

Foundations of R Software

Lecture 53

Some Examples of R Programming

Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

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

$$\frac{\sum_{i=1}^n x_i^2}{\sum_{i=1}^n y_i^2} \quad \text{and} \quad \sum_{i=1}^n \left(\frac{x_i}{y_i} \right)^2$$

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

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,**

$$g = \frac{\sum_{i=1}^n x_i^2}{\sum_{i=1}^n y_i^2} \quad \text{and} \quad h = \sum_{i=1}^n \left(\frac{x_i}{y_i} \right)^2$$

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)
```

Example 1

CONTD...

#Initialize the values to store squared values

x1 = 0

y1 = 0

z1 = 0

#Start of loop

for (i in 1:n)

{

Define x1, y1 and z1 to store their squares

x1[i] = x[i]^2

y1[i] = y[i]^2

z1[i] = (x[i]/y[i])^2

#End of loop

}

CONTD...

Example 1

CONTD...

```
# Obtain the sum of squared quantities
```

```
sum_square_x = sum(x1)
```

```
sum_square_y = sum(y1)
```

```
sum_square_z = sum(z1)
```

```
# Computation of g and h
```

```
g = sum_square_x/sum_square_y
```

```
h = sum_square_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]^2
    y1[i] = y[i]^2
    z1[i] = (x[i]/y[i])^2
  }
  sum_square_x = sum(x1)
  sum_square_y = sum(y1)
  sum_square_z = sum(z1)
  g = sum_square_x/sum_square_y
  h = sum_square_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]^2
    y1[i] = y[i]^2
    z1[i] = (x[i]/y[i])^2
  }
  sum_square_x = sum(x1)
  sum_square_y = sum(y1)
  sum_square_z = sum(z1)
  g = sum_square_x/sum_square_y
  h = sum_square_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]^2
+     y1[i] = y[i]^2
+     z1[i] = (x[i]/y[i])^2
+   }
+   sum_square_x = sum(x1)
+   sum_square_y = sum(y1)
+   sum_square_z = sum(z1)
+   g = sum_square_x/sum_square_y
+   h = sum_square_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 100 and 300 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 0.8996568 and 5.953203 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 100 and 300 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 0.8996568 and 5.953203 respectively

Example 1 (Alternative approach)

Input variables : x, y, n

Output variables: $g, h,$

$$g = \frac{\sum_{i=1}^n x_i^2}{\sum_{i=1}^n y_i^2} \quad \text{and} \quad h = \sum_{i=1}^n \left(\frac{x_i}{y_i} \right)^2$$

$$g = \text{sum}(x^2)/\text{sum}(y^2)$$

$$h = \text{sum}(x/y)^2$$

Example 2

Suppose we want to compute

$$f(x, y) = \frac{\left(\frac{x + \ln y}{y}\right)^2}{5 + \left(\frac{x + \ln y}{y}\right)^3} \left[\exp\left(\frac{x + \ln y}{y}\right) \right]^{\frac{2}{3}}$$

This can be written as

$$f(x, y) = \frac{(g(x, y))^2}{5 + (g(x, y))^3} \left[\exp(g(x, y)) \right]^{\frac{2}{3}}$$

where $g(x, y) = \frac{x + \ln y}{y}$

Example 2

Input variables : x, y

Output variables: : f

We break this function in two components –

- Compute $g(x, y)$ as a function and then
- compute $f(x, y)$ by calling $g(x, y)$.

Example 2

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

```
# Define input data vectors  
x  
y
```

CONTD...

Example 2

CONTD...

```
# define g(x,y)
g = function(x,y)
# Start of function
{
    (x+log(y))/y
# End of function
}
```

$$g(x, y) = \frac{x + \ln y}{y}$$

+++++

```
# define f(x,y)
f = function(x,y)
{
    (((g(x,y))^2)/(5+(g(x,y))^3)) * (exp(g(x,y)))^(2/3)
}
```

$$f(x, y) = \frac{(g(x, y))^2}{5 + (g(x, y))^3} \left[\exp(g(x, y)) \right]^{\frac{2}{3}}$$

Example 2: At a glance

```
# define g(x,y)
```

```
g = function(x,y)
{
  (x+log(y))/y
}
```

```
+++++
```

```
# define f(x,y)
```

```
f = function(x,y)
{
  (((g(x,y))^2)/(5+(g(x,y))^3))*(exp(g(x,y)))^(2/3)
}
```

```
# g(x,y) must have been defined earlier.
```

Example 2

R Console

```
> # define g(x,y)
> g = function(x,y)
+ # Start of function
+ {
+   (x+log(y))/y
+ # End of function
+ }
>
> # define f(x,y)
> f = function(x,y)
+ {
+   ((g(x,y))^2)/(5+(g(x,y))^3)*exp(g(x,y))^(2/3)
+ }
> |
```

Example 2

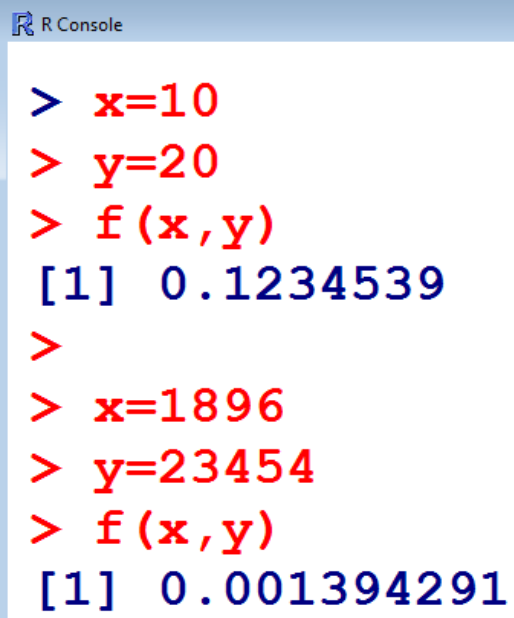
R Console

```
> g
function(x,y)
  # Start of function
{
  (x+log(y))/y
  # End of function
}
> f
function(x,y)
{
  (((g(x,y))^2)/(5+(g(x,y))^3))* (exp(g(x,y)))^(2/3)
}
```

Example 2

```
> x=10
> y=20
> f(x,y)
[1] 0.1234539

> x=1896
> y=23454
> f(x,y)
[1] 0.001394291
```

A screenshot of an R console window with a blue header bar containing the R logo and the text "R Console". The console shows two sets of commands and their outputs. The first set includes x=10, y=20, and f(x,y), resulting in [1] 0.1234539. The second set includes x=1896, y=23454, and f(x,y), resulting in [1] 0.001394291. The prompt character is a blue greater-than sign (>).

```
> x=10
> y=20
> f(x,y)
[1] 0.1234539
>
> x=1896
> y=23454
> f(x,y)
[1] 0.001394291
```

There is no need to calculate the value of $g(x,y)$.

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

Example 3

Suppose we want to compute

$$f(x) = \begin{cases} \exp\left(\frac{x + \ln(1 + x^3)}{x^2}\right) & \text{if } x > 0 \\ 10 & \text{if } x = 0 \\ \frac{2 + x^3}{x} & \text{if } x < 0 \end{cases}$$

and plot with line over a values of x as a sequence starting from -1 to 5 and increasing it by 0.2.

Example 3

Input variable : **x**

Output variable: **f**

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

```
# Define input data  
x
```

CONTD...

Example 3

CONTD...

```
f = function(x)
{
  if(x>0) {exp((x+log(1+x^3))/x^2)}

  else if(x==0) {10}

  else {(2+x^3)/x}
}
```

$$f(x) = \begin{cases} \exp\left(\frac{x + \ln(1 + x^3)}{x^2}\right) & \text{if } x > 0 \\ 10 & \text{if } x = 0 \\ \frac{2 + x^3}{x} & \text{if } x < 0 \end{cases}$$

CONTD...

Example 3

CONTD...

```
h = function()  
# Start of function  
{  
# Generation of data on x  
x = seq(-1,5,by=0.2)  
# Initialization of y to store values of f(x)  
y = 0
```

CONTD...

Example 3

CONTD...

```
# Generation of f(x) values corresponding to x
for(i in 1:length(x))
{
    y[i] = f(x[i])
}
# length(x) and length(y) must be same to plot
# y = f(x) with respect to x
plot(x, y, type = "l")
}
```

Example 3: At a glance

```
f = function(x)
{
  if(x>0) {exp((x+log(1+x^3))/x^2)}
  else if(x==0) {10}
  else {(2+x^3)/x}
}

h = function()
{
x = seq(-1, 5, by=0.2)
y = 0
for(i in 1:length(x))
{
  y[i] = f(x[i])
}
plot(x,y,type = "l")
}
```

Example 3

```
R Console
> f = function(x)
+ {
+   if(x>0) {exp((x+log(1+x^3))/x^2)}
+   else if(x==0) {10}
+   else {(2+x^3)/x}
+ }
>
> h = function()
+ {
+   x = seq(-1,5,by=0.2)
+   y = 0
+   for(i in 1:length(x))
+   {
+     y[i] = f(x[i])
+   }
+   plot(x,y,type = "l")
+ }
> |
```

Example 3

```
R Console

> f
function(x)
{
  if(x>0) {exp((x+log(1+x^3))/x^2)}
  else if(x==0) {10}
  else {(2+x^3)/x}
}
> h
function()
{
  x = seq(-1,5,by=0.2)
  y = 0
  for(i in 1:length(x))
  {
    y[i] = f(x[i])
  }
  plot(x,y,type = "l")
}
> |
```

Example 3

```
> f(123)
[1] 1.009126
```

```
> f(-123)
[1] 15128.98
```

```
> f(0)
[1] 10
```

```
> f(8)
[1] 1.249201
```

```
> f(-4)
[1] 15.5
```

```
> f(0)
[1] 10
```

R Console

```
> f(123)
[1] 1.009126
>
> f(-123)
[1] 15128.98
>
> f(0)
[1] 10
>
>
> f(8)
[1] 1.249201
> f(-4)
[1] 15.5
> f(0)
[1] 10
```

Example 3

> h()

