ADITYA ENGINEERING COLLEGE (A)
Aditya Nagar, ADB Road, Surampalem
Department of Information Technology

R Programming

Name of the Faculty: Mrs. S.Sushma, Assistant Professor, Department of IT

Subject: R Programming

Year & Semester: III - I

Topic: Functions in R: Syntax and Examples

Conventional Methods: Chalk & Talk

Teaching Methodology: Practical Demonstration

In R Programming, Functions are the programming artifacts that are supported by the R runtime environment to process the programming logic efficiently. R language supported both native function syntax to create a custom function and system define functions that do some predefined task. I explained examples of functions by Practical Demonstration. I explained R program function and object which takes zero or more parameter, to process some programming operations and provides the result as the return value. R program function is useful for reusability and intuitive code writing in R language.

References:

1.https://r4ds.had.co.nz/functions.html

2.https://www.dataquest.io/blog/write-functions-in-r/

Functions in R: Syntax and Examples-

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.

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, ...) {
   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.

Built-in Function

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

```
# 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 41 to 68.
print(sum(41:68))
```

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

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

new.function <- function(a) {

  for(i in 1:a) {

    b <- i^2

    print(b)

  }

}
```

Calling a Function

```
# 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)
```

Lazy Evaluation of Function

Arguments to functions are evaluated lazily, which means so they are evaluated only when needed by the function body.

```
# Create a function with arguments.

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

```
  print(a^2)
  print(a)
  print(b)
}
# Evaluate the function without supplying one of the arguments.
new.function(6)
```

Practical Demonstration of R function: