R Tutorial — Basic commands

Prerequisites:

• Download R

• Download R studio

If you have no experience with the R language, here are some resources for getting started:

• A (very) short introduction to R: https://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf

• Getting up to speed with R: https://rkabacoff.github.io/datavis/Rintro.pdf

1. R Overview

R is a programming language and software environment for statistical analysis, graphics representation and reporting. The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions.

Rstudio is an Integrated Development Environment (IDE) for R. There are two free versions *Rstudio Desktop* and *Rstudio Server*. You can download it from https://www.rstudio.com/products/rstudio/download/.

1.1. Features of R

The following are the important features of R:

• R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.

• R has an effective data handling and storage facility,

• R provides a suite of operators for calculations on arrays, lists, vectors and matrices.

• R provides a large, coherent and integrated collection of tools for data analysis.

• R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

2. Basic Syntax

As a convention, we will start learning R programming by writing a "Hello, World!" program.

```
# My first program in R Programming
myString <- "Hello, World!"
print ( myString)
```

```
## [1] "Hello, World!"
```

```
# Add two numbers
print(7 + pi)
```

```
## [1] 10.14159
```

2.1. Navigating directories

- `getwd()`: Returns the current working directory.
- `setwd()`: Set the working directory.
- `dir()`: Return the list of the directory.
- `sessionInfo()`: Return the session of the windows.
- `date()`: Return the current date.

```
# Show directory, session information, date
getwd()
dir()
sessionInfo()
date()
```

2.2. Installing packages

```
# install packages
install.packages("rmarkdown")
install.packages("knitr")
install.packages("lattice")
install.packages("ggplot2")
# load the required package
library(lattice)

# Use help() function for on-line help
help(print)
```

2.3. Clearing the console and environment

We clear console in R and RStudio, In some cases when you run the codes using "source" and "source with echo" your console will become messy. And it is needed to clear the console. So let's now look at how to clear the console. The console can be cleared using the shortcut key **ctrl + L**.

```
# Define a variable a=1
a <- 1
# Remove a variable a
rm(a)
# Remove all the variables in the current environment
rm(list=ls())
```

2.4. Assignment commands

In R, the assignment can be denoted in three ways:

1. `=` (Simple Assignment)
2. `<-` (Leftward Assignment)
3. `->` (Rightward Assignment)

This is an example:

```
# Assignment values to three variables in three ways
var1 = "Simple Assignment"
var2 <- "Leftward Assignment"
"Rightward Assignment" -> var3
```

```
# print the three variables
print(var1)
```

```
## [1] "Simple Assignment"
```

```
print(var2)
```

```
## [1] "Leftward Assignment"
```

```
print(var3)
```

```
## [1] "Rightward Assignment"
```

3. Data Types

In contrast to other programming languages like C and java in R, the variables are not declared as some data type. The variables are assigned with **R-Objects** and the data type of the R-object becomes the data type of the variable. There are many types of R-objects. The frequently used ones are:

- **Vectors**
- **Lists**
- **Matrices**
- **Arrays**
- **Factors**
- **Data Frames**

There are six data types of atomic vectors. We show the general types including logical, numeric and character.

```
# Logical type
v <- TRUE
print(class(v))
```

```
## [1] "logical"
```

```
# Numeric type
v <- 7
print(class(v))
```

```
## [1] "numeric"
```

```
# Integer type
v <- 2L
print(class(v))
```

```
## [1] "integer"
```

```
# Character type
v <- "TRUE"
print(class(v))
```

```
## [1] "character"
```

3.1. Vectors

Vector object is the simplest one. When you want to create vector with more than one element, you should use `c()` function which means to combine the elements into a vector.

```r
# Create a vector
vec1 <- c(1,2,3)
print(vec1)
```

```
## [1] 1 2 3
```

```r
# Get the class of the vector
print(class(vec1))
```

```
## [1] "numeric"
```

```r
# More examples
vec2 <- 1:10
print(vec2)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
vec3 <- seq(from = 1, to = 5, by = 1)
print(vec3)
```

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

```r
color <- c("red", "green", "yellow")
print(color)
```

```
## [1] "red"    "green"  "yellow"
```

3.2. List

A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

```r
# Create a list
list1 <- list(c(1, 4, 7), 32.5, cos)


# Print the list
print(list1)
```

```
## [[1]]
## [1] 1 4 7
##
## [[2]]
## [1] 32.5
##
## [[3]]
## function (x)  .Primitive("cos")
```

3.3. Matrices

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

```r
# Create a matrix.
M = matrix(c(1, 2:6), nrow = 2, ncol = 3, byrow = TRUE)
print(M)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
```

```
## [2,]    4    5    6
```

3.4 Arrays

While matrices are confined to two dimensions, arrays can be of any number of dimensions. The array function takes a dim attribute which creates the required number of dimension. In the below example we create an array with two elements which are 3x3 matrices each.

```
# Create an array.
a <- array(c('red','blue'), dim = c(3, 3, 2))
print(a)
```

```
## , , 1
##
##      [,1]   [,2]   [,3]
## [1,] "red"  "blue" "red"
## [2,] "blue" "red"  "blue"
## [3,] "red"  "blue" "red"
##
## , , 2
##
##      [,1]   [,2]   [,3]
## [1,] "blue" "red"  "blue"
## [2,] "red"  "blue" "red"
## [3,] "blue" "red"  "blue"
```

3.5. Factors

Factors are the R-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels. The labels are always character irrespective of whether it is numeric or character or Boolean etc. in the input vector. They are useful in statistical modeling.

Factors are created using the `factor()` function.The nlevels functions gives the count of levels.

```
# Create a vector
apple_colors <- c('green', 'green', 'yellow', 'red', 'red', 'red', 'green')

# Create a factor object
factor_apple <- factor(apple_colors)

# Print the factor
print(factor_apple)
```

```
## [1] green  green  yellow red    red    red    green
## Levels: green red yellow
```

```
# Print the number of levels of the factor
print(nlevels(factor_apple))
```

```
## [1] 3
```

3.6. Data Frames

Data frames are tabular data objects. Unlike a matrix in data frame each column can contain different modes of data. The first column can be numeric while the second column can be character and third column can be logical. It is a list of vectors of equal length.

Data Frames are created using the `data.frame()` function.

```r
# Create the data frame
BMI <- data.frame(gender = c("Male", "Female", "Male"),
                  height = c(151, 175, 192),
                  weight = c(49, 62, 78),
                  Age = c(42, 38, 26))

print(BMI)
```

```
##   gender height weight Age
## 1   Male    151     49  42
## 2 Female    175     62  38
## 3   Male    192     78  26
```

4. File Handling

In R Programming, handling of files such as reading and writing files can be done by using in-built functions present in R base package. In this article, let us discuss reading and writing of CSV files, creating a file, renaming a file, check the existence of the file, listing all files in the working directory, copying files and creating directories.

4.1. Creating a File

Using `file.create()` function, a new file can be created from console or truncates if already exists. The function returns a TRUE logical value if file is created otherwise, returns FALSE.

```r
# Create a file
# The file created can be seen in your working directory
file.create("DataVis.txt")
```

```
## [1] TRUE
```

4.2. Writing Into a File

`write.table()` function in R programming is used to write an object to a file. This function is present in utils package in R and writes data frame or matrix object to any type of file.

```r
# Write iris dataset into the txt file
write.table(x = iris[1:10, ], file = "DataVis.txt")
```

4.3. Renaming a File

The `file.rename()` function renames the file and return a logical value. The function renames files but not directories.

```r
# Rename file DataVis.txt to newDataVis.txt
file.rename("DataVis.txt", "newDataVis.txt")
```

```
## [1] TRUE
```

4.4. Reading a File

Using `read.table()` function in R, files can be read and output is shown as dataframe. This functions helps in analyzing the dataframe for further computations.

```r
# Reading txt file
new.iris <- read.table(file = "newDataVis.txt")
```

```
# Print
print(new.iris)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1  setosa
```

`read.csv()`: read.csv() is used for reading "comma separated value" files (".csv"). In this also the data will be imported as a data frame.

`read.csv2()`: read.csv() is used for variant used in countries that use a comma "," as decimal point and a semicolon ";" as field separators.

```
# R program to read a file in table format

# Write iris dataset into the txt file
write.table(x = iris[1:10, ], file = "DataVis.csv")

# Using read.csv()
myData = read.csv("DataVis.csv")
print(myData)
```

```
##    Sepal.Length.Sepal.Width.Petal.Length.Petal.Width.Species
## 1                                    1 5.1 3.5 1.4 0.2 setosa
## 2                                      2 4.9 3 1.4 0.2 setosa
## 3                                    3 4.7 3.2 1.3 0.2 setosa
## 4                                    4 4.6 3.1 1.5 0.2 setosa
## 5                                      5 5 3.6 1.4 0.2 setosa
## 6                                    6 5.4 3.9 1.7 0.4 setosa
## 7                                    7 4.6 3.4 1.4 0.3 setosa
## 8                                      8 5 3.4 1.5 0.2 setosa
## 9                                    9 4.4 2.9 1.4 0.2 setosa
## 10                                  10 4.9 3.1 1.5 0.1 setosa
```

`read.delim()`: This method is used for reading "tab-separated value" files (".txt"). By default, point (".") is used as decimal points.

`read.delim2()`: This method is used for reading "tab-separated value" files (".txt"). By default, point (",") is used as decimal points.

```
# R program reading a text file

# Read a text file using read.delim()
myData = read.delim("newDataVis.txt", header = T)

print(myData)
```

```
##    Sepal.Length.Sepal.Width.Petal.Length.Petal.Width.Species
## 1                                    1 5.1 3.5 1.4 0.2 setosa
```

```
## 2                                    2 4.9 3 1.4 0.2 setosa
## 3                                   3 4.7 3.2 1.3 0.2 setosa
## 4                                   4 4.6 3.1 1.5 0.2 setosa
## 5                                    5 5 3.6 1.4 0.2 setosa
## 6                                   6 5.4 3.9 1.7 0.4 setosa
## 7                                   7 4.6 3.4 1.4 0.3 setosa
## 8                                    8 5 3.4 1.5 0.2 setosa
## 9                                   9 4.4 2.9 1.4 0.2 setosa
## 10                                 10 4.9 3.1 1.5 0.1 setosa
```

`file.choose()`: In R it's also possible to choose a file interactively using the function `file.choose()`, and if you're a beginner in R programming then this method is very useful for you.

```
# R program reading a text file using file.choose()
myFile = read.delim(file.choose(), header = FALSE)

# If you use the code above in RStudio you will be asked to choose a file
print(myFile)
```

Other reading file functions:

`read_tsv()`: This method is also used for to read a tab separated (`\t`) values by using the help of readr package.

`read_lines()`: This method is used for the reading line of your own choice whether it's one or two or ten lines at a time. To use this method we have to import reader package.

`read_file()`: This method is used for reading the whole file. To use this method we have to import reader package.

4.5. Reading Files From Web

It's possible to use the functions `read.delim()`, `read.csv()` and `read.table()` to import files from the web.

```
# R program to read a file from the internet

# Using read.delim()
myData = read.delim("http://www.sthda.com/upload/boxplot_format.txt")

print(head(myData))
```

Reference:

https://www.tutorialspoint.com/r/index.htm

https://www.geeksforgeeks.org/file-handling-in-r-programming/?ref=lbp