Data Science Lab 1: Introduction to R

michaelferrie@edinburghcollege.ac.uk

1.1 Introduction

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories by John Chambers and colleagues.

R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control. R is cross platform and available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form.

1.2 R Installation

In order to install and work with R there are two main components needed, firstly the R language which is called the R-Base. You can think of this as the underlying code that makes R work. Once we install this we can use R with the command line only, which is fine if we are running code on a server but since we want to be able to work with R on our home PC, it is useful to have an editor. There are a variety of editors available for R, in this course we will use R-Studio. Follow these steps to install R on your operating system

Windows: Download R for Windows here

- Run the .exe file
- Go to the RStudio Download page
- Download RStudio Desktop Free
- Double click the file to install it
- Once R and RStudio are installed, open RStudio to make sure that you don't get any error messages.

Mac OS X: Go to CRAN and click on *Download R for (Mac) OS X*.

- Select the .pkg file for the version of OS X that you have and the file will download.

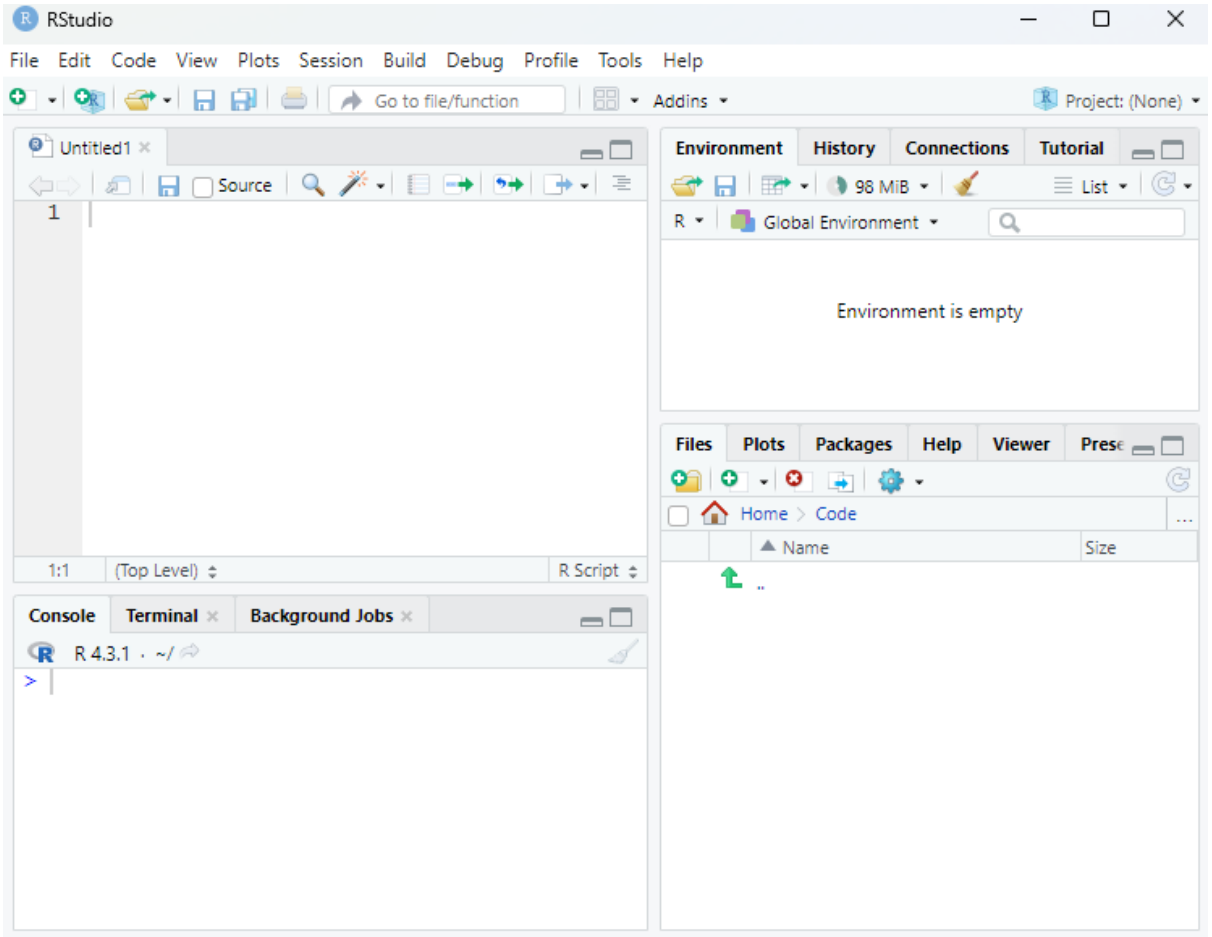
- Double click on the file that was downloaded and R will install.
- Go to the RStudio Download page.
- Under *Installers* select **RStudio *current version* ## - Mac OS X 10.6+ (64-bit)** to download it. Once it's downloaded, double click the file to install it.
- Once R and RStudio are installed, open RStudio to make sure it works and you don't get any error messages.

GNU/Linux: R is available through most Linux package managers. You can download the binary files for your distribution from CRAN. Or use your package manager (e.g. for Debian/Ubuntu run `sudo apt install r-base` and for Fedora run `sudo yum install R`).

- To install RStudio, go to the RStudio Download page.
- Under *Installers* select the version for your distribution.
- Once it's downloaded, double click the file to install it.

1.3 Getting Started

Once R and RStudio are installed, open RStudio to make sure that it works and there are no errors when you open it. In RStudio select the File menu and choose "New File" then "R Script". You should see the following four windows.



Top left **Code Editor** here you can open or save a program you are writing.
Bottom left **Console** type commands in here and R will interpret them.

Top right **History** keeps track of events while R studio is running.
Bottom right **Plots and Files** the output of R with the help function and file navigator

2.0 Let's learn R

Use the Console (bottom left) to enter the following commands and see how R interprets them.

Type the following into the console and then press the return key:

```
print("hello world")
```

R should return the following:

```
[1] "hello world"
```

A more detailed explanation of what is happening here. We are using the RStudio IDE (Integrated Development Environment). An IDE is a code editor with additional features to make software development easy.

We have written a single line R program, and when we use the `print` keyword and `()` parentheses. We are sending an argument to the print function. The print function is useful because it prints the text between the quotation marks out as text to the console.

When we run the program we are sending it to be interpreted by the R interpreter which then returns our output. We can use the R language to send more complex instructions to the interpreter, however this is a great start. It is important to understand that R is an interpreted language, meaning it is read one line at a time, in the order that you write it.

2.1 An R quiz

Open a word document or with a pen and paper, create a table like this:

1	5
2	6
3	7
4	8
9	10

Answer the following 10 questions and add your answer A, B, C or D to the table, enter the input values one line at a time and see what your final output is.

Q1 What is the output of the following example of the numerical date type?

```
> x <-12
> y <-4
> x / y
```

A. 3
B. Error
C. 4
D. [1] 3

Q2 What is the output of the following?

```
> age <- 87
> age + "2"
```

A. 3
B. Error
C. [1] 89
D. [1] 3

Additional information: Everything in R is an object - here are the basic data types and examples of each. Character - strings of letters or numbers "a", "swc" (strings always go in quotes). Numeric (real or decimal) 2, 15.5. Logical - boolean TRUE, FALSE (always written in capital letters).

Q3 What is the output of the following?

```
> x <- 3
> x <- (3 / 4)
> x
```

A. [1] 0.75
B. Error
C. 4
D. [1] 3

Additional information: Comments are ignored by R and are written with a # symbol, an object type can be checked using the `typeof()` function, another useful operator is the modulo `%%` which gives the remainder of a division. Use this information to help answer the next questions.

Q4 What is the output of the following?

```
> # Example
> x <- "dataset"
> typeof(x)
```

A. [1] 0.75
B. [1] “character”
C. 4
D. [1] 3

Q5 What is the output of the following?

```
> x = 10 %% 7
> x > 5
```

A. [1] "logical"
B. [1] FALSE
C. [1] TRUE
D. [1] 7

Q6 What is the output of the following?

```
> x = 10 %% 7
> x > 5
> typeof(x > 5)
```

A. [1] "logical"
B. [1] FALSE
C. [1] TRUE
D. [1] 7

Q7 What is the output of the following?

```
> x = 10 %% 7
> x > 5
> typeof(x > 10)
```

A. [1] "logical"
B. [1] FALSE
C. [1] TRUE
D. [1] 7

Q8 What is the output of the following?

```
> x = 7 %% 3
> x
```

A. [1] 3
B. [1] 2
C. [1] 4
D. [1] 1

Additional information: We can create *vectors*, the c character concatenates (joins things together). Use this information to help answer the following questions.

Q9 What is the output of the following?

```
> z <- c("Bob", "Alice", "Eve")
> z <- c(z, "Mike")
> z
```

A. [1] "Bob" "Alice" "Eve" "Mike"
B. [1] "Mike" "Bob" "Alice" "Eve"
C. [1] "Mike"
D. [1] z

Q10 What is the output of the following?

```
> z <- c("Bob", "Alice", "Eve")
> z <- c("Mike", z)
> z
```

A. [1] "Bob" "Alice" "Eve" "Mike"
B. [1] "Mike" "Bob" "Alice" "Eve"
C. [1] "Mike"
D. [1] z

2.2 More R basics

Hopefully the quiz got you used to the input and output of R as well as introducing some new concepts. If you ever need help with an R function you use the inbuilt help function. To get help with the `print` function enter the following:

```
help(print)
```

Or the `typeof` function or the combine function, try these out:

```
help(typeof)
help(c)
```

In R we can create lists of data called vectors, create a vector called my_numbers and then use the sum function to add them up, again - if we wanted help with sum we would just type:

```
help(sum)
my_numbers <- c(2,3,4)
sum(my_numbers)
```

Plot the vector, R is the language of data science and can easily visualise any data.

```
plot(my_numbers)
```

Use the vector to create a barplot:

```
barplot(my_numbers)
```

Add some details to the barplot, a title and a label to the x-axis with xlabel and make the colour red, modify the colour yourself, try making it green or blue:

```
barplot(my_numbers, main="My Numbers", xlab = "Total Numbers",
col="red"
```

2.2 Plotting vectors in R

So far we have only been entering commands into the console, which is good for learning but let's say we want to have a program that is more than one line. For this we should use the code editor (top left window).

In the code editor enter the following two lines to create two new vectors.

```
students <- c("Bob", "Alice", "Eve", "Mike")
exam_score <- c(60,70,80,90)
```

When we are entering code from the code editor RStudio lets us choose which line we start running the code from. With the mouse click to any point on line 1, to run this line press CTRL+ENTER.

You will notice that R runs the line and jumps to the end of the next line, so enter that in the same way. Also observe the events window in the top right of R studio, you can see that helpfully we have each of our vector names and the values displayed there, this is useful when working with larger datasets. Add a 3rd line to the R script to visualise the two vectors.

```
barplot(exam_score, names.arg = students, col ="darkgreen")
```

Now run this line to see the plot in the plots window, let's improve the plot, edit line 3 to look like the this:

```
barplot(exam_score, names.arg = students, xlab="Student Exam
Scores", col=c("darkblue","red", "darkgreen", "yellow"))
```

A more detailed explanation of this line - we are running the barplot function, and sending the following arguments to the function. The exam_score vector which is numeric, this will be used to make the bars in the barplot. Then we are using names.arg to give each of the bars a name, we can send the students vector to names.arg as our list of names. Each argument is separated by a comma. The xlab keyword sets a label for the X-axis and then we can create a vector of colours with the col keyword.

2.3 First challenge

Now you are an expert in R - consider the following table of data, it's a simple example of the number of fruits in a basket, can you take the data and create a bar chart as shown, you'll need two vectors.

Make the colours of the bar chart match approximately to the colours of the fruits, this colour chart will be useful: https://r-charts.com/colors/

Fruit	**Quantity**
Apples	10
Oranges	16
Plums	25
Grapes	40

Once you have the barplot displaying, add a label to the x-axis and add a `main()` title to the plot, use `barplot(help)` for examples and to see a list of the keyword arguments you can use in your plot.

You may want to come back to this simple program later for reference, to save the program click on the file menu and give the program a name such as `fruits_plot`, this might be useful later.