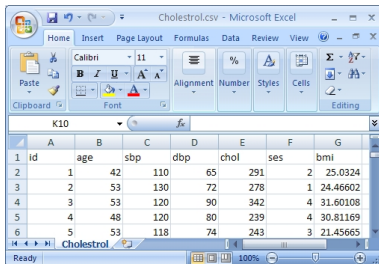
- As you use more specialized and advanced analysis you will have to load libraries into memory before you can call their methods
- Some libraries (e.g. `foreign`) are provided with the base distribution of R, but still need to be loaded into memory
- Other libraries need to be downloaded from repositories (e.g. CRAN) first

For example, if you wanted to analyse data arising from a longitudinal study, you might use Linear Mixed Models from the library `lme4`. In this case you would have to download this library from CRAN and then start your script file (the preamble) with the command:

```
library(lme4)
```

Cholesterol data in csv file

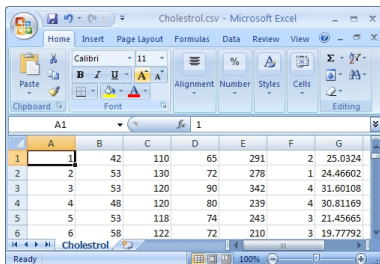
Now if we go back to our datafile, we note that there are column headings.



	A	B	C	D	E	F	G
	id	age	sbp	dbp	chol	ses	bmi
1	1	42	110	65	291	2	25.0324
2	2	53	130	72	278	1	24.46602
3	3	53	120	90	342	4	31.60108
4	4	48	120	80	239	4	30.81169
5	5	53	118	74	243	3	21.45665

What about if we don't have column headings.....

Cholesterol data in csv file



	A	B	C	D	E	F	G
1	1	42	110	65	291	2	25.0324
2	2	53	130	72	278	1	24.46602
3	3	53	120	90	342	4	31.60108
4	4	48	120	80	239	4	30.81169
5	5	53	118	74	243	3	21.45665
6	6	58	122	72	210	3	19.77792

R syntax: Reading in data without column headers

```
setwd("c:/Introduction to R/Basics")  
mychol.df<-read.csv("Chol.csv", header=FALSE)  
  
# No column headings so need to include  
names(mychol.df)<-c("id", "age", "sbp", "dbp",  
"ses", "bmi")
```

Viewing data in data frames and matrices

To view the first few lines in our data frame:

```
> head(mychol.df)
```

To view the last few lines in our data frame:

```
> tail(mychol.df)
```

To view whole dataset, type the name of the object at prompt:

```
> mychol.df
```

Or to view entire dataset in new window:

```
> edit(mychol.df)
```

or in Rstudio, just double click datafrae in workspace to see data

Warning/Pitfall:

Avoid using the `edit()` command in a script file, it stops the program until edit window is manually closed

Indexing

One of the best features in R is how object indices can be used to subset the data. For example:

- If we only want the first three columns:
`subset.chol.df<-mychol.df[,c(1:3)]`
- If we only want the rows 10 to 20:
`subset.chol.df<-mychol.df[c(10:20),]`
- If we only want all the rows **except** 10 to 20:
`subset.chol.df<-mychol.df[-c(10:20),]`
- If we want all patients in the 2nd socio-economic group:
`subset.chol.df<-mychol.df[mychol.df[,6]==2,]`
or...
`subset.chol.df<-mychol.df[mychol.df$ses==2,]`

Hint:

- ▶ R uses the `my.df[row, column]` format
- ▶ Columns can be referenced by col number, or name
- ▶ The absence of something before(after) the `,` means all rows(columns)

Generating plots

- R is very strong regarding plots (with thousands available from the various libraries)
- Let's start with a basic histogram
- Using our cholesterol data:

R syntax: Generating a histogram

```
hist(mychol.df$chol)
```

This will give us the very basic plot....

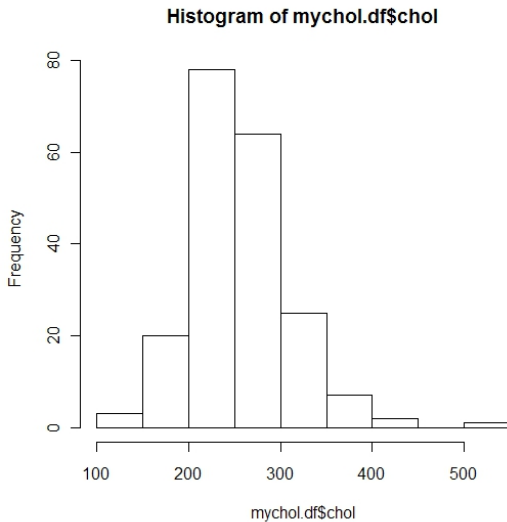
The \$ symbol

We can view the \$ symbol as 'containing' so:

```
mychol.df$chol
```

means that the cholesterol variable is 'contained in' the chol.df data frame

A very basic plot

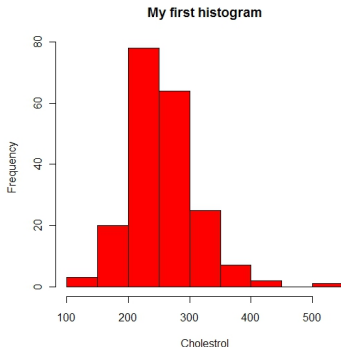


A bit better histogram

Now let's put some more features in the graph

R syntax: 'Pretty up' the histogram

```
hist(mychol.df$chol, main="My first  
histogram", xlab="Cholestrol", c="Red")
```

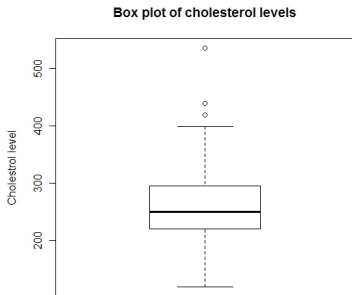


A simple boxplot

Similarly, to generate a simple boxplot..

R syntax: Simple box (and whisker) plots

```
boxplot(mychol.df$chol, main="Box plot of  
cholesterol levels", ylab="Cholesterol level")
```



Bivariate plots: Side-by-side boxplots

Here, I collapsed BMI into BMI classes (underweight, normal, overweight/obese)

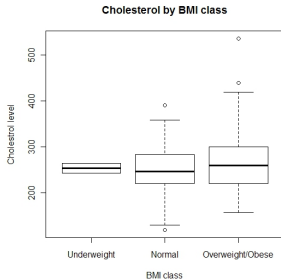
R syntax: Bivariate side-by-side boxplots

```
boxplot(chol~bmi.3class, data=mychol.df, main="Cholesterol by  
BMI class", ylab="Cholestrol level", xlab="BMI class")
```

Note the use of:

`chol~bmi.3class`

This is a very important feature in R. It is the formula format and you will see **much** more of it.



A basic scatter plot

R syntax: Scatterplots

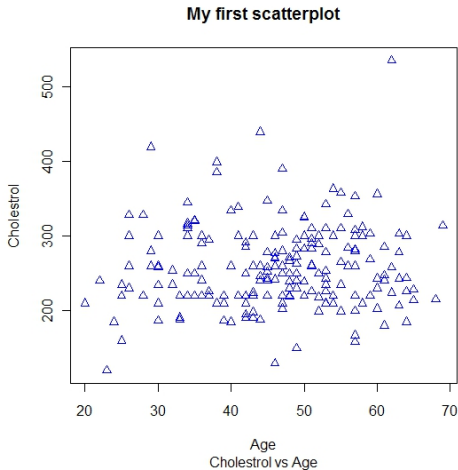
```
plot(x=age, y=chol, data=mychol.df, main="My first scatterplot",  
sub="Cholestrol vs Age", pch=2, col="Blue", xlab="Age",  
ylab="Cholestrol")
```

Hint:

- ▶ R doesn't care what order you specify function parameters
- ▶ If you don't specify a parameter, R will use the default
- ▶ `pch` stands for plot character
- ▶ By using `data=mychol.df` I don't have to prefix variable names with `chol.df$`
- ▶ Above, I included features not needed. The only things needed for scatter plots is `plot(x=myxvar, y=myyvar)`

The above R code chunk gives

A basic scatter plot



Some summary statistics for quantitative data

Centre

```
mean(mychol.df$age)
46.065
median(mychol.df$age)
47
```

Variability

```
sd(mychol.df$age)
10.81274
range(mychol.df$age)
20 69
```

Five number summary (min Q25, Median, Q75, Max)

```
fivenum(mychol.df$age)
20 38 47 54 69
```

More summary statistics

To get summary statistics by group....

```
# Mean age underweight  
mean(mychol.df$age[bmi.3class==1])  
# Mean age normal weight  
mean(mychol.df$age[bmi.3class==2])  
# Mean age overweight/obese  
mean(mychol.df$age[bmi.c3lass==3])
```

Hint:

There are libraries in R that provide a much better way of doing this (e.g. the `describeBy` function in the `psych` library. I just want to keep it to the libraries that come WITH R (don't need to be downloaded)

Summarizing categorical data

Univariate

Frequency table for a categorical variable:

R syntax: 'Univariate' frequency tables

```
table(mychol.df$bmi.3class)
```

Gives:

Underweight	Normal	Overweight/obese
2	93	105

Summarizing categorical data

Bivariate

Now to generate a cross-tabulation:

R syntax: 'Bivariate' cross-tabulation

```
# Cross tabulate BMI class by socio economic  
status table(mychol.df$bmi.class,  
mychol.df$ses)
```

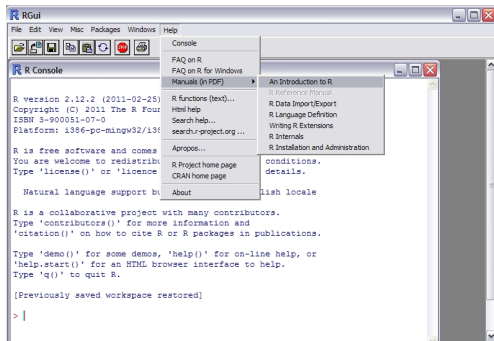
Gives:

BMI class	low	low-mid	mid	high-mid	high
Underweight	0	1	1	0	0
Normal	10	15	55	5	8
Overweight/obese	12	19	47	13	14

R for beginners

- R can be quite tricky when you first begin
- One of the ways to make this much easier is to understand and use the help
- I will talk about three main aspects of the R help system:
 - ① free introductory text
 - ② searching help
 - ③ HTML package help

Free introductory text



Searching help

Searching the help will identify methods associated with a key word. Two ways of using:

- ① *Help* → *Search help....* → *type in keyword*
- ② type in `help.search("t test")` This will identify libraries with this (or similarly named) functions

When we know the function name

Finally, if we know the function name (e.g. `t.test`) and want help in its properties use:

- `help("myfunction")` or `?("myfunction")`
For example: `help("t.test")`
- If we know what package the function comes, we can go to:
Help -> HTML help -> packages, and then we would choose the appropriate package (when you are more familiar with R, this will be your most common approach)

Getting R help in R-studio

As I have mentioned previously using R studio makes life a lot easier (it makes navigating and viewing different R windows much easier).

This is also true for help. You can look at your help window while seeing your syntax output etc.

I will show you how to do this in our hands-on session this afternoon.

Other help....

R is a VERY widely used package and there are **many online resources** for this package including:

- 1 Youtube walk-throughs (where people will actually show you how to do things)
- 2 Free text books and wiki-books (I am happy to distribute any of these I have if people are interested...especially those who want to become serious about R)
- 3 Help sites. I rarely come across a problem that somebody else hasn't encountered (and fixed) before. I usually just google my problem (an art to using the right words) to get to the right help page.

A final word...

- R is a comprehensive and comparatively 'low-level' language
- For those of you not used to coding/programming mastering R will take a little while
- However, it is free (open source), extendible, and is one of the strongest and most versatile statistical packages available
- For this reason it is the package of choice among many statisticians (few people look back after converting over to R)

THANK-YOU!!

Questions??