SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

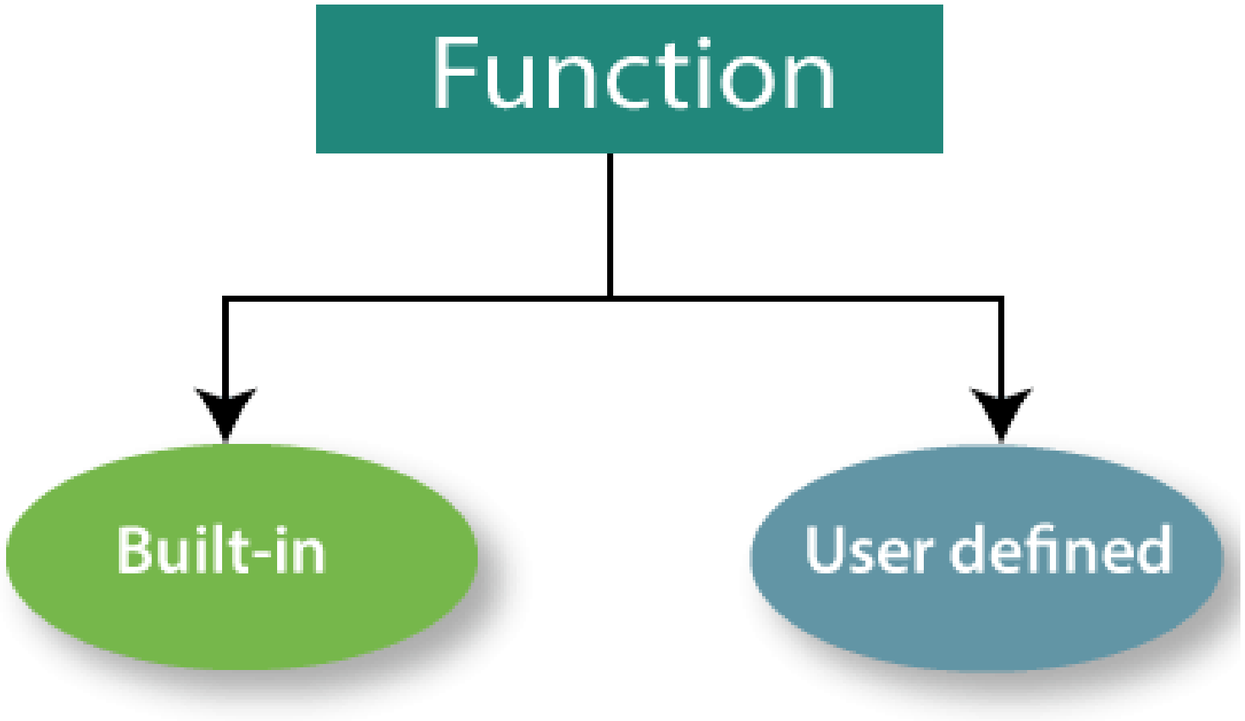
DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

COURSE NAME :24CS407 DATA ANALYTICS WITH R

II YEAR /IV SEMESTER

Unit 4- R PROGRAMMING BASICS

Topic : Functions, R packages


Function
Built-in
User defined

R - Functions

- 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

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.

Built-in Function

- 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.

- Simple examples of in-built functions are seq(), mean(), max(), sum(x) and paste(...) etc.

- They are directly called by user written programs. You can refer most widely used R functions.

Built-in Function

```
# 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))
```

[1] 32 33 34 35 36 37 38 39 40 41 42 43 44

[1] 53.5

[1] 1526

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)
```

[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36

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()
```

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```

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

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

# 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)
```

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

Calling a Function with Default Argument

We can define the value of the arguments in the function definition and call the function without supplying any argument to get the default result. But we can also call such functions by supplying new values of the argument and get non default result.

```
# 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)
```

```
[1] 18
[1] 45
```

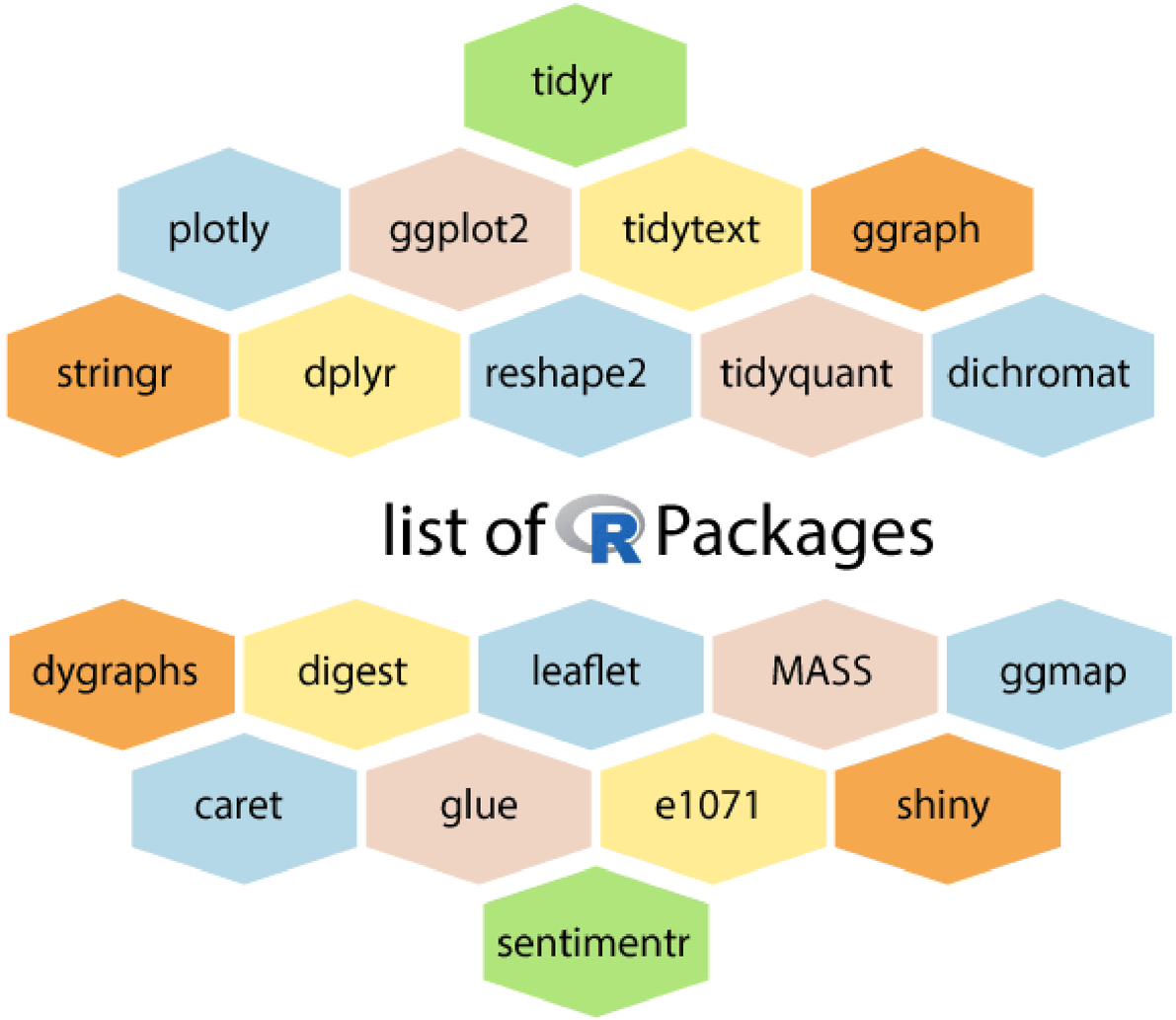
Lazy Evaluation of Function

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)
```

```
[1] 36
[1] 6
Error in print(b) : argument "b" is missing, with no default
```


tidyr
plotly
ggplot2
tidytext
ggraph
stringr
dplyr
reshape2
tidyquant
dichromat
list of R Packages
dygraphs
digest
leaflet
MASS
ggmap
caret
glue
e1071
shiny
sentimentr

R - Packages

- R packages are a collection of R functions, complied code and sample data. They are stored under a directory called "library" in the R environment. By default, R installs a set of packages during installation.

- More packages are added later, when they are needed for some specific purpose. When we start the R console, only the default packages are available by default. Other packages which are already installed have to be loaded explicitly to be used by the R program that is going to use them.

- All the packages available in R language are listed at R Packages.

R - Packages

- **Check Available R Packages**
- Get library locations containing R packages

.libPaths()

- When we execute the above code, it produces the following result. It may vary depending on the local settings of your pc.

[2] "C:/Program Files/R/R-3.2.2/library"

```
Packages in library 'C:/Program Files/R/R-3.2.2/library':

base                    The R Base Package
boot                    Bootstrap Functions (Originally by Angelo Canty
                        for S)
class                   Functions for Classification
cluster                 "Finding Groups in Data": Cluster Analysis
                        Extended Rousseeuw et al.
codetools               Code Analysis Tools for R
compiler                The R Compiler Package
datasets                The R Datasets Package
foreign                 Read Data Stored by 'Minitab', 'S', 'SAS',
                        'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...
graphics                The R Graphics Package
grDevices               The R Graphics Devices and Support for Colours
                        and Fonts
grid                    The Grid Graphics Package
KernSmooth              Functions for Kernel Smoothing Supporting Wand
                        & Jones (1995)
lattice                 Trellis Graphics for R
MASS                    Support Functions and Datasets for Venables and
                        Ripley's MASS
Matrix                  Sparse and Dense Matrix Classes and Methods
methods                 Formal Methods and Classes
mgcv                    Mixed GAM Computation Vehicle with GCV/AIC/REML
                        Smoothness Estimation
nlme                    Linear and Nonlinear Mixed Effects Models
nnet                    Feed-Forward Neural Networks and Multinomial
                        Log-Linear Models
parallel                Support for Parallel computation in R
rpart                   Recursive Partitioning and Regression Trees
spatial                 Functions for Kriging and Point Pattern
                        Analysis
splines                 Regression Spline Functions and Classes
stats                   The R Stats Package
stats4                  Statistical Functions using S4 Classes
survival                Survival Analysis
tcltk                   Tcl/Tk Interface
tools                   Tools for Package Development
utils                   The R Utils Package
```

Get the list of all the packages installed

library()

When we execute the above code, it produces the following result. It may vary depending on the local settings of your pc.

Packages in library 'C:/Program Files/R/R-3.2.2/library':

Get all packages currently loaded in the R environment

Search()

When we execute the above code, it produces the following result. It may vary depending on the local settings of your pc.

```
[1] ".GlobalEnv"         "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"    "Autoloads"         "package:base"
```

Install a New Package

There are two ways to add new R packages. One is installing directly from the CRAN directory and another is downloading the package to your local system and installing it manually.

Install directly from CRAN

The following command gets the packages directly from CRAN webpage and installs the package in the R environment. You may be prompted to choose a nearest mirror. Choose the one appropriate to your location.

```
install.packages("Package Name")
```

Install the package named "XML".

```
install.packages("XML")
```

Install package manually

Go to the link R Packages to download the package needed. Save the package as a .zip file in a suitable location in the local system.

Now you can run the following command to install this package in the R environment.

install.packages(file_name_with_path, repos = NULL, type = "source")

Install the package named "XML"

install.packages("E:/XML_3.98-1.3.zip", repos = NULL, type = "source")

Load Package to Library

Before a package can be used in the code, it must be loaded to the current R environment. You also need to load a package that is already installed previously but not available in the current environment.

A package is loaded using the following command –

library("package Name", lib.loc = "path to library")

Load the package named "XML"

install.packages("E:/XML_3.98-1.3.zip", repos = NULL, type = "source")

Assessment 1

References

1. João Moreira, Andre Carvalho, Tomás Horvath – “A General Introduction to Data Analytics” – Wiley -2018
2. https://www.tutorialspoint.com/r/r_functions.htm
3. https://www.tutorialspoint.com/r/r_packages.htm

Thank You