Tutorial for Assignment-2

R is a programming language and software environment for statistical analysis, graphics representation and reporting. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems like Linux, Windows and Mac. R is free software distributed under a GNU-style copy left, and an official part of the GNU project called GNU S.

Evolution of R

R was initially written by Ross Ihaka and Robert Gentleman at the Department of Statistics of the University of Auckland in Auckland, New Zealand. R made its first appearance in 1993. □ A large group of individuals has contributed to R by sending code and bug reports. □ □ Since mid-1997 there has been a core group (the "R Core Team") who can modify the R source code archive.

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.

R – Environment Setup

Windows Installation

You can download the Windows installer version of R from R-3.2.2 for Windows (32/64 bit) and save it in a local directory. As it is a Windows installer (.exe) with a name "R-version-win.exe". You can just double click and run the installer accepting the default settings. If your

Windows is 32-bit version, it installs the 32-bit version. But if your windows is 64-bit, then it installs both the 32-bit and 64-bit versions. After installation you can locate the icon to run the Program in a directory structure "R\R3.2.2\bin\i386\Rgui.exe" under the Windows Program Files. Clicking this icon brings up the R-GUI which is the R console to do R Programming.

Linux Installation

R is available as a binary for many versions of Linux at the location R Binaries. The instruction to install Linux varies from flavor to flavor. These steps are mentioned under each type of Linux version in the mentioned link. However, if you are in a hurry, then you can use yum command to install R as follows: $ yum install R Above command will install core functionality of R programming along with standard packages, still you need additional package, then you can launch R prompt as follows: $ R R version 3.2.0 (2015-04-16) -- "Full of Ingredients" Copyright (C) 2015 The R Foundation for Statistical Computing Platform: x86_64-redhat-linux-gnu (64-bit) R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under certain conditions. Type 'license()' or 'licence()' for distribution details. R is a collaborative project with many contributors. Type 'contributors()' for more information and 'citation()' on how to cite R or R packages in publications. Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help. Type 'q()' to quit R. > Now you can use install command at R prompt to install the required package. For example, the following command will install plotrix package which is required for 3D charts. > install("plotrix")

R – Basic Syntax

R Command Prompt 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) [1] "Hello, World!" 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. R Script File Usually, you will do your programming by writing your programs in script files and then you execute those scripts at your command prompt with the help of R interpreter called Rscript. So let's start with writing following code in a text file called test.R as under: # 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.

R – Data Types

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.

Q1. (i) The vector is the essential building block for handling multiple items in R. In a numeric sense, you can think of a vector as a collection of observations or measurements concerning a single variable, for example, the heights of 50 people or the number of coffees you drink daily. More complicated data structures may consist of several vectors. The function for creating a vector is the single letter c, with the desired entries in parentheses separated by commas. Vectors are the most basic R data objects and there are six types of atomic vectors. They are logical, integer, double, complex, character and raw.

Table 1: Vectors in R

Index	1	2	3	4
Values	20	50	10	100

>Creating a Vector

```
R> myvec <- c(1,3,1,42)
R> myvec
[1] 1 3 1 42
```

>Finding vector with length

```
R> length(x=c(3,2,8,1))
 [1] 4
 R> length(x=5:13)
 [1] 9
```

For example:

(a) Numeric Vector : c(4, 5, 6, 7)

(b) **Logical Vectors : c(TRUE, FALSE, TRUE)**
(c) **Character Vector : c("x", "yz","qwerty")**

(ii) Lists are the R objects which contain elements of different types like − numbers, strings, vectors and another list inside it. List is created using **list()** function.

For Example:

List containing numbers : list1 <- list(1,2,3)
List containing strings : list2 <- list("Sunday","Monday,"Tuesday")

(iii) The matrix is an important mathematical construct, and it's essential to many statistical methods. You typically describe a matrix A as an m × n matrix; that is, A will have exactly m rows and n columns. This means A will have a total of mn entries, with each entry ai, j having a unique position given by its specific row (i = 1, 2, . . ., m) and column (j = 1, 2, . . ., n). Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout. They contain elements of the same atomic types. A Matrix is created using the **matrix()** function.

Matrix Syntax : ***matrix(data, nrow, ncol, byrow, dimnames)***

A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

(iv) A function is a set of statements organized together to perform a specific task. 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
}
```

For example:

(a) Create a sequence of number from 1 to 10 : print(seq(1,10))

A function call is a request made by a program or script that performs a predetermined function.

For Example:

(a) pow(8, 2)

(v) A hash table, or associative array, is a well known key-value data structure. In R there is no equivalent, but you do have some options. You can use a vector of any type, a list, or an environment.

(vi) R consists of a number of data objects to perform various functions. There are 6 types of objects in R Programming. They include vector, list, matrix, array, factor, and data frame.

- **Vectors** are one of the basic R programming data objects. They are six types of atomic vectors- logical, integer, character, raw, double, and complex.
- **Lists** are data objects of R that contain various types of elements including strings, numbers, vectors, and a nested list inside it. It can also consist of matrices or functions as elements. It can be created with the help of the list() function.
- **Matrices** in R Programming are used to arrange elements in the two-dimensional layout. They contain elements of the same data type. They usually contain numeric values in order to perform mathematical operations.
- An **array** is used to store data in more than just 2 dimensions. It is used to store multi-dimensional data in the required format. It can be created with the help of an array() function.
- **Factors** are data objects that are used in order to categorize and store data as levels. They can be strings or integers. They are extremely useful in data analytics for statistical modeling. They can be created using factor() function.
- **Dataframe** is a 2-dimensional data structure wherein each column consists of the value of one variable and each row consists of a value set from each column.

Q2. (i)

Example 1: Write R program to add two vectors of integers type.

```
x = c(50, 40)
y = c(30, 10)
print("Original Vectors:")
print(x)
print(y)
print("After adding two Vectors:")
z = x + y
print(z)
```

Example 2: Write a R program to sort a Vector in ascending and descending order.

```
x = c(10, 20, 30, 25, 9, 26)
print("Original Vectors:")
print(x)
print("Sort in ascending order:")
print(sort(x))
print("Sort in descending order:")
```

```
print(sort(x, decreasing=TRUE))
```

(ii)

Example 1: Write a R program to create a list containing strings, numbers, vectors and a logical values.

```
list_data = list("Java", "C", c(5, 7, 9, 11), TRUE, 125.17, 75.83)
print("Data of the list:")
print(list_data)
```

Example 2: Write a R program to list containing a vector, a matrix and a list and give names to the elements in the list.

```
list_data <- list(c("Blue","White","Black"), matrix(c(1,3,5,7,9,11), nrow = 2),
list("Python", "C", "Java"))
print("List:")
print(list_data)
names(list_data) = c("Color", "Odd numbers", "Language(s)")
print("List with column names:")
print(list_data)
```

(iii) Matrix

Example 1: Write a R program to create a blank matrix.

```
m = matrix(, nrow = 10, ncol = 5)
print("Empty matrix of 10 rows and 5 columns:")
print(m)
```

Example 2: Write a R program to rotate a given matrix 90 degree clockwise rotation.

```
x =  matrix(1:9, 3)
print("Original matrix:")
print(x)
rotate = t(apply(x, 2, rev))
print("Rotate the said matrix 90 degree clockwise:")
print(rotate)
```

Data Frame

Example 1: Write a R program to create an empty data frame.

```
df = data.frame(Ints=integer(),
                Doubles=double(),
                Characters=character(),
                Logicals=logical(),
                Factors=factor(),
                stringsAsFactors=FALSE)
print("Structure of the empty dataframe:")
print(str(df))
```

Example2: Write a R program to get the structure of a given data frame.

```
exam_data = data.frame(
name = c('Ram', 'Sita', 'Ravana', 'James'),
score = c(13.5, 8, 16.5, 11),
attempts = c(1, 3, 2, 3),
qualify = c('yes', 'no', 'yes', 'no')
)
print("Original dataframe:")
print(exam_data)
print("Structure of the said data frame:")
print(str(exam_data))
```

(iv) Function

Example 1: Create a function to print squares of numbers in sequence.

```
new.function <- function(a)
{
for(i in 1:a)
        {
                b <- i^2
                print(b)
        }
}
```

Example 2: Write an R function, number.sequence(n), that returns a vector that has the form $\{2^2 \cdot 3, 3^2 \cdot 5, 4^2 \cdot 7, 5^2 \cdot 9, \ldots\}$ and has exactly n terms, as the sample output below suggests.

```
> number.sequence(5)
```

```
[1]  12  45 112 225 396
```

Function call

Example 1: 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 5 as an argument.
new.function(5)
```

Example 2: Create a function without an argument.

```
new.function <- function()
{
        for(i in 1:4) {
        b <- i^2
        }
}
# call the function new.function without supplying an argument.
new.function()
```

(v) Example 1: Create a hash table or Associative array using different R data structures.

```
library(plyr)
library(ggplot2)
unique_strings <- function(n){
string_i <- 1
string_len <- 1
ans<- character(n)
chars <- c(letters,LETTERS)
new_strings <- function(len,pfx){
for(i in 1:length(chars)){
if (len == 1){
ans[string_i] < <- paste(pfx,chars[i],sep='')
string_i < <- string_i + 1
} else {
new_strings(len-1,pfx=paste(pfx,chars[i],sep=''))
}
if (string_i > n) return ()
}
```

```
}
while(string_i <= n){
new_strings(string_len,'')
string_len <- string_len + 1
}
sample(ans)
}
```

(vi) Write proper R codes for all the data objects.

NB: For all the constructs write clear definition, explain specifically with its characteristics and mention the types (if applicable) along with proper R example.

Reference

https://cran.r-project.org/doc/manuals/r-release/R-lang.html#Function-calls

****