**Doctors as Data Scientists SESSION ONE**

**To take your first steps as a Data scientist we will use the computer language R, and the R studio software. R studio is an** IDE (an “integrated development environment” - an interface for working more easily with a computer language)

**We will use an online version of R studio, called R studio cloud, but if you want to practice more in your own time you can download a version because it is free “open source” software.**

**You will be provided with a link to an R studio cloud project called “Doctors as Data scientists” . You should then see a screen like the below, that will have a “start” button, in place of the “continue” button the first time you logon.**

**Click on the “start” button, or if its not the first time, the “continue” button**

**Graphical user interface, text, application

Description automatically generated**

**Then click on the “Doctors as Data Scientists project 1” link and you should see a screen like the below:**

**Graphical user interface, text, application, email

Description automatically generated**

**You are now ready to go with one potential additional thing to do if you are on a tablet:**

you have to turn off smart punctuation – if its turned on.

Settings > general > keyboard on an ipad. Otherwise “” come out in unrecognisable format

**To confirm everything is set up, try a simple sum such as the one in the screenshot and hit return**

**SAVING YOUR WORK TO AN R SCRIPT FILE**

Before we start, we need to set up an R script file, so that you can save and edit your work easily.

To save your work, go to the bottom right of your screen and click “new blank file” and select “R script” as below:

Graphical user interface, text

Description automatically generated

Save the file as something meaningful – I suggest “yourname\_Doctors\_As\_DataScientists” or similar. You will then see a new window pop up as below:

Graphical user interface, text, application, email

Description automatically generated

From now on, type your commands into the R script window, and hit the “run” button to see the results come out in the console window below.

*IMPORTANT NOTE – THE CURSOR NEEDS TO BE ON THE SAME LINE ASF THE COMMAND YOU WANT TO RUN IN ORDER TO RUN IT.*

You will be able to close your R script file using the x by the file name in the R script window and see your file in the bottom right window, where double clicking will re open the file.

Use this cross to close the R script and click the file name to open again. It will save automatically.

Graphical user interface, text, application, email

Description automatically generated

**EXERCISE ONE.** The basics of working with patient’s data using R with the R studio

In this first section you will learn two of the basic principles of being a data scientist – assigning elements (also called values) to vectors (also called variables in some contexts), and the importance of annotating your computer code.

To understand these concepts, imagine you wanted to calculate the total number of people in your clinic with diabetes. You look up the numbers and see that 334 have type1 and 3066 have type 2 diabetes.

From your R script file type the following command and then hit run:

334 + 3066

You should get the answer !

Now try another example, imagine you had just measured someone’s height as 1.7 metres and weight as 75 kilos and wanted to know their BMI. (you can google BMI calculators, but this is just an example to get used to R !)

From your R script file type the following and then hit run (note the \* is the multiplication sign in many computer languages to avoid confusion with x ):

75 / (1.70 \* 1.70)

*Tip – if you make a typo or other error you don’t have to type out the whole command again, use the up arrow to get your previous command*

But this is just using R as a calculator. The real power of computing with patients’ data comes from being able to assign values to vectors (or variables).

Try the following, hit run after each command.

height\_vector <- 1.7

weight\_vector <- 75

bmi\_vector <- weight\_vector / (height\_vector \* height\_vector)

Notice how you do not see any output in the console, because you are simply assigning a value to a vector using the <- command. But now if you type the vector name, and hit return:

bmi\_vector

You will see the result.

Now, imagine the patient comes back a year later and has gained some weight, but presumably remained the same height ! Now you can re-calculate the BMI without doing anything new with height:

weight\_vector <- 85

bmi\_vector <- weight\_vector / (height\_vector \* height\_vector)

Result- the new BMI, without doing anything with height.

Now try something similar with another calculation you may need to make using a patient’s details. E.g. their LDL-cholesterol to HDL-cholesterol ratio, using a HDL\_vector, and LDL\_vector. (you might need to google average HDL and LDL population values ). You might want to chose another example.

HDL\_vector <- *use a realistic HDL value here*

LDL\_vector <- *use a realistic LDL value here*

HDLLDLratio\_vector <- HDL\_vector / LDL\_vector

HDLLDLratio\_vector

ANNOTATING YOUR CODE

The second important principle of being a data scientist is to annotate your code. This is important so that you and others can work out what your computer programme (sometimes called a script when short and simple) is doing, when you come back to it. For this we use the # symbol to comment – most computer languages recognise the # symbol and interpret it as “ignore what follows”.

Annotate the script above with something like this:

#assign height in metres to height vector

height\_vector <- 1.7

#assign weight in kilos to weight vector

weight\_vector <- 75

#assign the value of BMI to the BMI vector

bmi\_vector <- weight\_vector / (height\_vector \* height\_vector)

#return result

bmi\_vector

**QUESTION 1**

**CREATE AND SAVE A NEW R SCRIPT CALLED *YOURNAME\_Q1\_SCRIPT*, (SEE SAVING YOUR WORK GUIDE ABOVE) THEN WRITE AND ANNOTATE A SHORT SCRIPT TO CALCULATE WAIST HIP RATIO FROM A WAIST AND A HIP MEASUREMENT, USING A SIMILAR SET OF COMMANDS AS THOSE FOR THE BMI SCRIPT ABOVE. RUN THE SCRIPT TO CHECK IT WORKS. THEN CHANGE ONE OF THE MEASUREMENTS AND ENTER IT INTO THE RELEVANT VECTOR AND RE RUN THE SCRIPT TO RETURN THE NEW VALUE OF THE RATIO. USE THE INTERNET TO FIND OUT REALISTIC RANGES OF HIP , WAIST AND WAIST HIP RATIO MEASUREMENTS TO ENSURE YOU USE REALISTIC VALUES.**

**Exercise 2 – basic functions.** The next concept to learn about computing in R is the concept of functions. Functions have the format of function() with the brackets containing the information about what you want the function to do. Functions can be thought of as commands – telling the computer what to do. Using commas in the brackets allows you to provide the function with more than one piece of information:

Lets start with a simple one print (), try

print(bmi\_vector)

And here’s a similar one, paste() try

paste(“the patient’s BMI is”, bmi\_vector)

TIP: If you see a + sign it is because R studio is expecting another command – e.g. after paste (it is suggesting you need to do something else to see the result) . You can hit escape to get back to the > cursor.

There are two ways of seeing the result.

First, try putting the result into a vector with

patient\_bmi\_vector <- paste(“the patient’s BMI is”, bmi\_vector)

patient\_bmi\_vector

*note how the paste() function has a statement in speech marks and a vector – an object that contains a piece of information, separated by a comma*

Second, use the print() function, which tells the computer to print a result to screen.

try

print(paste(“the patient’s BMI is”, bmi\_vector) )

notice the use of two sets of brackets – because the paste() function is within the print() function. In English the above command first pastes together the two items separated by the comma, then prints them to screen.

Have some more fun with this, by printing out more information about the patient, for example:

print(paste(“the patient’s BMI is”, bmi\_vector, “and their actual weight is”, weight\_vector) )

in the above function, there are four “items” to be pasted together using the paste() function, then printed to screen using the print() function: two pieces of text, that the computer knows are not commands because they are in speech marks, and two vectors, so the computer will print what is in the vectors not the actual words.

Try the following again, but this time having changed the weight\_vector and bmi\_vector

weight\_vector <- 105

bmi\_vector <- weight\_vector / (height\_vector \* height vector)

print(paste(“the patient’s BMI is”, bmi\_vector, “and their actual weight is”, weight\_vector) )

TIP do not copy and paste the above, but type out in R studio, the speech marks do not transcribe well.

**QUESTION 2:**

**CREATE AND SAVE A NEW R SCRIPT CALLED *YOURNAME\_Q2\_SCRIPT*, (SEE SAVING YOUR WORK GUIDE ABOVE)) AND THEN WRITE A TWO LINE SCRIPT THAT PLACES A HEALTHY LDL CHOLESTEROL LEVEL INTO A VECTOR, THEN PRINTS OUT A STATEMENT STATING THAT THE PATIENT’S LDL IS HEALTHY BECAUSE IT IS THAT VALUE**

**USE # TO COMMENT ON WHAT YOUR SCRIPT IS DOING AND THAT IT IS THE ANSWER TO QUESTION 2.**

**NOTE YOU WILL NEED THE PRINT() AND PASTE() FUNCTIONS AND SPEECH MARKS FOR TEXT**

We will come back to functions later

**EXERCISE 3: BOOLEAN OPERATORS**. Lets now practice a few simple “operations”. “Operators” are simple commands such as > (greater than), < (less than), | (or) , & (and). These are very important in computing, as essential all instructions to a computer have to be broken down into yes/no TRUE/FALSE type operations.

Try some very simple ones such as:

1 == 2

1 ==1

1 != 2

1 != 1

based on the answers what do you think the == and != operators mean ?

What about characters and the < and > operators ? Try “Jack” > “Bill” then “Betty” < “Jane”

**QUESTION 3: BY USING THE > AND < OPERATORS, FIND OUT IF THE PROGRAMME R RETURNS TRUE OR FALSE BECAUSE OF ALPHABETICAL ORDER OR LENGTH OF THE WORD. REMEMBER TO USE SPEECH MARKS TO TELL THE R PROGRAMME THAT YOU ARE USING TEXT ALSO KNOWN AS STRINGS.**

What if we wanted to know if a patient was heavier for their height compared to another patient ? First lets give two people a BMI, using our code from Exercise 1:

#assign height in metres to height vector

height\_vector <- 1.7

#assign weight in kilos to weight vector

weight\_vector <- 75

#assign the value of BMI to the BMI vector

patient1\_bmi <- weight\_vector / (height\_vector \* height\_vector)

Then change the values in your height and weight vectors for patient 2:

#assign height in metres to height vector

height\_vector <- 1.85

#assign weight in kilos to weight vector

weight\_vector <- 80

#assign the value of BMI to the BMI vector

patient2\_bmi <- weight\_vector / (height\_vector \* height\_vector)

Now you can use the Boolean operators to compare the two patient’s BMIs. Try playing with several of the operators, e.g.

patient1\_bmi < patient2\_bmi

patient1\_bmi == patient2\_bmi

patient1\_bmi != patient2\_bmi

To increase your familiarity with the R syntax try altering your patients’ BMI’s a few times using the script from exercise 1, and repeat your tests using the operators.

~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Bonus part to the exercise if you are starting to feel confident !**

Lets make a third patient, patient 3:

#assign height in metres to height vector

height\_vector <- 1.65

#assign weight in kilos to weight vector

weight\_vector <- 85

#assign the value of BMI to the BMI vector

patient3\_bmi <- weight\_vector / (height\_vector \* height\_vector)

Another interesting Boolean operator is the “pipe” | simple which in many computer languages means “OR”. So imagine we want to know if the BMI of person 3 is larger than the BMI of person 1 OR the BMI of person 2. In other words, we want to know if there is anyone with a BMI lower than that of person 3. Try the following, note the use of the pipe | operator:

patient3\_bmi > patient2\_bmi | patient1\_bmi

Which of course then leads us to the & operator. Which means “AND”. So imagine we want to know if the BMI of person 3 is larger than the BMI of person 1 AND the BMI of person 2. In other words, we want to know if there is anyone with a BMI higher than that of person 3. Try the following, note the use of the pipe | operator:

patient3\_bmi > patient2\_bmi & patient1\_bmi

**Exercise 4 – if and if else statements.** Now lets have some more fun and write a proper, short computer programme. Writing these kinds of statements is another important concept in computing:

First lets use the if statement. These statements have the format (also known as syntax)

If () {}

And the statements in the two types of brackets as follows:

If (some statement that could be true or false) {a function or operation of some sort that will be carried out if the statement in the first brackets is true. If the statement in the first brackets is not true, nothing will happen}

lets write an if statement, for example, to return whether someone is obese depending on their bmi. Try, using the print() function as the function to carry out, within the curly brackets:

if (patient1\_bmi <30) {print ("patient 1 is not obese")}

Now change it around to print a statement when the patient is obese (A BMI over 30)

if (patient1\_bmi >30) {print ("patient 1 is obese")}

what happens ? there is no output, because the bmi of patient 1 is under 30, and we have only told R to print something if the BMI is over 30.

So now we need to do something a little more sophisticated. Lets write an if else statement. An if else statement is just a logical extension of the if statement

If () {} else {}

In full English: If (some statement that could be true or false) {a function or operation of some sort that will be carried out if the statement in the first brackets is TRUE} else { a function or operation of some sort that will be carried out if the statement in the first brackets is NOT TRUE} try

if (patient1\_bmi < 30) {print ("patient1 is not obese")} else {print ("patient1 is obese")}

try a few more variations of this if else statement to ensure you are confident. Perhaps use a BMI of 25 and the term “overweight” rather than obese. Or use a HDL and LDL example and “the patient has/does not have high cholesterol” (you might want to google what a high cholesterol reading is)

**QUESTION 4.**

***QUESTION 2.1 MAKE A VECTOR FOR BMI CONTAINING A PATIENT’S BMI VALUE. THEN WRITE AN IF ELSE STATEMENT THAT PRINTS OUT WHETHER OR NOT THE PATIENT IS OBESE***

*make sure this statement works whether the patient is obese or not by changing the BMI value and re-running the script.*

*The script should be 3 lines long:*

*the two functions you need are*

*if (some condition is true) {do something} else {do something else}*

*print()*

*check with one of the teaching assistants that you are using the R script file and run button not the console and that your script is working as expected*

***QUESTION 2.2***

NOW ADD TO THE SCRIPT IN QUESTION 2.1 BY USING THE PASTE() FUNCTION TO MAKE THE PRINTED RESULT MORE INFORMATIVE. DO THIS BY PASTING TOGETHER SOME TEXT AND THE VALUE IN THE BMI VECTOR WITHIN YOUR PRINT() FUNCTIONS AND USING COMMAS TO SEPARATE THE VALUES/VECTORS/TEXT YOU WISH TO PASTE TOGETHER. THE FORMAT OF THE PRINT() FUNCTIONS WITHIN YOUR SCRIPT WILL BE

*Print (paste (“text”,bmi\_vector))*

***QUESTION 2.3***

*NOW LETS TRY SOMETHING MORE SOPHISTICATED – AN IF ELSE STATEMENT WITHIN AN IF ELSE STATEMENT. THIS WILL HELP REINFORCE YOUR LOGIC :*

*MODIFY THE FOLLOWING SCRIPT TO OUTPUT A DESCRIPTION OF A PATIENT’S BMI – WHETHER THEY ARE UNDERWEIGHT (BMI LESS THAN 18), NORMAL WEIGHT (BMI 18-25), OVERWEIGHT (BMI 25-30), OR OBESE (BMI OVER 30)*

*YOU CAN USE YOUR BMI\_VECTOR INSTEAD OF NUMBER. START BY ASSIGNING A BMI CATEGORY TO A RESULT VECTOR*

*Result <- “overweight”*

*Type result and hit run to check it*

THEN MODIFY THE FOLLOWING SCRIPT FROM THE R SCRIPT FILE. TO COMPLETE THE TASK YOU WILL NEED TO CHANGE THE NUMBERS INVOLVED AND THE WORDING IN SPEECH MARKS

*if (number < 10) {*

*if (number < 5) {*

*result <- "extra small"*

*} else {*

*result <- "small"*

*}*

*} else if (number < 100) {*

*result <- "medium"*

*} else {*

*result <- "large"*

*}*

*print(result)*

*CHECK THIS WORKS BY CHANGING THE VALUE IN THE BMI\_VECTOR AND RE-RUNNING THE SCRIPT FROM THE R SCRIPT FILE.*

*CHECK WITH A TEACHING ASSISTANT THAT ALL IS WORKING WELL.*

**END OF SESSION ONE – Phew ! OR, if you have time, increase your confidence with additional exercises:**

If () {} else if (){} else {}

try

if (patient1\_bmi >=18 & patient1\_bmi <= 25) {print ("patient 1 has a healthy weight")} else if (patient1\_bmi > 25 & patient1\_bmi <= 30) {print("patient 1 is overweight but not obese")} else {print("patient 1 is obese")}

try changing patient 1’s BMI quickly with

patient1\_bmi <- 35, for example and re running the command above.

Try expanding it even further:

if (patient1\_bmi >=18 & patient1\_bmi <= 25) {print ("patient 1 has a healthy weight")} else if (patient1\_bmi < 18) {print("patient 1 is underweight")} else if (patient1\_bmi > 25 & patient1\_bmi <= 30) {print("patient 1 is overweight but not obese")} else {print("patient 1 is obese")}

and you could even add in some more nuance by combining the print() and paste() functions again

if (patient1\_bmi >=18 & patient1\_bmi <= 25) {print(paste ("patient 1 has a healthy weight because their BMI is", patient1\_bmi))} else if (patient1\_bmi < 18) {print(paste("patient 1 is underweight because their BMI is ", patient1\_bmi))} else if (patient1\_bmi > 25 & patient1\_bmi <= 30) {print(paste("patient 1 is overweight but not obese because their BMI is", patient1\_bmi))} else {print(paste("patient 1 is obese because their BMI is", patient1\_bmi))}

this is a long statement, and has lots of brackets surrounding the statements and functions. If one bracket is missing or in the wrong place or there are too many brackets, it will not work, so scripts are often written out in the following format

If (something is true) {

do something } else if (something else is true) {

do something else } else {

do something else

}