Tutorial and Laboratories
Week2

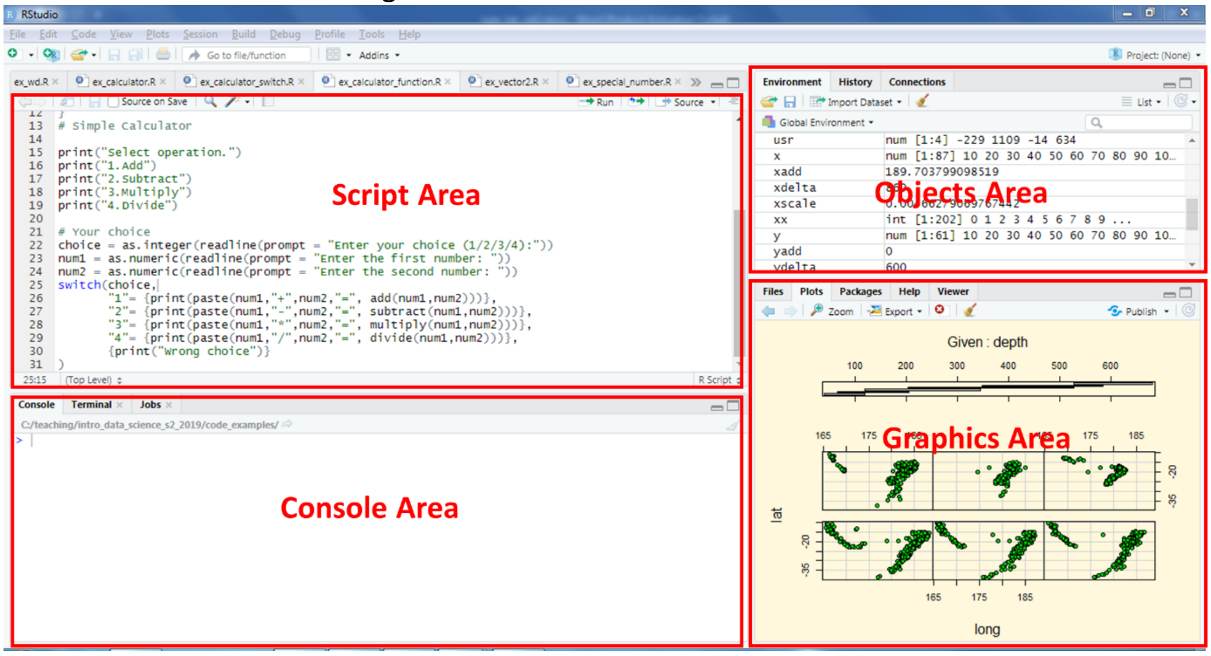
The purpose of this week's tutorial and lab exercises is to introduce you to the R programming language, which we will use extensively throughout the unit. The *"Definition of Done"* for this tutorial and lab is to:

1- Get familiar with R environment and RStudio interface,
2- Get your hands on doing simple operations (e.g. mathematical, logical, etc.) by using Console window in RStudio,
3- Gain the ability to write and run simple R scripts,
4- Get yourself familiar with creating variables and assign values to them and
5- Acquire skills to write small R scripts, run them and inspect the output.

Install R and RStudio

To setup the environment on your computer, you need to install R and RStudio. Please follow the instructions from the lecture notes of week 1. The slides are only for Windows and MAC operating systems. If you are using lunix, you can follow the instructions in this link: https://linuxconfig.org/rstudio-on-ubuntu-18-04-bionic-beaver-linux

Please note: In this unit, we use RStudio on Windows to do all our programming practices. The instructions in this document assume you are working on Windows operating system, so if you are using any other operating system, please adjust accordingly.
To open RStudio, go to start menu, write RStudio and click on the RStudio icon. You will see a screen similar to the following:



```
}
# Simple Calculator

print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")

# Your choice
choice = as.integer(readline(prompt = "Enter your choice (1/2/3/4):"))
num1 = as.numeric(readline(prompt = "Enter the first number: "))
num2 = as.numeric(readline(prompt = "Enter the second number: "))
switch(choice,
       "1"= {print(paste(num1,"+",num2,"=", add(num1,num2)))},
       "2"= {print(paste(num1,"-",num2,"=", subtract(num1,num2)))},
       "3"= {print(paste(num1,"*",num2,"=", multiply(num1,num2)))},
       "4"= {print(paste(num1,"/",num2,"=", divide(num1,num2)))},
       {print("wrong choice")}
)
```

```
C:/teaching/intro_data_science_s2_2019/code_examples/
> 
```

```
usr      num [1:4] -229 1109 -14 634
x        num [1:87] 10 20 30 40 50 60 70 80 90 10...
xadd     189.703799098519
xdelta   [illegible]
xscale   0.00[illegible]6627906976744 2
xx       int [1:202] 0 1 2 3 4 5 6 7 8 9 ...
y        num [1:61] 10 20 30 40 50 60 70 80 90 10...
yadd     0
ydelta   600
```

We have 4 main areas within the RStudio interface:

1. **Console Area**: This shows the output of code you run. Also, you can directly write line-by-line commands in the console. Code entered directly in R console cannot be traced later. This is where script area comes to use.

2. **Script Area**: As the name suggest, here you get space to write multiple lines of codes. To run those lines, simply select them and press (Ctrl + Enter). Alternatively, you can click on little 'Run' button at top right corner of the script area.

3. **Environment/Objects Area**: This space displays the set of external objects added. This includes data set, variables, vectors, functions etc. To check if data has been loaded properly in R, always look at this area.

4. **Graphical Area**: This space displays the plots or graphs created during exploratory data analysis. Not just graphs, you could select packages, seek help with embedded R's official documentation.

Interacting with RStudio

- Most of the time, you will be shifting between the Script area and the Console are, so that RStudio provides simple shortcuts for this transition, simply use the Ctrl+1 and Ctrl+2 shortcuts to jump between the script and the console windows.
- To create new script, go to File >> new file >> R script, or simply press Ctrl+Shift+N.
- Practice the following shortcuts:
  - Ctrl+Return(Enter) to run lines from editor
  - Alt+Shift+k for RStudio keyboard shortcuts
  - Ctrl+r to browse the command history
  - Alt+Shift+j to navigate between code sections
  - tab for auto-completion
  - Ctrl+1 to skip to editor
  - Ctrl+2 to skip to console
  - Ctrl+8 to skip to the environment list
  - Code Folding:
    - Alt+l collapse chunk
    - Alt+Shift+l unfold chunk
    - Alt+o collapse all
    - Alt+Shift+o unfold all
  - Alt+"-" for the assignment operator <-
  - Esc, if you want to exist a running program

How to install R Packages?

To install a package, simply type the following command in the console area:

install.packages("package name")

Also, you can do this by using "install" button in the top left corner of the packages window, for example, type in the console window:

```
> install.packages("tidyverse")
```

Basic Computations

Exercise 1

Perform the following simple operations in the console window and inspect the result:

```
> 2 + 3

> 6 - 10

> 3 * 4

> 10 / 4

> 12 ** 4

> 12 ^ 4

> log(1:5)

> print("Introduction to Data Science.")

> print(paste("I am ", 25, "years old!"))

> 5 == 2

> 30 > 20

> 10 <= 2

> 5 != 4

> readline(prompt= "what is your age?")

> age
```

- What has happened after "readline(prompt= "what is your age?")" in the console window? Why? and
- Why did you get an error message in the last line?

Variables and Data Types

Variables are placeholders to refer to values stored in memory. These values can fall into the following categories:

- Numeric,
- Character,
- Logical,
- Complex and
- Integer

You can assign values to variables using the assignment operator <-, like this:

```
> x <- 1/40
```

And now the variable x contains the decimal value 0.025:

```
> x

[1] 0.025
```

Look up at the top right pane of RStudio, in the "objects area", and you'll see that this has appeared in the "Environment" pane.

Our variable x can be used in place of a number in any calculation that expects a number.

```
> log(x)
```

What is the result?

```
> sin(x)
```

What is the result?

The right hand side of the assignment can be any valid R expression.

It is also possible to use the `=` operator for assignment:

```
> x = 1/40
```

...but this is much less common in R community. The most important thing is to **be consistent** with the operator you use. There are occasionally places where it is less confusing to use <- than =, and it is the most common symbol used in the community. So I would recommend using <-.

Exercise 2

```
> x <- 120
```

Does the assignment print a value?

Can the variable be reassigned with different values?

What will x contain after running this?

```
> x <- x + 1
```

Note that the right hand side is fully evaluated before the assignment occurs.

Also notice that variable names can contain letters, numbers, underscores and periods. They cannot start with a number nor contain spaces at all. Different people use different conventions for long variable names, these include

- periods.between.words
- underscores_between_words
- camelCaseToSeparateWords

What you use is up to you, but **be consistent**.

Commenting

Use # signs (Shift+3 on your keyboard) to comment. We add comments with code in the script area. Anything to the right of a # is ignored by the R interpreter.

Exercise 3

- In current directory of your workspace, create another directory for the exercises and name it “exercises_wk2”.

- Change the current working directory of the RStudio to the newly created folder, using the following command in the console window:

  ```
  > setwd("path to exercises_wk2 folder")
  ```

- Create a new script, and do the following:
  - Save it to the "erercises_wk2" directory, by pressing Ctrl+S, and name it, "variables.R", please note the extension of the file (.R),
  - Add comment to describe the objective/s of this script, *e.g. practicing on variables in R*
  - Print a message to ask the user to enter his age and save the age to an integer variable called "age"
  - Add 2 to the age variable
  - 2. Multiply the result by 3.
  - 3. Subtract 6 from it
  - 4. Divide what you get by 3.
- Print the final age
- What is the final age? Is it the same? Or different?

Exercise 4

This exercise shows you the different usability options of the ***paste*** function:

```
> paste("Leo", "the", "lion")
> paste("a", "b")
> paste("a", "b", sep="")
> paste(1:5)
> paste("a", 1:5)
> paste("a", 1:5, sep="")
> paste(1:5, collapse="")
> paste(letters[1:5], collapse="")
```

- What are the respective effects of the parameters sep and collapse?