LAB100

Week 07: RStudio as a Scientific Calculator.

Your aim for this workshop is to learn how to use to RStudio to perform the functions of a scientific calculator or a spreadsheet. Upon completing this workshop you should be able to:

- Set-up RStudio and create a new R Script.
- Perform simple and scientific calculations.
- Assign objects to memory.
- Construct vectors and use them for calculations.
- Use the help files.

1 Getting Started

Open RStudio from your desktop or from your start menu. The first job is to change the working directory to where you are going to store all of your LAB100 material. Doing this will make saving and loading files and data sets much easier.

STEP 1 Click on the 'Files' tab in the bottom right window.

STEP 2 Click on the three-dots button on the right hand side of this window.

STEP 3 In the new window, locate the file that you wish to save all of your LAB100 workshops. For example, John Smith may select his university filestore H:\smithj44\LAB100 as his working directory.

STEP 4 Click 'OK' and the contents of this file will appear in the 'Files' window

STEP 5 Next, click on 'More' and then on 'Set As Working Directory'. This will send a command to the 'Console' window that says that you have changed you directory.

It is essential to keep a record of all your R work such that it is easier to identify and correct errors in your code. To do this we open and save an R script:

STEP 1 Click on 'File' menu item at the top of the RStudio window.

STEP 2 Go to 'New' and click on 'R Script'. This will open a new script editor in the top right window called 'Untitled1'.

STEP 3 To save this script, click on 'File' again and then on 'Save As...'.

STEP 4 Call the script 'Workshop wk07' and click on 'Save'. Note the appearance of the new script in your working directory in the bottom right window.

Type the following code into your editor:

```
3+2
```

To execute this command, either highlight the section of code or place the cursor on the line you wish to calculate and click on the 'Run' button at the top of the script window. This will submit your code to the 'Console' below and you will see:

```
> 3+2
  [1] 5
```

Alternatively, the shortcut command for submitting code is Ctrl+Enter for Windows and Linux computers or Command+Enter for Mac users.

2 RStudio as a Basic Calculator

At its most basic level, R can be used as a calculator. It uses the BODMAS order of calculation: first Brackets (`()`), then powers Of (`^`), Division (`/`), Multiplication (`*`), Addition (`+`) and finally Subtraction (`-`). To illustrate this ordering, type and run the following code into your script and compare what is happening in each pair of calculations.

```
3 - 4 * 5 + 6
(3 - 4) * (5 + 6)
4 / 3 * 0.2
4 / (3 * 0.2)
5^2 + 1
5^(2 + 1)
```

3 RStudio as a Scientific Calculator

For more scientific calculations, RStudio has many in-built arithmetic functions. Some of the most common commands are presented in the table below.

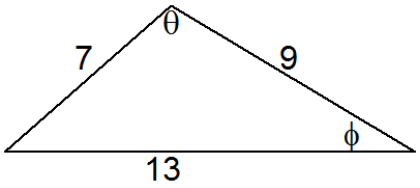
For example, consider the triangle below. Rearrange the cosine rule to calculate the angle θ.

```
acos( (7^2 + 9^2 - 13^2)/(2*7*9) )
```

Note that all of the trigonometric calculate angles in radians, not degrees.

The following table contains some numerical constants that may be useful to use or to help interpret what the console returns.

Command	Mathematical Operation
`sqrt( )`	Square root
`sin( )` , `cos( )` , `tan( )`	Trigonometric functions
`asin( )` , `acos( )` , `atan( )`	Inverse Trigonometric functions
`sinh( )` , `cosh( )` , `tanh( )`	Hyperbolic functions
`exp( )` , `log( )`	Exponential and natural logarithm
`factorial( )`	Factorial ($n! = 1 \cdot 2 \cdot \ldots \cdot n$)



Cosine Rule:

$$a^2 = b^2 + c^2 - 2bc\cos(\theta)$$

Command	Value
`pi`	$\pi \approx 3.141592653589793$
`Inf`	Infinity (∞)
`e`	Short hand for $\times 10$, e.g. $0.03 =$ `3e-2` and $4100 =$ `4.1e3` . NOT EXPONENTIAL!
`NaN`	Not a Number, resulting from calculations such as $0/0$ or $\infty - \infty$

4 Assigning to Memory

For larger calculations, it is very difficult to ensure that a long line of code will contain no mistakes. In this case it is useful to assign certain values a name, known as an object, that is used in future lines of code. To do this we use the assign operation `<-` , which is the 'less than' key followed by the 'minus' key. Type the following example into you script and execute the commands.

```
x <- 4
X <- 30
x
log(X + 1)
```

Take care with typing this example as RStudio is case sensitive. After executing, notice that the two new objects appear under the 'Workspace' tab in the top-right window. This displays everything that is stored in the RStudio memory. Now type the following into your script.

```
x <- 100
x
```

Run this code and you will notice that the value for the object `x` has been overwritten.

5 Vectors

Quite often the same calculation will be performed on different numbers. Instead of considering each calculation separately, which you would normally do with a hand-held calculator, the whole calculation can be performed simultaneously through the use of vectors. The table below describes four methods of constructing vectors. Type and run the examples as these will be used later in this section.

Command	Description	Example
`c`	Combines a sequence of numbers into a vector.	`A <- c(1, 3.5, -66, pi)`
`rep`	Repeats the same number.	`B <- rep(5, times=4)`
`seq`	Creates a sequence of equally spaced numbers from and to specified values.	`C <- seq(from=0, to=pi, by=pi/8)`
`:`	Creates a sequence of integers	`D <- 3:10`

Once the vectors have be submitted to the memory, we can treat the objects as values and write the code as before. Try the following examples to illustrate how to perform multiple calculations with one command.

```
2 * sin(C) * cos(C)
```

```
log(1 + C/pi)
```

Alternatively, we may be interested in a particular summary of a set of numbers contained within a vector. The table below contain some functions that takes a vector as an input argument and returns a single summary value.

Command	Description	Example
`sum`	Takes a numeric vector and calculates the summation.	`sum(C)`
`length`	Determines the length of the vector	`length(C)`
`mean`	Calculates the average value of the vector	`mean(C)`
`min`	Returns the minima of the input values	`min(A)`
`max`	Returns the maxima of the input values	`max(A)`

For instance if we want to calcuate

$$\sum_{n=3}^{10} n$$

given we have already assigned the vector `D` to take the sequence `3:10` we can calculate the required sum using

```
sum(D)
```

while if we wanted to obtain

$$\sum_{n=3}^{10} \sqrt{n}$$

we would first create a vector containing

$$\sqrt{3}, \sqrt{4}, \ldots, \sqrt{10}$$

and then use `sum` e.g.

```
E <- sqrt(D)
sum(E)
```

6 Help Files

If you wish to understand what a particular function does and what arguments it requires, at your disposal are the help files. click on the 'Help' tab in the bottom-right window and type the function you wish to know more about in the search box on the right-hand side. For example, look at the help file for the function `seq` . A help file will contain a description of the function, a list of arguments that are taken as inputs to the function, details about how the function operates and finally some examples of its usage.

Quiz 1: Translating Equations

What is the correct command to calculate:

$$6\,\frac{5+1}{(4-2)^3}$$

(**A**) `6 * ((5 + 1)/(4 - 2))^3`

(**B**) `6 * (5 + 1)/((4 - 2)^3)`

(**C**) `6(5 + 1)/(4 - 2)^3`

(**D**) `6 * 5 + 1/(4 - 2)^3`

(**E**) `(6 * 5 + 1)/((4 - 2)^3)`

Quiz 2: Summing using vectors (i)

Use **R** to calculate

$$\sum_{n=1}^{50} n^2$$

The answer is? (**A**) 1625625 (**B**) 42925 (**C**) 63625 (**D**) 3126250 (**E**) 46375

Quiz 3: Summing using vectors (ii)

Use **R** to calculate

$$\sum_{n=1}^{50} (-1)^n n^2$$

The answer is? (**A**) 46375 (**B**) 1250 (**C**) 775 (**D**) 2025 (**E**) 1275

Quiz 4: Summing using vectors (iii)

Use **R** to calculate

$$\sum_{n=1}^{50} \frac{1}{\log(1+\exp(0.5n))}$$

The answer to 1 decimal place is (**A**) 7.3 (**B**) 7.4 (**C**) 7.5 (**D**) 7.6 (**E**) 7.7

Quiz 5: Approximation of π

π can be approximated by the series

$$\sum_{n=0}^{N} \frac{2^{(n+1)}(n!)^2}{(2n+1)!}$$

The smallest value of N that ensures the approximation is correct to 5 decimal places is:
(**A**) 6 (**B**) 9 (**C**) 13 (**D**) 16 (**E**) 28
