Practical 1 – Basics of R

AIM : To study R language

R PROGRAMMING

R Data Types

The R programming language the variables are not declared as some data types.

The variables are assigned with R object and data types of R object becomes data type of variables

There are many types of R objects

Vectors

List

Matrices

Array

Factors

Data frames

PRACTICAL NO. 1

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

The simplest of R object is vectors and there are six datatypes of these atomic vectors.

The R object are built upon this atomic vectors.

Types of R vectors

Logical

Numeric

Integer

Complex

Character

Raw

Class

- Class gives data type of the vector. R possesses a simple generic function mechanism which can be used for an object-oriented style of programming.
- Method dispatch takes place based on the class of the first argument to the generic function.

Six data-types of atomic vectors are:

1) logical: Create or test for objects of type "logical", and the basic logical constants.

```
> z= TRUE
> class(z)
[1] "logical"
```

2) numeric: Creates or coerces objects of type "numeric". is.numeric is a more general test of an . object being interpretable as numbers.

```
> a=10
> class(a)
[1] "numeric"
```

3) integer: Creates or tests for objects of type "integer".

```
> b=12L
> class(b)
[1] "integer"
```

4) complex: Basic functions which support complex arithmetic in R, in addition to the arithmetic operators +, -, *, /, and ^.

```
> a= 3+2i
> class(a)
[1] "complex"
```

5)character: Create or test for objects of type "character".

```
> name="revati"
> class(name)
[1] "character"
```

charToRaw()

- Conversion and manipulation of objects of type "raw".

6) raw: Creates or tests for objects of type "raw".

```
> x="hello"
> charToRaw(x)
[1] 68 65 6c 6c 6f
> r=charToRaw(x)
> class(r)
[1] "raw"
```

Sequence Generation

seq()

- Generate regular sequences.
- seq is a standard generic with a default method.
- seq.int is a primitive which can be much faster but has a few restrictions.
- seq_along and seq_len are very fast primitives for two common cases.

```
> seq()
[1] 1
> q=seq(2,36,by=2)
> q
 [1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
```

C function

c()

Description

- This is a generic function which combines its arguments.
- The default method combines its arguments to form a vector.
- All arguments are coerced to a common type which is the type of the returned value, and all attributes except names are removed.

```
> x=c(5,3,8,23,56)
> x
[1]  5  3  8 23 56
```

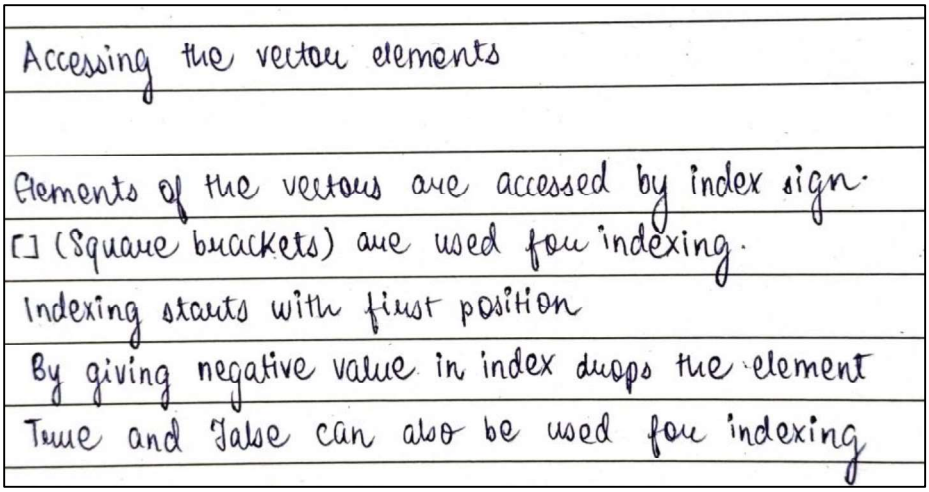
To create a sequence

Colon

:

- It is used to create a regular sequence

```
> z=1:20
> z
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
> z=5:25
> z
 [1]  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
```




```
> weekdays=c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
> weekdays
[1] "Sunday"    "Monday"    "Tuesday"   "Wednesday" "Thursday"  "Friday"    "Saturday"
```

Accessing element at specific position:

```
> weekdays[4]
[1] "Wednesday"
```

Accessing element at different position:

```
> weekdays[c(7,1)]
[1] "Saturday" "Sunday"
```

To drop elements of specific index:

```
> weekdays[c(-2,-4)]
[1] "Sunday"   "Tuesday"  "Thursday" "Friday"   "Saturday"

> weekdays [c(FALSE,TRUE,FALSE,FALSE,FALSE,TRUE,TRUE)]
[1] "Monday"   "Friday"   "Saturday"
```

Vector element re-cycling

```
> v1=c(2,4,7)
> v2=c(3,5,2,2,4,54,2,24)
> v1+v2
[1]  5  9  9  4  8 61  4 28
Warning message:
In v1 + v2 :
  longer object length is not a multiple of shorter object length
> v3=c(5,45,22,24,13,134,11,6,44,4,1,67,104)
> v3
 [1]   5  45  22  24  13 134  11   6  44   4   1  67 104
```

Sorting

sort()

Description

- Sort (or order) a vector or factor (partially) into ascending or descending order.
- For ordering along more than one variable, e.g., for sorting data frames, see sort

Sorting Number

- In Ascending Order

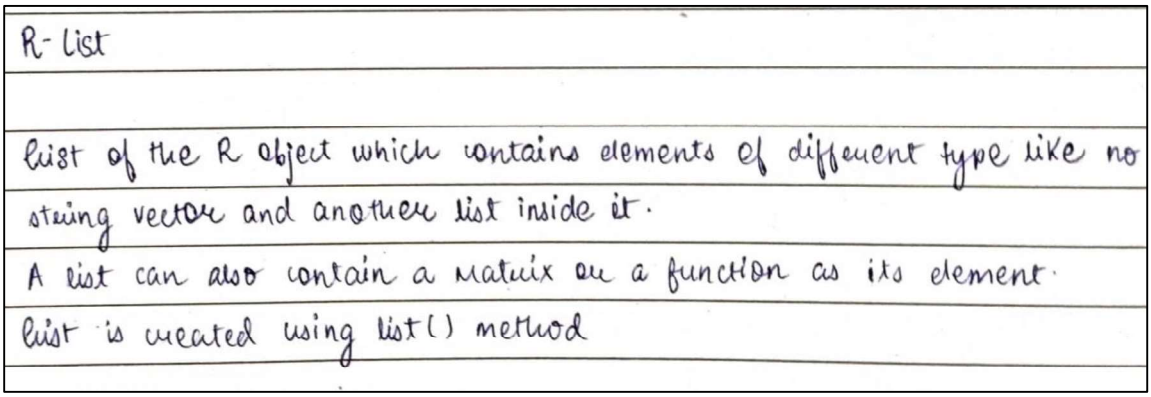
```
> sortedv3=sort(v3)
> sortedv3
 [1]   1   4   5   6  11  13  22  24  44  45  67 104 134
```

- In Descending Order

```
> sortedv3d=sort(v3,decreasing=TRUE)
> sortedv3d
 [1] 134 104  67  45  44  24  22  13  11   6   5   4   1
```

Sorting characters

```
> colors=c("violet","indigo","blue","green","yellow","orange","red")
> colors
[1] "violet" "indigo" "blue"   "green"  "yellow" "orange" "red"
> sortedcolors=sort(colors)
> sortedcolors
[1] "blue"   "green"  "indigo" "orange" "red"    "violet" "yellow"
```


R- List

List of the R object which contains elements of different type like no string vector and another list inside it.
A list can also contain a matrix or a function as its element.
List is created using list() method

List

Description

- Functions to construct, coerce and check for both kinds of R lists.

Matrix

Description

- Matrix creates a matrix from the given set of values.
- As.matrix attempts to turn its argument into a matrix.

```
> list1=list(c("jan","feb","march"),matrix(c(2,4,5,6),nrow=2),list("green",15.5))
> list1
[[1]]
[1] "jan"   "feb"   "march"


[[2]]
     [,1] [,2]
[1,]    2    5
[2,]    4    6


[[3]]
[[3]][[1]]
[1] "green"


[[3]][[2]]
[1] 15.5
```

Accessing list elements by index.

```
> list1[[3]][2]
[[1]]
[1] 15.5

> list1[3]
[[1]]
[[1]][[1]]
[1] "green"

[[1]][[2]]
[1] 15.5
```

Name the list

- Function to get or set the names of an object.

```
> names(list1)=c("lst_quarter","matrix","inner_list")
> list1
$lst_quarter
[1] "jan"   "feb"   "march"

$matrix
     [,1] [,2]
[1,]    2    5
[2,]    4    6
```

```
$inner_list
$inner_list[[1]]
[1] "green"


$inner_list[[2]]
[1] 15.5
```

Accessing list elements

- Access names of the list by using "$ "

```
> list1$matrix[1,2]
[1] 5
> list1$inner_list[2]
[[1]]
[1] 15.5
> list1$inner_list[1]
[[1]]
[1] "green"
```

Creating l1 and l2 to perform Merging

```
> l1=list("red","yellow","green")
> l2=list(10,8,7)
> l1
[[1]]
[1] "red"
```

```
[[2]]
[1] "yellow"

[[3]]
[1] "green"

> l2
[[1]]
[1] 10

[[2]]
[1] 8

[[3]]
[1] 7
```

Merging List l1 and l2

```
> mergedlist=c(l1,l2)
> mergedlist
[[1]]
[1] "red"

[[2]]
[1] "yellow"
```

```
[[3]]
[1] "green"


[[4]]
[1] 10


[[5]]
[1] 8


[[6]]
[1] 7
```

Unlist a list.

- Given a list structure x, unlist simplifies it to produce a vector which contains all the atomic components which occur in x.

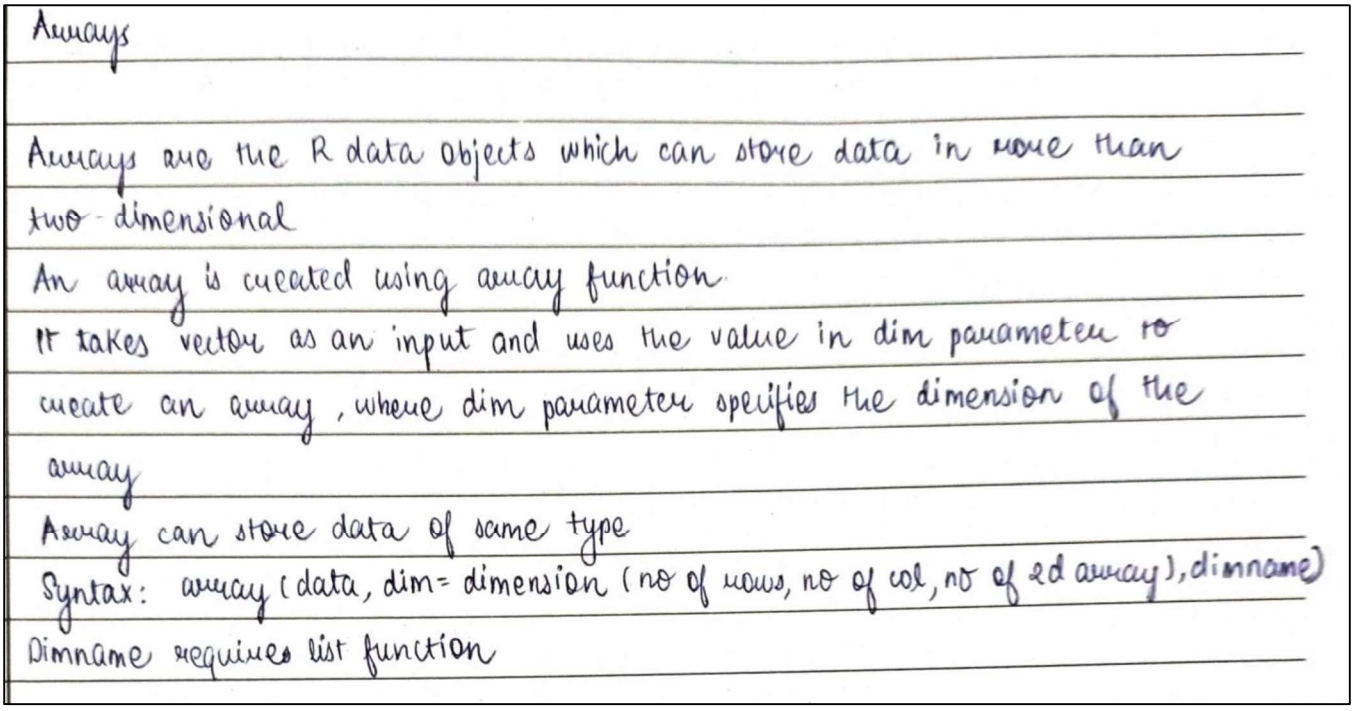
```
Create a list l3:
> l3=list(6,9,12)
> l3
[[1]]
[1] 6


[[2]]
[1] 9


[[3]]
[1] 12
```

Unlist l3:

```
> unlist (l3)
[1]  6  9 12
> v5=unlist (l3)
> v5
[1]  6  9 12
```



Array

Description

- Creates or tests for arrays.

Arguments

data:	a vector (including a list or expression vector) giving data to fill the array. Non-atomic classed objects are coerced by as.vector.
dim:	the dim attribute for the array to be created, that is an integer vector of length one or more giving the maximal indices in each dimension.

dimnames : Either NULL or the names for the dimensions.
This must a list (or it will be ignored) with one component for each dimension, either NULL or a character vector of the length given by dim for that dimension. The list can be named, and the list names will be used as names for the dimensions. If the list is shorter than the number of dimensions, it is extended by NULLs to the length required.

Usage

- array(data = NA, dim = length(data), dimnames = NULL)

Creating an array.

```
> v1=c(3,7,5)
> v2=c(4,2,6,9,8,1)
> array1=array(c(v1,v2),dim=c(3,3,2))
> array1
, , 1

     [,1] [,2] [,3]
[1,]    3    4    9
[2,]    7    2    8
[3,]    5    6    1

, , 2
     [,1] [,2] [,3]
[1,]    3    4    9
[2,]    7    2    8
[3,]    5    6    1
```

Naming the rows, column and matrix

```
> rnames=c("r1","r2","r3")
> cnames=c("c1","c2","c3")
> mnames=c("m1","m2")
> array1=array(c(v1,v2),dim=c(3,3,2),dimnames=list(rnames,cnames,mnames))
> array1
, , m1

   c1 c2 c3
r1  3  4  9
r2  7  2  8
r3  5  6  1

, , m2

   c1 c2 c3
r1  3  4  9
r2  7  2  8
r3  5  6  1
```

Function factor is used to encode a vector as a factor.

factor()

Description

- The function factor is used to encode a vector as a factor (the terms ‘category’ and ‘enumerated type’ are also used for factors).
- If argument ordered is TRUE, the factor levels are assumed to be ordered. For compatibility with S there is also a function ordered.

```
>data=c("north","south","east","north","south","east","east","east","north","south","north","south","north")

> data

 [1] "north" "south" "east"  "north" "south" "east"  "east"  "east"  "north" "south" "north" "south" "north"

> factor(data)

 [1] north south east  north south east  east  east  north south north south north

Levels: east north south
```

Data Frames

Data frame is a table or 2D array like structure in which each column contain value one variable of each row contain one set of values from each column.

Characteristics of Data Frames are:

Column name should not be empty

Row name should be unique

The data stored in a data frame can be of numeric factors or character type

Each column should contain same no of data frames items.

Data frames is created with data.frame function

- Creating a Data Frame/Table.

```
>empdata=data.frame(empID=c(101:105),empName=c("Revati","Shruti","Vidhi","Ananya",
"Sakshi"),Salary=c(18000,14000,20000,23000,25000),startDate=as.Date(c("2018-03-
25","2019-02-24","2020-5-21","2018-03-25","2018-03-2")),stringsAsFactors=FALSE)

> empdata

  empID empName Salary  startDate

1   101  Revati  18000 2018-03-25

2   102  Shruti  14000 2019-02-24

3   103   Vidhi  20000  2020-5-21

4   104  Ananya  23000 2018-03-25

5   105  Sakshi  25000  2018-03-2
```

str()

Description

- Compactly display the internal structure of an R object, a diagnostic function and an alternative to summary (and to some extent, dput).
- Ideally, only one line for each ‘basic’ structure is displayed.
- It is especially well suited to compactly display the (abbreviated) contents of (possibly nested) lists.
- The idea is to give reasonable output for any R object.
- It calls args for (non-primitive) function objects.

```
> str(empdata)

'data.frame':   5 obs. of  4 variables:

 $ empID    : int  101 102 103 104 105

 $ empName  : chr  "Revati" "Shruti" "Vidhi" "Ananya" ...

 $ Salary   : num  18000 14000 20000 23000 25000

 $ startDate: chr  "2018-03-25" "2019-02-24" "2020-5-21" "2018-03-25" ...
```

summary()

Description

- summary is a generic function used to produce result summaries of the results of various model fitting functions.
- The function invokes particular methods which depend on the class of the first argument.

```
> summary(empdata)
    empID       empName             Salary       startDate
 Min.   :101   Length:5           Min.   :14000   Length:5
 1st Qu.:102   Class :character   1st Qu.:18000   Class :character
 Median :103   Mode  :character   Median :20000   Mode  :character
 Mean   :103                      Mean   :20000
 3rd Qu.:104                      3rd Qu.:23000
 Max.   :105                      Max.   :25000
```

- Extracting data from Data Table (Data Frame)

data.frame()

Description

- The function data.frame() creates data frames, tightly coupled collections of variables which share many of the properties of matrices and of lists, used as the fundamental data structure by most of R's modeling software.

```
> result=data.frame(empdata$empName,empdata$Salary)
> result
  empdata.empName empdata.Salary
1          Revati         18000
2          Shruti         14000
3           Vidhi         20000
4         Ananya           23000
5          Sakshi         25000
```

```
> result2=empdata[3:5,]
> result2
  empID empName Salary  startDate
3   103   Vidhi  20000 2020-05-21
4   104  Ananya  23000 2018-03-25
5   105  Sakshi  25000 2018-03-02


> result3=empdata[c(2,3),c(2,4)]
> result3
  empName  startDate
2  Shruti 2019-02-24
3   Vidhi 2020-05-21
```

- Expanding data frames

Adding columns to the data frame

```
> empdata$dept=c("IT","Finance","Testing","Development","HR")
> empdata
  empID empName Salary  startDate        dept
1   101  Revati  18000 2018-03-25          IT
2   102  Shruti  14000 2019-02-24     Finance
3   103   Vidhi  20000 2020-05-21     Testing
4   104  Ananya  23000 2018-03-25 Development
5   105  Sakshi  25000 2018-03-02          HR
```

- Creating data frame and combining it.

```
> empnewdata=data.frame(empID=c(106:108),empName=c("Raj","Ved","Rishi"),
Salary=c(12000,19000,24000), startDate=as.Date(c("2020-01-5","2018-01-2","2019-5-21")),
dept=c("HR","IT","Development"),stringsAsFactors=FALSE)

> empnewdata

  empID empName Salary  startDate        dept

1   106     Raj  12000 2020-01-05          HR

2   107     Ved  19000 2018-01-02          IT

3   108   Rishi  24000 2019-05-21 Development
```

rbind()

Description

- Take a sequence of vector, matrix or data-frame arguments and combine by *c*olumns or *r*ows, respectively. These are generic functions with methods for other R classes.

```
> employeedetails=rbind(empdata,empnewdata)

> employeedetails

  empID empName Salary  startDate        dept

1   101  Revati  18000 2018-03-25          IT

2   102  Shruti  14000 2019-02-24     Finance

3   103   Vidhi  20000 2020-05-21     Testing

4   104  Ananya  23000 2018-03-25 Development

5   105  Sakshi  25000 2018-03-02          HR

6   106     Raj  12000 2020-01-05          HR

7   107     Ved  19000 2018-01-02          IT

8   108   Rishi  24000 2019-05-21 Development
```

Practical 2 – Matrix Operations

AIM : To perform Matrix operations

Code :

```
m1=matrix(c(1,2,3,4),nrow=2); m2=matrix(c(5,4,3,1),nrow=2)
m1m2=m1%*%m2; d=det(m1); inv=solve(m1); t=t(m1)

m1
m2
m1+m2
m1-m2
m1m2
d
inv
t
```

Output:

```
> m1
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> m2
     [,1] [,2]
[1,]    5    3
[2,]    4    1
> m1+m2
     [,1] [,2]
[1,]    6    6
[2,]    6    5
> m1-m2
     [,1] [,2]
[1,]   -4    0
[2,]   -2    3
> m1m2
     [,1] [,2]
[1,]   17    6
[2,]   26   10
> d
[1] -2
> inv
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
> t
     [,1] [,2]
[1,]    1    2
[2,]    3    4
>
```

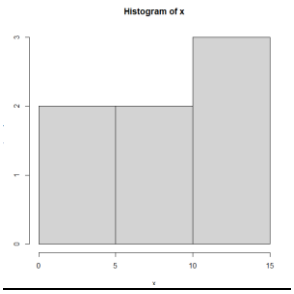
Practical 3 – Plot Histogram

AIM : To plot Histogram

Code :

```
x=c(2,5,8,11,10,12,15)
hist(x)
```

Output:



Practical 4 – Statistical functions

AIM : To find mean, median, quantiles, range

Code :

```
x=c(2,5,7,3,6)
mean=mean(x)
median=median(x)
q=quantile(x)
r=range(x)
mean
median
q
r
```

Output:

```
> mean
[1] 4.6
> median
[1] 5
> q
  0%  25%  50%  75% 100%
   2    3    5    6    7
> r
[1] 2 7
>
```

Practical 5 – Standard Deviation and Variance

AIM : To find SD and Variance

Code :

```
x=c(1,5,7,5,3)
stddev=sd(x)
v=var(x)
stddev
v
```

Output:

```
> stddev
[1] 2.280351
> v
[1] 5.2
>
```

Practical 6 – Skewness and Kurtosis

AIM : To observe Skewness and Kurtosis

Code :

```
install.packages("moments")
library(moments)
x=c(1,2,3,5,8)
skewness(x)
kurtosis(x)
```

Output:

```
> skewness(x)
[1] 0.6216349
> kurtosis(x)
[1] 2.032468
>
```

Practical 7 – Z test

AIM : To perform C test

Code :

```
mu=20
m=20.42
sd=2
N=33
zcal=((m-mu)/(sd/sqrt(N)))
zcal
```

Output:

```
[1] 1.206358
>*
```

Practical 8 – Chi Square test

AIM : To perform Chi Square test

Code :

```
x=matrix(c(20,30,40,20), nrow=2)
df=data.frame(x)
chi=chisq.test(df)
chi
```

Output:

```
        Pearson's Chi-squared test with Yates' continuity correction

data:  df
X-squared = 6.7836, df = 1, p-value = 0.0092

>
```

Practical 9 – Linear Regression

AIM : To perform Linear Regression using matrix

Code :

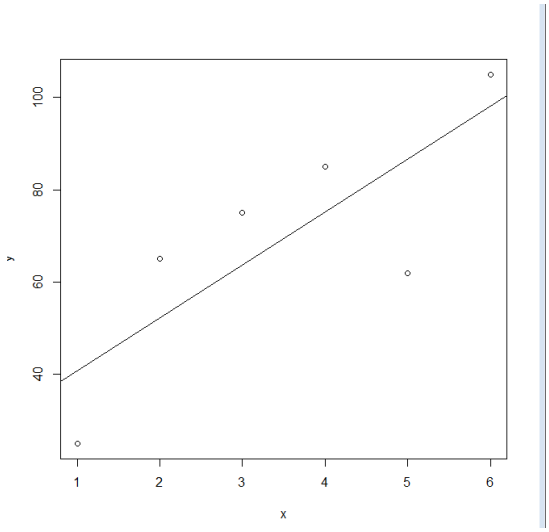
```
setwd("E:/");
M=matrix(c(1,2,3,4,5,6,25,65,75,85,62,105),ncol=2);
Mdata=data.frame(M);
x=Mdata$X1;
y=Mdata$X2;
reg=lm(y~x);
reg;
plot(x,y,abline(reg));
```

Output:

```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      29.40         11.46

> plot(x,y,abline(reg));
>
```



Practical 10 – Binomial and Normal Distribution

AIM : To perform Binomial and Normal Distribution

Binomial Distribution:

Binomial Distribution model deals with finding the probability of success of a random experiment

Example 1:

To find probability density function at each point

R built-in function used :

dbinom(x,size,prob)

where **x** is a vector of numbers

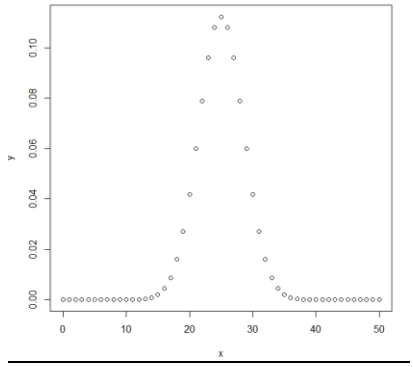
size is the number of trials

prob is the probability of success of each trial

Code:

```
x=seq(0,50,by=1);
x;
y=dbinom(x,50,0.5);
y;
plot(x,y);
```

Output :



Normal Distribution:

Normal Distribution curve is a bell-shaped curve for sufficiently high sample size

Example 3:

To find height of probability distribution at each point for given mean and given sd

R built-in function used :

dnorm(x,mean,sd)

where **x** is a vector of numbers

mean is the mean of distribution

sd is the standard deviation of distribution

Code:

```
x=seq(-10,10,by=0.1);

y=dnorm(x,mean=2.5,sd=0.5);

plot(x,y);
```

Output :