Practical No. 1 Date: / / 201

Aim: Introduction to R programming language.

Objectives:

- To understand R & its features.
- To install & use R.
- To study R basics.

Theory:

What is R?

R is a popular programming language for data science and Statistical Computing. It is used by professionals and data experts to map marketing trends, develop financial and climate models, and much more.

R is used in many fields and industries for both small and big data applications. It is open-source and backed by a huge community that creates new tools and packages every day.

It was named after first names of its authors Ross Ihaka and Robert Gentleman. The project was considered in 1992 in Auckland, New Zealand. Initial version was released in 1995 and stable beta version in 2000.

R Features

- Open source language.
- Relates to other languages.
- Vast Community.
- Supports Extensions.
- Extremely comprehensive.
- Outstanding graphs.
- Advanced statistical language.
- Cross-platform compatible.

R Limitations

- Easy to learn, hard to master.
- Command-line interface daunting at first.
- Poorly written code hard to read/maintain.
- Poorly written code is slow.

R Environment Setup

Windows Setup

You can download the latest Windows installer version of R from

`https://cran.r-project.org/bin/windows/base/`

You can just double click and run the installer accepting the default settings. This will install 32-bit version for 32-bit Windows or both 32-bit & 64-bit version for 64-bit Windows.

After installation you can locate the Rgui.exe in a directory structure R\R-3.x.y\bin\i386\ under Windows Program Files to run R-GUI which is the R console to do R Programming.

Another option is to set path of R. exe and Rscript.exe in Environment Variable as C:\Program Files\R\R-3.x.y\bin\ and do R Programming from command prompt.

Linux Setup

R is available as a binary for many versions of Linux. The instruction to install R in Linux varies from flavor to flavor. As an example in Ubuntu 14.04 LTS you can use following command to install R:

```
$ sudo get-apt install r-base-core
```

This will download and install core functionalities of R but you can extend it by installing other packages as required.

R Execution

Using Interpreter

Once you have R environment setup, then it's easy to start your R command prompt by just typing the following command at your command prompt:

```
$ R
```

This will launch R interpreter and you will get a prompt > where you can start typing your program as follows:

```
myString = "Hello, World!"
print(myString)
```

Here first statement defines a string variable *myString*, where we assign a string *"Hello, World!"* and then next statement *print()* is being used to print the value stored in variable *myString*. The output will be displayed on next line as

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

Using Script File

You can write your programs in script files and then execute those scripts at your command prompt with the help of R interpreter called *Rscript*. For Example, write following code in a text file called test.R:

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

Save the above code in a file test.R and execute it at Linux command prompt as given below. Even if you are using Windows or other system, syntax will remain same.

```
$ Rscript test.R
```

When we run the above program, it produces the following result.

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

R Comments

Comments are like helping text in your R program and they are ignored by the interpreter while executing your actual program.

Single comment is written using # in the beginning of the statement as follows:

```
# My first program in R Programming
```

R Input Handling

To read input from user R provides readline() and readLines() methods. Let us see readline() with the help of an example,

```
> no = readline(prompt = "Enter Number : ")
> print(no)
```

The output will look like

```
Enter Number: 10
[1] 10
```

Note that, readline() should be used in interpreter mode, in script file we prefer readLines() method.

In script file say testinp.R, readLines() can be used as follows

```
cat("Enter Number : ")
no = readLines("stdin",1)
cat(no)
```

After executing testing.R, the output will look like

```
Enter Number: 10
[1] 10
```

Note that, in readLines() method, first argument represents connection object (default is "stdin") and second argument represents maximal number of lines to read.

R Datatypes

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

The simplest of these objects is the vector object and there are six data types of these atomic vectors, also termed as six classes of vectors. The other R-Objects are built upon the atomic vectors.

Data Type	Example
Logical	TRUE, FALSE
Numeric	5, 6.7
Integer	0L, 2L, 34L
Complex	3+2i
Character	'H', "TRUE", "123"
Raw	"Hello" is stored as 48 65 6c 6c 6f

R Variables

A variable provides us with named storage that our programs can manipulate. A variable in R can store an atomic vector, group of atomic vectors or a combination of many R-objects. Following are rules for R variable:

- A valid variable name consists of letters, numbers and the dot(.) or underline(_) characters (no other special character is allowed).
- The variable name must starts with a letter or the dot not with a number or underscore.
- The variable name can start with a dot(.) but the dot(.)should not be followed by a number (For example, .2var_name).

The variables can be assigned with values using leftward, rightward and equal to operator. The values of the variables can be printed using print() or cat()function. The cat() function combines multiple items into a continuous print output.

```
# Assignment using equal operator.
var.1 = c(0,1,2,3)
# Assignment using leftward operator.
var.2 <- c("learn","R")
# Assignment using rightward operator.
c(TRUE,1) -> var.3
print(var.1)
cat ("var.1 is ", var.1 ,"\n")
cat ("var.2 is ", var.2 ,"\n")
cat ("var.3 is ", var.3 ,"\n")
```

When we execute the above code, it produces the following result:

```
[1] 0 1 2 3
var.1 is 0 1 2 3
var.2 is learn R
var.3 is 1 1
```

The vector c(TRUE,1) has a mix of logical and numeric class. So logical class is coerced to numeric class making TRUE as 1.