Welcome to Maths Tutorial Session-2

Introduction to Programming

Running R Script

- You can click on "Source“ to run a R script.
- The source () function instructs R to read the text file and execute its contents.

source(“myScript.R”)

[Ctrl+Shift+S]

- Optional parameter echo=TRUE will echo the script lines before they are executed

source("myScript.R", echo=TRUE)

[Ctrl+Shift+Enter]

If Statement

- An if statement consists of a Boolean expression followed by one or more statements.
- Syntax :

```
if(boolean_expression) {
  // statement(s) will execute if the boolean expression is
true.
}
```

If Statement

- If statements operate on length-one logical vectors.

Syntax for If statements

if(cond1=true) { cmd1 } else { cmd2 }

Example:

```
if(1==0) {
    print("True condition")
} else {
    print("False condition")
}
```

OUTPUT:

```
False condition
```

If-else Statement

- If else statements operate on vectors of variable length.

Syntax

ifelse(test, true_value, false_value)

Example:

```
 x <- 1:10                    # Creates sample data
ifelse(x<5 | x>8, x, 0)
```

OUTPUT:

```
[1]  1  2  3  4  0  0  0  0  9 10
```

ifelse() function

- ifelse() function is nothing but a vector equivalent form of if..else.
- ifelse(expression, yes, no)
- **expression**– A logical expression, which may be a vector.
- **yes** – What to return if expression is TRUE.
- **no** – What to return if expression is FALSE.

Example:

```
a = c(1,2,3,4)
ifelse(a %% 2 == 0,"even","odd")
```

Nested if...else statement

- Only one statement will get executed depending upon the test expressions.
- **Syntax:**

```
 if (expression1) {
       statement1
 } else if (expression2) {
       statement2
 } else if (expression3) {
       statement3
 } else {
```

Contd..

- Example:

```
x <- c("what","is","truth")
if("Truth" %in% x){
  print("Truth is found the first time")
} else if ("truth" %in% x) {
  print("truth is found the second time")
} else {
  print("No truth found")
}
```

For Loop

- For loop in R executes code statements for a particular number of times.

Syntax:

```
for (val in sequence) {
        statement
}
```

Example 1:

```
vec <- c(1,2,3,4,5)
 for (val in vec) {
print(val)
```

Example 2 :

```
v <- LETTERS[1:4]
 for ( i in v) {
print(i)
```

While Loop

- The While loop executes the same code again and again until a stop condition is met.

Syntax:

```
while (test_expression) {
    statement
}
```

- Example 1: Example 2:

```
 z <- 0                     v <- c("Hello","while loop")
while(z < 5) {                      cnt <- 2
  z <- z + 2                    while (cnt < 7){
  print(z)                    print(v)
```

break statement

A break statement is used inside a loop to stop the iterations and flow the control outside of the loop.

Example:

```
        num <- 1:5
        for (val in num) {
        if (val == 3){
         break
        }
      print(val)
}
```

next statement

- A next statement is useful when you want to skip the current iteration of a loop alone.

Example:

```
       num <- 1:5
        for (val in num) {
        if (val == 3){
         next
       }
      print(val)
}
```

output 1,2,4,5

switch function

- switch function is more like controlled branch of if else statements.

Syntax :

```
switch (expression, list)
```

Example 1:

```
switch(2, "apple", “ball" , "cat")
```

Example 2:

```
color = "green"
```

scan() function

- scan() function helps to read data from console.
- reading data from console

  x <- scan()

Example:

```
# Reading in numeric data
x <- scan()
1: 3 5 6
4: 3 5 78 29
8:
Read 7 items
```

Contd..

- # Reading in string data
- # empty quotes indicates character input

```
 y <- scan(what=" ")
1: red blue
3: green red
5: blue yellow
7:
Read 6 items
```

- y
- [1] "red" "blue" "green" "red" "blue" "yellow“

R - Function

- A function is a set of statements organized together to perform a specific task. R has a large number of in - built functions and the user can create their own functions.

- Elements of user –defined functions

1. Function definition.
2. Function call.

Function Definition

- An R function is created by using the keyword **function**. The basic syntax of an R function definition is as follows:

```
function_name <- function(arg_1,arg_2,...arg_N)
{
Function
body
}
```

Built - in Functions

- R has many in-built functions which can be directly called in the program without defining them first. We can also create and use our own functions referred as user defined functions.
- Example:

  seq()

  mean()

  max()

  sum()

- They are directly called by user written programs.

Example

- # Create a sequence of numbers from 32 to 44.

```
print(seq(32,44))
```

- # Find mean of numbers from 25 to 82.

```
print(mean(25:82))
```

- # Find sum of numbers frm 1 to 5.

```
print(sum(1:5))
```

User - defined Function

- We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions.

```
# Create a function to print squares of numbers in sequence.

   new.function <-function(a) {

            for(i in 1:a) {

                      b <-i^2

                      print(b)

                    }

                          }

# Call the function new.function supplying 6 as an argument.

       new.function(6)
```

Calling a Function without an Argument

```
# Create a function without an argument.

new.function <- function() {
  for(i in 1:5) {
  print(i^2)
  }
     }

# Call the function without supplying an argument.
new.function()
```

Calling a Function with Argument Values (by position and by name)

- The arguments to a function call can be supplied in the same sequence as defined in the function or they can be supplied in a different sequence but assigned to the names of the arguments.

Example:

```
# Create a function with arguments.
new.function <- function(a,b,c) {
 result <- a*b+c
 print(result)
   }
```

Contd..

```
# Call the function by position of arguments.
new.function(5,3,11)

# Call the function by names of the arguments.
new.function(a=11,b=5,c=3)
```

Result:

```
[1] 26
[1] 58
```

Contd..

Example:

```
# Create a function with arguments.
new.function <- function(a = 3,b =6) {
  result <- a*b
  print(result)
       }
# Call the function without giving any argument.
new.function()
# Call the function with giving new values of the argument.
new.function(9,5)
```

Example 2

```
# define a simple function
myFirstFun<-function(n)
{
  n*n             #  compute the square of integer n
}

# define a value
k<-10
```

Example 3

```
# we define the function and specify the exponent, second
argument directly

MyFourthFun <- function(n, y = 2)

{

  n^y  #  compute the power of n to the y

}

MyFourthFun(2,3) # specify both args

MyFourthFun(2)   # or just first'
```

Tasks.. Or Knowledge Check

- what will be the output of print.

```
     num <- 1:5
      for (val in num) {
      next
     break
    print(val) }
```

- A. Error
- B. output 3,4,5
- C. Program runs but no output is produced

Work yourself – Display commands

Q1.

```
a=4

b=5

sum=a+b

………………..
```

Write a single command to display the following output:

```
The sum of  4 and 5 is 9
```

Assignment

- Write a R script to check whether a person is eligible to vote or not.
- Write a R script to print the numbers until it is less than 15 and end the loop if it encounters number 12.
- Write a R script to find sum of natural numbers.