Introduction to R Software

Swayam Prabha

Lecture 39

Programming in R

Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

Slides can be downloaded from

http://home.iitk.ac.in/~shalab/sp

Steps to write a programme

- A programme is a set of instructions or commands which are written in a sequence of operations i.e., what comes first and what comes after that.

- The objective of a programme is to obtain a defined outcome based on input variables.

- The computer is instructed to perform the defined task.

Steps to write a programme

- **Computer is an obedient worker but it has its own language.**

- **We do not understand computer's language and computer does not understand our language.**

- **The software help us and works like an interpreter between us and computer.**

Steps to write a programme

- We say something in software's language and software informs it to computer.

- Computer does the task and informs back to software.

- The software translates it to our language and informs us.

Steps to write a programme

- **Programme in R is written as a function using `function`.**

- **Write down the objective, i.e., what we want to obtain as an outcome.**

- **Translate it in the language of R.**

- **Identify the input and output variables.**

- **Identify the nature of input and output variables, i.e., numeric, string, factor, matrix etc.**

Steps to write a programme

- Input and output variables can be single variable, vector, matrix or even a function itself.

- The input variables are the component of `function` which are reported in the argument of `function()`

- The output of a `function` can also be input to another `function.`

- The output of an outcome can be formatted as per the need and requirement.

Steps to write a programme

Tips:

- **Loops usually slower the speed of programmes, so better is to use vectors and matrices.**
- **Use # symbol to write comment to understand the syntax.**
- **Use the variable names which are easy to understand.**
- **Don’t forget to initialize the variables.**

Example 1

Suppose we want to compute

$$\frac{\sum_{i=1}^{n} x_i^3}{\sum_{i=1}^{n} y_i^3} \quad \text{and} \quad \sum_{i=1}^{n} \left(\frac{x_i}{y_i} \right)^3$$

Data $x_1, x_2, \ldots, x_n$ $y_1, y_2, \ldots, y_n$

`x, y`**:** **Two data vectors**

Example 1

Input variables : `x`, `y`, `n` **(if `x` and `y` have different number of observations, choose different numbers, say `n1` and `n2`)**

Output variables: `g`, `h`,

$$g = \frac{\sum_{i=1}^{n} x_i^3}{\sum_{i=1}^{n} y_i^3} \quad \text{and} \quad h = \sum_{i=1}^{n} \left(\frac{x_i}{y_i} \right)^3$$

We need summation, so use `sum` function or alternatively compute it through vectors.

Example 1

```
# Remove all data
rm(list = ls())

# Define input data vectors, for example
x = c(10,20,30)
y = c(1,2,3)

++++++START OF FUNCTION+++++++++
example1 <- function(x,y)

# Start of function body
{
# First give all other input variables

# Computation of number of observations
n <- length(x)
```

CONTD..

Example 1

CONTD…

```
 #Initialize the values to store cube values
x1 <- 0
y1 <- 0
z1 <- 0

#Start of loop
for (i in 1:n)
{
# Define x1, y1 and z1 to store their cubes
  x1[i] <- x[i]^3
  y1[i] <- y[i]^3
  z1[i] <- (x[i]/y[i])^3
#End of loop
  }
```

CONTD…

Example 1

CONTD…

```
# Obtain the sum of cube quantities
sum_cube_x <- sum(x1)
sum_cube_y <- sum(y1)
sum_cube_z <- sum(z1)

# Computation of g and h
g <- sum_cube_x/sum_cube_y
h <- sum_cube_z

# Format the output
cat("The value of g and h are", g, "and", h,
"\n", )
}
++++++END OF FUNCTION++++++++
```

Example 1: At a glance

```
example1 <- function(x,y)
{
  n <- length(x)
  x1 <- 0
  y1 <- 0
  z1 <- 0
  for (i in 1:n)
  {
    x1[i] <- x[i]^3
    y1[i] <- y[i]^3
    z1[i] <- (x[i]/y[i])^3
  }
  sum_cube_x <- sum(x1)
  sum_cube_y <- sum(y1)
  sum_cube_z <- sum(z1)
  g <- sum_cube_x/sum_cube_y
  h <- sum_cube_z
  cat("The value of g and h are", g, "and", h,
         "respectively", "\n")
}
```

Example 1

R Console

```
> example1 <- function(x,y)
+ {
+   n <- length(x)
+   x1 <- 0
+   y1 <- 0
+   z1 <- 0
+   for (i in 1:n)
+   {
+     x1[i] <- x[i]^3
+     y1[i] <- y[i]^3
+     z1[i] <- (x[i]/y[i])^3
+   }
+   sum_cube_x <- sum(x1)
+   sum_cube_y <- sum(y1)
+   sum_cube_z <- sum(z1)
+   g <- sum_cube_x/sum_cube_y
+   h <- sum_cube_z
+   cat("The value of g and h are", g, "and", h,
+          "respectively", "\n")
+ }
```

Example 1

R Console

```
> example1
function(x,y)
{
  n <- length(x)
  x1 <- 0
  y1 <- 0
  z1 <- 0
  for (i in 1:n)
  {
    x1[i] <- x[i]^3
    y1[i] <- y[i]^3
    z1[i] <- (x[i]/y[i])^3
  }
  sum_cube_x <- sum(x1)
  sum_cube_y <- sum(y1)
  sum_cube_z <- sum(z1)
  g <- sum_cube_x/sum_cube_y
  h <- sum_cube_z
  cat("The value of g and h are", g, "and", h,
             "respectively", "\n")
}
> |
```

Example 1

```
> x=c(10,20,30)
> y=c(1,2,3)
> example1(x,y)
The value of g and h are 1000 and 3000
respectively

> x=c(67,87,26,85,6,45)
> y=c(54,64,22,94,20,88)
> example1(x,y)
The value of g and h are 0.862584 and 6.972778
respectively
```

Just by changing the values of `x` and `y`, one can get required different outcomes.

Example 1

R Console

```
> x=c(10,20,30)
> y=c(1,2,3)
> example1(x,y)
The value of g and h are 1000 and 3000 respectively
>
> x=c(67,87,26,85,6,45)
> y=c(54,64,22,94,20,88)
> example1(x,y)
The value of g and h are 0.862584 and 6.972778 respectively
> |
```