R-PROGRAMMING - 1

Introduction: Use of the statistical software's is must for handling large data sets and complex procedures of analysis. Several statistical packages are available such as:

- SPSS
- SAS
- S-PLUS
- R
- MINITAB. . . .

Why R?

- R is open source and runs on UNIX, Windows and Macintosh.
- Programming Language.
- Very good computing performance.
- Excellent built-in-help system.
- Excellent graphing capabilities.
- Easy to extend with user written function.
- Can be modified by users and its development is open to contributors.
- Scripting and interfacing facilities.

Installing R:

The latest version of R for windows is R.3.1.3.

To install R on your PC, first go to the site called CRAN (Comprehensive R Archive Network) (or) you can type its full address http://cran.r-project.org/, once there you need to 'Download and Install R' by running the appropriate pre-compiled binary distributions. Click to choose between Linux, Mac OS and Windows, and then follow the instructions. You want the 'base' package and you want to run the setup program which will have a name like R*.exe (on a PC). When asked say you want to 'Run'.

Working with R:

When R is installed properly, we see R icon on our desktop, click on the R icon to start R. The data analysis in R proceeds as an interactive dialogue with the interpreter. As soon as we type command at the prompt (>), and press the enter key, the interpreter responds by executing the command. To end the session type q().

Saving, Storing and Retrieving work:

Workplace in R: when you quit the session, just say "yes" to "save work space image." When you restart R all the data and variables from previous session will be available. To retrieve the R command you have used previously, just use up arrow key.

In WORD file:

1. Create a file in WORD to save commands say, "comword" and create a separate file for output in WORD, say result.out.
2. Copy R-commands from R-console and paste these in comword. Once you paste delete the prompt (<) from all the commands. Make sure that each command ends with semicolon. The commands saved in this way can be easily reused to continue the incomplete session or to carry out similar analysis with appropriate changes by using copy and paste facility.
3. Copy, the outputs generated and paste it in result.out. To copy the graph generated in graphical window right click on the graph, select the option "save as bitmap" and then paste it in result.out. The tables generated in R usually need formatting hence it is better to paste the table in excel first and then paste the formatted tables in WORD.

R-preliminaries:

Data Types:

The usual data types available in R are known as modes. The modes are:

1. Logical (Boolean True / False)
2. Numeric (Integers and Real)
3. Complex (Real + Imaginary)

Arithmetic operators:

R includes the usual arithmetic operations with usual hierarchy:

Operator	Function
+	Addition
−	Subtraction
*	Multiplication
/	Division
^	Exponentiation

Note:

1. The hierarchy of the operators is maintained as usual.
2. Spaces are not required to separate the elements of arithmetic operators.

Assignment operators:

R uses the assignment operator < − (assigns from right to left) to give a data object (or any other object) its value. The operator − > (assigns from left to right) may also be used.

Example:

- x < −2; this command assigns the value 2 to the object x.
- x^2− > y; this command assigns the value x^2 to the object y.

Note:

1. If the command is not complete we see "+" at the end of the line.
2. Commands are separated by a semicolon (;) or a new line.
3. Comments will be lines starting with a hash mark (#). i.e., text to the right of # will be ignored by interpreter.

Comparison operators:

Following are the common comparison operators:

Operator	Comparison
>	is greater than
<	is less than
>=	is greater or equal
<=	is less or equal

==	is equal to
! =	is not equal to
!	logical not
&	logical and
|	logical or

Standard functions:

Following are the standard functions that are found on most calculators that are available in R:

Name	Operation
sqrt()	square root
abs()	absolute value
sin(),cos(),tan()	trig functions (radians)
pi	the number Π = 3.1415926 . . .
exp(),log()	exponential and logarithm
gamma()	Euler's gamma function
factorial()	factorial function
choose	combination

Functions:

Many mathematical and statistical functions are available in R. A function has a name typed, and then followed by a pair of parentheses. Arguments are added inside this pair of parentheses as needed.

Note: Some function can be used as a method of data input in R.

1. *assign function*:

   Example: assign("x", 3). Here assign is the name of the function. The first argument ("x") is the name of the variable to which the assignment is to be made. The second argument is the value to be assigned.

2. *combine function (or) c function*: The most useful R command for quickly entering small data sets is the c function. This function combines terms together.

   Example y < −c(1, 2, 3, 9, 15, 17); c function constructs a vector. Even though we have to use the c function to

construct a vector, once we have assigned the vector, we may reassign its value to other data objects.

The c function can also be used to construct a vector of character strings. Example: names < –c("Rama", "Lakshmi", "Gowri");

3. ***sequence operator and seq function:*** The sequence operator (:) generates consecutive numbers while the sequence function does same thing but more flexibly.

   Example:

   - 1:4 command gives the output as [1] 1 2 3 4
   - 4:1 command gives the output as [1] 4 3 2 1
   - -1:2 command gives the output as [1] -1 0 1 2

   Note:

   a. The sequence operator has high priority within an expression.

      Example:

      - 2 * 1 : 4 is a vector, where the output is [1] 2 4 6 8
      - seq(2,8,by=2); specifies an interval and increment. where the output would be [1] 2 4 6 8
      - seq(0,1,length=3); specifies interval and the number of elements. Where the output would be [1] 0.0 0.5 1.0

   b. Parameters to seq() and many other R functions can also be given in named form, in which case the order on which they appear is irrelevant. The first two parameters of sequence function are named as from and to.

      Example:

      - seq(from = 1, to=30); (or) seq(to=30, from=1); (or) 1:30 will give the same output.
      - seq(length=15, to=30, from=0); (or) seq(to=30, length=15, from=0); are equivalent.

4. ***scan*** function: We explain this function with an example. Suppose the body weights (in grams) of 12 rats used in a study of vitamin deficiencies are given: 103, 125, 112, 153, 124, 106, 141, 117, 121, 130, 95. To create a data object wt using scan function we

type the command: wt=scan() Following is the response to their command

1:

Now type the data separated by spaces, when all the 12 values are input, press the enter key twice. The output will look like 1: 103 125 112 153 124 106 141 117 121 115 130 95

13:

Read 12 items

5. rep function: In order to enter the data containing repeated values, rep function is useful.

   Example:

   - Enter the data set in R, 1 1 1 2 2 3 3 3 7 9 9 The R-command is: x<-c(rep(1,3), rep(2,2), rep(3,3), 7, rep(9,2)); The first argument to rep() function is the number to be repeated and the second argument is the number of times it is to be repeated.
   - Suppose we have a vector x and we want to repeat each of its elements equal number of times say 4 then the R command is rep(x, each=4);

6. ***data.frame*** function: A data frame corresponds to what other statistical packages call a "data matrix" or a "data set". It is a list of vectors or factors of the same length, which are related "across", such that data in the same position comes from the same experimental unit. It is possible to create a data frame with the data.frame function. The vectors so included in the data frame must be of same length, or if one of them is shorter, it is "recycled" an appropriate number of times. Example:

   - Suppose we have four vectors x = (1, 2, 3, 4), y = (2, 3, 4), z = (10, 15) and w = (10) we create the following data frames (x, z), (x, w) and (x, y)

     R commands: x<-1:4; y<-2:4; z<-c(10,15); w<-10;

     i. d<-data.frame(x,z);

        output:

```
   x  Z
1  1  10
2  2  15
3  3  10
4  4  15
```

The first column of the output represents row (individual or subject) number. x and z are labeled columns corresponding to vectors x and z. Length (number of components) of vectors z = 2. Length of vector x = 4 = 2 × 2. Hence z is repeated 2 times to create the data.frame(x, z).

ii. d1<-data.frame(x,w);

output:

```
   x  W
1  1  10
2  2  10
3  3  10
4  4  10
```

The first column of the output represents row (individual or subject) number x and w. Length (number of components) of vector w = 1, length of vector x = 4 = 4 × 1. Hence w is repeated 4 times to create the data.frame(x, w).

iii. d2<-data.frame(x,y);

output: Error in data.frame(x,y); arguments imply differing number of rows:4,3. Observe that longer vector is not a multiple of the shorter. Hence we get the error message.

- Example: Following table shows the age distribution of cases of certain disease reported during a year in a community. We enter the data using a data.frame function.

Sl.No	mid age	no.of cases

1	10	5
2	20	10
3	30	120
4	40	22
5	50	13
6	60	5

R command using data.frame function:

```
midx<-seq(10,60,by=10);
fr<-c(5,10,120,22,13,5);
fr.dist<-data.frame(midx,fr);
fr.dist;
```

output:

```
  midx  fr
1   10   5
2   20  10
3   30 120
4   40  22
5   50  13
6   60   5
```

Note: Suppose we want to change the names of the columns or rows, this can be done by using colnames function and rownames function.

```
colnames(fr.dist)<-c(''midage'',''no.of  cases'')
fr.dist;
```

output:

```
  midage  no.of cases
1     10            5
2     20           10
3     30          120
4     40           22
5     50           13
6     60            5
```

7. ***matrix function***: This is another function used to create a data frame. Consider the example that was used in data.frame function,

The data can be input either by row wise or column wise using a matrix function. fr.dist<-matrix(c(seq(10,60,by=10), 5,10,120,22,13,5),nrow=6,byrow=F); fr.dist

(OR)

fr.dist<-
matrix(c(10,5,20,10,30,120,40,22,50,13,60,5),nrow=6,byrow=T);
fr.dist;

The columns of data set are combined by c function. This is first argument to matrix function. The second argument gives the number of rows. Either we have to give the number of rows or number of columns (ncol=2). The third argument is byrow=F (Some versions accept it as default)

output:

	[,1]	[,2]
[1,]	10	5
[2,]	20	10
[3,]	30	120
[4,]	40	22
[5,]	50	13
[6,]	60	5

To change the row and column names:

rownames(fr.dist)<-c("1","2","3","4","5","6");
colnames(fr.dist)<-c("midage","no.of cases")
fr.dist;

output:

	midage	no.of cases
1	10	5
2	20	10
3	30	120
4	40	22
5	50	13
6	60	5

Matrix operations can easily be performed in R using a few simple function like:

Name	Operation
dim()	dimension of the matrix (number of rows and columns.
as.matrix()	used to coerce an argument into a matrix object.
%*%	matrix multiplication.
t()	matrix transpose.
det()	determinant of a square matrix.
solve()	matrix inverse; also solves a system of linear equations.
eigen()	computes eigen values and eigen vectors.

8. class function: This function is useful in deciding the class of the data object.

Examples:

- a<-c(1,9,6,8);
  class(a);
  # output of this command is "numeric"
- a<-1:4;
  class(a);
  # output of this command is "integer"
- a<-c("x","y");
  class(a);
  # output of this command is "character"
- class(c);
  # output of this command is "function"
- d<-data.frame(x=1:4,y=5:8);
  class(d);
  # output of this command is "data.frame"
- d<-matrix(c(1:4,5:8),ncol=2);
  class(d);
  # output of this command is "matrix"

Importing data from excel: Enter the data in excel with headers (column names) then save file as tab delimited text file with name, say rawdata, on local disk C; and close the file. To import the file in R console, we use read.table function.

R command is:

```
data<-read.table("C:/rawdata.txt",header=T);
```

Resident data sets:

The data sets that come with R or one of the distributed R packages are known as resident data sets or built-in-data sets. We can use data function to access resident data sets in the base package of R.

Example:

```
data(rivers); # read data with data().
rivers; # print out the values of data set.
help(rivers); # brief description of the data rivers.
length(rivers); # how many observations.
data(); # To get list of all data sets in the base packages.
```

Note: There are resident data sets in other packages (libraries), too.

We can see the list of all data sets in all packages by issuing the command: data(package=.package(all.available=TRUE));

Some useful built-in functions

Consider x<-c(4,5,2,13);

1. length function: gives the number of elements in the vector.
   length(x); output is [1] 4
2. max function: gives the maximum element of data.
   max(x); output is [1] 13
3. min function: gives the minimum element of data.
   min(x); output is [1] 2
4. range function: gives the range of data.
   range(x); output is [1] 2 13
5. sum function: gives the sum of the values.
   sum(x); output is [1] 24

6. cumsum function: gives the cumulative sum of the values.
   cumsum(x); output is [1] 4 9 11 24
7. mean function: gives the mean of data.
   mean(x); output is [1] 6
8. median function: gives the median of data.
   median(x); output is [1] 4.5
9. var function: gives the variance of data.
   var(x); output is [1] 23.3333
10. sort function: This function sorts a vector in increasing order of magnitude.
   sort(x); output is [1] 2 4 5 13
   Note: To sort the variable values in decreasing order of magnitude R command is sort(x, decreasing=T); output is [1] 13 5 4 2
11. diff function: This function creates a vector of differences by subtracting previous element from next element.
   diff(x); output is [1] 1 -3 11
   Note: To get the list of functions and variables in R, you can use help as follows: click help on menu bar of R console, Select Manuals (pdf)-Introduction to R-Function-variable index.

Graphics with R:

R has a remarkable variety of graphics. There are two kinds of graphical functions: the high level function, which creates a new graph, and low level function, which adds elements to an already existing graph. The graphs are produced with respect to graphical parameters, which are defined by default and can be modified with the par function. Lists of the functions used for standard plots are as below:

Sl.No.	Function	Name of the plot
1	plot()	Scatter plot (and more)
2	hist()	Histogram
3	boxplot()	Box-and-whiskers plot
4	stripchart()	Strip chart
5	barplot()	Bar diagram

6	stem()	Stem-and-leaf display

Arguments to plot() function: The plot function, hist function and boxplot function take extra arguments that control the graphic.

Some arguments to plot() function:

Sl.No.	Argument	Explanation
1	main=	Title
2	xlab=	Label for x-axis
3	ylab=	Label for y-axis
4	xlim=	Specifies x-limits (useful for multiple plots)
5	ylim=	Specifies y-limits (useful for multiple plots)
6	type=	Type of plot: "p" for points, "l" for lines, "o" for lines through points
7	pch=	The style of the points (pch can be a number between 0 and 20)
8	lty=	The style of the lines plotted
9	cex=	Magnification factor

Low level plotting functions:

Sl.No.	Function	Explanation
1	lines()	Lines
2	abline()	Line given by intercept and slope
3	points()	Points
4	text()	Text in plot
5	legend()	List of symbols

Measure of Central Tendency:

Mean in R is computed using the function mean.

Example: For the following data 70, 78, 66, 65, 50, 53, 48, 88, 95, 80, 85, 84, 81, 63, 68, 73, 75, 84, 49, and 77. Compute the mean and median.

The R-codes is as follow:

```
scores <- c(70, 78, 66, 65, 50, 53, 48, 88, 95, 80, 85, 84, 81, 63, 68, 73, 75, 84, 49, 77)
mean(scores)
[1] 71.6
median(scores)
[1] 74
```

MEASURES OF SKEWNESS AND KURTOSIS

Skewness and Kurtosis in R are available in the moments package. To install the moments packages install.packages("moments").

To load the these packages

```
library(moments)
```

Example: Obtain skewness and kurtosis for the following data: 19.09, 19.55, 17.89, 17.73, 25.15, 27.27, 25.24, 21.05, 21.65, 20.92, 22.61, 15.71, 22.04, 22.60, and 24.25.

The R-codes is as follows:

```
library(moments)
x<-c(19.09, 19.55, 17.89, 17.73, 25.15, 27.27, 25.24, 21.05, 21.65, 20.92, 22.61, 15.71, 22.04, 22.60, 24.25)
skewness(x)
[1] -0.01565162
kurtosis(x)
[1] 2.301051
```

Conclusion: Negatively skewed and platykurtic.

Conclusion: In this session we have seen how we can enter the data to R-software, access the data files in R, and compute the measures of central tendency, skewness and kurtosis.