Introduction to R and Rstudio

Andrew Parnell

28 June 2016

Introduction

Welcome to Practical 1, an introduction to using R and Rstudio. In this practical we will:

- Learn some of the basic R commands and Rstudio workflow
- Learn how to load data and run some simple summary statistics
- Learn how to create some simple plots
- Install packages and extend the functionality of R

You should follow and run the commands shown in the grey boxes. At various points you will see a horizontal line in the text which indicates a question you should try to answer, like this:

Exercise
What words does the following command print to the console?

```
print("Hello World")
```

Make sure to store all your answers in a `.R` script file (more on how to do this below).

Top 10 things for a newbie to know about R and Rstudio

1. R is the underlying programme which does all the hard stuff. Rstudio is the front-end which makes programming in R easier.
2. Rstudio has 4 windows. The two most important are the *Console* window, where you can type commands directly in and hit Enter (try typing `2+2` in the console window), and the *Source* window which is where you can create your own .R files which store all your commands.
3. The two lesser used windows are the Environment/History window which shows the objects you have created and the previous commands you've typed into the Console window, and the Files/Plots/Packages/Help/Viewer window that has lots of tabs and contains exactly what you'd expect from the labels!
4. The quickest way to get properly started with Rstudio is to go to File > New File > R script. This will create a blank R script for you to start working on. Type your commands in here line by line (try e.g. `x = 2 + 2` and then `print(2 * x)` on the next line), then highlight the text and click on Run at the top middle of the window. Congratulations - you have just run your first R script file!
5. You can put comments in your script file with the `#` command. It's a good idea to put comments everywhere in your script file
6. To get help with a command, type `?command` in the Console window. If you don't know the name of the command type `??command`

7. Most lines in your script file will look like `object = function(stuff)` where `stuff` is some data you want to manipulate, `function` is some set of operations you want to perform, and `object` is the where the output will be stored
8. Some good web links to get started include: QuickR, a really nice reference website for R, and StackOverflow which has loads of code for more difficult queries.
9. The most useful function for reading in data is `read.table`, which will read in almost any standard format (tab-separated, comma separated, etc)
10. R becomes most useful when you load in other people's packages. There are a huge number of them listed here. They cover almost everything that anyone might want to do with data, and all contain manuals and help pages.

Exercise 1

You can use the function `read.csv` to load in the prostate data with:

```
prostate = read.csv('https://goo.gl/ntQuXO')
```

What do the functions `str` and `head` do when applied to the `prostate` data object?

Accessing parts of a data set

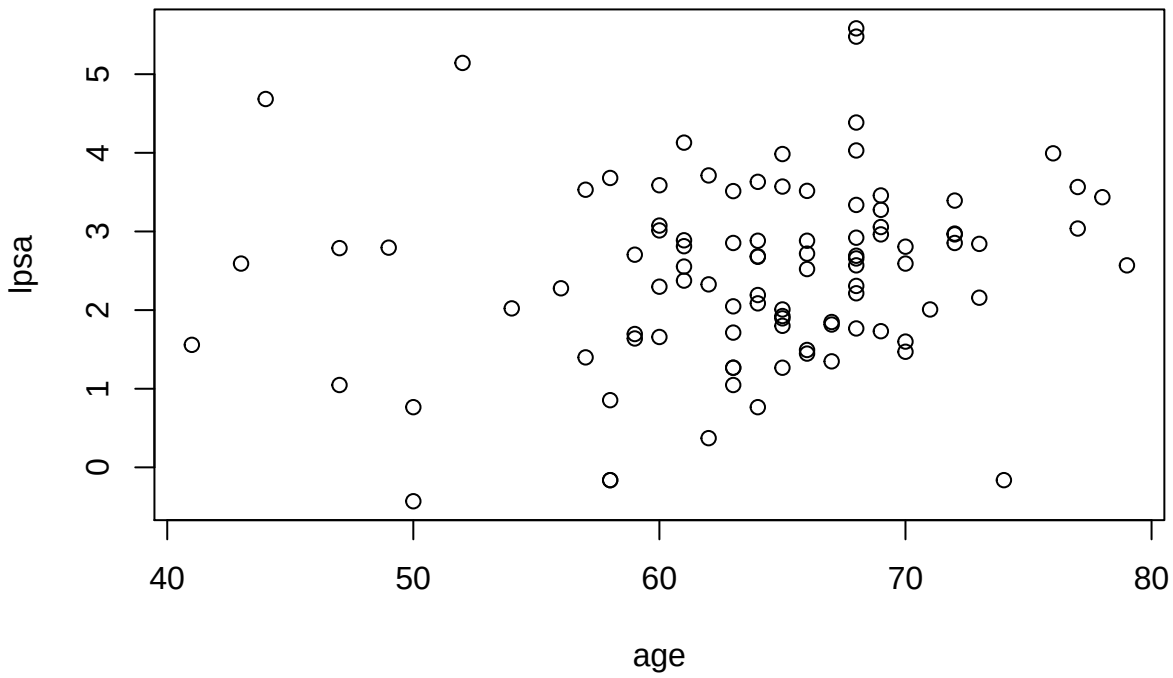
You can use a number of ways to index into a data frame (and other objects) in R. For example:

```
lpsa = prostate$lpsa
lpsa = prostate[,9]
lpsa = prostate[,'lpsa']
```

should all give the same result. The first type uses the `$` notation to access a variable by name. The square brackets in the second and third example allow greater control, for example `prostate[1:10,2:3]` selects the first 10 rows and the columns 2 and 3.

A nice way to refer to the data in an object while keeping code easy to understand is using the `with` function. For example, you can plot the data combining the `plot` and the `with` functions:

```
with(prostate, plot(age, lpsa))
```



Exercise 2

Have a look at the help file for plot (via `?plot`). What happens when you add in the extra argument `xlab = "Age"` to the plot command? See if you can add in y-axis labels too.

Running simple summary statistics on a data set

R has many useful functions for creating simple summary statistics. Perhaps the simplest is `summary`:

```
summary(prostate)
```

```
##      lcavol           lweight           age             lbph        
##  Min.   :-1.3471   Min.   :2.375   Min.   :41.00   Min.   :-1.3863  
##  1st Qu.: 0.5128   1st Qu.:3.376   1st Qu.:60.00   1st Qu.:-1.3863  
##  Median : 1.4469   Median :3.623   Median :65.00   Median : 0.3001  
##  Mean   : 1.3500   Mean   :3.629   Mean   :63.87   Mean   : 0.1004  
##  3rd Qu.: 2.1270   3rd Qu.:3.876   3rd Qu.:68.00   3rd Qu.: 1.5581  
##  Max.   : 3.8210   Max.   :4.780   Max.   :79.00   Max.   : 2.3263  
##       svi              lcp             gleason          pgg45       
##  Min.   :0.0000   Min.   :-1.3863   Min.   :6.000   Min.   :  0.00
```

```
##  1st Qu.:0.0000   1st Qu.:-1.3863   1st Qu.:6.000   1st Qu.:  0.00
##  Median :0.0000   Median :-0.7985   Median :7.000   Median : 15.00
##  Mean   :0.2165   Mean   :-0.1794   Mean   :6.753   Mean   : 24.38
##  3rd Qu.:0.0000   3rd Qu.: 1.1787   3rd Qu.:7.000   3rd Qu.: 40.00
##  Max.   :1.0000   Max.   : 2.9042   Max.   :9.000   Max.   :100.00
##       lpsa           train
##  Min.   :-0.4308   Mode :logical
##  1st Qu.: 1.7317   FALSE:30
##  Median : 2.5915   TRUE :67
##  Mean   : 2.4784   NA's :0
##  3rd Qu.: 3.0564
##  Max.   : 5.5829
```

If we want more detailed information we can use any of the following functions: `mean`, `sd`, `median`, `IQR`, `range`, etc, etc.

Some functions (like `plot` above) have multiple *arguments*. For example, the `quantile` function allows us to create percentiles:

```
quantile(prostate$lpsa, prob=c(0.05,0.95))
```

```
##       5%      95%
## 0.686687 4.180670
```

Exercise 3

Try out some of the functions listed above on the different variables. What is the median of `age`? What is the standard deviation of `lcavol`? What is the interquartile range of the `gleason` grade?

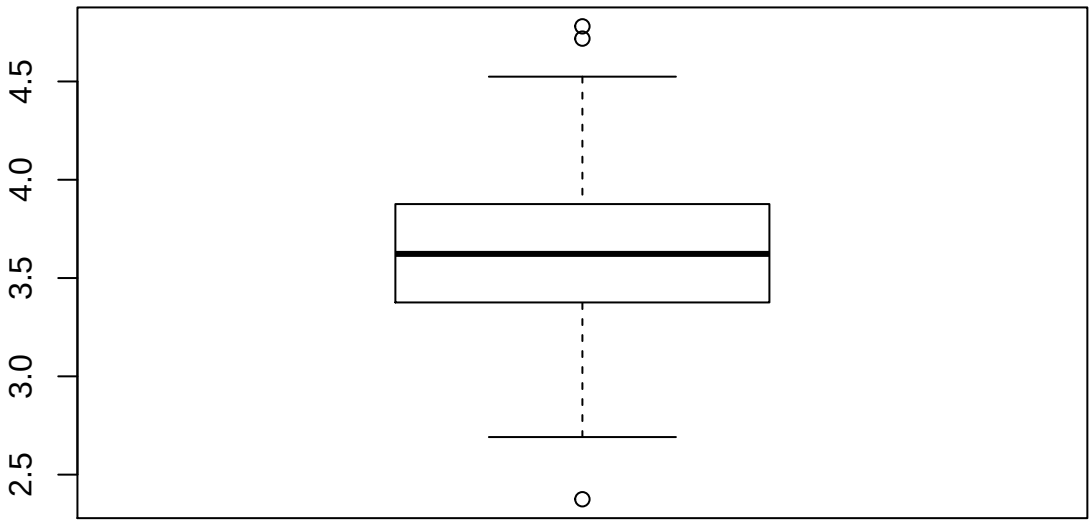
Plots

We saw above the `plot` function to create simple scatter plots. There are two other simple plots which are very useful. The first is `hist` which creates a histogram:

```
hist(prostate$lbph)
```

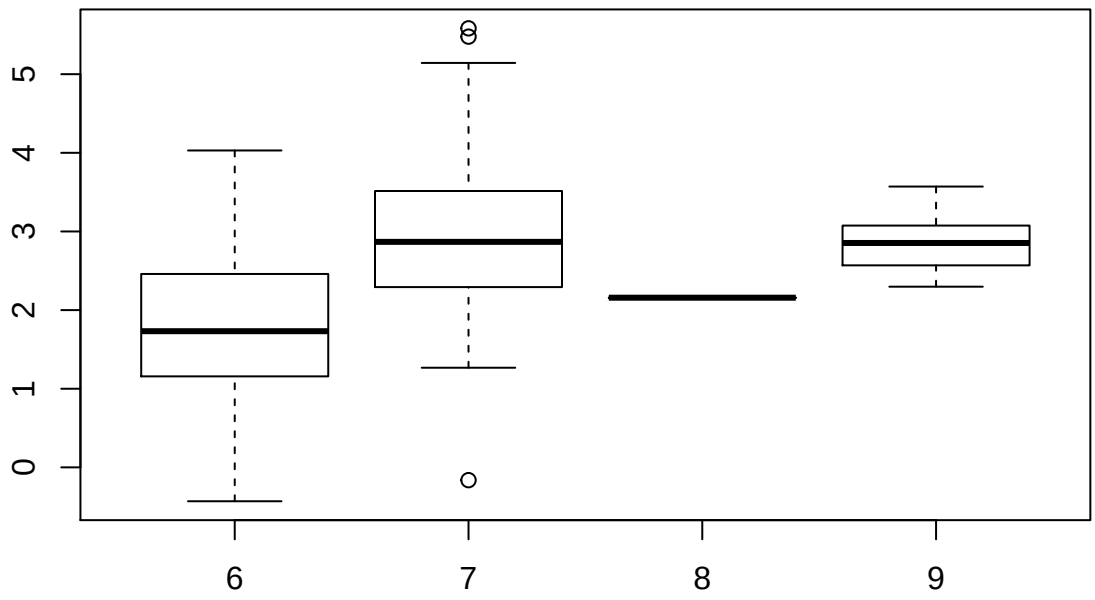
The second is `boxplot`:

```
boxplot(prostate$lweight)
```



A nice thing about the boxplot function is that you can give it a formula instead of a single variable:

```
boxplot(lpsa ~ gleason, data = prostate)
```



The first variable gives the values we want to plot and the second gives the groups for the different boxes.

Exercise 4

Look at the help file for `hist` and try to add some extra arguments in. Useful ones include `breaks=30`, `freq=FALSE`, and `main="X"`. What do these all do?

Loading in other packages

Sometimes we want to use functions that other people have written in their own packages. We can load in their packages withe the `library` command, e.g.

```
library(MASS)
```

Sometimes we'll get an error if we're trying to load in a package that we haven't downloaded yet, in that case use:

```
install.packages('package_name')
```

R will then download it and, assuming all goes well, `library(package_name)` will then load in the functions from that package.

Exercise 5

Visit the R package web page and see if you can find a package that interests you. Install it and have a look at the manual (this will be on the web page). See if you can follow or run some of the examples in the manual.
