Tutorial and Laboratories

Week3

The purpose of this week's tutorial and lab exercises is to introduce you to different important components of the R programming language, which are essential to do data analysis. These components include vectors as a data structure and conditional and loop statements. The objectives of this tutorial and lab are:

1- Get yourself familiar with creating, accessing and operating on 'vectors'
2- Acquire skills to write scripts with different conditional statements.
3- Acquire skills to write scripts with loops statements.

Vectors

A vector is a data type for collection of items that are stored together under a name and has a size, which is the number of these items. The **c( )** function can be used to create vectors of homogenous objects by concatenating them together.

To access elements in a vector, you can use the square brackets, i.e **[ ]** with index/indices to refer to the items you would like to access. The indices can be either positive or negative. Also they can be either scalar or combination of numbers in another vector or sequence.

- Practice creating the following vectors in the console window of the RStudio:

```
> x <- c(0.5, 0.6)        ## numeric
> x <- c(TRUE, FALSE)     ## logical
> x <- c(T, F)            ## logical
> x <- c("a", "b", "c")   ## character
> x <- 9:29               ## integer
> x <- c(1+0i, 2+4i)      ## complex
```

Exercise 1 (Vectors) 

The weights of five people before and after a diet program are given in the below table:

people	Jack	Dione	Reda	Sally	Adam
Before	78	72	78	79	105
After	67	65	79	70	93

- Read the "before" and "after" values in two different vectors called before and after. Also create a vector with the participants' names.
- Evaluate the amount of weight lost for each participant
- What is the average weight lost?
- Extract the name of the person who lost the most. You may practice using a built-in function ('**which'**) as following:
  - **which(condition)**,

  where 'which' returns the index of the item who met the passing condition.
- Is anyone gaining more weight with this diet program? Which one? You may check that with using built-in function ('**any**'), as following,
  - **any(condition)**

Exercise 2 (Sub-setting) 

The built-in vector **LETTERS** is a constant that contains the uppercase letters of the English alphabet. Generate a vector of

- the first 12 letters,
- the odd `numbered' letters,
- the English vowels (*i.e.* a, e, i, o, u, y) and
- the (English) consonants.

Exercise 3 (Unsupervised Activity) (vectors with random values) 

The function '**rnorm**' generates normal random variables. For instance, rnorm(10) gives a vector of 10 independently and identically distributed (i.i.d) standard normal values. Practice the following:

- Generate 30 standard normal values, and store them as x,
- Obtain the entries in x which are less than 1,
- Extract the entries between 0.5 and 1 and
- Generate the entries whose absolute value is larger than 1.5.

Conditional statements

Conditional statements can be used in cases when we want to execute lines of codes based on a Boolean expression. The most used conditional statements is as following:

```
if (Boolean expression){
        # block of code
}else{
        # another block of code
}
```

Multiple and nested "if…else" blocks can be used.

Exercise 4 (Conditional Statements) 

We need to use "if...else" blocks to create a simple calculator that can do the basic arithmetic operations, e.g. addition, subtraction, multiplication and division.

- Create a new R script and name it "simple_calculator_with_if.R". Write the following code to the newly created script:
- Add the following code to this script and run it as before.

```
# 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: "))

# Which operation?
if(choice == 1){
  print(paste(num1,"+",num2,"=", num1+num2))
}else if(choice == 2){
  print(paste(num1,"-",num2,"=", num1-num2))
}else if(choice == 3){
  print(paste(num1,"x",num2,"=", num1*num2))
}else if(choice == 4){
  print(paste(num1,"/",num2,"=", num1/num2))
}else{
  print("Wrong choice!")
}
```

Run the code different times with a different choice every time and check the output.

Loops

Most of the programming languages provides a way to repeat the execution of set of statements many times. In R, we can do this using one of the following loop sentences:

1- for loop
2- while loop
3- repeat

Please refer to the lecture of the third week for more information about the loops. Then practice the following example in RStudio and ensure that you have understood the concept of the loop statements.

```
a <- 1:10
b <- 1:10
res <- numeric(length=length(a))
for(i in a){
  res[i] = a[i] + b[i]
}
```

What about just using `res <- a + b`, is it the same result?

Exercise 5 (Unsupervised Activity) (loop statements) 

In the next exercise, imagine we have a column in a data base, where this column holds emails of 10 users, we need to check the validity of these email. By validity, we mean to check, for example, an email record has '@' character or not.

To do so:

1- we will read the emails of the 10 users into a vector as following:

```
emails <- c('myname@mycompany.com',
            'my@mycompany.com',
            'name@mycompany.net',
```

```
'work@mycompany.org',
'mywork#mycompany.net',
'myproduct#mycompany.com',
'myCoffee2mycompany.org',
'mycar@mycompany.com',
'mypet@mycompany.com',
'hobbies!mycompany.com')
```

2- Build a for loop that will go through these emails one by one to check if each of them has '@' character as a test of the validity.

TIP

To check the existence of a sub-string in a string, you can use the *grepl* function. The syntax of this function is as following:

```
valid <- grepl('@', element)
```