UNIT - I

INTRODUCTION TO ANAIYTICS AND R PROGRAMMING

INTRODUCTION TO R

What is R?

R is a flexible and powerful open-source implementation of the language s (for statistics) developed by John chambers and others at Bell labs.

Why R?

Five Reasons to learn and use R:

→ R is open source and completely free. R Community members regularly contribute Packages to increase R's functionality.

→ R is as good as commercially available statistical like spss, SAS, and Minitab.

→ R has extensive statistical and graphing capabilities. R provides hundreds of built-in statistical functions as well as its own built-in programming language.

→ R is used in teaching and performing computational - statistics. It is the language of choice for many academics who teach computational statistics.

R uses:

→ R combines aspects of functional and object-oriented programming.

→ R can use in interactive mode.

→ It is an interpreted language rather than a compiled one.

→ Finding and fixing mistakes is typically much easier in R than in many other languages.

R Features:

→ programming language for graphics and statistical computations.

→ Available freely under the GNU public license.

→ Used in data mining and statistical analysis.

→ Included time series analysis, linear and nonlinear modeling among others.

→ Very active community and package contributions.

→ Very little programming language knowledge necessary.

First R program : using R as Calculator:

R commands can run in two ways:

1) Type at console and press enter to see the output. output will get at console only in R Studio.
2) Open new R script file and write the command, keep the curser on the same line and press Ctrl + enter or click on Run. Then see the output at console along with command.

At console:

R as a calculator, typing commands directly into the R Console. Launch R and type the following code, pressing <Enter> after each command.

Type an expression on console.

R allow 3 assignment operations are:
<- or = for assignment and == to test equality.

Example:

```
> 2*2 ## Multiplication
  [1] 4
```

```
> 2/2   ## Division
[1] 1
> 2+2   ## addition
[1] 4
> 2-2   ## Subtraction
[1] 0
> 2^2   ## Exponentiation
[1] 4
> q()   ## to quit
> y <- 3* Exp(x)
> x <- 3* exp(x)
```

R Expression :

At >prompt type the R expression and press enter.

R output :

R labels each output value with a number in Square brackets. As far as R is concerned, an individual number is a one element vector. The [1] is simply the index of the first element of the vector.

Variable (object) Names:

Certain variable names are reserved for particular purposes. Some reserved

Symbols are : c q t c D F I T

meaning of c q t c D F I T

? ## to see help document.

?c ## c means combine values into a vector or list.

?q ## q means Terminate an R session.

?t ## t means matrix Transpose.

?c ## c means Sets contrast for a factor.

?D ## D means Symbolic and Algorithmic Derivatives of Simple Expressions

?F ## f means & logical vector character strings.

c("T", "TRUE", "True", "true") are regarded as true, c("F", "FALSE", "false", "false") as false, and all others as NA

>F ## [1] FALSE

?I ## Inhibit Interpretation / conversion of objects.

Working on variables :

Operators in R :

Arthimetic Operators.

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
^ or **	Exponentiation
x %% y	modulus (x mod y) 5 %% 2 is 1
x %/% y	integer division 5 %/% 2 is 2

Logical operators

Operator	Description
<	lessthan
<=	less than or equal to
>	greater than
>=	greater than or equal to
!=	not equal to
==	Exactly equal to
!x	Not x

x/y	x or y
x & y	x AND y
is TRUE (x)	test if x is TRUE

Getting Help in R:

> help (function), or use the

>? keyword checks in all packages --- function shortcut.

>?? Keyword checks in content of all packages either of above all will open the R documentation.

Comment Notation in R:

is used to comment a line in R Script

List of objects:

To see a listing of the objects in your workspace, you can use the ls() function.

To get more detail, use ls.str() > ls()

[1] "A" "acctdata" "address" "B" "b1"

[6] "balance" "c" "careersat" "chisquare" "colnames".

INTRODUCTION TO VARIOUS DATA TYPES:

In Analytics the data is classified as Quantitative (numeric) and Qualitative (character/ factor) on very broad level.

→ Numeric Data :- It includes 0~9, "." and "_ve" sign.

→ Character Data :- Everything except Numeric data type is character.

Ex:- Names, Gender etc.

"1, 2, 3...." are Quantitative Data while "Good", "Bad" etc. are Qualitative Data. We can convert Qualitative Data into Quantitative Data using ordinal values.

List of Data types in R

Data Type		Verify.
Logical	True, False	V <- TRUE Print (class (v)) [1] "logical"
Numeric	12.3, 5, 999	V <- 23.5 print (class(v)) [1] "Numeric"
Integer	2L, 34L, 0L	V <- 2L Print (class (v)) [1] "integer"

Complex	3+2i	v<- 2+5i Print (class(v)) [1] "complex"
character	"a", "good", "TRUE", 23.4	v <- "TRUE" print (class(v)) [1] "character"
Raw	"Hello" is stored as 48 65 6c 6c 6f	v <- charToRaw ("Hello") print (class(v)) [1] "raw"

Matrices:

A matrix is a two-dimensional rectangular data set. Matrices must be heterogeneous ie, the type of data in a given vector of all not in the same class.

Matrix can be created in three ways:

→ Matrix(): A vector input to the matrix function

→ using rbind() and cbind() functions

→ using dim() to the existing vector.

Creating a matrix using matrix():

```
# Create a matrix.
M = Matrix (c('a','a','b','c','b','a'), nrow=2, ncol=3,
           byrow = TRUE)
Print (M)
  [,1] [,2] [,3]
  [1,] "a"  "a"  "b"
  [2,] "c"  "b"  "a"
```

Creating a matrix using rbind() or cbind():

First create two vectors and then Create a matrix using rbind(). It binds the two vectors data into two rows of matrix.

Ex: To create a matrix having the data as:

6, 2, 10 & 1, 3, -2

Step 1: Create two vectors as xr1, xr2

```
> xr1 <- c (6,2,10)
> xr2 <- c (1,3,-2)
> x <- rbind (xr1, xr2) ## binds the
       vectors into rows of a matrix
                                (2x3)
> x
  [,1] [,2] [,3]
xr1  6    2    10
xr2  1    3    -2
```

Array:

Array 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 creating an array:

```
> my.array <- array (1:24, dim = c (3,4,2))
```

In the above example, "my.array" is the name of the array we have given. There are 24 units in this array mentioned as "1:24" and are divided in three dimensions "(3,4,2)".

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. It displays data along with header information.

To retrieve data in a particular cell:

Enter its row and column coordinates in the single square bracket "[]" operator.

Example:-

To retrieve the cell value from the first row, second column of mtcars.

```
> mtcars [1,2]
```

mt cars [row, column]

```
# Create the data frame.
> BMI <- data.frame (gender = c("male", "male"
  "female"), height = c(152, 171.5, 165) weight =
  c(81,93,78), Age = c(42, 38,26))
> print (BMI)
```

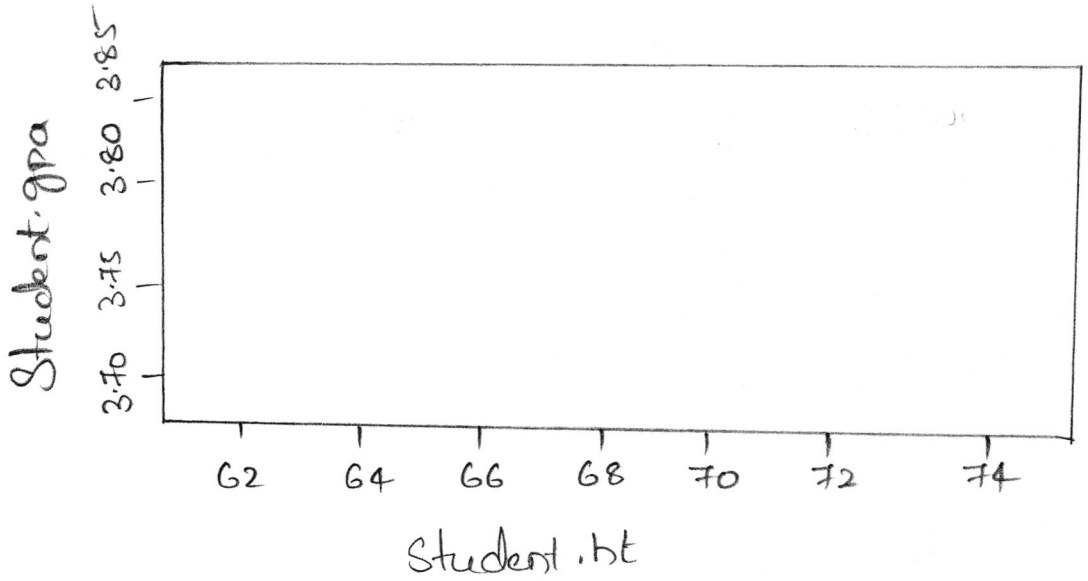
Gender	Height	Weight	Age
Male	152.0	81	42
Male	171.5	93	38
Female	165.0	78	26

Data 1	Height GPA
66	3.80
62	3.78
63	3.88
70	3.72
74	3.69

```
> student.ht <- c(66, 62, 63, 70, 74)
> student.gpa <- c(3.80, 3.78, 3.88, 3.72, 3.69)
> student.data1 <- data.frame(student, ht, student.gpa)
> student.data
```

Student.ht	Student.gpa
66	3.80
62	3.78
63	3.88
70	3.72
74	3.69

```
> plot(student.ht, student.gpa)
```



DATE:

Date () function is used to access the data and time in R

Sys.Date () : The current System date. This function returns a Data object.

class (Data)

A string in this format is treated as a character unless cast to a Data type.

```
> class ("2010-06-16")
> class (as.Date ("2010-06-16"))
```

You can also pass in dates in other formats and cast them as strings by specifying the format in use.

```
> as.Date ('02/03/2004', '%m/%d/%y')
```

To format data information in a wide variety of string formats, use the Strftime function.

READING DATASETS USING R :

We can import Dataset from various sources having various file types :

Example :-

→ .csv or . txt format
→ Big data tool - Impala
→ CSV File.

The Sample data can also be in Comma Separated values (CSV) format. Each cell inside such data file is Separated by a Special character, which usually is a comma, although other characters can be used as well. The first row of the data file should contain the column names instead of the actual data. Here is a Sample of the Expected format.

```
col1, col2, col3
100, a1, b1
200, a2, b2
300, a3, b3
```

After we copy and paste the data above in a file named "mydata.csv" with a text editor, we can read the data with the function read.csv.

In R data can read in two ways either from local disc or web.

From disc:

The data file location is known on local disc use: