CM 2062 - Statistical Computing with R
Lab Sheet 7

Writing functions in R

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.

In R, a function is an object so the R interpreter is able to pass control to the function, along with arguments that may be necessary for the function to accomplish the actions.

The function in turn performs its task and returns control to the interpreter as well as any result which may be stored in other objects.

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, ...) {
   Function body
}
```

Function Components

The different parts of a function are-

Function Name - This is the actual name of the function. It is stored in R environment as an object with this name.

Arguments - An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional; that is, a function may contain no arguments. Also arguments can have default values.

Function Body - The function body contains a collection of statements that defines what the function does.

Return Value - The return value of a function is the last expression in the function body to be evaluated.

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.

R in-built Functions

Simple examples of in-built functions are **mean()**, **max()**, **sum()** and **seq()** ... etc. They are directly called by user written programs.

Examples

```
# Find mean of numbers from 25 to 82.
mean(25:82)
[1] 53.5

# Find sum of numbers frm 41 to 68.
sum(41:68)
[1] 1526

# Create a sequence of numbers from 32 to 44.
seq(32,44)
[1] 32 33 34 35 36 37 38 39 40 41 42 43 44
```

User-defined Functions

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. Below is an example of how a function is created and used.

Examples

```
new.function <- function(a, b) {
  result <- a * b
  result
}

> new.function(2, 3)
[1] 6

> new.function(4, 5)
[1] 20

new.function <- function(a, b, c) {
  result <- a * b + c
  result
}

> new.function(2, 3, 1)
```

```
[1] 7
> new.function(4, 5, 2)
[1] 22

pow <- function(x, y) {
  # function to print x raised to the power y
   result <- x^y
   print(result)
   print(paste(x,"raised to the power", y, "is", result))
}

> pow(2, 3)
[1] 8
[1] "2 raised to the power 3 is 8"

> pow(4, 5)
[1] 1024
[1] "4 raised to the power 5 is 1024"
```

Example

```
x <- c(1, 5, 4, 2, 8, 6, 7, 9, 10)
# R in-built function to find the mean is
mean(x)
[1] 5.777778

# Let's write a user defined function to find the mean.

MEAN <- function(x) {
  M <- sum(x)/length(x)
  M
}

> MEAN(x)
[1] 5.777778

> x <- rnorm(100, mean=2, sd=1)
> MEAN(x)
[1] 2.082384
```

Exercise

Write a function to get the temperature from Celsius for a given Fahrenheit value according to the below formula.

$$C = (F - 32) * (5/9)$$

Find the temperature from Celsius for 32 F, 50 F from your created function.

Example

```
y <- 2
g <- function(x){
  x * y
}

f <- function(x){
  y^2 + g(x)
}
f(3)
[1] 10
```

Exercise 1
Write a function to get sum of squares of deviation.

$$\sum (x - \mu)^2$$

Consider x= 1:10.

Exercise 2
Consider the following data set where examination marks for three students are given for three subjects.

Name	Sub1	Sub2	Sub3
A1	45	78	67
B1	56	45	87
C1	78	90	43

(a) Create a data frame call "marks" and use "Name" variable as raw names.

(b) Suppose you want to calculate the total mark obtained by each student. Write a function to get the total score for each student separately.

Loops in R

In R programming, we require a control structure to run a block of code multiple times. Loops come in the class of the most fundamental and strong programming concepts. A loop is a control statement that allows multiple executions of a statement or a set of statements. The word 'looping' means cycling or iterating.

A loop asks a query, in the loop structure. If the answer to that query requires an action, it will be executed. The same query is asked again and again until further action is taken. Any time the query is asked in the loop, it is known as an iteration of the loop. There are two components of a loop, the control statement, and the loop body. The control statement controls the execution of statements depending on the condition and the loop body consists of the set of statements to be executed.

There are three types of loop in R programming:

- For Loop
- While Loop
- Repeat Loop

For Loop in R

It is a type of control statement that enables one to easily construct a loop that has to run statements or a set of statements multiple times. For loop is commonly used to iterate over items of a sequence. It is an entry controlled loop, in this loop the test condition is tested first, then the body of the loop is executed, the loop body would not be executed if the test condition is false.

R – For loop Syntax:

```
for (value in sequence){

  statement

}
```

Example 1

Program to display numbers from 1 to 5 using for loop in R.

```
for (i in 1: 5){

    print(i)

}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

Here, for loop is iterated over a sequence having numbers from 1 to 5. In each iteration, each item of the sequence is displayed.

Exercise 1
Write a R programme to print numbers from 1 to 20.

Exercise 2
Consider the numbers from 1 to 10. Write a R programme to print the square of these numbers.

Example 2
Program to display days of a week.

```
week <- c('Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday')

for (day in week){

  print(day)

}
```

```
[1] "Sunday"
[1] "Monday"
[1] "Tuesday"
[1] "Wednesday"
[1] "Thursday"
[1] "Friday"
[1] "Saturday"
```

In the above program, initially, all the days(strings) of the week are assigned to the vector week. Then for loop is used to iterate over each string in a week. In each iteration, each day of the week is displayed.

Exercise 3
Write a R programme to display 12 months in a year.

Exercise 4
Consider the names "Max", "Tina", "Lindsey", "Anton", "Sharon". Write a R programme to display the name and the number of the characters in each name.

Nested for-Loop in R

In Nested For Loop in R, R makes use of the control structures to manage the execution of the expression, one such control structure is Nested For Loop a similar to basic 'for' loop executes. It can be defined as placing one 'for' loop inside the first 'for' loop is called as nesting or loop of loops in some terms, which takes the responsibility of two loops such that the outer loop controls the number of repetition of the whole inner detailed information until it is false, in other words, the inner loop executes n-times of every execution of the outer for loop and also it's a great tool to work with R Programming Language.

Syntax:

```
for (variable in sequence) {
for (variable in sequence)
{
expression      # expression statements can be a single
statement of R or group of statements.

} }
```

Example 1

```
for (i in 1:3) {
  for (j in 1:2) {
    print(paste("i =", i, "j= ",j))
  }
}
```

```
[1] "i = 1 j=  1"
[1] "i = 1 j=  2"
[1] "i = 2 j=  1"
[1] "i = 2 j=  2"
[1] "i = 3 j=  1"
[1] "i = 3 j=  2"
```

Example 2

```
for(i in 1:5)
{
  for(j in 1:2)
  {
    print(i*j);
  }
}
```

```
[1] 1
[1] 2
[1] 2
[1] 4
[1] 3
[1] 6
[1] 4
[1] 8
[1] 5
[1] 10
```

Exercise 1

Write and R programme to get below results.

```
   (0,0) (0,1) (0,2)
(1,0) (1,1) (1,2)
(2,0) (2,1) (2,2)
(3,0) (3,1) (3,2)
```

Example 3

Creating a matrix using nested for loop.

```
res <- matrix(nrow=4, ncol=4)

for(i in 1:nrow(res))
{
  for(j in 1:ncol(res))
  {
    res[i,j] = i*j
  }
}

print(res)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    2    4    6    8
[3,]    3    6    9   12
[4,]    4    8   12   16
```

if condition in for loop

Below is an example to count the number of even numbers in a vector.

```
x <- c(2,5,3,9,8,11,6)
count <- 0

for (val in x) {
  if(val %% 2 == 0)
    count = count+1
}
print(count)
```

```
[1] 3
```

break statement in R for loops

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

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

```
[1] 1
[1] 2
```

Exercise 1
Consider $i = 1 : 10$ and write a R programme to display the i^2 only for $i <= 4$. **The Use of**

"next" in R for Loop
"next" discontinues a particular iteration and jumps to the next cycle. In fact, it jumps to the evaluation of the condition holding the current loop.

```
x <- 1:5
for (i in x) {
  if (i == 2){
    next
  }
  print(i)
}
```

[1] 1
[1] 3
[1] 4
[1] 5

Exercise 1
Let $i = 1 : 10$ and write a R programme to get i^2 except $i = 1, 5, 7$.