MATH1019 Linear Algebra and Statistics for Engineers

Laboratory 1

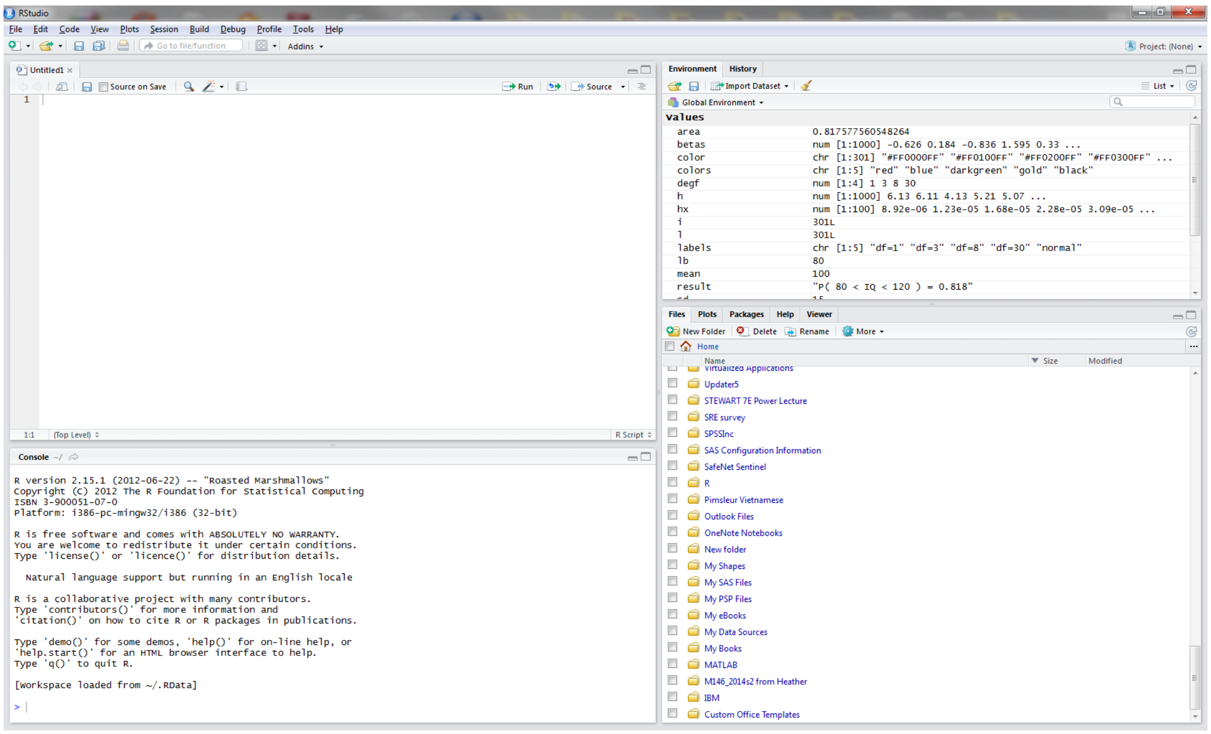
1 Introduction

R provides an extremely powerful environment in which you can perform statistical analysis and produce graphics. It is actually a complete programming language. *R* is one of the most widely used statistics programming languages among data scientists and is supported by a vibrant community of contributors.You can use it for simple purposes or very complex ones. In this tutorial, we will only cover some of the basics of *R*.

R is a command driven language, so to make our lives easier we will be using an interface to *R* called *RStudio*. *R* can be freely downloaded from: https://www.r-project.org/ and *RStudio* can be downloaded from: https://www.rstudio.com/ .

2 Getting started

Before you begin, you should create a working directory where all your data files will be stored. When you log in to a Curtin Computer, you will have access to your own drive, the I-drive. OK. We are ready to begin! When you open *RSudio*, you will see a window similar to the following:



```
Values
  area        0.817577560548264
  betas       num [1:1000] -0.626 0.184 -0.836 1.595 0.33 ...
  color       chr [1:301] "#FF0000FF" "#FF0100FF" "#FF0200FF" "#FF0300FF" ...
  colors      chr [1:5] "red" "blue" "darkgreen" "gold" "black"
  degf        num [1:4] 1 3 8 30
  h           num [1:1000] 6.13 6.11 4.13 5.21 5.07 ...
  hx          num [1:100] 8.92e-06 1.23e-05 1.68e-05 2.28e-05 3.09e-05 ...
  i           301L
  l           301L
  labels      chr [1:5] "df=1" "df=3" "df=8" "df=30" "normal"
  lb          80
  mean        100
  result      "P( 80 < IQ < 120 ) = 0.818"
```

```
R version 2.15.1 (2012-06-22) -- "Roasted Marshmallows"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-pc-mingw32/i386 (32-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.

[Workspace loaded from ~/.RData]

> 
```

3.1 Creating a project folder

First we will create an *RStudio* project. Click on the **Project** icon in the top right hand corner, and follow the menu items **Project->New Project->Existing Directory**. Using the **Browse** button, select the directory you created earlier, and click on **Create Project**. Next time you use *RStudio,* you can access your folder by either double-clicking on the *.Rproj* file in Windows Explorer; or navigate to the *.Rproj* file in *RStudio* by using the menu sequence **File->Open Project**.

3.2 Structure of *RStudio* window

The left hand pane is the console where you type in commands and see the output. The **Workspace** tab in the top right window shows all the active objects in your current workspace. The **History** tab shows a list of the commands that you have typed so far. The bottom right window has a number of tabs, the first of which is the **Files** tab that shows you all the files and folders in your default workspace. Any plots that you produce will show up in the **Plots** tab, and the **Help** tab contains help on *R* commands.

4 Simple *R* Commands

In its most basic form, *R* can be used as a simple calculator. Consider the following arithmetic operators:

- Addition: +
- Subtraction: -
- Multiplication: *
- Division: /
- Exponentiation: ^
- Modulo: %%

The last two might need some explaining: - The ^ operator raises the number to its left to the power of the number to its right: for example 3^2 is 9. The %% operator (modulo) returns the remainder of the division of the number to the left by the number on its right, for example 5 modulo 3 or 5 %% 3 is 2.

At the prompt ('>') in the console, type in

```
> 10 + 2
```

Followed by 'Enter' to see the result. Now type in

```
> 10 + 2 * 3 + (10 + 2) * 3 - 5^4/2
```

The precedence of operators in R is the same as what you'd learn in an Algebra course or on your calculator. Work out the above expression by hand to make sure you understand.

R contains lots of functions, and many of them have names that you would expect:

```
> exp(1) - log(1) - log10(10) + sqrt(5)
```

Functions enclose their *arguments* inside left '(' and right parentheses ')', and we'll see later that many functions can have several named arguments.

4.1 Exercises

Use the console to calculate the following:

1. $(\frac{2}{3} \times 8 - 1)^{2/3}$
2. $\log_2(4096)$
3. $e^{2+\cos(\frac{\pi}{2})}$
4. $(2.3^2 + 5.4^2 - 2 \times 2.3 \times 4.5 \times \cos(\frac{\pi}{8}))^{1/2}$

5 Assignment to Variables

A basic concept in (statistical) programming is called a **variable**. A variable allows you to store a value (e.g. 4) or an object (e.g. a function description) in *R*. A variable provides us with named storage that our programs can manipulate and easily access the value or the object that is stored within this variable. A valid variable name consists of letters, numbers and the dot or underline characters. The variable name starts with a letter or the dot not followed by a number.

Variable Name	Validity	Reason
var_name2.	valid	Has letters, numbers, dot and underscore
var_name%	invalid	Has the character '%'. Only dot(.) and underscore allowed.
2var_name	invalid	Starts with a number
.var_name	valid	Can start with a dot(.) but the dot(.)should not be followed by a number.
var.name	valid	Starts with a letter and the dot(.) is not be followed by a number.
.2var_name	invalid	The starting dot is followed by a number making it invalid.
_var_name	invalid	Starts with _ which is not valid

Variable names are case-sensitive, so *Name*, *name*, and *NAME* can all be different variables (although *R* allows this, for clarity in your code it is better if you don't use all three).

The assignment character is <-, and it is constructed by using two separate keystrokes (< and -), or in *RStudio*, just type Alt- (the Alt key and the minus sign – together). If you want to write two or more *R* commands on the same line of the console, separate them by a semi-colon (;). Typing the name of the variable and then pressing **Enter** will tell you the contents of the variable. For example:

```
> A <- 15; B <- 3
> C <- A/B
> C
```

Variables can also be assigned values using the rightward (->) and equal to (=) operator. The values of the variables can also be printed using **print()** or **cat()** function. The **cat()** function combines multiple items into a continuous print output. For example:

```
> print(C)
> cat("The value of C is",C,"\n")
```

6 Basic Data Types in R

R works with numerous data types. Some of the most basic types to get started are:

- Decimals values like 4.5 are called **numerics**.
- Natural numbers like 4 are called **integers**. Integers are also numerics.
- Boolean values (TRUE or FALSE) are called **logical**.
- Text (or string) values are called **characters**.

In contrast to other programming languages like C and java, in *R* the variables are not declared as some data type. The variables are assigned with *R*-Objects and the data type of the *R*-object becomes the data type of the variable. There are many types of *R*-objects. The frequently used ones are:

- Vectors
- Lists
- Matrices
- Arrays
- Factors
- Data Frames

6.1 Vectors

In addition to the scalar quantities that we've worked with so far, we can also work with more complex objects such as vectors and matrices. When you want to create a vector with more than one element, one way is use the **c()** function which means to combine the elements into a vector.

```
> apple <- c('red','green',"yellow")
> print(apple)
```

```
> x <- c(0,pi/4,pi/2,3*pi/4,pi,5*pi/4,3*pi/2,7*pi/4,2*pi)
> x
```

We saw above that functions can take scalar arguments, but they can also take vector arguments, and the function will operate on each element of the vector, as follows:

```
> y <-sin(x)
> y
```

Another way of creating a vector is to use the operator **:** (a colon), which creates a sequence of values:

```
> z <- 1:10
> z
```

We can also select a subset of the elements of a vector by using the indexing operator []. So, if we wanted to select the first three elements of x above, we could do this in the following two ways:

```
> x[1:3]
> x[c(1,2,3)]
```

The second statement can be used to select any subset of elements in any order.

Exercises

Use the console to do the following:

1. Select the first and sixth elements of x.
2. Construct a vector that has all but the first element of y

6.2 Data Frames

Data frames are tabular data objects. Each column can contain different modes of data. The first column can be numeric while the second column can be character and third column can be logical. It is a list of vectors of equal length.

Data Frames are created using the **data.frame()** function.

```
> # Create the data frame.
> BMI <-       data.frame(
     gender = c("Male", "Male","Female"),
     height = c(152, 171.5, 165),
     weight = c(81,93, 78),
     Age = c(42,38,26)
 )

> print(BMI)
```

7 Simple Plots in R

R has extensive graphics capabilities that can be a bit daunting at first, but if you start with simple plots and extend them, it’s a bit less intimidating. Here’s a simple plot using the function **plot()**:

```
> plot(x,y)
```

The default is just points, but let’s draw **b**oth points and lines:

```
plot(x,y,type="b")
```

We can also add a title, some colour and informative labels:

```
plot(x,y, type="b", xlab = "time(hours)", ylab = "scaled temperature diffe
rence", main="Temperature anomaly", col="blue")
```

R functions, in general, take several arguments, some of which are optional (this means that there will be some default values associated with them if not explicitly specified). The **plot** function takes lots of arguments. Type *?plot* or *help(plot)* into the console, and then take a look at the **Help** tab in the lower right window to explore these arguments.

If you are using the latest version of *RStudio*, you should notice that as you start typing commands in the console, it will show you commands that start with the letters that you have typed so far. Once you have typed the name of a function and a left parenthesis, press the *Tab* key to get a list of the arguments that the function takes.

Next, we will do a simple plot using the data set **"mtcars",** available in the *R* environment, to create a basic scatterplot. Let's use the columns "wt" and "mpg" in mtcars.

```
> input <- mtcars[,c('wt','mpg')]
```

mtcars is a *dataframe* and in the notation above we are choosing a subset of that *dataframe*. We can select specific rows and columns of this *dataframe*. The selection of rows is separated by a comma from the selection of columns. As we have not specified any rows, this means that all the rows will be selected. We have specified two columns to be selected, namely: *wt* and *mpg*. Now let’s have a look at the first few elements of this *dataframe*.

```
> head(input)
```

Our x values will be the column of values corresponding to **wt**, and for y values we will use the column corresponding to **mpg**. To extract these columns, we will use the **$** operator. So the simplest version of the plot function would be:

```
> plot(x = input$wt,y = input$mpg)
```

Exercise

Construct a plot of *wt* against *mpg* for cars with weight between 2.5 to 5 and mileage between 15 and 30 and label it appropriately. The title of the plot should be: “Weight vs Mileage”.