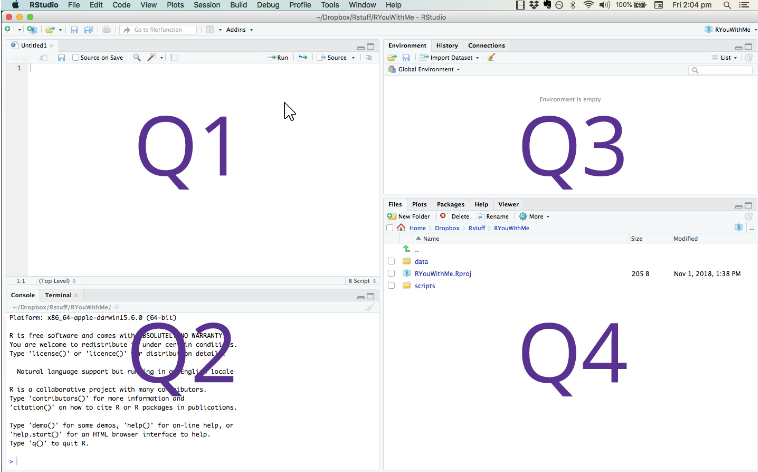
Practical 1 – Introduction to R (Lab Version)

Open RStudio – this should make the following interface appear



Q1 - contains: ***script, data, command to run script***  
Your written R Scripts will appear here, but also data (for instance a spreadsheet you might have imported) – also any variables can be inspected here. Data variables and code will appear in their separate tabs  
Q2 - contains: ***console***  
This is where you can run commands. In the tab “console” you can run R commands over any data imported or variables created, in the “Terminal” console you can make operating system commands  
Q3 - contains: ***environment***  
Essentially this is a visual representation of all the current variables available in the session, clicking on any one of them will make that variable appear in Q1  
Q4 - contains: ***files, plots, packages, help as tabs***  
Files just shows the files in a folder, plots displays any graphs generated by your code (unless specifically sent to a file as a graphic or a pdf), and packages shows the packages currently being used

At this point you should start by just playing with R commands in Q2

After R is started, there is a console awaiting for input. At the prompt (>), you can enter numbers and perform calculations.

> 1 + 2   
[1] 3

Task 1:

Try to do some simple calculations here. For instance convert pounds to dollars or rupees. Convert today’s temperature into Fahrenheit.

**Variables**

We assign values to variables with the assignment operator "=", however, in R, a more common assignment operator is “<-“

Just typing the variable by itself at the prompt will print out the value. We should note that another form of assignment operator "<-" is also in use.

> x = 1   
> x   
[1] 1

**Task 2**

See if you can convert the weight 15st 7lbs into kg. You will need to do two calculations. Convert stones to pounds (multiply by 14) then add the pounds, then turn it into kilograms by dividing by 2.2. See if you can do that. You might need to create some variables (e.g stones, pounds) to be able to do this.

**Functions and Data Types**

R functions are invoked by its name, then followed by the parenthesis, and zero or more arguments.

Lets start off with a very basic but extremely powerful function c(). It will convert a set of values (separated by commas) into a data type called a *vector*. A vector, in other programming languages would be called an “array” – that is to say, a set of variables of the same data type.

> c(1, 2, 3)   
[1] 1 2 3

But here is where things get interesting. Lets convert this into a variable for instance

>myvector<-c(1,2,3)

But see what happens if you multiply that by 2 by typing:

> myvector\*2

What do you see? Essentially each item in the vector is multiplied by 2! This is quite different from what you might expect in other programming languages. For instance in javascript – to do that you would need to write

let myvector=[1,2,3];  
for (let i=0;i<myvector.length;i++) console.log(myvector[i]\*2);

Here you can see what makes R so powerful, but also, a little opaque. When an operation is applied to an array in other languages – normally some kind of loop syntax is required. In R, whenever that operation is applied.

Now lets try the function paste() which is used in R to concatenate strings. Try

> paste("a", "b")

The output gives us:   
[1] "a b"

Nothing very surprising there. But now try this

> paste(myvector, "ok")

[1] "1 ok" "2 ok" "3 ok"

Again, the function is run over every element in the vector individually.

**Task 3**

Ask (at least) two people sitting next to you their names. Then create a vector of their names. Then use the paste function to say “Hi!” to all those people

If you are finding this a bit weird – don’t worry – that is R – it is a weird language, even though it is fantastically powerful – and for statistics and data science there is nothing more powerful. But if you are thinking at this point: “But I am doing cybersecurity – what use is this to me?” – I will answer – well do you not think security teams make statistics of DDoS attacks? Are not the number of requests calculated on a time axis – will it not be important to plot these as graphs? All of this will need to be done with some understanding of statistics – and while there are other routes, I would suggest that your ability to represent them will be massively enhanced by knowledge of R.

**Task 4**

However, the ultimate goal of R is to make sense of data – to give it meaning. Now let us try a more real world objective. Choose your favorite sport – and your favorite player or team in it. If its football, find how many goals your team has scored in the last 5 games. If it is cricket, how many runs a team or player has scored in their last 5 innings. Then turn it into a vector – for instance

goals<-c(0,2,1,3,2)

or

runs<-c(22,44,0,106,45)

Then run the summary function over it. What you should get is something like this:

summary(goals)  
 Min. 1st Qu. Median Mean 3rd Qu. Max.   
 0.0 1.0 2.0 1.6 2.0 3.0

What was the average score, in your case? What was the maximum, and what was the minimum?

We will cover the meanings of 1st Qu and 3rd Qu when we have done a a bit more statistical theory.

As an aside at this point, if you need to access individual elements of a vector – you can use the traditional square bracket notation. In the example above we could use *goals*[1] to get the first element of the *goals* vector. Note this is also, a change from most other programming languages which zero index their arrays (that is to say, they start at 0). R always uses a 1 to reference the 1st element of a vector.

However, rather than summarize, we might wish to get these results separately. Here we can introduce the functions mean() for the the average, median(), in very basic terms to get the middle of the distribution, and sd(), to get the standard deviation, the best known measure of the variability of a distribution. (For instance, student heights will have a very low standard deviation because it is unlikely that we find anyone below 1.2 m or anyone above 2 m. However, the standard deviation in marks at the end of the course might vary a lot – some people might score very highly, others very low, and so this will have a much wider standard deviation).

**Task 5**

Get the average (mean), the median and the standard deviation (sd) of the vector of goals or runs that you did in task 4

The other data structures we might wish to look at are the ***matrix***, the ***list*** and the ***dataframe*.** Of these the most relevant for us is the dataframe – and it is the one into which much of the data we look at will be put.

To help us get started with data frames, the best way might be to open a built-in dataframe in R which is **mtcars**

To see what’s in mtcars, just type that name at the prompt

>mtcars

Here you will see all the data in that dataset. However to see it more easily, type in **head(mtcars)** or **tail(mtcars) –** this will get you the first and last 5 rows in the dataframe respectively. But you can also run the summary function on this data too. If you run that, you will see the summary of the data on a column by column basis. But you can also query the data on a column by column basis. You can see the columnames by typing the command **names(mtcars) –** however you can query individual columns by using the dollar notation. For instance type:

>mtcars$mpg

This then reads all the values in the mpg column in the mtcars data frame as a vector. Moreover, you could find the average mpg of all the cars in this sample by typing

>mean(mtcars$mpg)

**Task 6**

Get the mean horse power of the cars?  
Find the standard deviation of horse power in the distribution  
What is the highest horse power in the distribution

R is not only very good at interrogating data, but is also very good at visualizing it. In this very simple introduction we wont go into much depth, but just try running the barplot() function over any vector. This can be one of the vectors earlier in this tutorial (e.g. ***goals*** and ***runs***) but you could also try it over any of the numerical columns in the mtcars dataframe. For instance, have a look at the mpg of the cars.

Finally, we will look at how to import data into R – since what we really want to do is investigate our own data, or new data. Take the file london\_boroughs.csv and open it in R Studio as follows

In the files section click on the file

Graphical user interface, application

Description automatically generated

And then choose “Import Dataset”

This will bring up a dialog which previews what the data will look like when imported into R

Table

Description automatically generated with medium confidence

In this case just click on the “Import” button

In reality, what R Studio has done for us, is run 3 separate commands

|  |  |
| --- | --- |
| library(readr) | #invoking the importation library |
| london\_boroughs <- read\_csv("london\_boroughs.csv") | #reading the text in the csv file and converting it to a dataframe called london\_boroughs |
| View(london\_boroughs) | #showing the data in tabular format |

Here is a lot of really interesting data – now play with it and see if you can find some interesting facts about London!