

# Introduction to Programming in R

## From basics to Advanced

**Md. Neaz Ali**

M.Sc in Statistics

Department of Statistics

Islamic University, Kushtia – 7003, Bangladesh

Email: [neazali77@gmail.com](mailto:neazali77@gmail.com)

## Contents

Chapter 1 Introduction to R Software .....	3
1.1 Obtaining R and RStudio .....	3
1.2 RStudio Interface .....	3
1.3.1 Console .....	3
1.3.2 Editor .....	3
1.3.3 Workspace, History .....	3
1.3.4 File, Plots, Packages, Help .....	4
1.4 Starting out – setting a working directory .....	4
1.5 R as a big calculator .....	4
1.6 A few important points on R .....	4
1.7 Operators in R .....	5
1.7.1 Arithmetic Operators .....	5
1.7.2 Logical Operators .....	5
1.7.3 Relational Operators .....	5
1.7.4 Assignment Operators .....	6
1.7.5 Miscellaneous Operators .....	6
1.8 Variables in R programming .....	6
1.9 Data Types in R Programming .....	6
1.10 Useful Functions in R e.g. rm(), ls() .....	7
1.11 Data Sets in R and save R Session .....	7
1.12 Installing new R Libraries .....	8
Chapter 2 Objects in R .....	9
2.1 Types of R Objects .....	9
2.2 Attributes of R Objects .....	10
2.2.1 Basic Attributes .....	10
2.2.2 Other Attributes, dimension .....	10
2.3 Creating and Accessing Objects .....	10

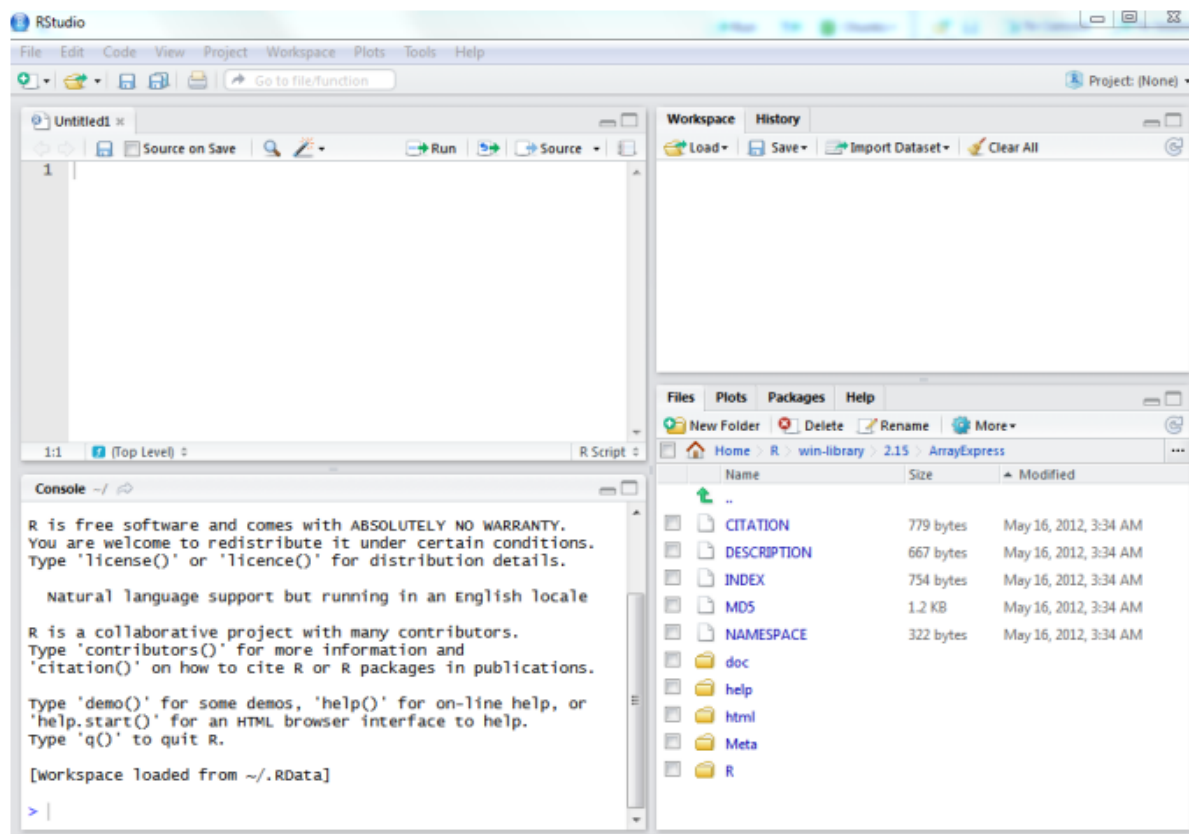
# Chapter 1 Introduction to R Software

## 1.1 Obtaining R and RStudio

R can be downloaded from the website: [Download R-4.4.2 for Windows. The R-project for statistical computing.](#) and R-Studio can be downloaded from the website: [RStudio Desktop - Posit](#)

## 1.2 RStudio Interface

RStudio is a free and an open source integrated development environment for R. On startup R Studio brings up a window with 3 or 4 panels. If you only see 3 panels, click on *File* → *New* → *New R Script*.



The bottom left panel “console” is the exact same as the standard R console. RStudio just loads your local version of R. You can specify a different version of R (if you have multiple versions of R running on your machine) by clicking on *Tools* → *Options* and selecting R version.

### 1.3.1 Console

RStudio has a nice console features

- Start typing a command, for example `fi`, press the TAB key, it will suggest function that begin with `fi`
- Select `fi` for `fishers.test`

### 1.3.2 Editor

The top left panel is an editor which can be used to edit R scripts (.R), plain text (.txt), html web files or any other files. There are several nice features to this text editor which we will describing in the following sections. But for this time being note, that it highlights R code, and that the code is searchable.

### 1.3.3 Workspace, History

On the top right there is a tab menu workspace and history.

- It lists the object in the current R session. You can load, save or “Clear All” objects for a workspace.
- There is the option to **Import Dataset**.

- The history panel lists all of the command that have been types or input in the console.

### 1.3.4 File, Plots, Packages, Help

On the bottom right there is a tab menu Files, Plots, Packages, and Help.

- **Files** is the file browser, which allows you to create a new folder, rename a folder or delete a folder.
- The **Plots** window displays plots generated in R. Simply type the following command into the Console window

```
plot(1:10)
plot(rnorm(10), 1:10)
```

- **Packages** lists all of the packages installed in you computer. The packages with a tick marked are those loaded in your current R session. Click on a package name to view help on that package. Note that you can **install packages or check for updates**.

```
install.packages("packagename")
# load the package
library(packagename)
```

- The **Help** menu provides an extensive R help.

```
?mean()
help(mean)
```

## 1.4 Starting out – setting a working directory

The first thing to do when starting an R session, is to ensure that you will be able to find your data and also that your output will be saved to a useful location on your computer hard-drive. Therefore, set a **working directory**.

There are numerous ways to set the **working directory**. To change directory:

1. In the classic R interface. Use the file menu, to change directory *File* → *Change dir*.
2. If you start R by clicking on an R icon. You may wish to change the default start location by right mouse clicking on the R icon on the desktop/start menu and changing the **Start In** property.
3. In RStudio Tools → **Set Working Directory**.
4. Use the **File** browser window to view the contents of a directory and navigate to the directory you wish to set as you home directory.
5. The commands to set the working directory

```
# What is my current directory
getwd()
# To Change the directory
setwd("File Path")
```

6. Creating a R Project: Top right corner of the window and following along with your desired name of the project and the directory to create a R project.

## 1.5 R as a big calculator

Type the following into an R session.

```
2 + 2
## [1] 4
2 * 2
## [1] 4
2 * 100 / 4
## [1] 50
```

## 1.6 A few important points on R

Elementary commands: *expressions* are evaluated, printed and value lost; *assignments* evaluate expression, passes value to a variable, but not automatically printed

```
2 * 5 ^ 2
```

```
## [1] 50
x = 2 * 5 ^ 2
print(x)
## [1] 50
```

## 1.7 Operators in R

Operators are the symbols directing the compiler to perform various kinds of operators between the operands e.g.

```
2 + 4
```

Here 2 and 4 are operands and (+) is the operator which is performing addition operation on the both operand 2 and 4.

R support majorly four kinds of binary operators between a set of operands. Namely they are

- Arithmetic Operators
- Logical Operators
- Relational Operators
- Assignment Operators
- Miscellaneous Operators

### 1.7.1 Arithmetic Operators

Arithmetic operators modulo using the specified operator between operands, which may be either scalar value, complex numbers, or vectors. The most common arithmetic operators are

- Addition (+): 5+4
- Subtraction (-): 4-2
- Multiplication (\*): 7\*2
- Division (/): 10/2
- Power (^): 2^5
- Modulo (%): 24 %% 5

### 1.7.2 Logical Operators

Logical operators in R simulate element-wise decision operations, based on the specified operator between the operands, which are then evaluated either a TRUE or FALSE value.

- Element-wise Logical AND operator (&): TRUE & TRUE = TRUE (if both side are true then result is true).
- Element-wise Logical OR operator (|): TRUE | FALSE = TRUE (if at least one side is true then result is true).
- NOT operator (!): A unary operator that negates the status of the elements of the operand. ! TRUE = FALSE.
- Logical AND operator (&&): Returns TRUE if both the first element of the operands is True.
- Logical OR operator (||): Return TRUE if either of the first elements of the operands is True.

### 1.7.3 Relational Operators

The Relational Operators in R carry out comparison operations between the corresponding elements of the operands. Returns a Boolean TRUE value if the first operand satisfies the relation compared to the second.

- Less than (<): 5 < 10 (TRUE)
- Greater than (>): 5 > 10 (FALSE)
- Less than equal to (<=): 5 <= 10 (TRUE)
- Greater than equal to (>=): 5 >= 10 (FALSE)
- Not equal to (!=): 5 != 10 (TRUE)
- Equal to (==): 5 == 10 (FALSE)

### 1.7.4 Assignment Operators

Assignment operators in R are used to assigning values to various data objects in R. the objects may be integers, vectors, functions, or dataframes etc. There are two kinds of assignment operators: Left and Right

- Left assignment (<-): `vec1 <- c("ab", TRUE)`
- Right assignment (->): `c("ab", TRUE) -> vec2`
- Or Simply equal to (=): `vec1 = c("ab", TRUE)`

**N.B:** Operations in programming flows from right to left and up to bottom manner.

### 1.7.5 Miscellaneous Operators

Miscellaneous operators are the mixed operators in R that simulate the printing of sequences and assignment of vectors, either left or right – handed.

- **%in% Operator:** Checks if an element belongs to a list and returns a Boolean value TRUE if value is present else FALSE e.g.,

```
val = 0.1
list1 = c(TRUE, 0.1, "apple")
print(val %in% list1)
```

- **%% Operator:** This operator is used to multiply a matrix with its transpose. Transpose of the matrix is obtained by interchanging the rows to columns and columns to rows.

```
mat = matrix(c(1,2,3,4,5,6),nrow=2,ncol=3)
print(mat)
print(t(mat))
pro = mat %% t(mat)
print(pro)
```

## 1.8 Variables in R programming

A variable is a memory allocated for the storage of specific data and the name associated with the variable is used to work around this reserved block. The name given to a variable is known as its variable name. Usually a single variable stores only the data belonging to a certain data type.

```
var1 = "hello"
print(var1)
```

Rules for Variable Naming:

- A valid variable name consists of a combination of alphabets, numbers, dot(.), and underscore(\_) characters. Example: `var.1_` is valid
- Apart from the dot and underscore operators, no other special character is allowed. Example: `var$1` or `var#1` both are invalid
- Variables can start with alphabets or dot characters. Example: `.var` or `var` is valid
- The variable should not start with numbers or underscore. Example: `2var` or `_var` is invalid.
- If a variable starts with a dot the next thing after the dot cannot be a number. Example: `.3var` is invalid
- The variable name should not be a reserved keyword in R. Example: `TRUE`, `FALSE`, etc.

**N.B.:** Always use relative names for variables for better understanding the work of the variable.

## 1.9 Data Types in R Programming

**R Data types** are used to specify the kind of data that can be stored in a variable. For effective memory consumption and precise computation, the right data type must be selected. Each R data type has its own set of regulations and restrictions. Variables are not needed to be declare with a data type in R, data type even can be changed. Data types in R are:

- Numeric – (3, 6.7, .121)
- Integer – (2L, 42L; where 'L' declares this as an integer)
- Logical – (TRUE)

- Complex – (7+5i; where ‘I’ is imaginary number)
- Character – (“a”, “B”)
- Raw – (as.raw(55); raw creates a raw vector of the specified length)
- String – (“Hello World”)

Basic Data Types	Values	Examples
Numeric	Set of all real numbers	<pre>"numeric_value &lt;- 3.14"</pre>
Integer	Set of all integers, Z	<pre>"integer_value &lt;- 42L"</pre>
Logical	TRUE and FALSE	<pre>"logical_value &lt;- TRUE"</pre>
Complex	Set of complex numbers	<pre>"complex_value &lt;- 1 + 2i"</pre>
Character	"a", "b", "c", ..., "@", "#", "\$", ..., "1", "2", ...etc	<pre>"character_value &lt;- "Hello Geeks"</pre>
raw	as.raw()	<pre>"single_raw &lt;- as.raw(255)"</pre>

## 1.10 Useful Functions in R e.g. rm(), ls()

ls(); List all the objects currently defined in your R workspace.

➤ ls(pattern = "my\_var")

rm(); Removes the specified objects.

➤ rm("var1")

➤ rm(list = ls())

## 1.11 Data Sets in R and save R Session

For R session you may visit this site: [R tutorials, R session, enter, leave, quit](#)

Both the R core installation and contributed R package contain datasets, which are useful example data when learning R. To list all available datasets:

data()

To load a dataset, for example, the dataset women which gives the average heights and weights for 15 American women aged 30-39.

```
data(women)
ls()
ls(pattern = "s")
```

To finish up today, we will save our R session and history.

1. R session One can either save one or more R object in a list to a file using *save()* or save the entire R session using *save.image()*.

```
save(women, file = "women.RData")
save.image(file = "entiresession.RData")
```

2. To load this into R, start a new R session and use the *load()*

```
rm(women)
ls(pattern = "women")
load("women.RData")
ls(patterns = "women")
```

3. R history records the commands history in an R session. To view most recent R commands in a session

```
history()
help(history)
history(100)
savehistory(file = "L2.Rhistory")
```

## 1.12 Installing new R Libraries

Packages in [R Programming language](#) are a set of [R functions](#), compiled code, and sample data. These are stored under a directory called "library" within the R environment. By default, R installs a group of packages during installation. Once we start the R console, only the default packages are available by default. Other packages that are already installed need to be loaded explicitly to be utilized by the R program that's getting to use them. There are several thousand R packages and >500 Bioconductor packages (also called libraries) available. These are not installed by default, so we have to select and install additional packages that will be of use to us. To install packages from cran use the following command

```
install.packages("packageName") # e.g. packageName = ggplot2
```

To load a package into the session use the following command

```
library(packageName) # e.g. packageName = ggplot2
```

Update packages

```
install.packages("lme4")
update.packages("lme4")
library(help = lme4)
```

Or

```
library(lme4) # Load a package
## Or the alternative
require(lme4)
sessionInfo() # List all packages loaded in the current R session
library() # List all installed packages
```

Unload any packages

```
search()
detach(package:lme4)
search()
```



## Chapter 2 Objects in R

Everything (variable, functions etc) in R is an object. Every object has a class e.g., I am an object of the class human. R creates and manipulates *objects*: variables, matrices, strings, functions, etc . *Objects* are stored by name during an R session.

During a R session, you may create many objects, if you wish to list the objects you have created in the current session use the command

```
objects()
ls()
# To Remove objects
rm(x, y, z) # where x, y, and z are objects
```

### 2.1 Types of R Objects

Objects can be thought of as a container which holds data or a function. The most basic form of data is a single element, such as a single numeric or a character string. However, one can't do statistics on a single number! Therefore, there are many other objects in R.

1. A *vector* is an ordered collection of numerical, character, complex or logical objects. Vectors are collection of atomic (same data type) components or models. For example

```
Vec1 = 1:10
Vec1
Vec2 = LETTERS[1:10]
Vec2
Vec3 = vec2 == "D"
Vec3
```

2. A *matrix* is a multidimensional collection of data entries of the same type. Matrices have two dimensions. It has rownames and colnames.

```
mat1 = matrix(vec1, ncol = 2, nrow = 5)
parint(mat1)
dim(mat1)
colnames(mat1) = c("A", "B")
rownames(mat1) = paste("N", 1:5, sep = " ")
print(mat1)
```

3. A *list* is an ordered collection of objects that can be of different modes (e.g. numeric vector, array, etc).

```
a = 20
newList1 = list(a, vec1, mat1)
print(newList1)
newList1 = list(a = a, myVec = vec1, mat = mat1)
print(newList1)
```

4. Though a *data.frame* is a restricted list with class data.frame, it may be regarded as a matrix with columns that can be of different modes. It is displayed in matrix form, row by columns. (Its like an excel spreadsheet)

```
df1 = as.data.frame(mat1)
df1
```

5. A *factor* is a vector of categorical variables, it can be ordered or unordered

```
charVec = rep(LETTERS[1:3], 10)
print(charVec)
table(charVec)
fac1 = factor(charVec)
print(fac1)
```

```
attributes(fac1)
levels(fac1)
class(fac1)
```

## 2.2 Attributes of R Objects

### 2.2.1 Basic Attributes

The most basic and fundamental properties of every objects is its mode and length. These are intrinsic attributes of every object. Example of mode are “logical”, “numeric”, “character”, “list”, “expression”, “name/symbol” and “function”

```
x = 3
mode(x)
x = "apple"
mode(x)
x = 3.1416
x + 2
x == 2
x = x == 2
mode(x)
```

All R objects have mode

```
# vectors of different modes Numeric
x = 1:10
mode(x)
class(x)
x = matrix(rnorm(50), nrow = 5, ncol = 10, byrow = TRUE) # ?matrix()
mode(x)
x = LETTERS[1:5]
mode(x)
length(x)
class(x)
```

### 2.2.2 Other Attributes, dimension

```
x = matrix(5:14, nrow = 2, ncol = 5)
x
attributes(x)
```

In summary

Object	Modes	Allow >1 Modes*
vector	numeric, character, complex or logical	No
matrix	numeric, character, complex or logical	No
list	numeric, character, complex, logical, function, expression, ...	Yes
data frame	numeric, character, complex or logical	Yes
factor	numeric or character	No
array	numeric, character, complex or logical	No

\*Whether object allows elements of different modes. For example all elements in a vector or array have to be of the same mode. Whereas a list can contain any type of object including a list.

## 2.3 Creating and Accessing Objects