PRACTICAL MANUAL

ON

COMPUTER ORIENTED STATISTICAL TECHNIQUES

S.Y.B.Sc.IT

COMPILED BY: MR. NITESH N. SHUKLA

CHANDRABHAN SHARMA COLLEGE OF ARTS, COMMERCE & SCIENCE

INDEX

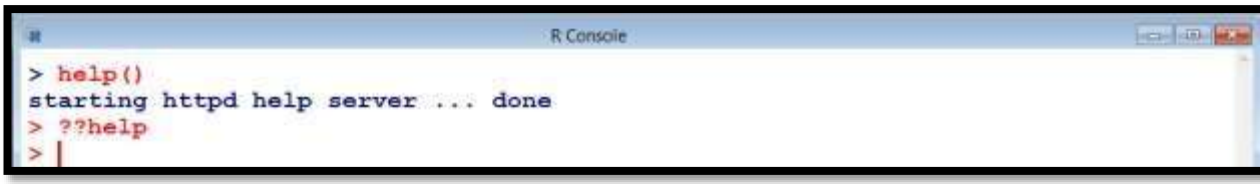
PRACTICAL No.-1

Aim:-

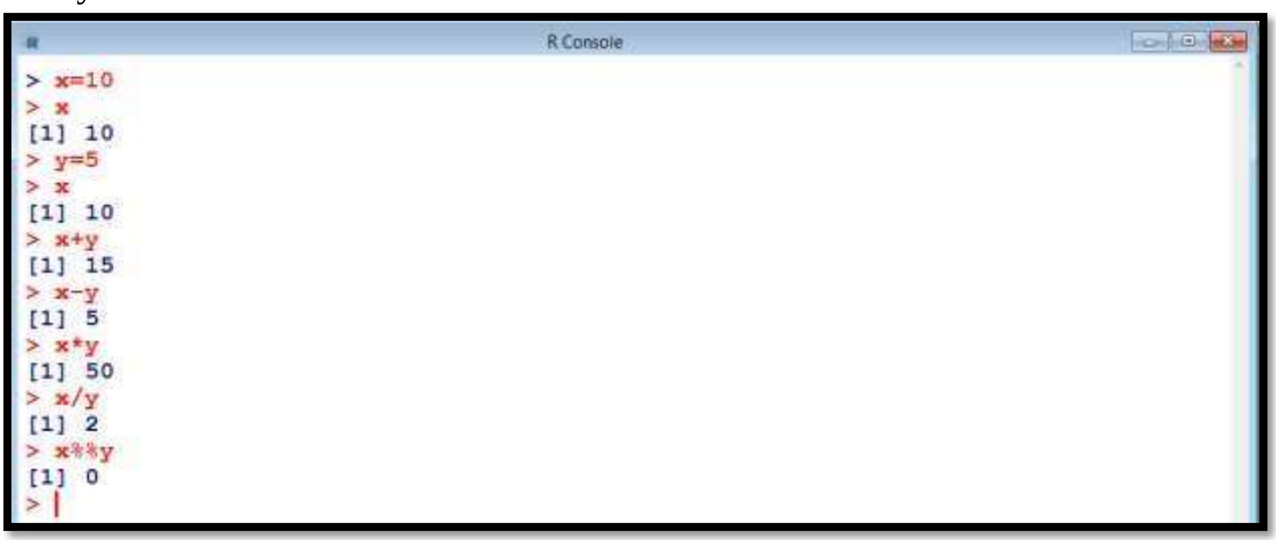
Using R execute the basic commands, array, list and frames.

Solution:-

1. Using R Execute the Basic Commands #For Help help() or
??help



#Define Variable To Perform Arithematic Operation x=10 x
y=5 x x+y x-y x*y
x/y
x%%y



#Use of c() in R

"c" stands for "combine". You are combining items together into one vector.

x=c(1,2,3,3.3,1.8) x
y=c(1,2,3,3.3,1.8, TRUE) y

z=c(11, d = list(1:3))
z

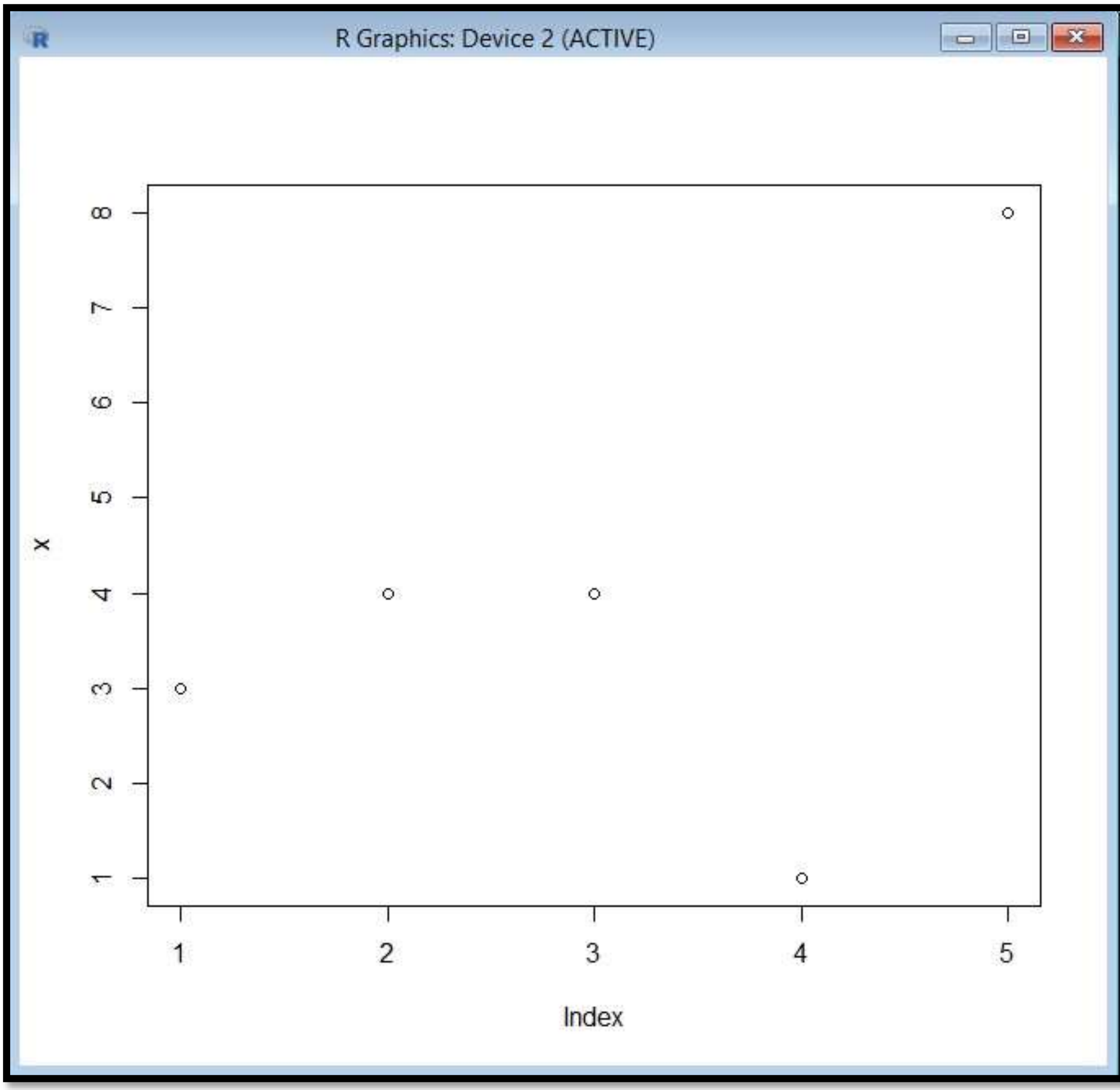
```
R Console
> x=c(1,2,3,3.3,1.8)
> x
[1] 1.0 2.0 3.0 3.3 1.8
> y=c(1,2,3,3.3,1.8, TRUE)
> y
[1] 1.0 2.0 3.0 3.3 1.8 1.0
> z=c(11, d = list(1:3))
> z
[[1]]
[1] 11

$d
[1] 1 2 3

> |
```

#For Finding Mean and Standard Deviation
x=c(3,4,4,1,8) x
#For Mean mean(x)

#For Standard Deviation
sd(x)

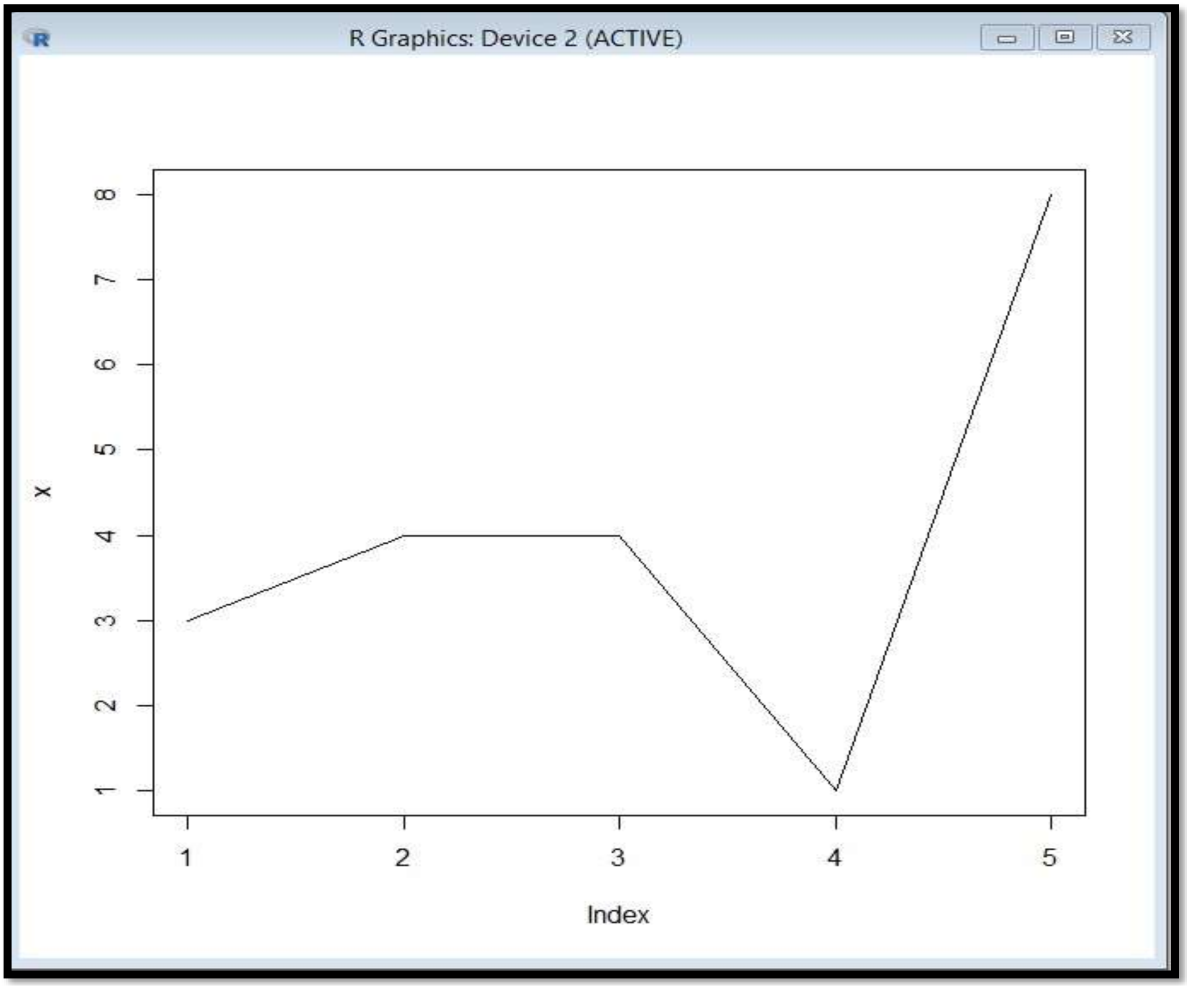
```
R Console
> x=c(3,4,4,1,8)
> x
[1] 3 4 4 1 8
> mean(x)
[1] 4
> sd(x)
[1] 2.54951
> |
```

#For Plotting The Graph
plot(x)



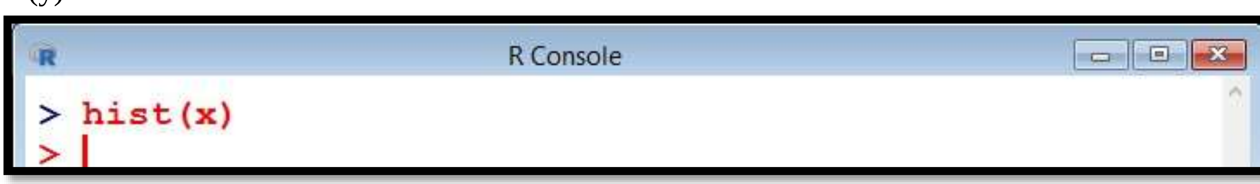
#For Plotting The Line Graph

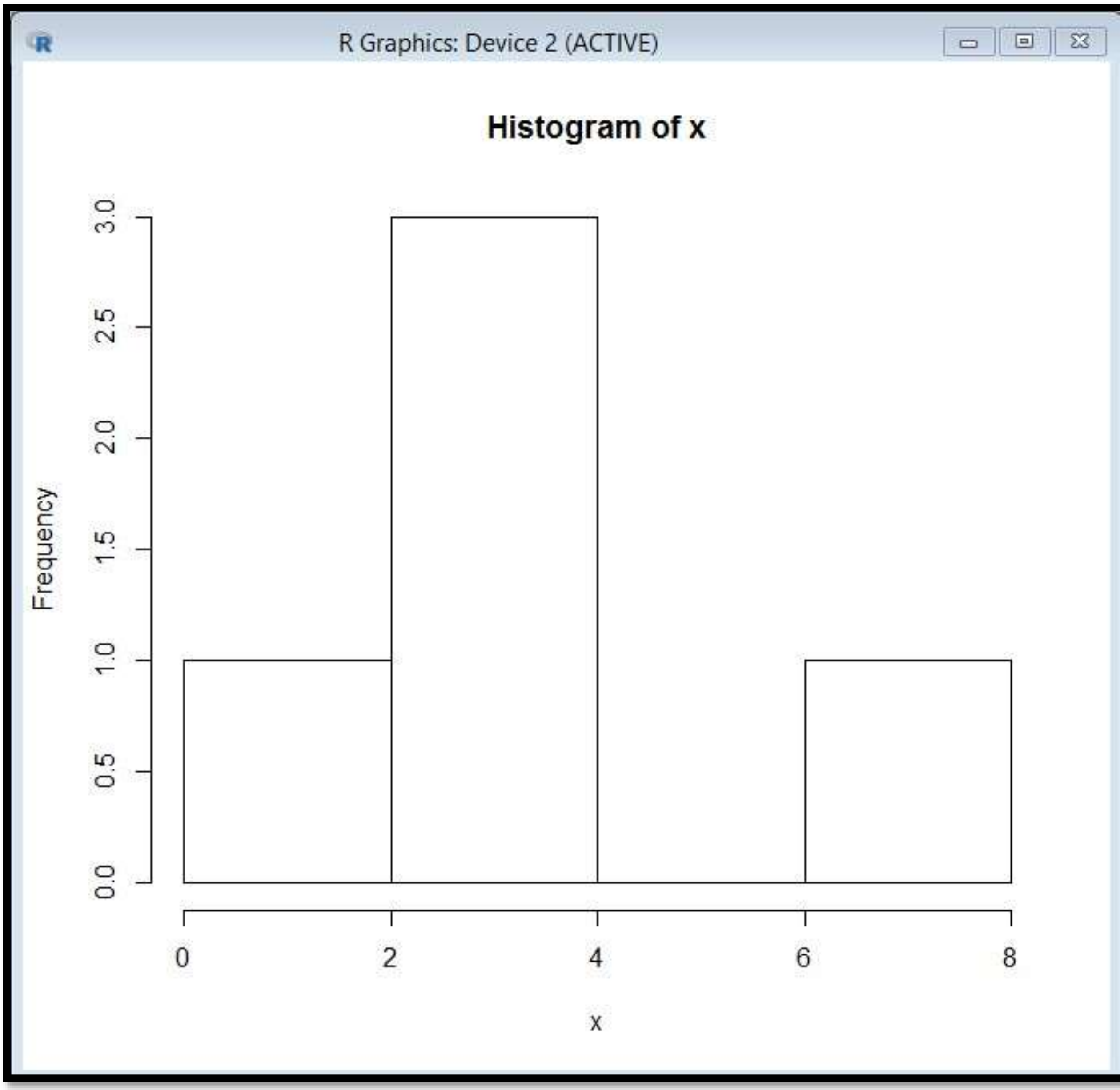
plot(x,type='l')



#For Plotting Histogram

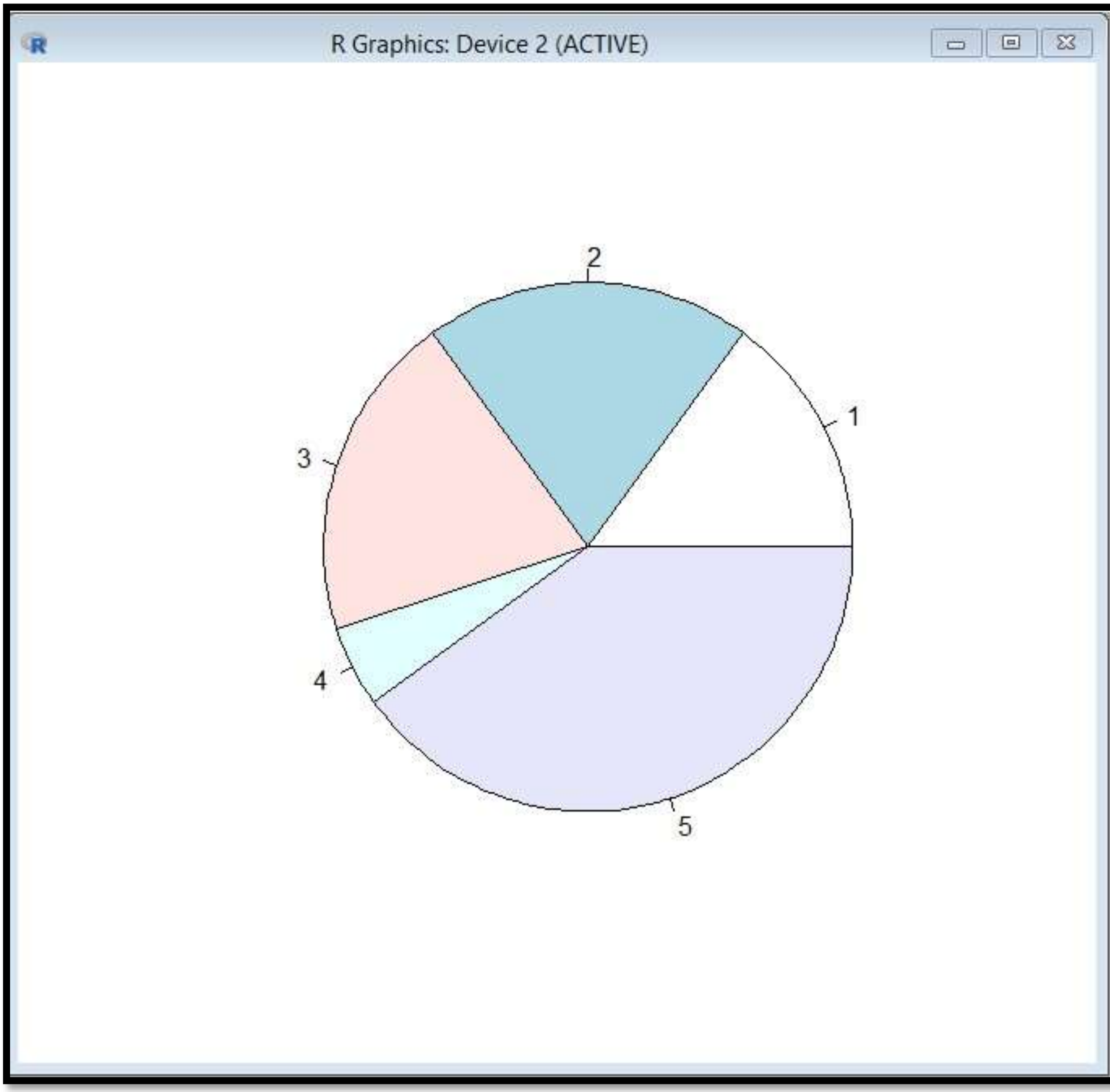
hist(y)





#For Plotting Pie Chart

pie(x)



2. Using R Create Array and Perform Some Operation on it.

An array is created using the **array ()** function. It takes vectors as input and uses the values in the **dim** parameter to create an array.

A. Creates an array of two 3x3 matrices each with 3 rows and 3 columns.

arr1=c(1,2,3) arr2=c(5,10,15,20,25,30)
myarr=array(c(arr1,arr2),dim(3,3))
print(myarr)

```
R Console
> arr1=c(1,2,3)
> arr2=c(5,10,15,20,25,30)
> myarr=array(c(arr1,arr2),dim=c(3,3))
> print(myarr)
     [,1] [,2] [,3]
[1,]    1    5   20
[2,]    2   10   25
[3,]    3   15   30
>
```

B. Naming Columns and Rows and Array

```
R Console
> arr1=c(1,2,3)
> arr2=c(5,10,15,20,25,30)
> colname=c("col1","col2","col3")
> rowname=c("row1","row2","row3")
> myarr=array(c(arr1,arr2),dimname=list(rowname,colname),dim=c(3,3))
> print(myarr)
     col1 col2 col3
row1    1    5   20
row2    2   10   25
row3    3   15   30
>
```

C. Accessing Array Elements

#Print the 2nd row of the array

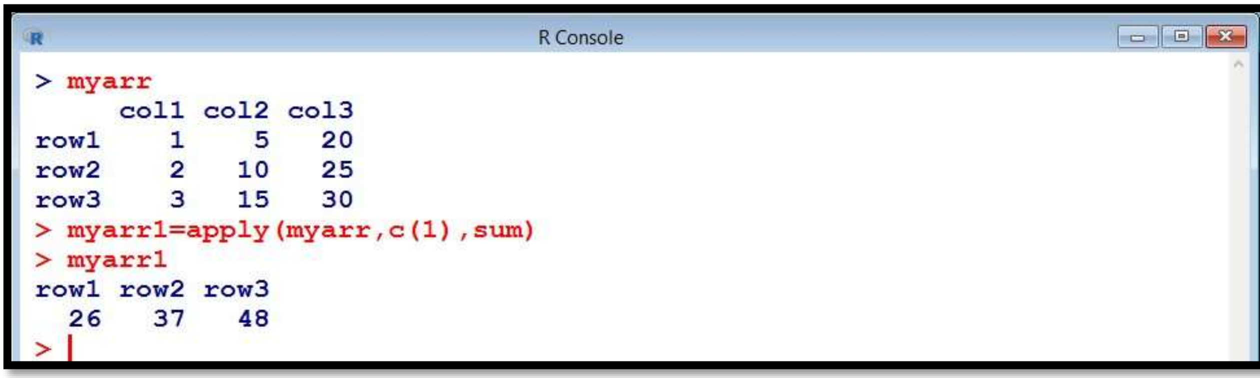
print(myarr[2,])

#Print the Element of 1st row and 3rd column of an array print(myarr[1,3])

```
> print(myarr)
     col1 col2 col3
row1    1    5   20
row2    2   10   25
row3    3   15   30
> print(myarr[2,])
col1 col2 col3
   2   10   25
> print(myarr[1,3])
[1] 20
>
```

D. Calculations across Array Elements

We can do calculations across the elements in an array using the **apply()** function. **# Use apply to calculate the sum of the rows across all the matrices.**



```
R Console
> myarr
     col1 col2 col3
row1    1    5   20
row2    2   10   25
row3    3   15   30
> myarr1=apply(myarr,c(1),sum)
> myarr1
row1 row2 row3
  26   37   48
>
```

3. Using R Create a List

List is created using list() function.

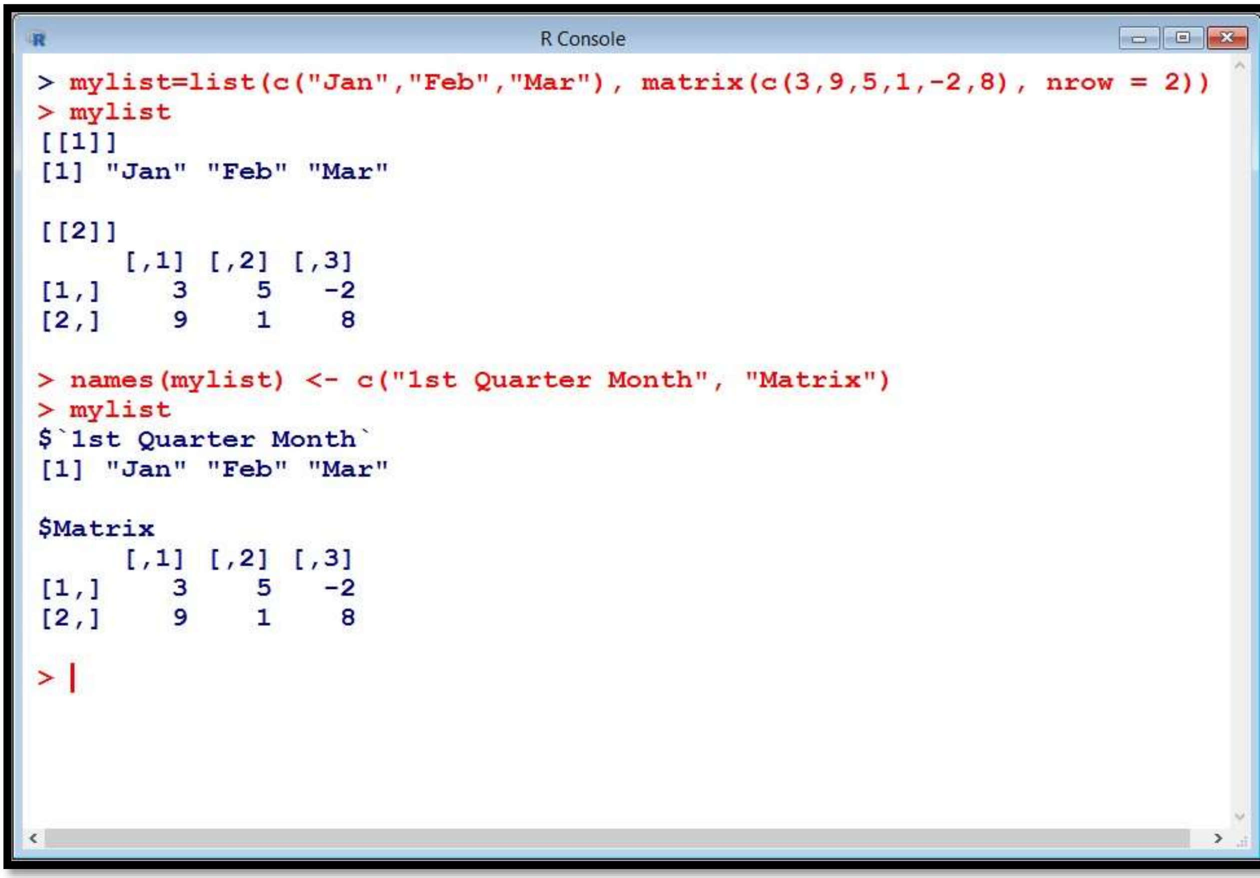
A.Create a list containing strings, numbers, vectors and logical values.

list_data = list ("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)
print(list_data)

```
> list_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)
> print(list_data)
[[1]]
[1] "Red"

[[2]]
[1] "Green"

[[3]]
[1] 21 32 11

[[4]]
[1] TRUE

[[5]]
[1] 51.23

[[6]]
[1] 119.1

>
```

<u>Naming List Elements</u>

mylist=list(c ("Jan","Feb","Mar"), matrix(c (3, 9,5,1,-2,8), nrow = 2))
mylist
names (mylist) <- c ("1st Quarter Month", "Matrix") mylist

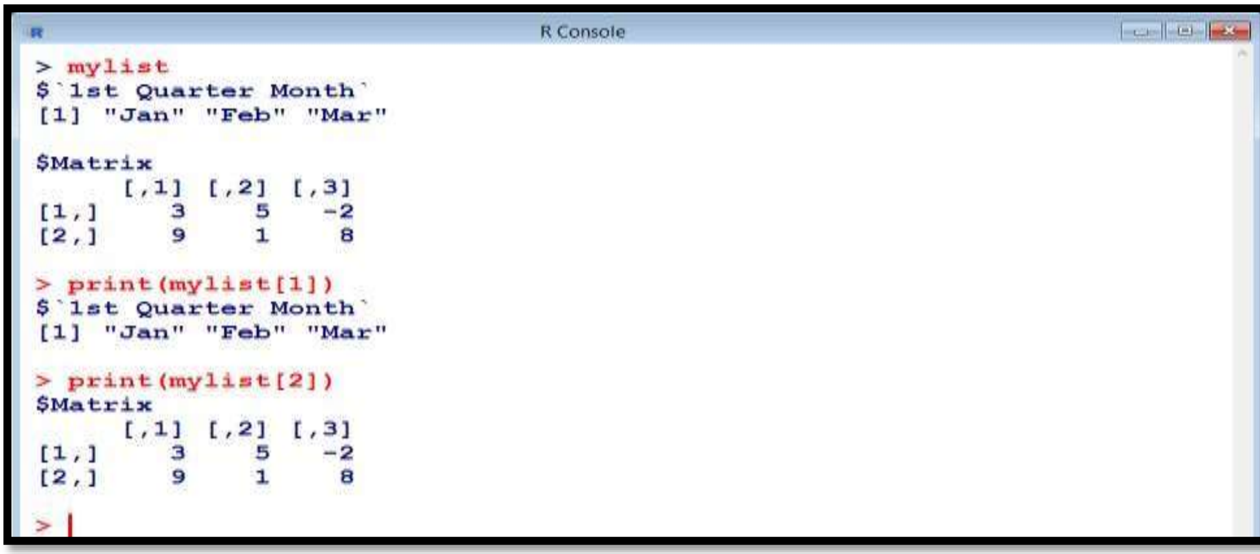


C. Accessing List Elets

Access the first element of the list.

print(mylist[1]) **# Access the thrid element.**

print(mylist[2])

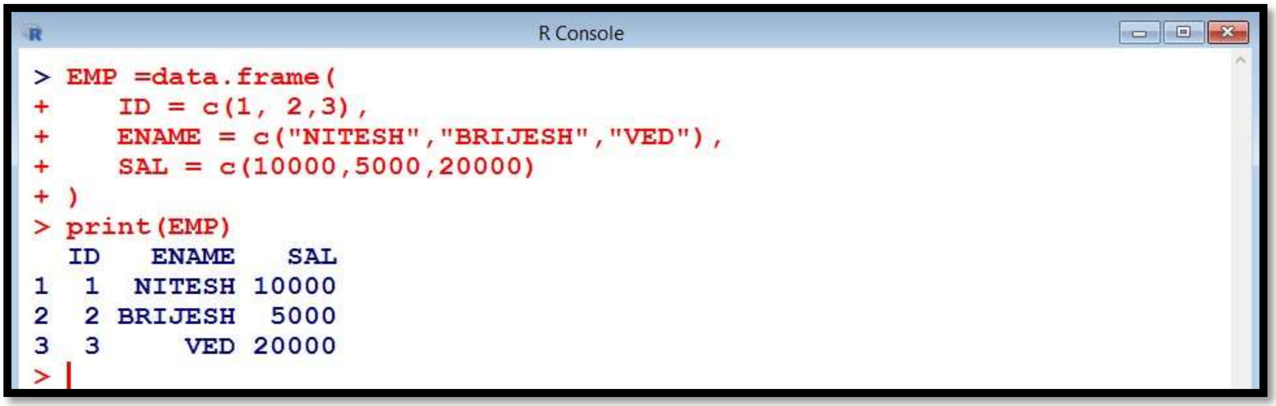


4. DATA FRAMES

Data frames are tabular data objects. Data Frames are created using the **data.frame()** function.

A. Create the data frame.

```
EMP =data.frame(
  ID = c(1, 2,3),
  ENAME = c("NITESH","BRIJESH","VED"),
  SAL = c(10000,5000,20000)
)
print (EMP)
```



```
> EMP =data.frame(
+     ID = c(1, 2,3),
+     ENAME = c("NITESH","BRIJESH","VED"),
+     SAL = c(10000,5000,20000)
+ )
> print(EMP)
  ID   ENAME   SAL
1  1  NITESH 10000
2  2 BRIJESH  5000
3  3     VED 20000
>
```

B. Get the Structure of the Data Frame

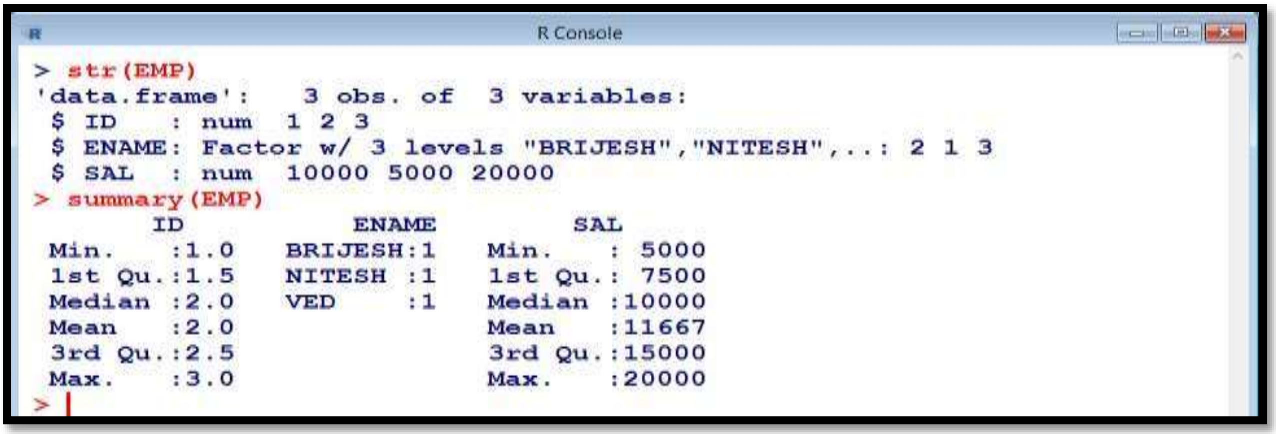
The structure of the data frame can be seen by using str() function.

str(EMP)

C. Summary of Data in Data Frame

The statistical summary and nature of the data can be obtained by applying summary () function.

summary(EMP)



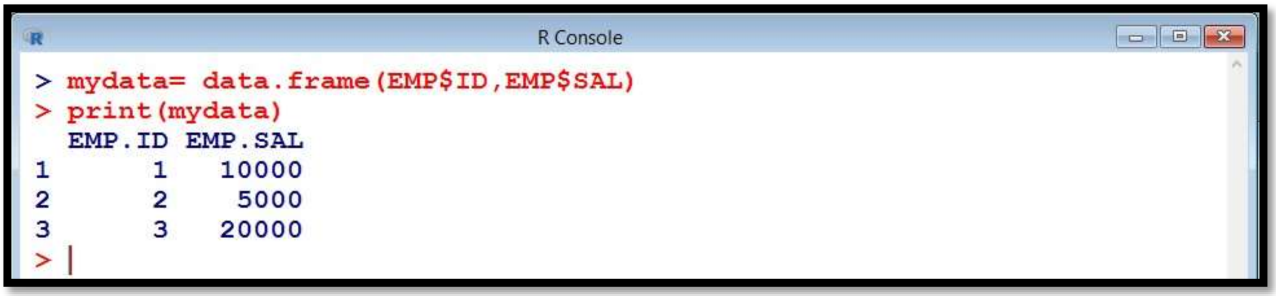
```
> str(EMP)
'data.frame':   3 obs. of  3 variables:
 $ ID   : num  1 2 3
 $ ENAME: Factor w/ 3 levels "BRIJESH","NITESH",..: 2 1 3
 $ SAL  : num  10000 5000 20000
> summary(EMP)
       ID            ENAME        SAL
 Min.   :1.0   BRIJESH:1   Min.   : 5000
 1st Qu.:1.5   NITESH :1   1st Qu.: 7500
 Median :2.0   VED    :1   Median :10000
 Mean   :2.0               Mean   :11667
 3rd Qu.:2.5               3rd Qu.:15000
 Max.   :3.0               Max.   :20000
>
```

D. Extract Data from Data Frame

Extract specific column from a data frame using column name. # Extract Specific columns.

mydata= data.frame(EMPID,EMPSAL)

print(mydata)



 # Extract first two rows.

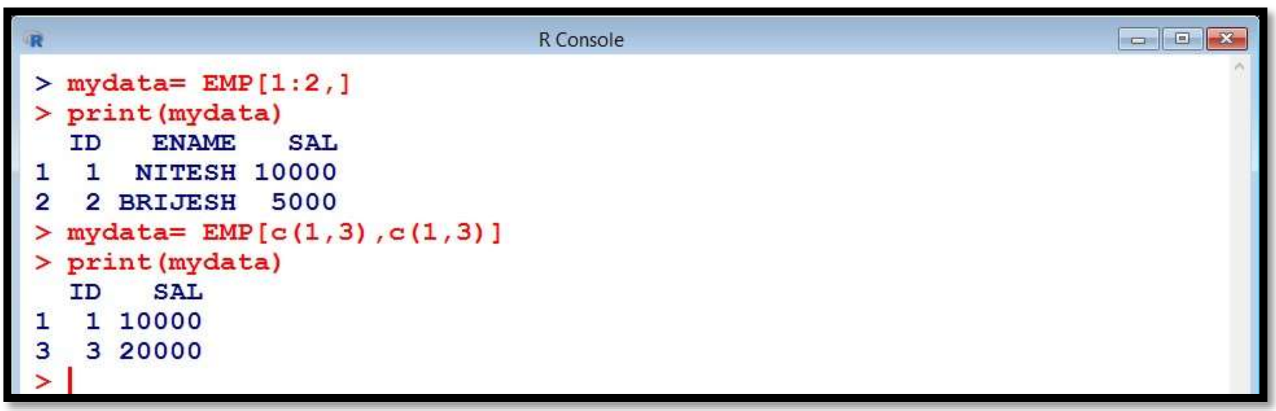
mydata= EMP[1:2,]

print(mydata)

 # Extract 1st and 3rd row with 1st and 3rd column

mydata= EMP[c(1,3),c(1,3)]

print(mydata)



E. Expand Data Frame

A data frame can be expanded by adding columns and rows.

I. Add Column

Just add the column vector using a new column name.

 # Add the "dept" coulmn.

EMP$DEPT = c("IT","Operations","HR")

EMP1=EMP

print(EMP1)

```
R Console
> EMP$DEPT = c("IT","Operations","HR")
> EMP1=EMP
> print(EMP1)
  ID   ENAME   SAL       DEPT
1  1  NITESH 10000         IT
2  2 BRIJESH  5000 Operations
3  3     VED 20000         HR
>
```

II. **Add Row**

To add more rows permanently to an existing data frame, we need to bring in the new rows in the same structure as the existing data frame and use the rbind() function. EMP2 =data.frame(

ID = c(4, 5,6),

ENAME = c("SANDEEP","ARVIND","ALOK"),

SAL = c(15000,10000,20000)

)

print(EMP2)

EMPFinal=rbind(EMP1,EMP2)

print(EMPFinal)

```
R Console
> EMP2 =data.frame(
+    ID = c(4, 5,6),
+    ENAME = c("SANDEEP","ARVIND","ALOK"),
+    SAL = c(45000,30000,35000),
+    DEPT = c("IT","HR","MGR")
+ )
> print(EMP2)
  ID   ENAME   SAL DEPT
1  4 SANDEEP 45000   IT
2  5  ARVIND 30000   HR
3  6    ALOK 35000  MGR
> EMPFinal=rbind(EMP1,EMP2)
> EMPFinal
  ID   ENAME   SAL       DEPT
1  1  NITESH 10000         IT
2  2 BRIJESH  5000 Operations
3  3     VED 20000         HR
4  4 SANDEEP 45000         IT
5  5  ARVIND 30000         HR
6  6    ALOK 35000        MGR
>
```

PRACTICAL NO:-2

AIM:-

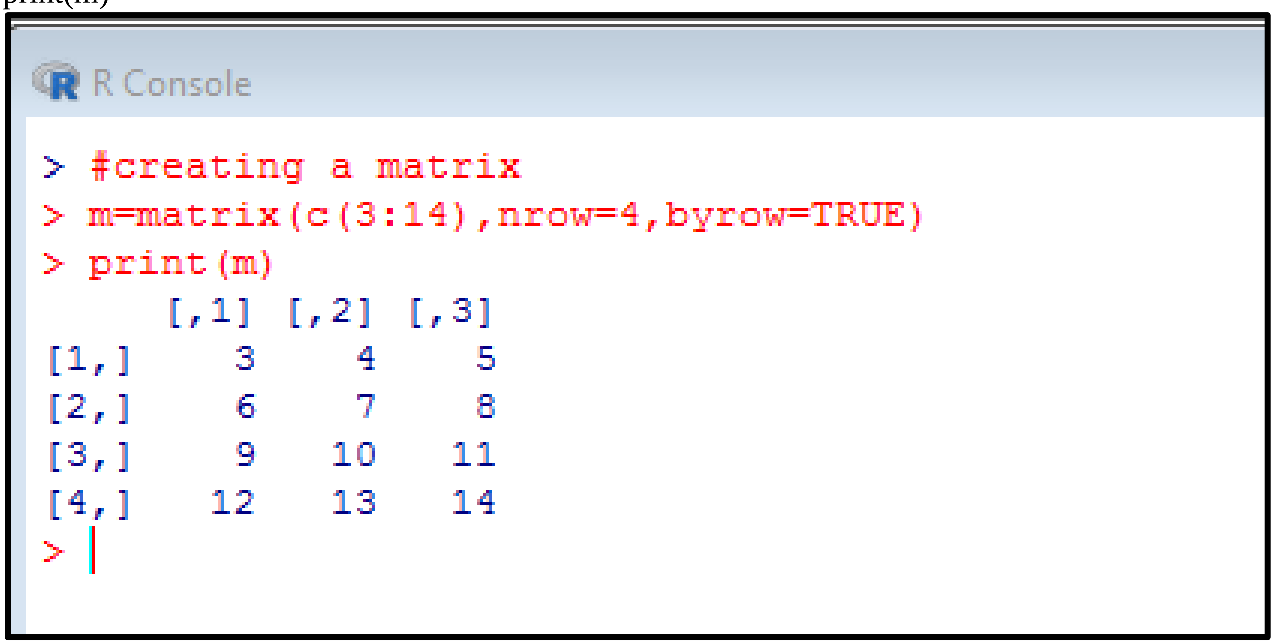
Create a matrix using R and perform the operations addition, inverse, transpose and multiplication operations.

solution:-

#**creating a matrix**

m=matrix(c(3:14),nrow=4,byrow=TRUE)

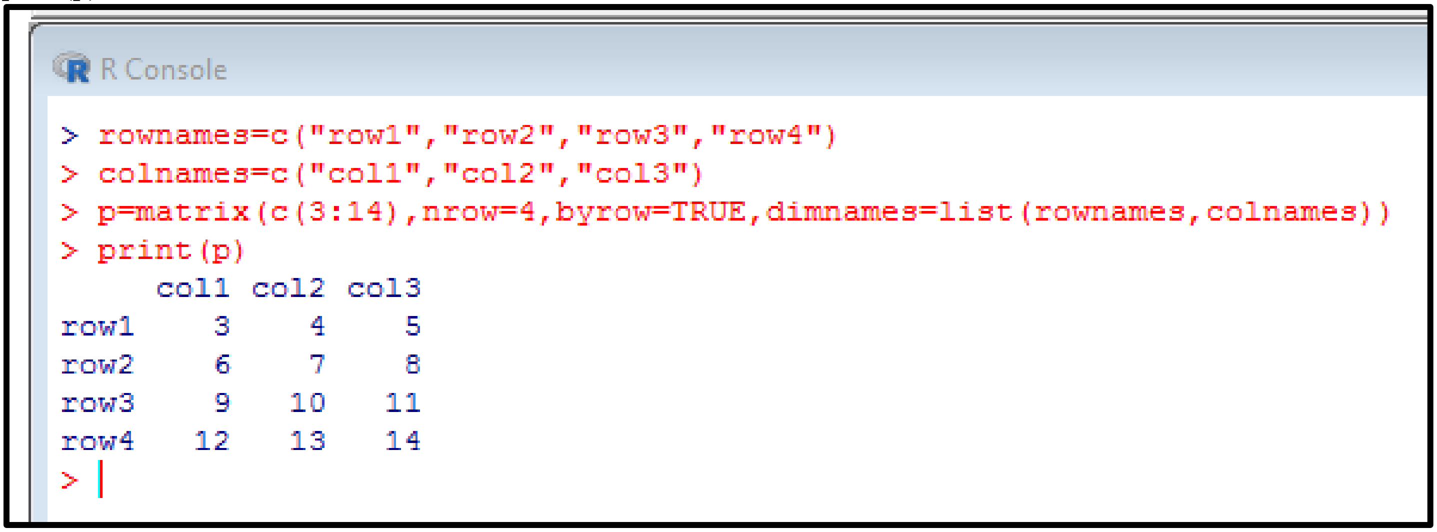
print(m)



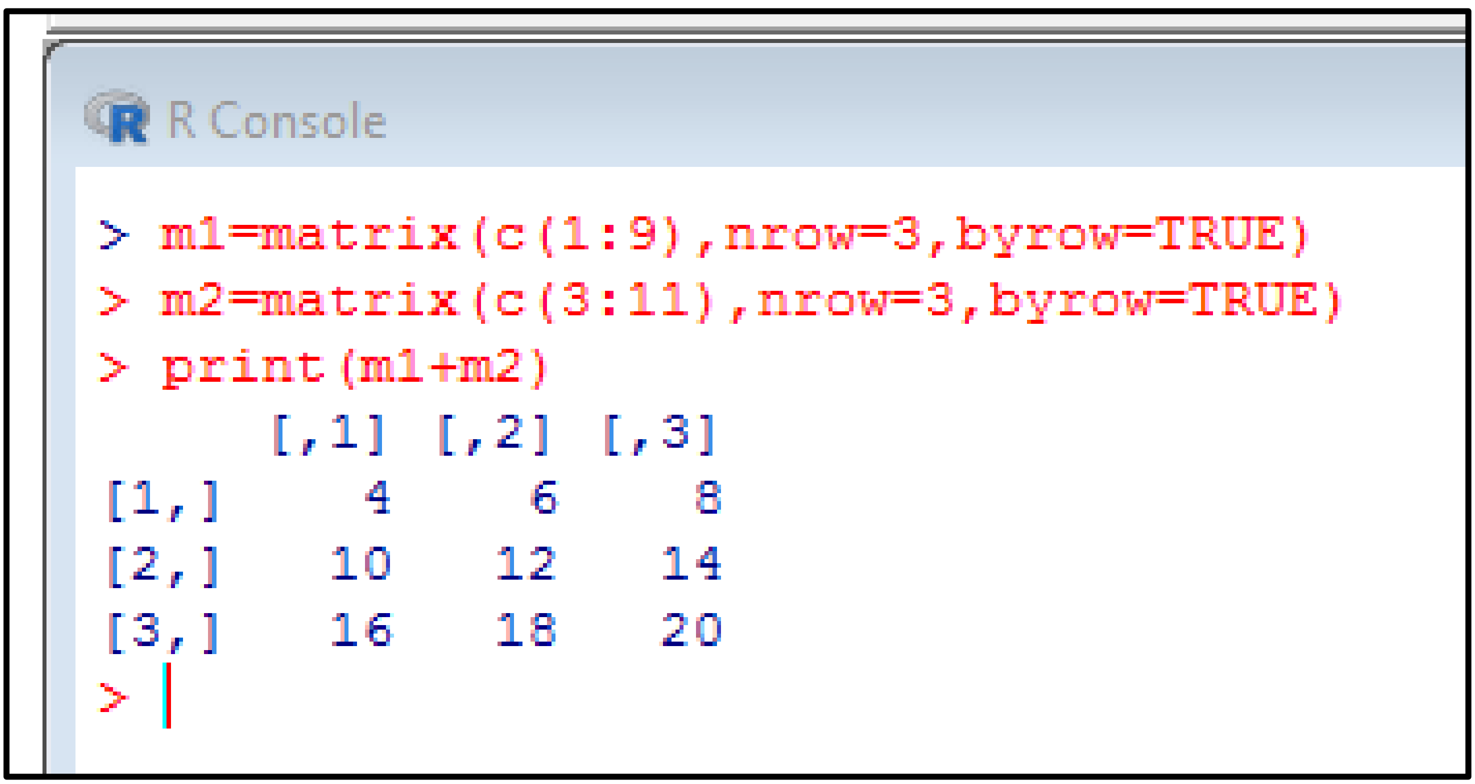
#Naming columns and rows to the Matrix

rownames=c("row1","row2","row3","row4")

colnames=c("col1","col2","col3")
p=matrix(c(3:14),nrow=4,byrow=TRUE,dimnames=list(rownames,colnames))
print(p)



#**<u>To performing addition</u>**
m1=matrix(c(1:9),nrow=3,byrow=TRUE)
m2=matrix(c(3:11),nrow=3,byrow=TRUE)
print(m1+m2)

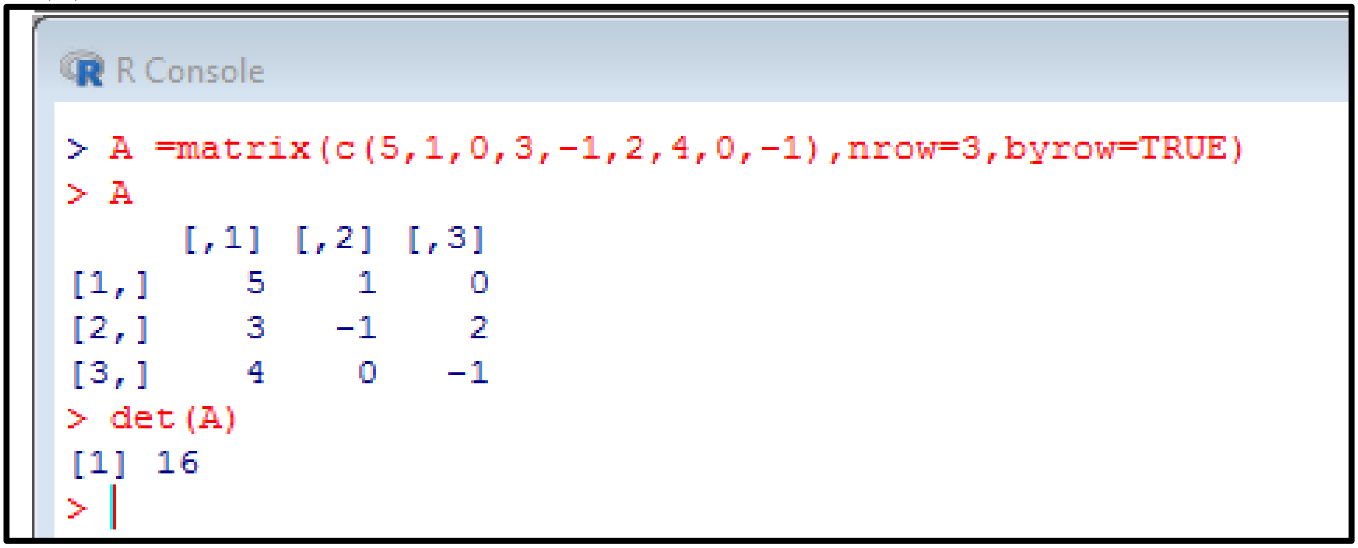


To find inverse of a matrix

First find the determinant of that matrix then perform inverse operation

1. finding determinant

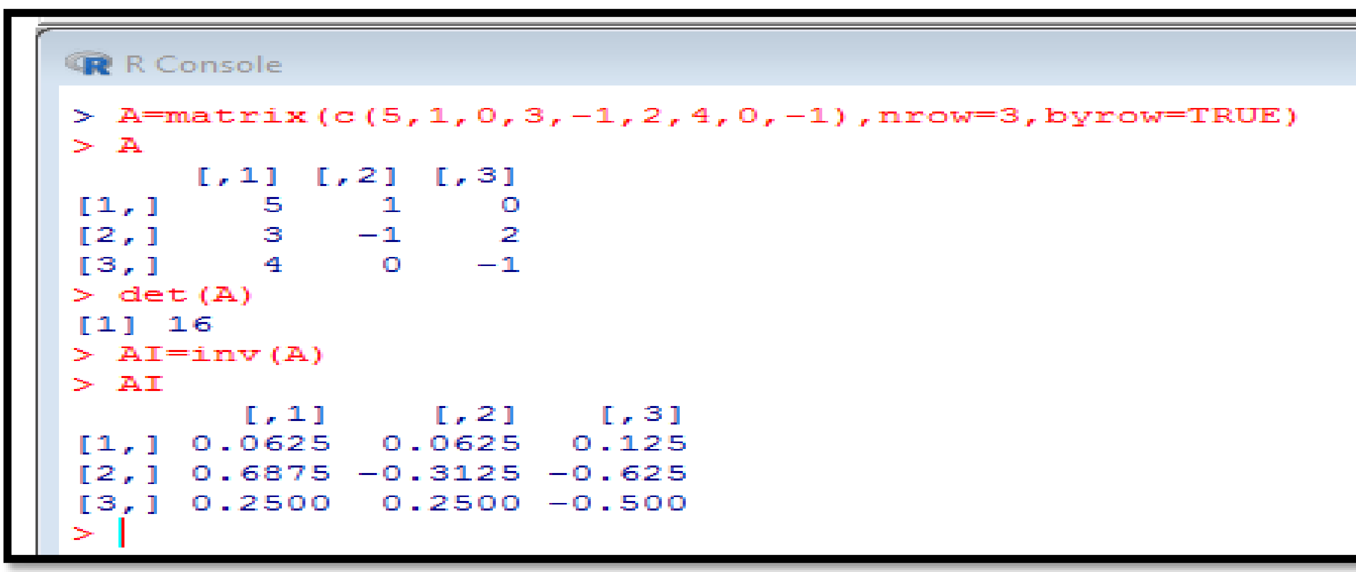
A =matrix(c(5,1,0,3,-1,2,4,0,-1),nrow=3,byrow=TRUE)
A
det(A)



```
R Console
> A =matrix(c(5,1,0,3,-1,2,4,0,-1),nrow=3,byrow=TRUE)
> A
     [,1] [,2] [,3]
[1,]    5    1    0
[2,]    3   -1    2
[3,]    4    0   -1
> det(A)
[1] 16
>
```

2. **Finding inverse**

A=matrix(c (5, 1, 0, 3,-1, 2, 4, 0,-1), nrow=3, byrow=TRUE)
A
det (A)
AI=inv (A)
AI



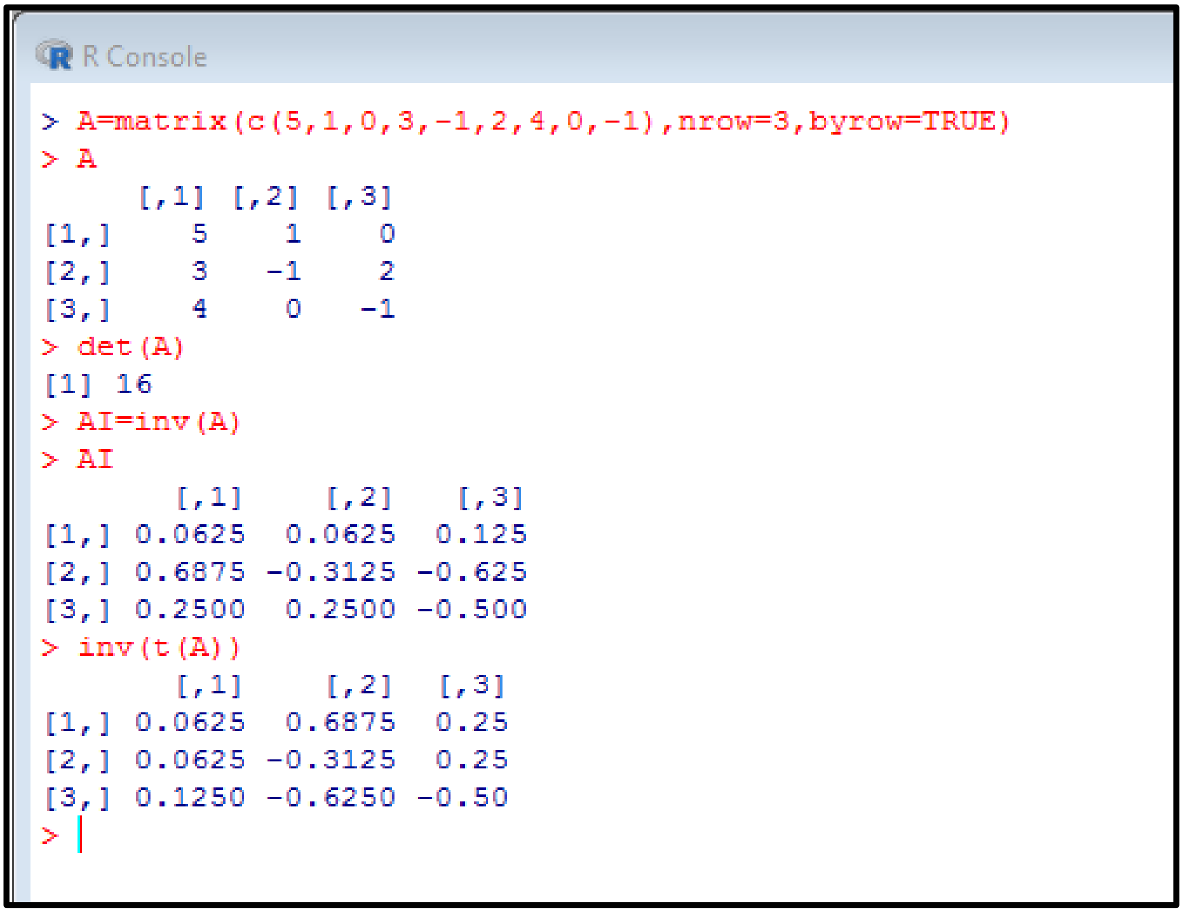
```
R Console
> A=matrix(c(5,1,0,3,-1,2,4,0,-1),nrow=3,byrow=TRUE)
> A
     [,1] [,2] [,3]
[1,]    5    1    0
[2,]    3   -1    2
[3,]    4    0   -1
> det(A)
[1] 16
> AI=inv(A)
> AI
       [,1]    [,2]   [,3]
[1,] 0.0625  0.0625  0.125
[2,] 0.6875 -0.3125 -0.625
[3,] 0.2500  0.2500 -0.500
>
```

#**To find transpose of a matrix**

A=matrix(c(5,1,0,3,-1,2,4,0,-1),nrow=3,byrow=TRUE)

A
det(A)
AI=inv(A)
inv(t(A))

To perform multiplication operation



```
R Console
> A=matrix(c(5,1,0,3,-1,2,4,0,-1),nrow=3,byrow=TRUE)
> A
     [,1] [,2] [,3]
[1,]    5    1    0
[2,]    3   -1    2
[3,]    4    0   -1
> det(A)
[1] 16
> AI=inv(A)
> AI
       [,1]    [,2]   [,3]
[1,] 0.0625  0.0625  0.125
[2,] 0.6875 -0.3125 -0.625
[3,] 0.2500  0.2500 -0.500
> inv(t(A))
       [,1]    [,2]  [,3]
[1,] 0.0625  0.6875  0.25
[2,] 0.0625 -0.3125  0.25
[3,] 0.1250 -0.6250 -0.50
>
```

M1=c(1:9)
m2=c(3:11)
res1=matrix(m1,nrow=3,byrow=FALSE)
res2=matrix(m2,nrow=3,byrow=TRUE)
print(res1)
print(res2)
cat("multiplication of matrix\n")
res1*res2

R Console

```
> m1=c(1:9)
> m2=c(3:11)
> res1=matrix(m1,nrow=3,byrow=FALSE)
> res2=matrix(m2,nrow=3,byrow=TRUE)
> print(res1)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> print(res2)
     [,1] [,2] [,3]
[1,]    3    4    5
[2,]    6    7    8
[3,]    9   10   11
> cat("multiplication of matrix\n")
multiplication of matrix
> res1*res2
     [,1] [,2] [,3]
[1,]    3   16   35
[2,]   12   35   64
[3,]   27   60   99
> 
```

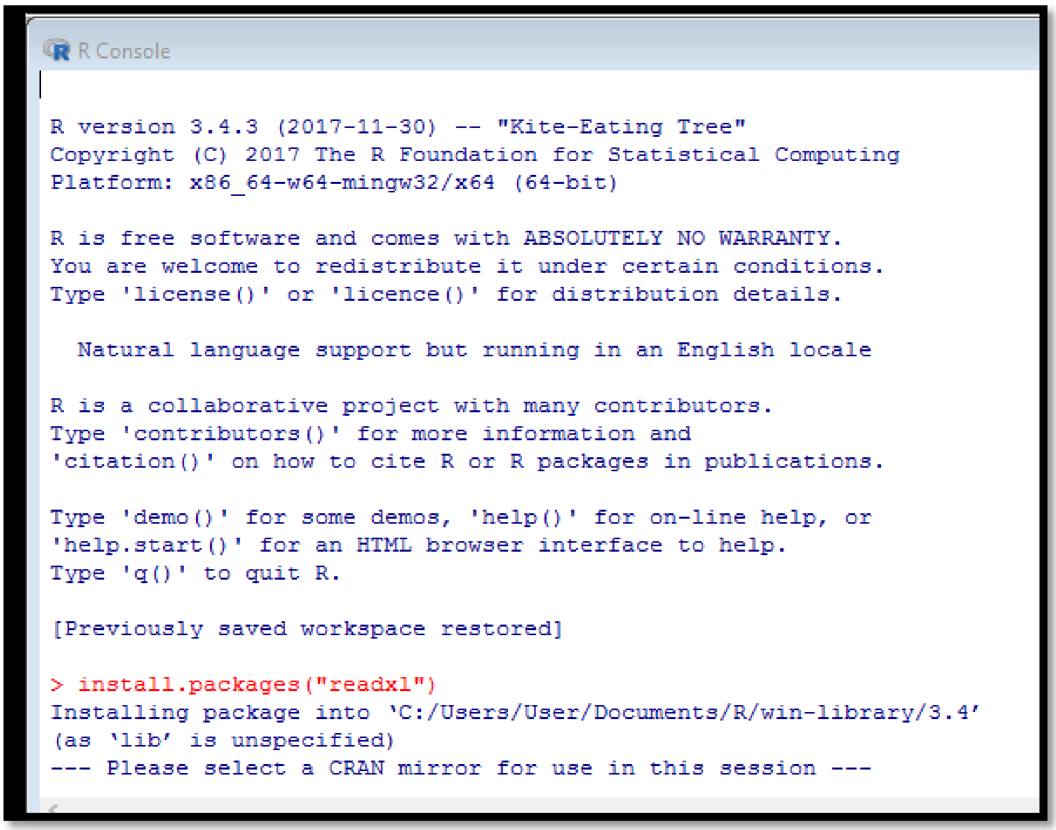
PRACTICAL NO.3

AIM: -

Using R Execute the statistical functions: mean, median, mode, quartiles, range, Inter quartile range histogram. Using R import the data from Excel file and Perform above functions.

<u>SOLUTION:</u>-

Microsoft Excel is the most widely used spreadsheet program which stores data in the .xls or .xlsx format. First install readxl package.



```
R Console

R version 3.4.3 (2017-11-30) -- "Kite-Eating Tree"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

  Natural language support but running in an English locale

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.

[Previously saved workspace restored]

> install.packages("readxl")
Installing package into 'C:/Users/User/Documents/R/win-library/3.4'
(as 'lib' is unspecified)
--- Please select a CRAN mirror for use in this session ---
```

<u>Create file in excel</u>

	ID	FNAME	LNAME	SALARY
1	ID	FNAME	LNAME	SALARY
2	1	Moni	pal	25000
3	2	saniya	mirza	12000
4	3	ram	mourya	14000
5	4	sachin	gupta	22000
6	5	namrta	bhore	20000
7				

<u>Then in R console</u>

mydata=read_excel (file.choose ())

mydata

R Console

```
> mydata=read_excel(file.choose())
> mydata
# A tibble: 5 x 4
     ID FNAME  LNAME  SALARY
  <dbl> <chr>  <chr>   <dbl>
1  1.00 Moni   pal     25000
2  2.00 saniya mirza   12000
3  3.00 ram    mourya  14000
4  4.00 sachin gupta   22000
5  5.00 namrta bhore   20000
>
```

<u>To perform mean, median, mode following commands are used:</u>

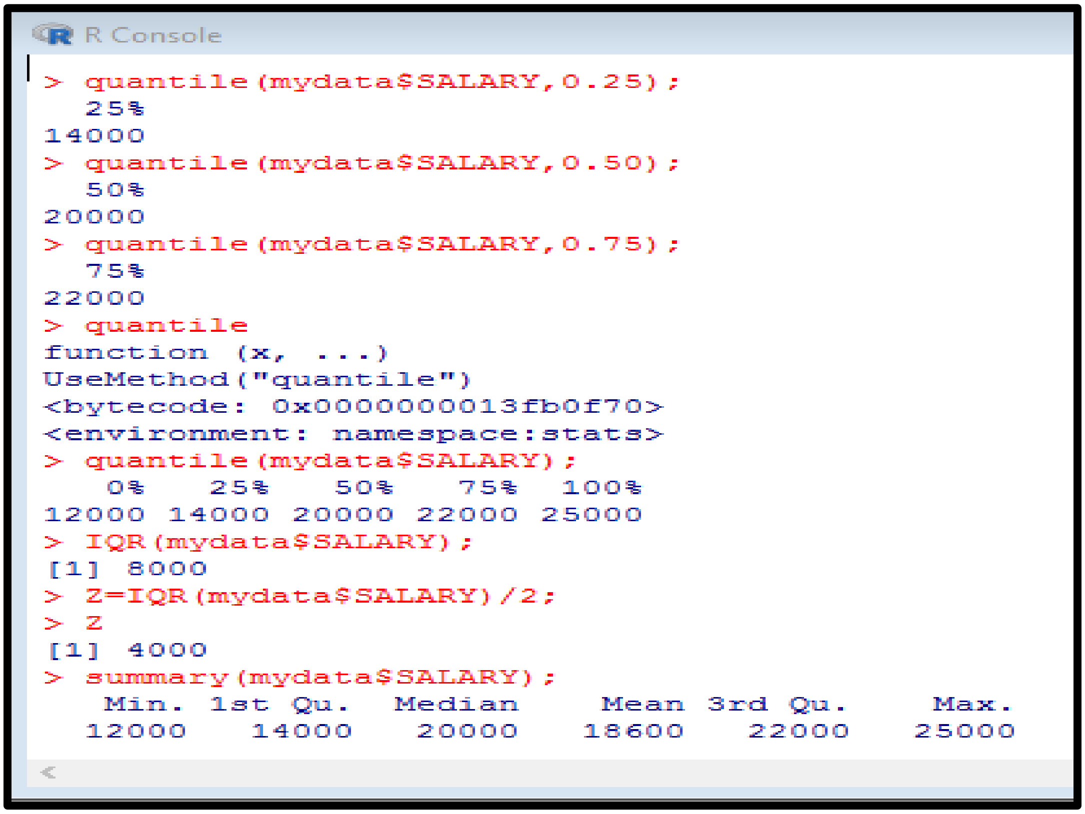
mean (mydata)

mean (mydata$SALARY)
median (mydata$SALARY)
names=(table(mydata$SALARY))[(mydata$SALARY)==max(table(mydata$SALARY))]

R Console

```
> mydata=read_excel(file.choose())
> mydata
# A tibble: 5 x 4
     ID FNAME  LNAME  SALARY
  <dbl> <chr>  <chr>   <dbl>
1  1.00 Moni   pal     25000
2  2.00 saniya mirza   12000
3  3.00 ram    mourya  14000
4  4.00 sachin gupta   22000
5  5.00 namrta bhore   20000
> mean(mydata)
[1] NA
Warning message:
In mean.default(mydata) : argument is not numeric or logical: returning NA
> mean(mydata$SALARY)
[1] 18600
> median(mydata$SALARY)
[1] 20000
> names=(table(mydata$SALARY))[table(mydata$SALARY)==max(table(mydata$SALARY))]
> names(table(mydata$SALARY))[table(mydata$SALARY)==max(table(mydata$SALARY))]
[1] "12000" "14000" "20000" "22000" "25000"
> 
```

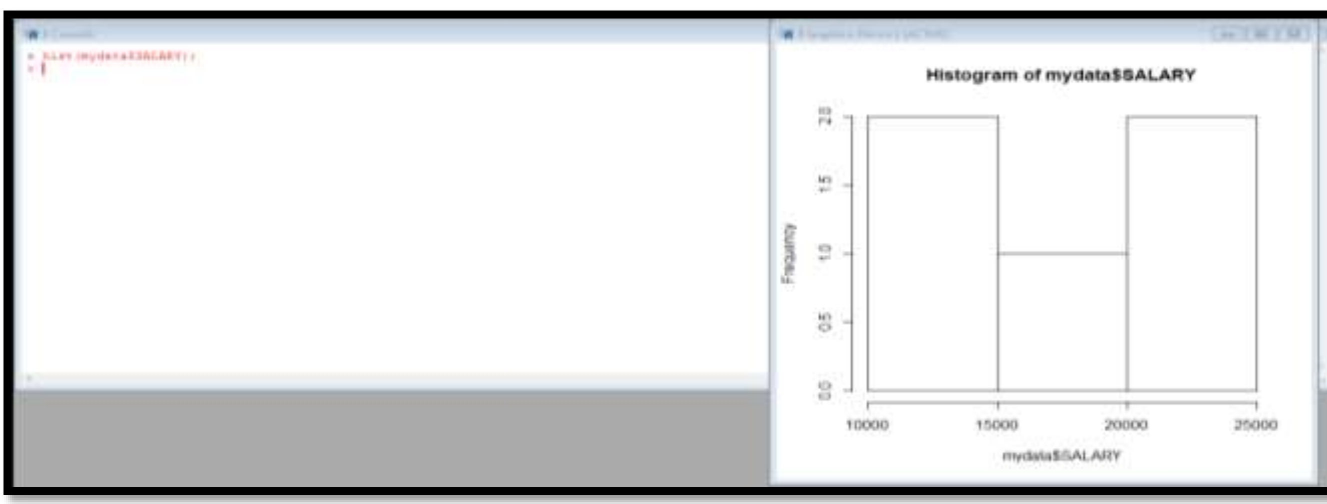
<u>To perform quartiles, inter quartile following commands are used:</u>
quantile (mydata$SALARY, 0.25)
quantile (mydata$SALARY, 0.50)
quantile (mydata$SALARY, 0.75)
quantile



```
R Console
> quantile(mydata$SALARY,0.25);
  25%
14000
> quantile(mydata$SALARY,0.50);
  50%
20000
> quantile(mydata$SALARY,0.75);
  75%
22000
> quantile
function (x, ...)
UseMethod("quantile")
<bytecode: 0x0000000013fb0f70>
<environment: namespace:stats>
> quantile(mydata$SALARY);
   0%   25%   50%   75%  100%
12000 14000 20000 22000 25000
> IQR(mydata$SALARY);
[1] 8000
> Z=IQR(mydata$SALARY)/2;
> Z
[1] 4000
> summary(mydata$SALARY);
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12000   14000   20000   18600   22000   25000
```

To perform histogram:

hist(mydata$SALARY)



PRACTICAL NO:-4

AIM: -

Using R import the data from Excel / .CSV file and Calculate the standard deviation, variance, co-variance.

Solutionn:-

First Create a file in excel

	A	B	C	D	E	F
1	id	name	salary	start_date	dept	
2	1	Rick	623.3	1/1/2012	IT	
3	2	Dan	515.2	9/23/2013	Operations	
4	3	Michelle	611	11/15/2014	IT	
5	4	Ryan	729	5/11/2014	HR	
6	5	Gary	843.25	3/27/2015	Finance	
7	6	Nina	578	5/21/2013	IT	
8	7	Simon	632.8	7/30/2013	Operations	
9	8	Guru	722.5	6/17/2014	Finance	
10						

Then install package "readxl"
Type Command in R control
mydata=read_excel(file.choose())
mydata

```
R Console
> mydata=read_excel(file.choose())
> mydata
# A tibble: 5 x 4
     ID FNMAE      SALARY DEPT
  <dbl> <chr>       <dbl> <chr>
1  1.00 Alisha      25000 IT
2  2.00 Zahara      26000 HR
3  3.00 Monica      24000 IT
4  4.00 Jayshree    23000 HR
5  5.00 Swapna      27000 IT
>
```

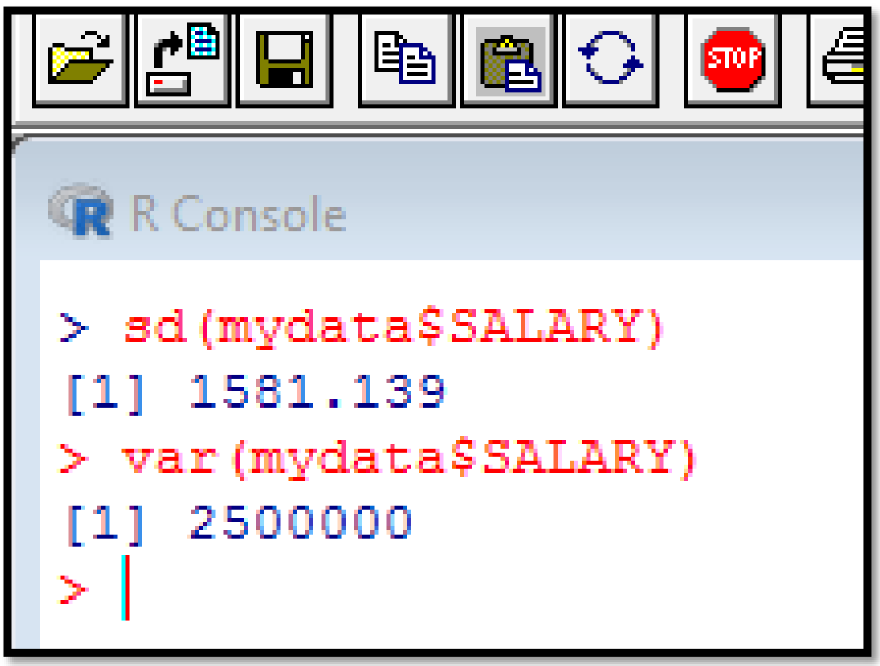
1. To Find standard deviation
The standard deviation of an observation variable is the square root of its variance.

The variance is a numerical measure of how the data values are dispersed around the mean.
In particular, the sample variance is defined as:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Similarly, the population variance is defined in terms of the population mean μ and population size N:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$



2. To find co variance

The covariance of two variables *x* and *y* in a data set measures how the two are linearly related.
A positive covariance would indicate a positive linear relationship between the variables, and a negative covariance would indicate the opposite.
The sample covariance is defined in terms of the sample means as:

$$s_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

Similarly, the population covariance is defined in terms of the population mean μ_x, μ_y as:

$$\sigma_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$$

R Console

```
> mydata=read_excel(file.choose())
> mydata
# A tibble: 5 x 4
     ID FNMAE     SALARY DEPT
  <dbl> <chr>      <dbl> <chr>
1  1.00 Alisha     25000 IT
2  2.00 Zahara     26000 HR
3  3.00 Monica     24000 IT
4  4.00 Jayshree   23000 HR
5  5.00 Swapna     27000 IT
> cov(mydata$ID,mydata$SALARY)
[1] 250
>
```

PRACTICAL NO.5

AIM:-

Using R import the data from Excel/ .CSV and draw the skewness.

SOLUTION:-

To draw skewness first installs package “e1071”.

Install.packages(“e1071”)

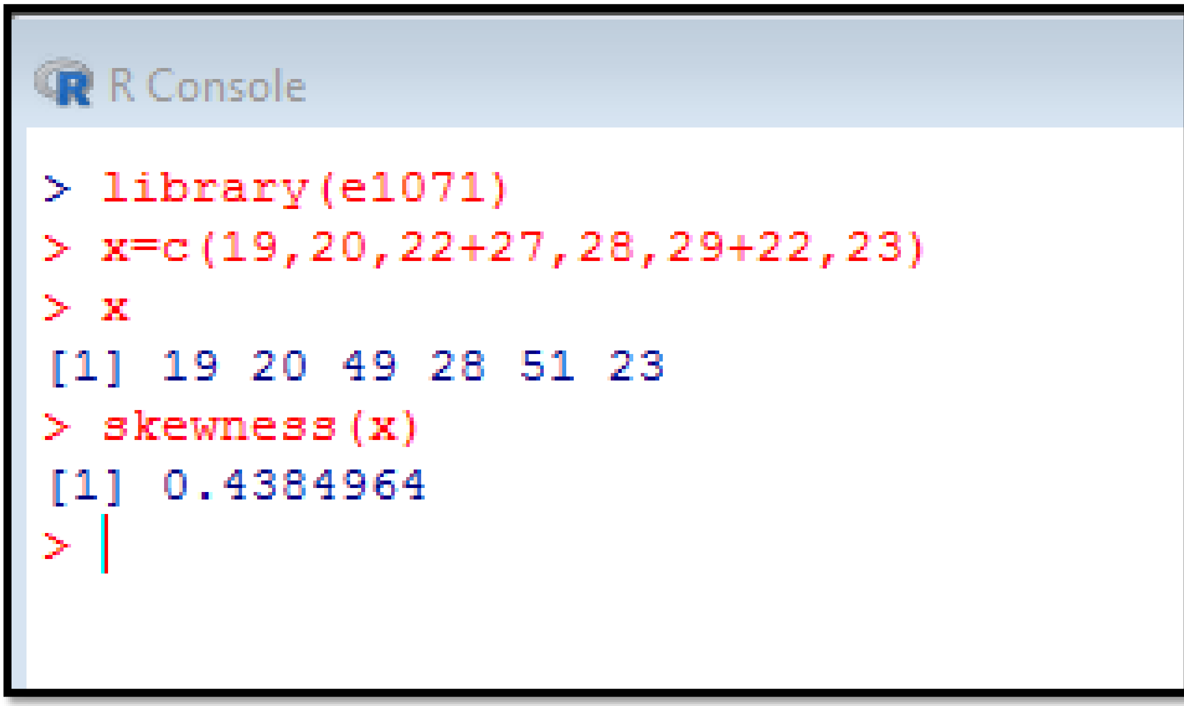
Library(e1071)

```
R Console
> install.packages("e1071")
Installing package into 'C:/Users/User/Documents/R/win-library/3.4'
(as 'lib' is unspecified)
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.4/e1071_1.6-8.zip'
Content type 'application/zip' length 894434 bytes (873 KB)
downloaded 873 KB

package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\User\AppData\Local\Temp\RtmpS6H8rW\downloaded_packages
> library(e1071)
> |
```

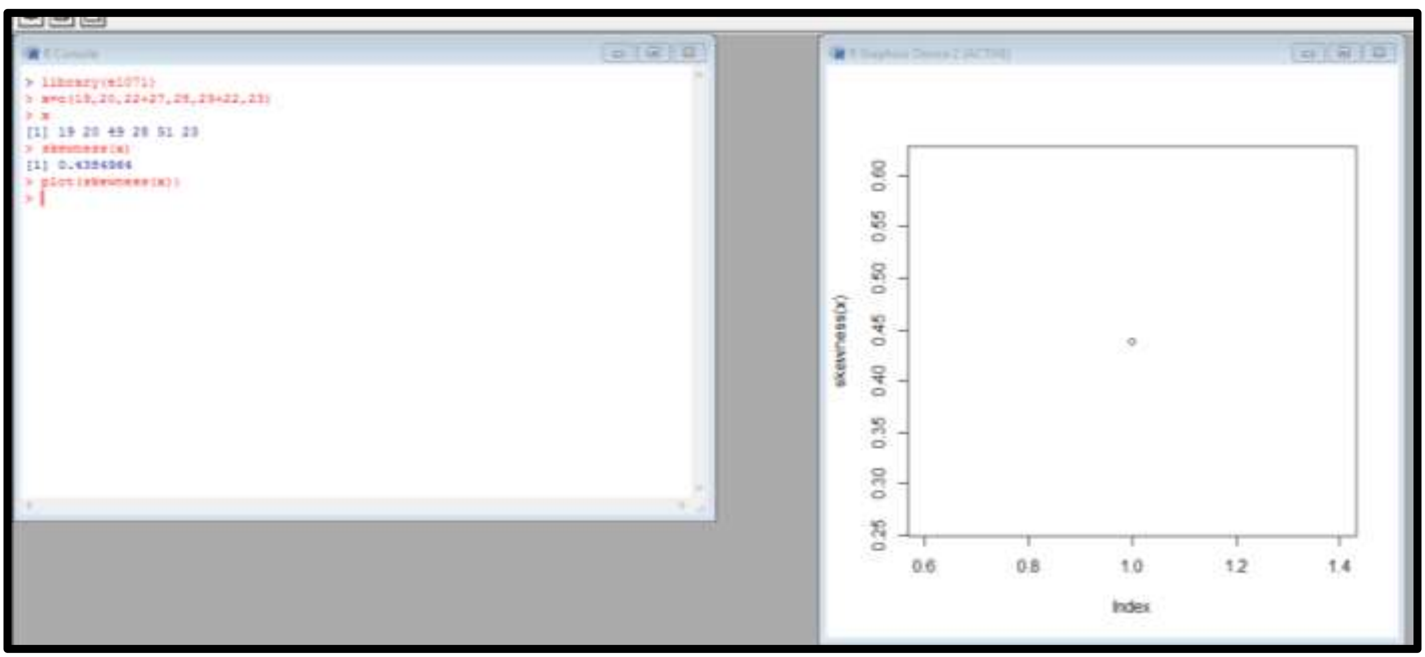
The skewness of a data population is defined by the following formula, where μ_2 and μ_3 are the second and third central moments.

$$\gamma_1 = \mu_3 / \mu_2^{3/2}$$

Then following commands are used for skewness



For plotting skewness



Practical No:06

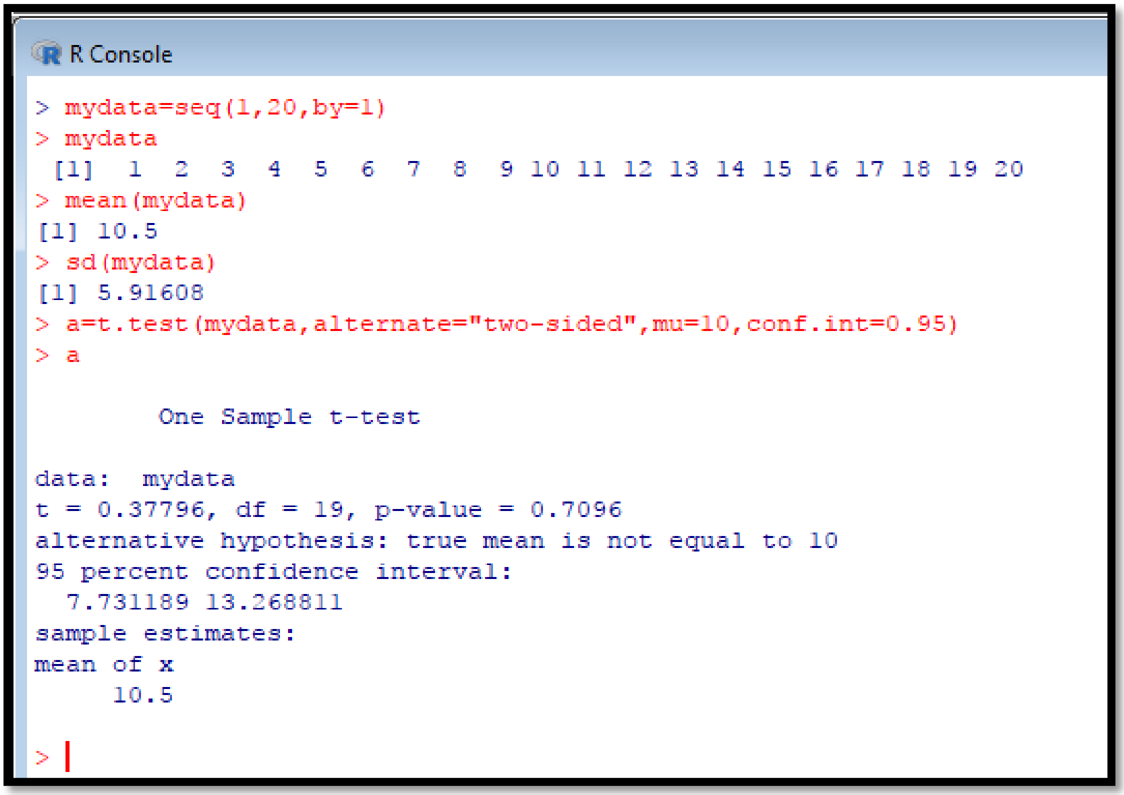
AIM:

Import the data from Excel / .CSV and perform the hypothetical testing.

SOLUTION:

T_test:

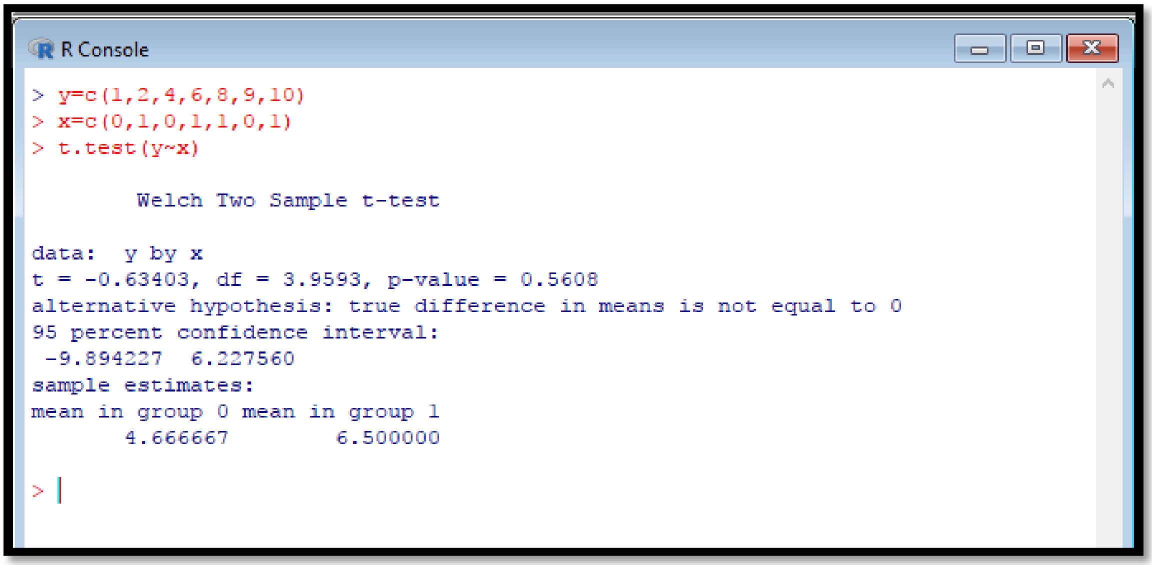
The **t.test ( )** function produces a variety of t-tests. Unlike most statistical packages, the default assumes unequal variance and applies the Welsh DF modification.




```
> mydata=seq(1,20,by=1)
> mydata
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
> mean(mydata)
[1] 10.5
> sd(mydata)
[1] 5.91608
> a=t.test(mydata,alternate="two-sided",mu=10,conf.int=0.95)
> a

        One Sample t-test

data:  mydata
t = 0.37796, df = 19, p-value = 0.7096
alternative hypothesis: true mean is not equal to 10
95 percent confidence interval:
  7.731189 13.268811
sample estimates:
mean of x
     10.5

> 
```

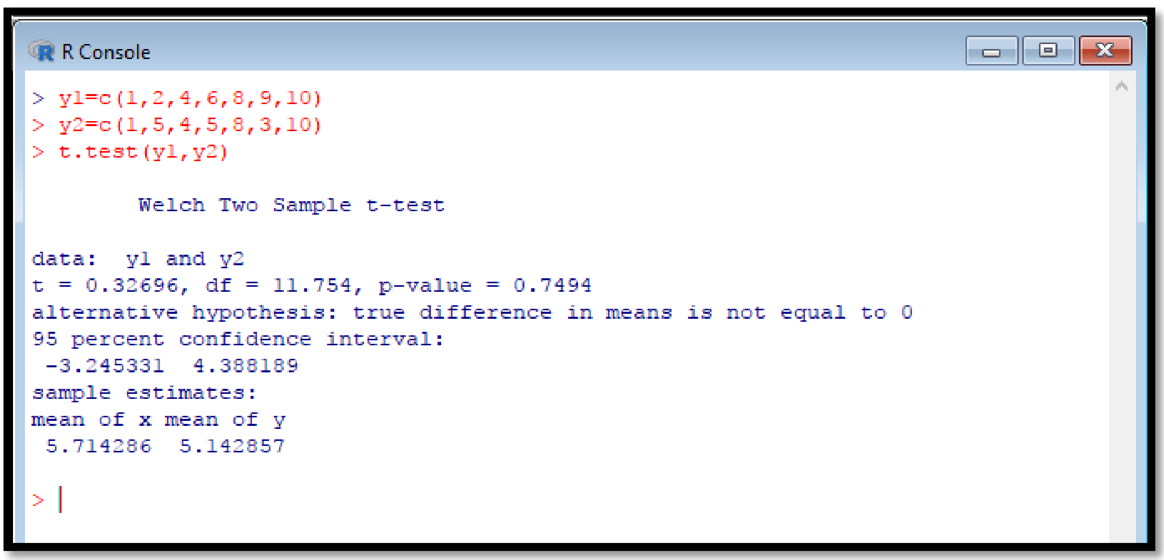
 # Independent 2-group t-test

y=c(1,2,4,6,8,9,10)
x=c(0,1,0,1,1,01)
t.test(y~x) # where y is numeric and x is a binary factor



```
R Console
> y=c(1,2,4,6,8,9,10)
> x=c(0,1,0,1,1,0,1)
> t.test(y~x)

        Welch Two Sample t-test

data:  y by x
t = -0.63403, df = 3.9593, p-value = 0.5608
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -9.894227  6.227560
sample estimates:
mean in group 0 mean in group 1
       4.666667        6.500000

>
```

 # independent 2-group t-test
y1=c(1,2,4,6,8,9,10)
y2=c(1,5,4,5,8,3,10)
t.test(y1,y2) # where y1 and y2 are numeric



```
R Console
> y1=c(1,2,4,6,8,9,10)
> y2=c(1,5,4,5,8,3,10)
> t.test(y1,y2)

        Welch Two Sample t-test

data:  y1 and y2
t = 0.32696, df = 11.754, p-value = 0.7494
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.245331  4.388189
sample estimates:
mean of x mean of y
 5.714286  5.142857

>
```

 # paired t-test
y1=c(1,2,4,6,8,9,10)
y2=c(1,5,4,5,8,3,10)
t.test(y1,y2,paired=TRUE) # where y1 & y2 are numeric

```
R Console
> y1=c(1,5,4,6,8,6,10)
> y2=c(1,5,4,5,8,3,10)
> t.test(y1,y2,paired=TRUE)

        Paired t-test

data:  y1 and y2
t = 1.3333, df = 6, p-value = 0.2308
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.4772479  1.6201051
sample estimates:
mean of the differences
              0.5714286

>
```

<u>one sample t-test</u>

y1=c(1,2,4,6,8,9,10)
t.test(y,mu=3) # Ho: mu=3

```
R Console
> y=c(1,2,4,6,8,9,10)
> t.test(y,mu=3)

        One Sample t-test

data:  y
t = 2.0528, df = 6, p-value = 0.08591
alternative hypothesis: true mean is not equal to 3
95 percent confidence interval:
 2.478899 8.949673
sample estimates:
mean of x
 5.714286

>
```

<u>2. Problem</u>

Suppose the manufacturer claims that the mean lifetime of a light bulb is more than 10,000 hours. In a sample of 30 light bulbs, it was found that they only last 9,900 hours on average. Assume the population standard deviation is 120 hours. At .05 significance level, can we reject the claim by the manufacturer?

<u>SOLUTION:</u>-

The null hypothesis is that $\mu \geq 10000$. We begin with computing the test statistic.
We then compute the critical value at .05 significance level.

```
R Console
>  xbar = 9900
>  mu0 = 10000
>  sigma = 120
>  n = 30
>  z = (xbar-mu0)/(sigma/sqrt(n))
>  z
[1] -4.564355
> alpha = .05
>  z.alpha = qnorm(1-alpha)
>  -z.alpha
[1] -1.644854
> 
```

2. Problem:-

Suppose the food label on a cookie bag states that there is at most 2 grams of saturated fat in a single cookie. In a sample of 35 cookies, it is found that the mean amount of saturated fat per cookie is 2.1 grams. Assume that the population SD is 0.25 grams. At 0.05 significance level, can we reject the claim on food label?

SOLUTION:-

The null hypothesis is that mu<=2.

```
R Console
> xbar=2.1
> mu0=2
> sigma=0.25
> n=35
> z=(xbar-mu0)/(sigma/sqrt(n))
> z
[1] 2.366432
> alpha=0.05
> z.alpha=qnorm(1-alpha)
> z.alpha
[1] 1.644854
> 
```

The test statistics 2.3664 is greater than the critical value of 1.6449. Hence, at .05 significance level, we reject the claim that there is at most 2 grams of saturated fat in a cookie.

ONE SAMPL T-TEST

The R function t.test () can be used to perform both one and two sample t-tests on vectors of data. The function contains a variety of options and can be called as follows:

> t.test(x, y = NULL, alternative = c ("two.sided", "less", "greater"), mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95)

PROBLEM:-

An outbreak of Salmonella-related illness was attributed to ice cream produced at a certain factory. Scientists measured the level of Salmonella in 9 randomly sampled batches of ice cream. The levels (in MPN/g) were: 0.593 0.142 0.329 0.691 0.231 0.793 0.519 0.392 0.418 Is there evidence that the mean level of Salmonella in the ice cream is greater than 0.3 MPN/g?

SOLUTION:-

Let be the mean level of Salmonella in all batches of ice cream. Here the hypothesis of interest can be expressed as:
H0: = 0.3
Ha: > 0.3
Hence, we will need to include the options alternative="greater", mu=0.3. Below is the relevant R-code:

TWO-SAMPLE T-TEST
Problem:-
6 subjects were given a drug (treatment group) and an additional 6 subjects a placebo (control group). Their reaction time to a stimulus was measured (in ms). We want to perform a two-sample t-test for comparing the means of the treatment and control groups.
Solution:
Let mu1 be the mean of the population taking medicine and mu2 the mean of the untreated population n. Here the hypothesis of interest can be expressed as:
H0: mu1-mu 2=0
Ha: mu1-mu 2

```
R Console

>  Control = c(91, 87, 99, 77, 88, 91)
>  Treat = c(101, 110, 103, 93, 99, 104)
>  t.test(Control,Treat,alternative="less", var.equal=TRUE)

        Two Sample t-test

data:  Control and Treat
t = -3.4456, df = 10, p-value = 0.003136
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -6.082744
sample estimates:
mean of x mean of y
 88.83333 101.66667

> t.test(Control,Treat,alternative="less")

        Welch Two Sample t-test

data:  Control and Treat
t = -3.4456, df = 9.4797, p-value = 0.003391
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -6.044949
```

```
R Console
        Two Sample t-test

data:  Control and Treat
t = -3.4456, df = 10, p-value = 0.003136
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -6.082744
sample estimates:
mean of x mean of y
 88.83333 101.66667

> t.test(Control,Treat,alternative="less")

        Welch Two Sample t-test

data:  Control and Treat
t = -3.4456, df = 9.4797, p-value = 0.003391
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -6.044949
sample estimates:
mean of x mean of y
 88.83333 101.66667

>
```

R Console

```
> xbar=9900
> muo=10000
> sigma=120
> n=30
> z=(xbar-muo)/(sigma/sqrt(n))
> z
[1] -4.564355
> alpha=0.05
> zalpha=qnorm(1-alpha)
> zalpha
[1] 1.644854
> 
```

<u>**Practical No:07**</u>

<u>Aim:</u>

Import the data from Excel / .CSV and perform the Chi-squared Test.

<u>Solution:</u>

Assume f_{ij} is the observed frequency count of events belonging to both *i*-th category of *x* and *j*-th category of *y*. Also assume e_{ij} to be the corresponding expected count if *x* and *y* are independent. The null hypothesis of the independence assumption is to be rejected if the p-value of the following Chi-squared test statistics is less than a given significance level *α*.

$$\chi^2 = \sum_{i,j} \frac{(f_{ij} - e_{ij})^2}{e_{ij}}$$

R Console

```
> x=matrix(c(24,289,100,9,13,565),nrow=2)
> x
     [,1] [,2] [,3]
[1,]   24  100   13
[2,]  289    9  565
> View(x)
> chisq.test(x)

        Pearson's Chi-squared test

data:  x
X-squared = 635.25, df = 2, p-value < 2.2e-16

> 
```

PRACTICAL NO: 08

AIM:

Using R perform the binomial and normal distribution on the data.

SOLUTION:

NORMAL DISTRIBUTION:

Following is the description of the parameters used in normal distribution:

X=vector of quantiles.

p= vector of probabilities.

N=number of observations. If length (n) > 1, the length is taken to is the number required.

Mean= vector of means.

SD=vector of standard deviations.

dnorm ():-This function gives height of the probability distribution at each point for a given mean and standard deviation.

x=seq(-5,5,by=0.1)

x

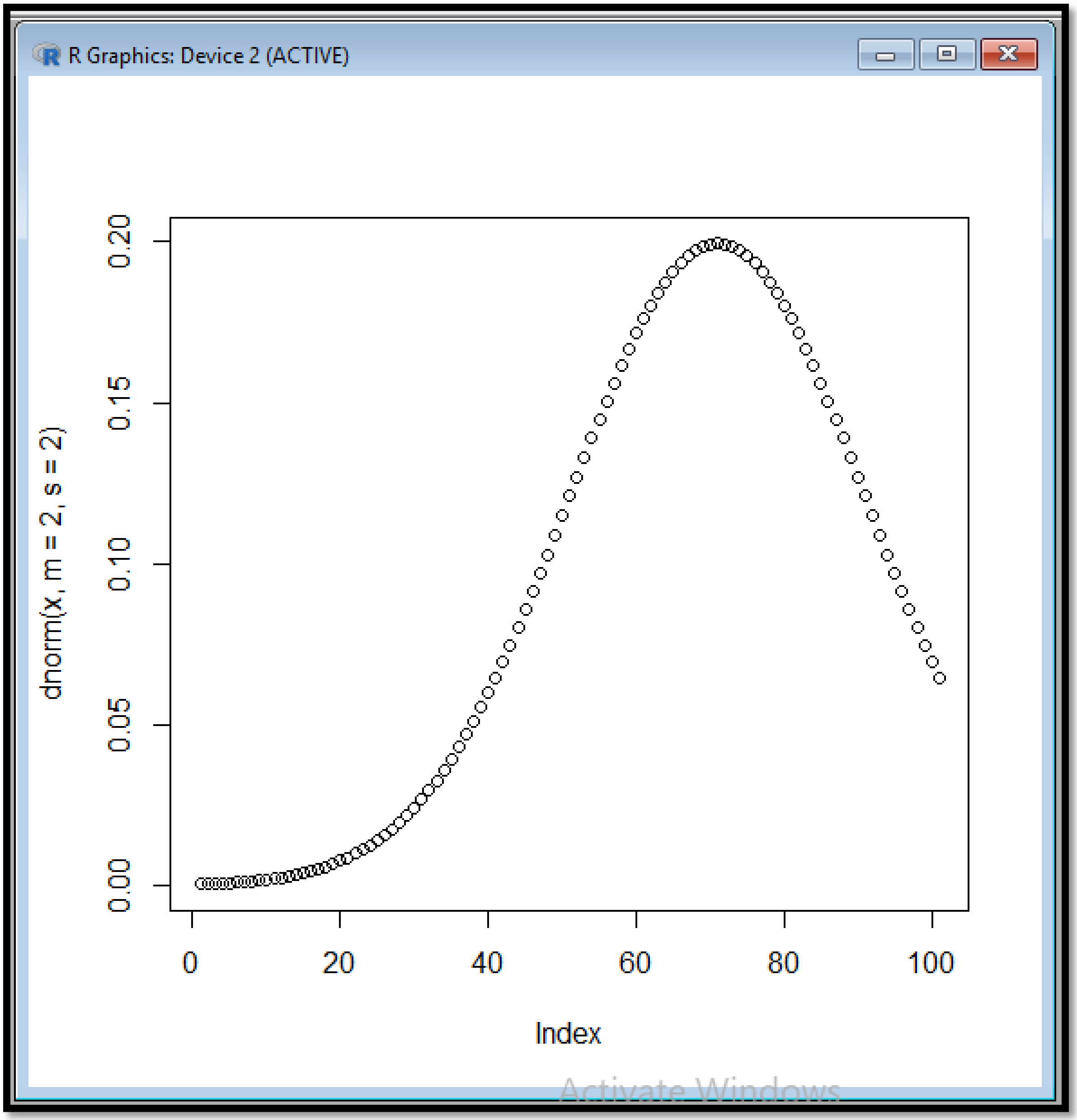
mean(x)

sd(x)

dnorm(x,m=2,s=2)

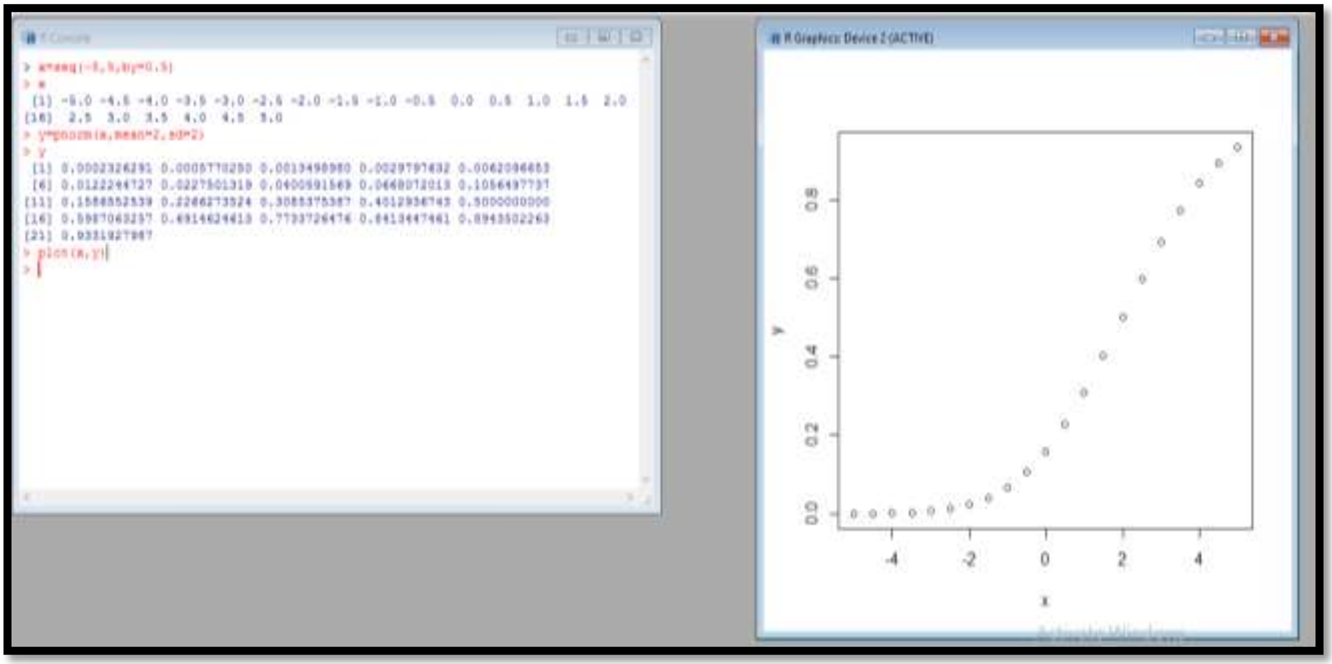
```
> x=seq(-5,5,by=0.1)
> x
  [1] -5.0 -4.9 -4.8 -4.7 -4.6 -4.5 -4.4 -4.3 -4.2 -4.1 -4.0 -3.9 -3.8 -3.7 -3.6
 [16] -3.5 -3.4 -3.3 -3.2 -3.1 -3.0 -2.9 -2.8 -2.7 -2.6 -2.5 -2.4 -2.3 -2.2 -2.1
 [31] -2.0 -1.9 -1.8 -1.7 -1.6 -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6
 [46] -0.5 -0.4 -0.3 -0.2 -0.1  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9
 [61]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4
 [76]  2.5  2.6  2.7  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9
 [91]  4.0  4.1  4.2  4.3  4.4  4.5  4.6  4.7  4.8  4.9  5.0
> mean(x)
[1] 2.639528e-16
> sd(x)
[1] 2.930017
> dnorm(x,m=2,s=2)
  [1] 0.0004363413 0.0005191406 0.0006161096 0.0007293654 0.0008612845
  [6] 0.0010145240 0.0011920441 0.0013971292 0.0016334095 0.0019048810
 [11] 0.0022159242 0.0025713205 0.0029762662 0.0034363833 0.0039577258
 [16] 0.0045467813 0.0052104674 0.0059561218 0.0067914846 0.0077246736
 [21] 0.0087641502 0.0099186772 0.0111972651 0.0126091100 0.0141635189
 [26] 0.0158698259 0.0177372964 0.0197750208 0.0219917980 0.0243960093
 [31] 0.0269954833 0.0297973530 0.0328079074 0.0360324372 0.0394750792
 [36] 0.0431386594 0.0470245387 0.0511324623 0.0554604173 0.0600045003
 [41] 0.0647587978 0.0697152832 0.0748637328 0.0801916637 0.0856842960
 [46] 0.0913245427 0.0970930275 0.1029681344 0.1089260885 0.1149410703
 [51] 0.1209853623 0.1270295282 0.1330426249 0.1389924431 0.1448457764
 [56] 0.1505687161 0.1561269667 0.1614861798 0.1666123014 0.1714719275
 [61] 0.1760326634 0.1802634812 0.1841350702 0.1876201735 0.1906939077
 [66] 0.1933340584 0.1955213470 0.1972396655 0.1984762737 0.1992219570
 [71] 0.1994711402 0.1992219570 0.1984762737 0.1972396655 0.1955213470
 [76] 0.1933340584 0.1906939077 0.1876201735 0.1841350702 0.1802634812
 [81] 0.1760326634 0.1714719275 0.1666123014 0.1614861798 0.1561269667
 [86] 0.1505687161 0.1448457764 0.1389924431 0.1330426249 0.1270295282
 [91] 0.1209853623 0.1149410703 0.1089260885 0.1029681344 0.0970930275
 [96] 0.0913245427 0.0856842960 0.0801916637 0.0748637328 0.0697152832
[101] 0.0647587978
> |
```

plot(dnorm(x,m=2,s=2))



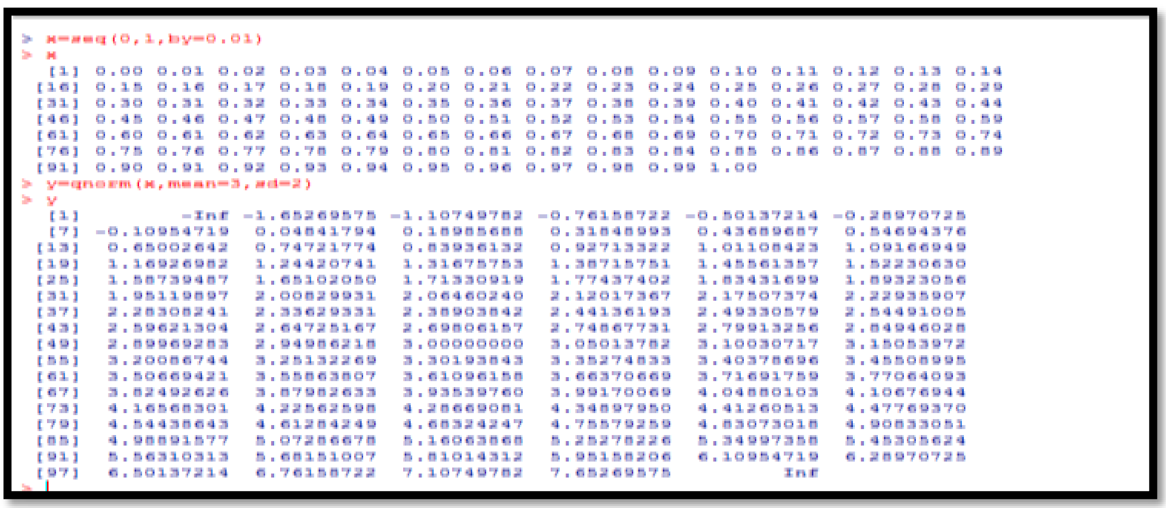
Pnorm() :- This function gives the probability of a normally distributed random number to be less the value of a given number.

Pnorm(x,mean=2,sd=2)



<u>qnorm():-</u> This function takes the probability value and gives a number whose cumulative value matches the probability value.

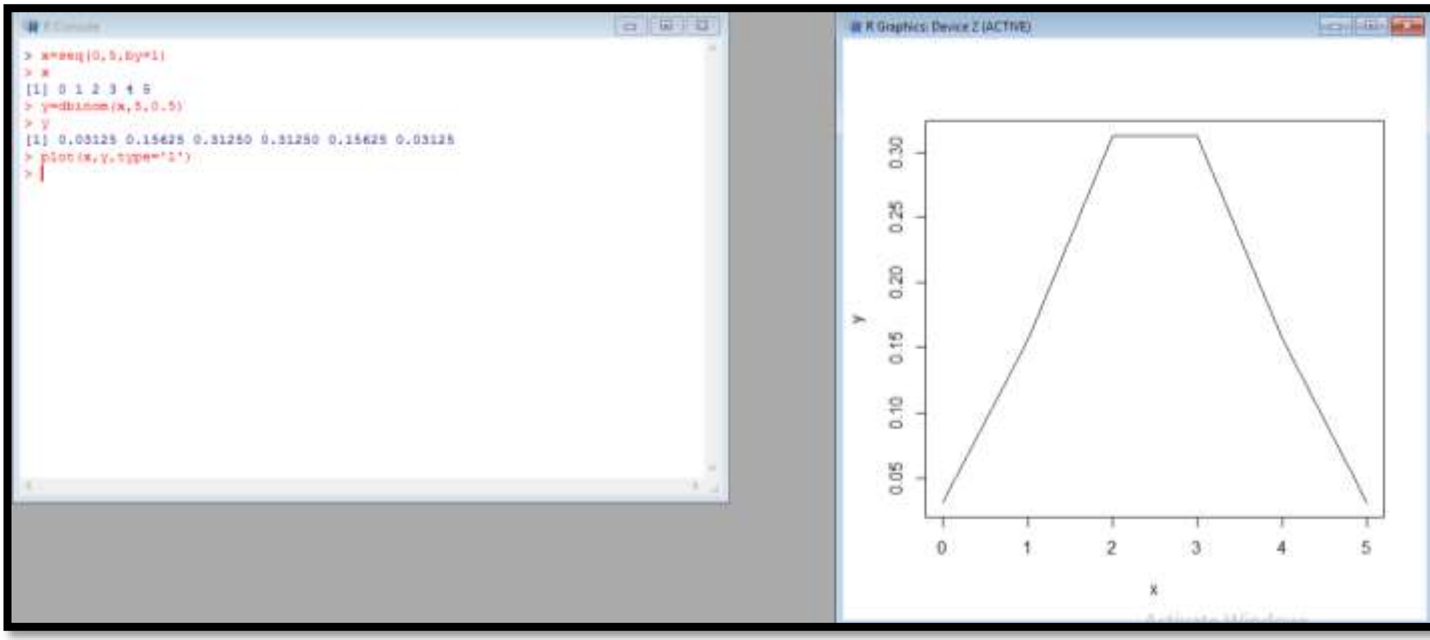
qnorm(x,mean=3,sd=2)



```
> x=seq(0,1,by=0.01)
> x
  [1] 0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10 0.11 0.12 0.13 0.14
 [16] 0.15 0.16 0.17 0.18 0.19 0.20 0.21 0.22 0.23 0.24 0.25 0.26 0.27 0.28 0.29
 [31] 0.30 0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.40 0.41 0.42 0.43 0.44
 [46] 0.45 0.46 0.47 0.48 0.49 0.50 0.51 0.52 0.53 0.54 0.55 0.56 0.57 0.58 0.59
 [61] 0.60 0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.70 0.71 0.72 0.73 0.74
 [76] 0.75 0.76 0.77 0.78 0.79 0.80 0.81 0.82 0.83 0.84 0.85 0.86 0.87 0.88 0.89
 [91] 0.90 0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98 0.99 1.00
> y=qnorm(x,mean=3,sd=2)
> y
  [1]        -Inf -1.65269575 -1.10749782 -0.76158722 -0.50137214 -0.28970725
  [7] -0.10954719  0.04841794  0.18985688  0.31848993  0.43689687  0.54694376
 [13]  0.65002642  0.74721774  0.83936132  0.92713322  1.01108423  1.09166949
 [19]  1.16926982  1.24420741  1.31675753  1.38715751  1.45561357  1.52230630
 [25]  1.58739487  1.65102050  1.71330919  1.77437402  1.83431699  1.89323056
 [31]  1.95119897  2.00829931  2.06460240  2.12017367  2.17507374  2.22935907
 [37]  2.28308241  2.33629331  2.38903842  2.44136193  2.49330579  2.54491005
 [43]  2.59621304  2.64725167  2.69806157  2.74867731  2.79913256  2.84946028
 [49]  2.89969283  2.94986218  3.00000000  3.05013782  3.10030717  3.15053972
 [55]  3.20086744  3.25132269  3.30193843  3.35274833  3.40378696  3.45508995
 [61]  3.50669421  3.55863807  3.61096158  3.66370669  3.71691759  3.77064093
 [67]  3.82492626  3.87982633  3.93539760  3.99170069  4.04880103  4.10676944
 [73]  4.16568301  4.22562598  4.28669081  4.34897950  4.41260513  4.47769370
 [79]  4.54438643  4.61284249  4.68324247  4.75579259  4.83073018  4.90833051
 [85]  4.98891577  5.07286678  5.16063868  5.25278226  5.34997358  5.45305624
 [91]  5.56310313  5.68151007  5.81014312  5.95158206  6.10954719  6.28970725
 [97]  6.50137214  6.76158722  7.10749782  7.65269575         Inf
> 
```

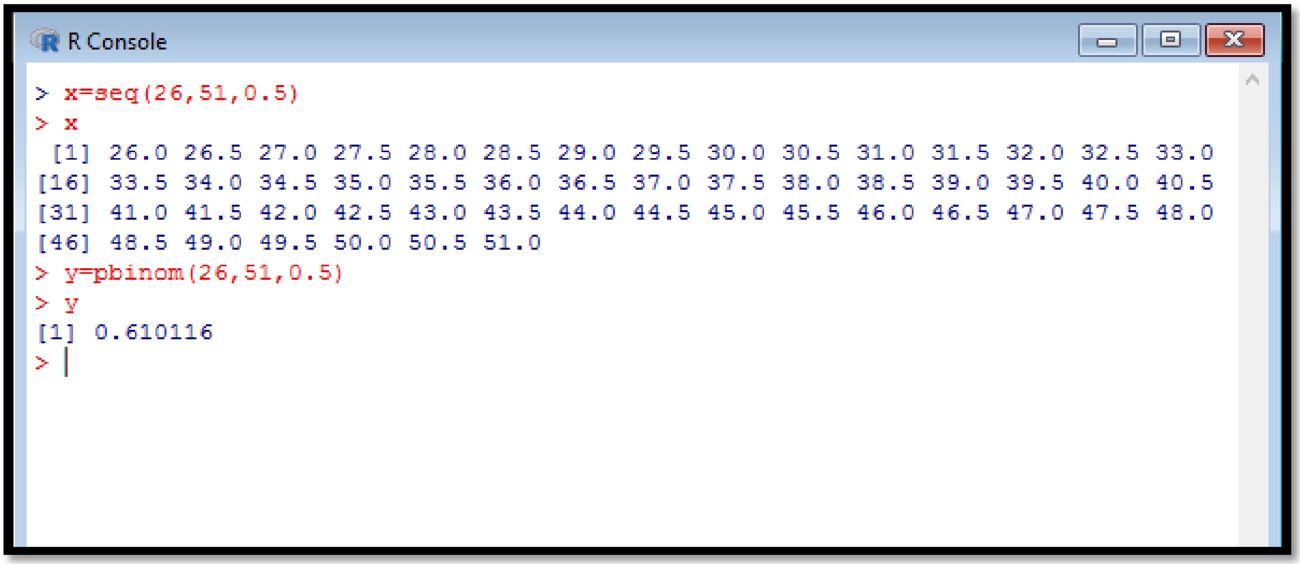
<u>BINOMIAL DISTRIBUTION</u>
It deals with finding the probability of success of an experiment.

Following is the description of the parameters used in binomial distribution:
X=vector of numbers.
p=vector of probabilities.
N=number of observations.
Size = number of trials.
Prob =Probability of success of each trial.
dbinom() :- This function gives the probability density distribution at each point.
x=seq(0,5,by=1)
x
y=dbinom(x,size,prob)
y
plot(x,y,type='l')



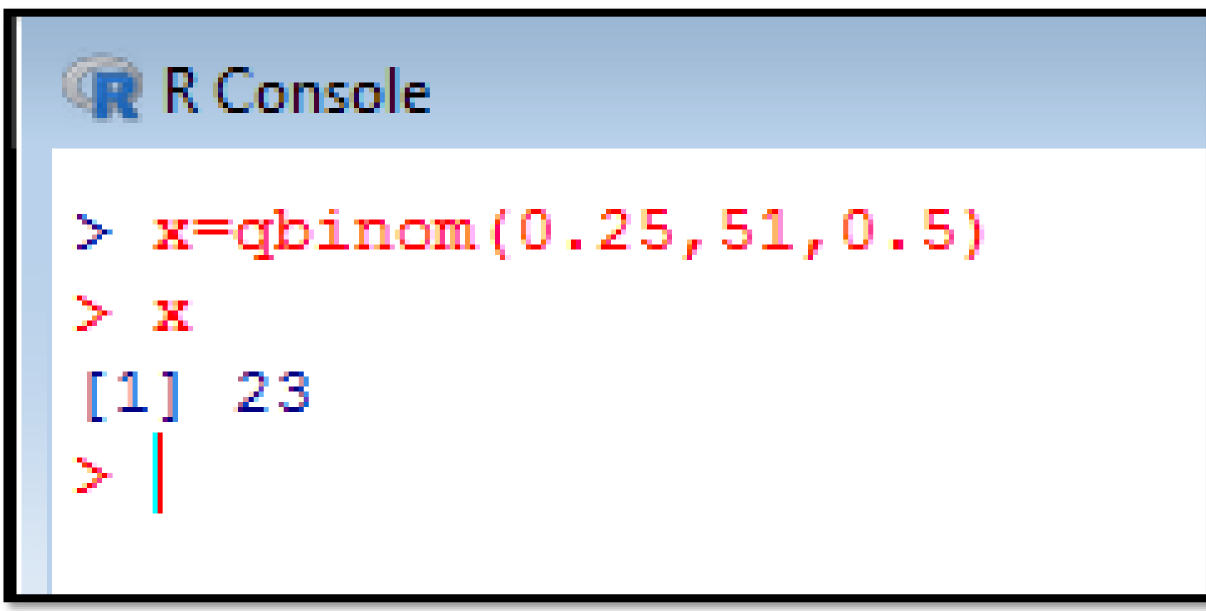
Pbinom() :- This function gives the cumulative probability of an event.

Pbinom(x,size,prob)



<u>Qbinom()</u> :- This function takes the probability value and gives a no. whose cumulative value matches the probability value.

qbinom(x,size,prob)



<u>rbinom() :-</u> This function generates number of random values of given probability from a given sample.

rbinom(x,size,prob)

R Console

```
> x=rbinom(10,150,0.4)
> x
 [1] 71 61 71 73 59 56 59 65 62 70
>
```

PRACTICAL NO. 09

AIM:-

Perform the Linear Regression using R

SOLUTION:-

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

The general mathematical equation for a linear regression is:

y = ax+b

Following is the description of the parameters used:

Y is the response variable.

X is the predictor variable.

A and **b** are constants which are called the coefficients.

Steps to Establish a Regression

1. Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
2. Create a relationship model using the **lm ()** functions in R.
3. Find the coefficients from the model created and create the mathematical equation using these.
4. Get a summary of the relationship model to know the average error in prediction. Also called **residuals**.
5. To predict the weight of new persons, use the **predict ()** function in R.

lm() Function

This function creates the relationship model between the predictor and the response variable.

Syntax

The basic syntax for **lm()** function in linear regression is:

```
lm(formula,data)
```

Following is the description of the parameters used:

- **formula** is a symbol presenting the relation between x and y.
- **data** is the vector on which the formula will be applied.

```
x=c(151,174,138,186,128,136,179,163,152,131)
y=c(63,81,56,91,47,57,76,72,62,48)
relation=lm(y~x)
relation
```

```
R Console
> x=c(151,174,138,186,128,136,179,163,152,131.)
> y=c(63,81,56,91,47,57,76,72,62,48)
> relation=lm(y~x)
> relation

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   -38.4551       0.6746

>
```

<u>predict() Function</u>

<u>Syntax</u>

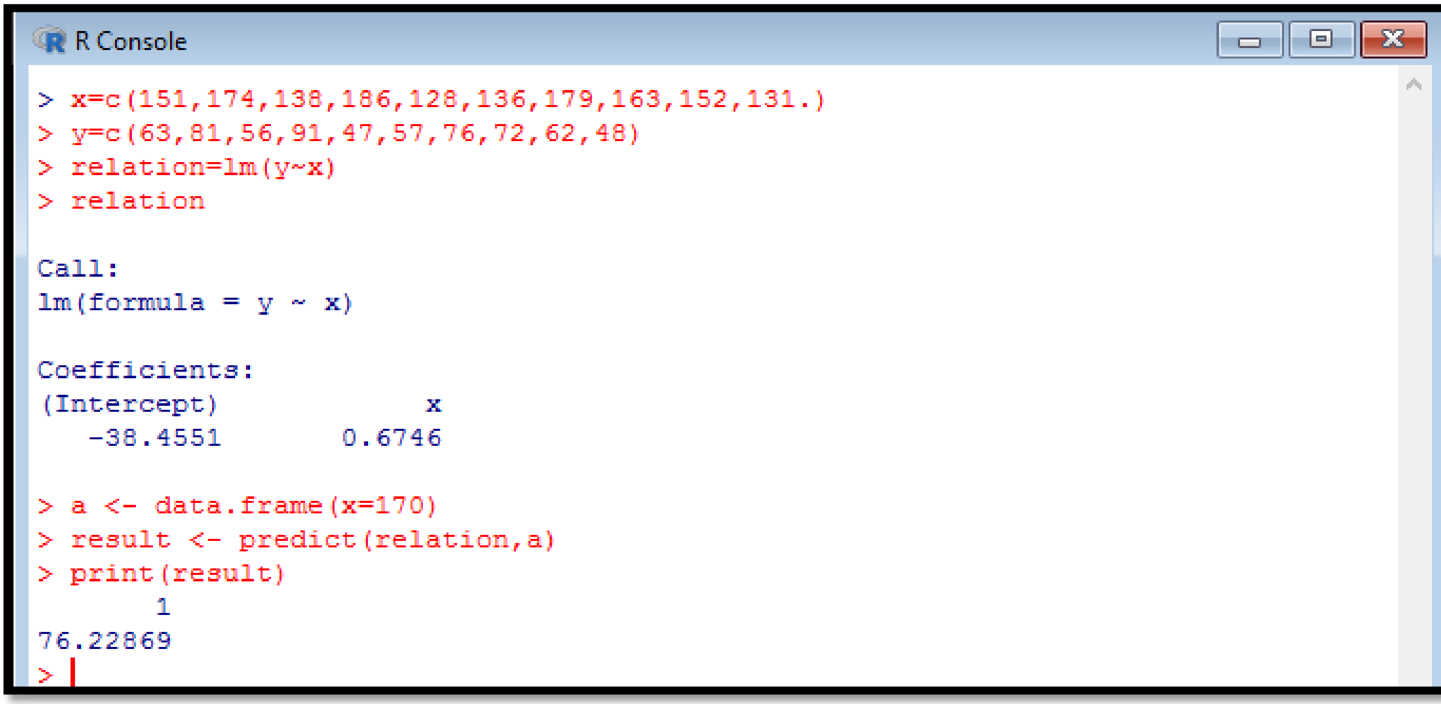
The basic syntax for predict () in linear regression is:

predict(object, newdata)

Following is the description of the parameters used:

- **object** is the formula which is already created using the lm() function.
- **newdata** is the vector containing the new value for predictor variable.

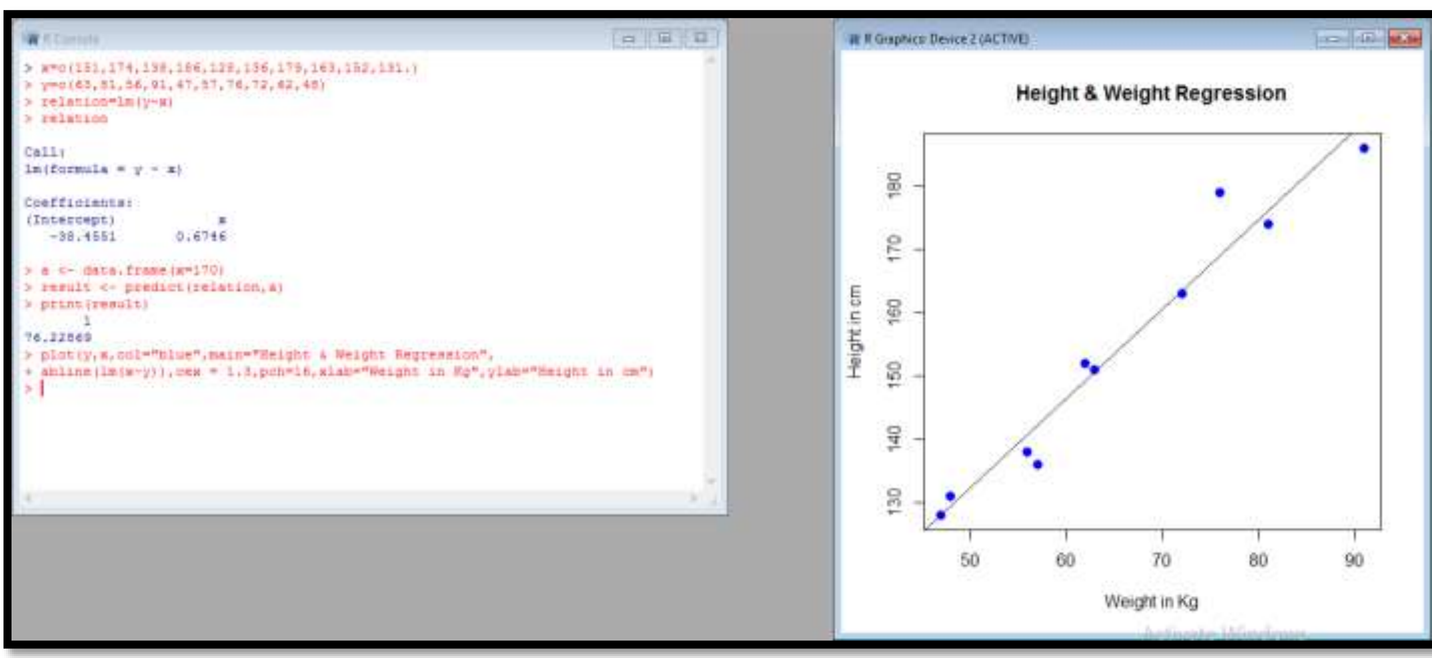
```
x=c(151,174,138,186,128,136,179,163,152,131)
y=c(63,81,56,91,47,57,76,72,62,48)
relation=lm(y~x)
relation
a=data.frame(x=170)
result=predict(relation,a)
result
```



```
> x=c(151,174,138,186,128,136,179,163,152,131.)
> y=c(63,81,56,91,47,57,76,72,62,48)
> relation=lm(y~x)
> relation

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   -38.4551       0.6746

> a <- data.frame(x=170)
> result <- predict(relation,a)
> print(result)
       1
76.22869
>
```

Visualize the Regression Graphically

plot(y,x,col=”blue”,main=”Height & Weight Regression”,abline(lm(x~y)),cex=1.3,pch=16,xlab=”Weight in Kg”,ylab=”Height in cm”)



PRACTICAL NO.10

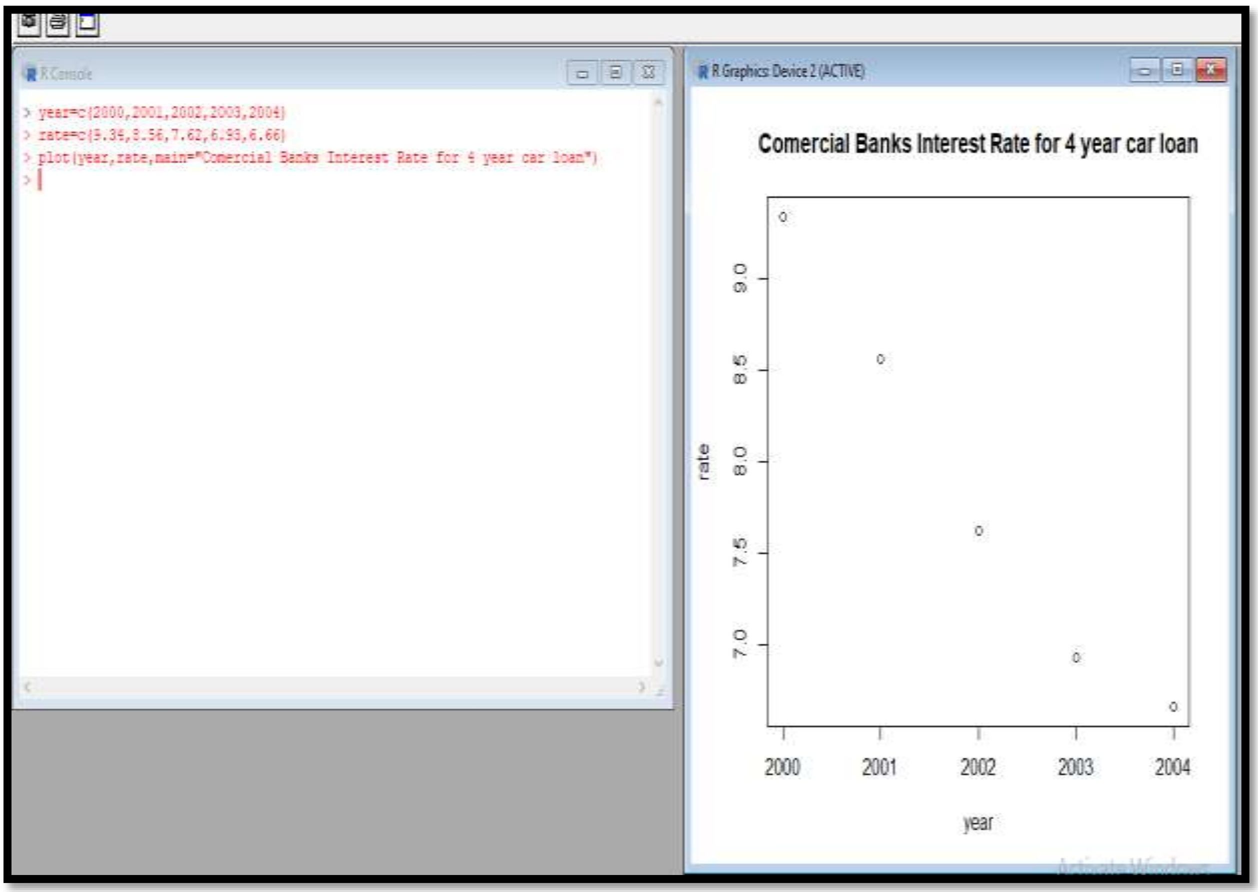
AIM :-

Compute the linear Least Square Regression.

SOLUTION:-

Five pairs of numbers, enter them in manually. Each of the five pairs consists of a year and the mean interest rate:

year=c(2000,2001,2002,2003,2004)
rate=c(9.34,8.56,7.62,6.93,6.66)
plot(year,rate,main="Commercial Banks Interest Rate for i year car loan")



The way that this relationship is defined in the lm command is that you write the vector containing the response variable, a tilde ("~"), and a vector containing the explanatory variable:

cor(year,rate)
fit=lm(rate~year)
fit
attribute(fit)
fit$coefficients[i]

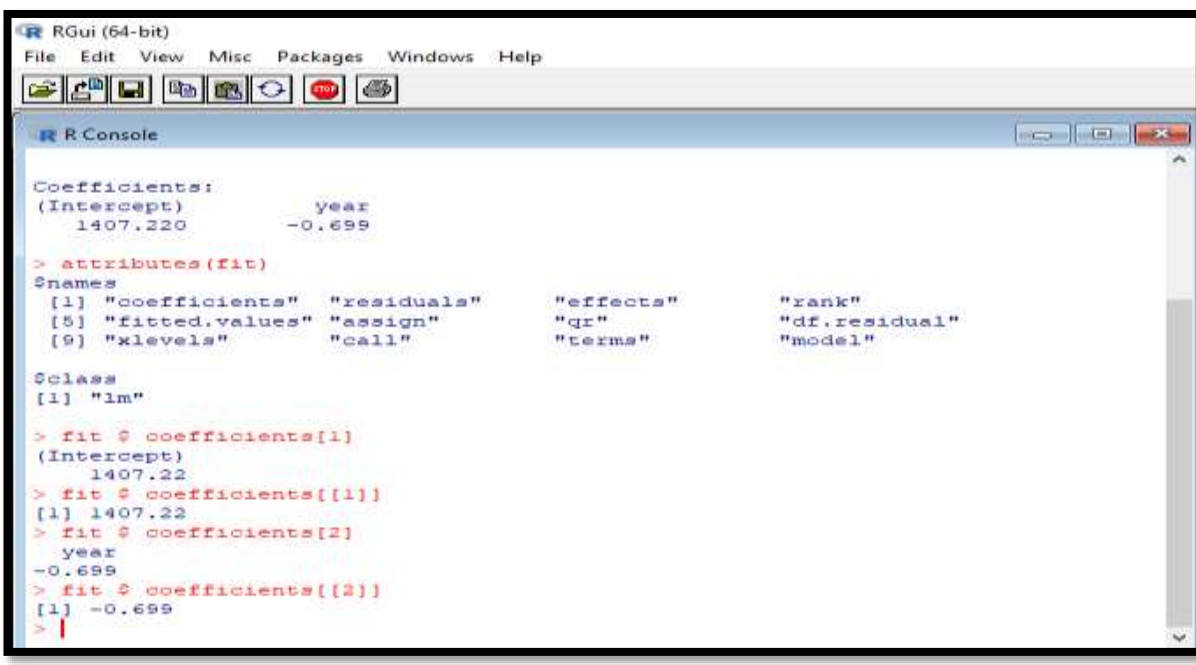
If you would like to know what else is stored in the variable you can use the attributes command:

fit$coefficient[1]

fit$ coefficient[[1]]

fit$coefficient[2]

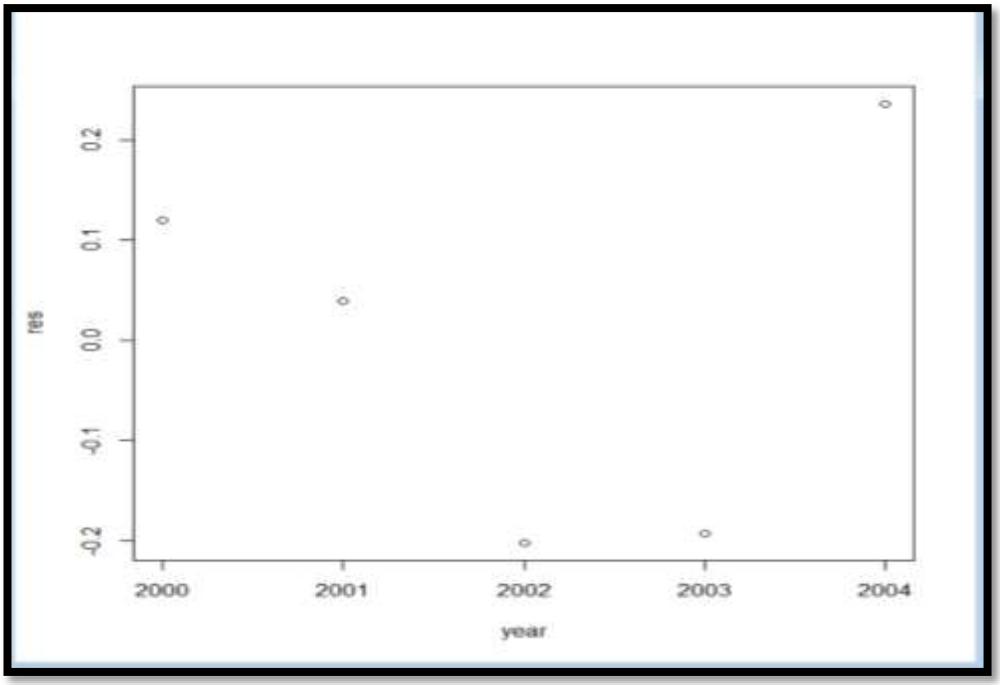
fit$ coefficient[[2]]



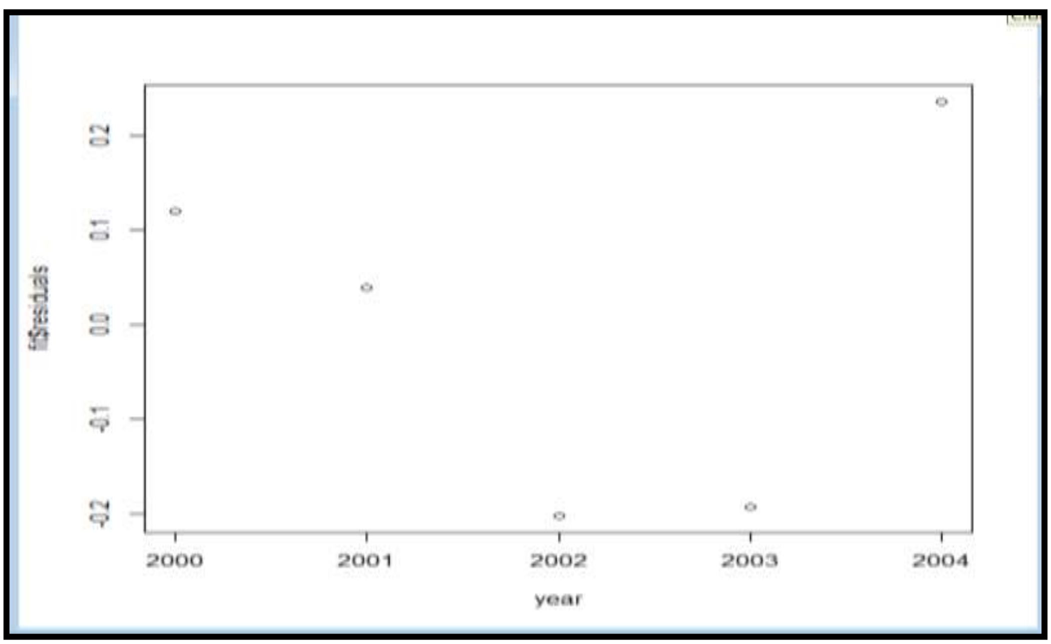
One of the things you should notice is the coefficients variable within fit. You can print out the y-intercept and slope by accessing this part of the variable:

fit$ coefficient[[2]]*2015+ fit$coefficient[[1]]
res=rate-(fit$ coefficient[[2]]*year + fit$coefficient[[1]])
res
plot(year,res)

```
names
[1] "coefficients"  "residuals"     "effects"       "rank"
[5] "fitted.values" "assign"        "qr"            "df.residual"
[9] "xlevels"       "call"          "terms"         "model"

class
1] "lm"

 fit $ coefficients[1]
Intercept)
   1407.22
 fit $ coefficients[[1]]
1] 1407.22
 fit $ coefficients[2]
 year
0.699
 fit $ coefficients[[2]]
1] -0.699
 fit $ coefficients[[2]]*2015+fit $ coefficients[[1]]
1] -1.265
 res=rate-(fit $ coefficient[[2]]*year+fit $ coefficient[[1]])
 res
1]  0.120  0.039 -0.202 -0.193  0.236
 plot(year,res)
```



That is a bit messy, but fortunately there are easier ways to get the residuals. Two other ways are shown below:

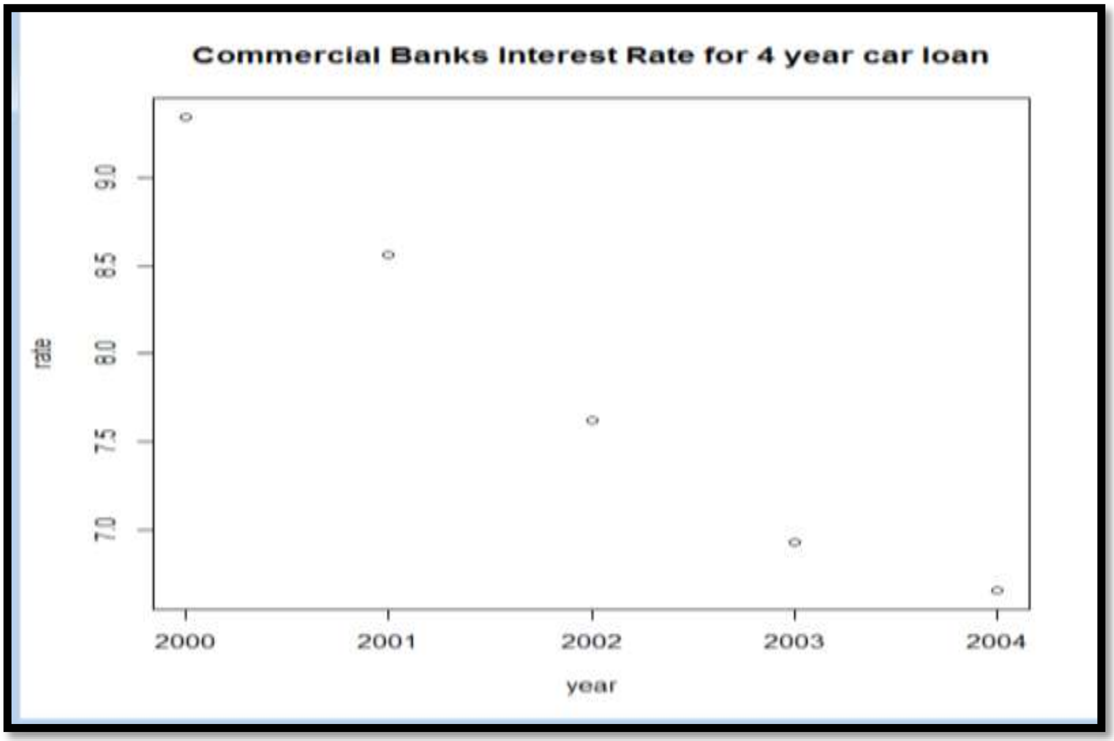


residuals(fit)
fit$residual
plot(year,fit$residuals)

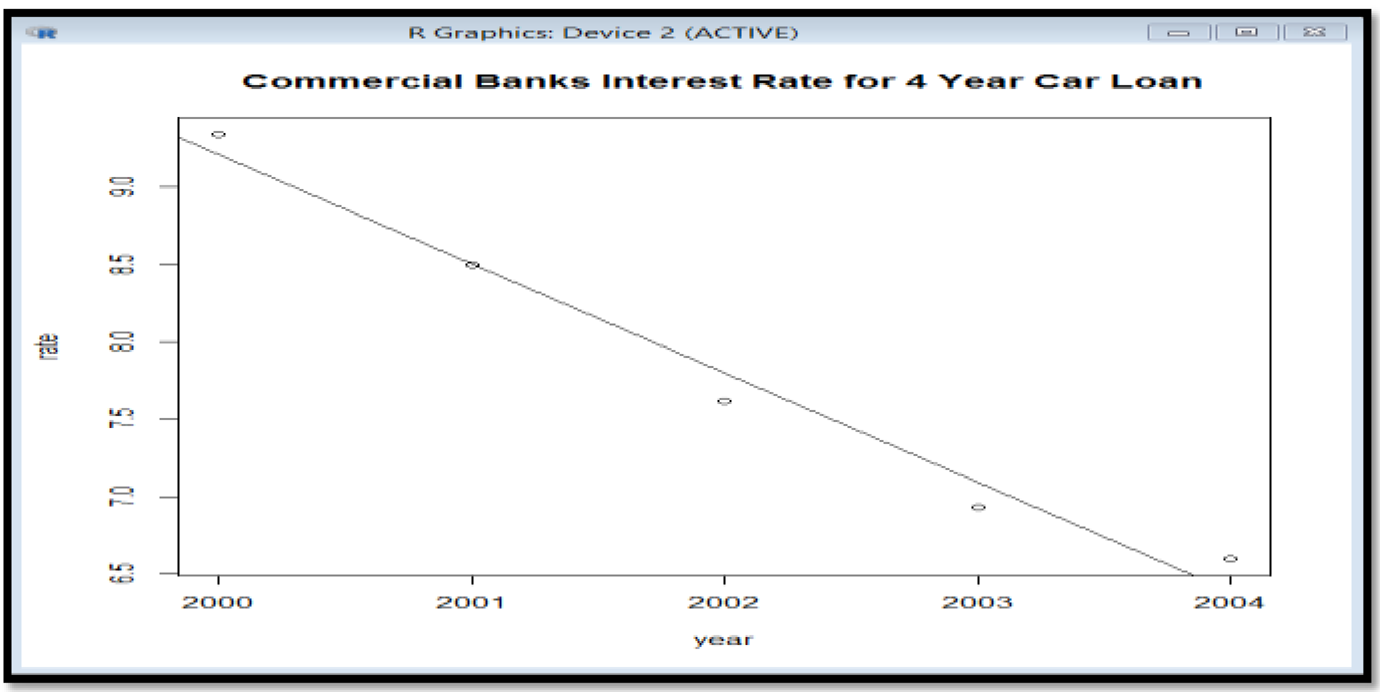
```
> residuals(fit)
     1      2      3      4      5
 0.120  0.039 -0.202 -0.193  0.236
> fit $ residuals
     1      2      3      4      5
 0.120  0.039 -0.202 -0.193  0.236
> plot(year,fit $ residuals)
>
```

If we want to plot the regression line on the same plot as your scatter plot you can use the abline function along with your variable fit:

```
> plot(year,fit $ residuals)
> plot(year,rate,main="Commercial Banks Interest Rate for 4 year car loan")
>
```



abline(fit)



Results of an F-test by asking R for a summary of the fit variable:

summary(fit)

```
> summary(fit)

Call:
lm(formula = rate ~ year)

Residuals:
     1      2      3      4      5 
 0.120  0.039 -0.202 -0.193  0.236 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)   
(Intercept) 1407.22000  141.40701   9.952  0.00216 **
year          -0.69900    0.07063  -9.896  0.00219 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2234 on 3 degrees of freedom
Multiple R-squared:  0.9703,	Adjusted R-squared:  0.9604 
F-statistic: 97.94 on 1 and 3 DF,  p-value: 0.002194

> 
```