

# **ANNAMALAI**



# **UNIVERSITY**

## **FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING**

**B.E. (COMPUTER SCIENCE AND ENGINEERING)**

**[DATA SCIENCE]**

**VI – SEMESTER**

**DSCP607 – DATA ANALYSIS WITH R LAB**

**Name :** \_\_\_\_\_

**Reg. No. :** \_\_\_\_\_

# ANNAMALAI



# UNIVERSITY

## **FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**B.E. (COMPUTER SCIENCE AND ENGINEERING) [DATA SCIENCE]**

**VI - SEMESTER**

### **DSCP607 – DATA ANALYSIS WITH R LAB**

Certified that this is the bonafide record work done by  
Mr./Ms. \_\_\_\_\_  
Reg.No. \_\_\_\_\_ of B.E. (Computer Science and Engineering)  
[Data Science] in the DSCP607 – DATA ANALYSIS WITH R LAB  
during even semester of the academic year 2021 – 2022.

**Staff-in-charge  
Examiner**

**Internal**

**Place: Annamalai Nagar  
Examiner**

**External**

### **List of Exercises**

1. Learning R-Basic Mathematical and General Commands.
2. Write a R program to perform the Matrix Operations such as Addition (+), Subtraction (-), Multiplication (%\*%), Transpose (t), Inverse (solve()) and Diagonal of a Matrix (diag) using matrix, rbind and cbind commands.
  - a) Input matrix is fixed.
  - b) Get the input matrix from Keyboard)
3. Create a data frame called student data, explore the structure of the data and process it using
  - a) data.frame
  - b) read.table
4. a)Write a R program to compute Interquartile Range (IQR) for a given data.  
b)Write a R program to compute Interquartile Range (IQR) for Sepal Length of Iris data.
5. Write a R program to generate Frequency Distributions of MT Car's Carburettors and Air Quality's Temperatures from its Data Sets.
6. a)Write a R program to construct Univariate Normal Density and to predict whether A Person is Adult or not Based on Height  
b)Write a R program to construct multivariate Normal Density and to predict whether A Person is Adult or not Based on Height and Weight
7. Write a R program to analyze the Linear and Nonlinear Relationship between two variables in the different data sets (Women Data and MTcars Data) Using Covariance, Pearson and Spearman Correlation coefficients.
8. Write a R program to Analyze the Linear an Nonlinear Relationship Between the Continuous Variables of Iris Data Using Multiple Correlation coefficients.
9. Write a R program to analyze Baye's Rule and to predict whether A person is male or female .
10. Write a R program to find prediction of rainfall using sample mean and population with US precipitation cities data.
11. Write a R program to test the hypothesis which proves the mileage is better for manual cars than cars with automatic transmission using two means from MTcars dataset.
12. Write a R program to predict the mileage of car based on weight of car using simple linear regression.

EX.NO	DATE	NAME OF THE EXPERIMENT	PAGE	SIGN
1		Learning R-Basic Mathematical and General Commands.	1	
2		Perform the Matrix Operations such as Addition (+), Subtraction (-), Multiplication (%*%), Transpose (t), Inverse (solve()) and Diagonal of a Matrix (diag).	8	
3		Creating Data frame for student data	14	
4A		Interquartile Range (IQR) for a given data	18	
4B		Interquartile Range (IQR) for Sepal Length of Iris data.	21	
5		Frequency Distributions of MT Car's Carburetors and Air Quality's Temperatures	22	
6A		Univariate Normal Density	25	
6B		Multivariate Normal Density	29	
7		Linear and Nonlinear Relationship between two variables sets Using Covariance, Pearson and Spearman Correlation coefficients.	32	
8		Linear and Nonlinear Relationship Between the Continuous Variables of Iris Data Using Multiple Correlation coefficients	37	
9		Bayes' Rule	42	
10		Prediction of rainfall	46	
11		MT cars dataset	49	
12		Simple linear regression.	52	

<b>EX.NO: 1</b>	<b>R BASIC COMMANDS</b>
<b>DATE :</b>	

## AIM

To perform the basic mathematical operations in R programming.

### R Programming Basic Commands:

```

1+1          #Addition
10-2         #Subtraction
2*2          #Multiplication
100/10       #Division
10^2         #Exponentiation abs(-10)  #abs
function in R ceiling(4.5) #Ceiling and Floor
Function in R floor(4.5)
sqrt(10)     #Square Root Fun in R
exp(2)       #Exponentiation Fun in R pi
#Print pi value.
log(100)     #natural logs(i.e. base e)
log(100, base=10) #base 10 logs
(2+2)*2      #Expression
1+(2*2)
2==2        # Relational operation
5<=6
5>=6
x<- c(2,5,7,8,9) #Creating a Vector with integer vales using c fun in R x
> 5
mean(c(1,2,3,4)) #Return mean value
x<- c("apple","banana") #Creating a Vector with strings using c fun in R x
Y<- 10+10
Z<- -10+10
Y/Z x<-c(1,2,3)
y<-rnorm(4,5)  #random deviation function x;y

```

```

getwd()          #Current working directory
search()         #base packages x<-1:10
#creating a vector with 1 to 10.
x
ls( )            #list the used variables save(x, y, file = "xy.RData")# x and y
values are printed in a file called xy.RData.
save.image()     # creating ".RData" in current working directory load("xy.RData")
# Loading "xy.RData" in current working directory unlink("xy.RData")
lsx<-c(TRUE,FALSE)      #logical vector y<-
c("a","b","c")          #character vector z<-c(1,2,3,4)
#numeric integer vector m<-c(1,2,1.5,1.7)
#numeric real/double vector lsx; y; z; m
class(x); typeof(x); mode(x)
is.numeric(c("a","b"))  #Test whether the given data are numeric or not is.numeric(c(5,6))
as.character(c(1,2,3))  #Converted the given into character form as.numeric(c("c","4","b"))
#Converted the given into numeric form sum(c(FALSE,TRUE,TRUE))
c(1,2,3,4,5,6,7,8)
1:10
seq(1,8, by =2)         #Generates the sequence from:to by rep(1,5)
#Replicate Elements of Vectors and Lists rep(c(1,2,3),5)
x<-c(1,2,3,4,5)
names(x)<-c("a","b","c","d","e") #Names of x x
x[c(1,2,3)] #by numeric position
x[x<3]      #by logical vector x
d=x[x<3]    #by logical vector d
x[c("b","c")] # by name f=x[c("b","c")]
f
y<-matrix(c(1,2,3,4,5,6),byrow=TRUE,ncol=2) #Creating a matrix
y          class(y)

```

```

dim(y)
nrow(y)      #no of rows
ncol(y)      #no of column
rownames(y)<-c("a","b","c")
colnames(y)<-c("col1","col2")
y            y["a",]
y[c(1,2),] #Creating a list
x = list(name="Arun Patel", nationality="Indian", height=5.5, grades=c(95,45,80))
x class(y) x$name x$hei
#Creating a Data frame
z<-data.frame(var1=1:9,var2=letters[1:9])
z
View(z)
#Reading the Data from csv file
data=read.csv("d:\\sample.csv",header=T,sep=",")
data nrow(data) ncol(data) data head(data)
#Creating a function hw.f1 <- function()
{ hw <- "Hello
World" hw } hw.f1()
Install.packages("Matrix") dependencies=TRUE
Install.packages("Matrix", dependencies=TRUE)
d=library(Matrix) d x<-c(1,2,NA,4) x x<-
c("a","b",NA,"c") x is.na(x) na.omit(x)
library(MASS)      #user survey data from MASS
package data(survey) #load an internal data set data()

mydata<-survey
names(mydata)
str(mydata) x <-
c(1,2,3,4,5)
range(x)

```

```
dat<-data.frame(x=c(1,2,3,4,5),y=c(1,1,0,1,1)) dat
```

```
dat$z<-dat$x +dat$y
```

```
dat dat$z<-dat$x+10
```

```
rm(x)          #removing a variable x
```

## OUTPUT:

```
> #Learning R Programming Basic Commands
>
> 1+1          #Addition
[1] 2

> 10-2         #Subtraction
[1] 8

> 2*2          #Multiplication
[1] 4

> 100/10       #Division
[1] 10

> 10^2         #Exponentiation
[1] 100

> abs(-10)     #abs function in R
[1] 10

> ceiling(4.5) #Ceiling and Floor Function in R
[1] 5

> floor(4.5)
[1] 4

> sqrt(10)     #Square Root Fun in R
[1] 3.162278

> exp(2)       #Exponentiation Fun in R
[1] 7.389056
```

---



```

> pi          #Print pi value.
[1] 3.141593

> log(100)    #natural logs(i.e. base e)
[1] 4.60517

> log(100, base=10) #base 10 logs
[1] 2

> (2+2)*2     #Expression
[1] 8

> 1+(2*2)
[1] 5

> 2==2        # Relational operation
[1] TRUE

> 5<=6
[1] TRUE

> 5>=6
[1] FALSE

> x<- c(2,5,7,8,9) #Creating a Vector with integer vales using c fun in R

> x > 5
[1] FALSE FALSE TRUE TRUE TRUE

> mean(c(1,2,3,4)) #Return mean value
[1] 2.5

> x<- c("apple","banana") #Creating a Vector with strings using c fun in R

> x
[1] "apple" "banana"

> Y<- 10+10

> Z<- -10+10

> Y;Z
[1] 20
[1] 0

> x<-c(1,2,3)

> y<-rnorm(4,5) #random deviation function

> x;y
[1] 1 2 3
[1] 5.204597 5.691761 6.216607 4.489959

> getwd()      #Current working directory
[1] "C:/Users/91805/Documents"

> search()     #base packages
[1] ".GlobalEnv" "tools:rstudio" "package:stats" "package:graphics" "package:grDevices" "package:utils"
[7] "package:datasets" "package:methods" "Autoloads" "package:base"

> x<-1:10      #creating a vector with 1 to 10.

> x
[1] 1 2 3 4 5 6 7 8 9 10

> ls()         #list the used variables
[1] "d" "f" "lsx" "m" "x" "y" "z" "z"

> save(x, y, file = "xy.RData")# x and y values are printed in a file called xy.RData.

> save.image() # creating ".RData" in current working directory

> load("xy.RData") # Loading "xy.RData" in current working directory

> unlink("xy.RData")

> lxx<-c(TRUE,FALSE) #logical vector

> y<-c("a","b","c") #character vector

> z<-c(1,2,3,4) #numeric integer vector

> m<-c(1.2,1.5,1.7) #numeric real/double vector

> lxx; y; z; m
[1] TRUE FALSE
[1] "a" "b" "c"
[1] 1.0 2.3 4.0
[1] 1.2 1.5 1.7

> class(x); typeof(x); mode(x)
[1] "integer"
[1] "integer"
[1] "numeric"

> is.numeric(c("a","b")) #Test whether the given data are numeric or not
[1] FALSE

> is.numeric(c(5,6))
[1] TRUE

```

```

> as.character(c(1,2,3))      #Converted the given into character form
[1] "1" "2" "3"

> as.numeric(c("c","4","b")) #Converted the given into numeric form
[1] NA  4 NA

> sum(c(FALSE,TRUE,TRUE))
[1] 2

> c(1,2,3,4,5,6,7,8)
[1] 1 2 3 4 5 6 7 8

> 1:10
[1] 1 2 3 4 5 6 7 8 9 10

> seq(1,8, by =2)             #Generates the sequence from:to by
[1] 1 3 5 7

> rep(1,5)                    #Replicate Elements of Vectors and Lists
[1] 1 1 1 1 1

> rep(c(1,2,3),5)
[1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3

> x<-c(1,2,3,4,5)

> names(x)<-c("a","b","c","d","e") #Names of x

> x
a b c d e
1 2 3 4 5

```

---

```

> x[c(1,2,3)]                #by numeric position
a b c
1 2 3

> x[x<3]                      #by logical vector
a b
1 2

> x
a b c d e
1 2 3 4 5

> d=x[x<3]                    #by logical vector

> d
a b
1 2

> x[c("b","c")]              # by name
b c
2 3

> f=x[c("b","c")]

> f
b c
2 3

> y<-matrix(c(1,2,3,4,5,6),byrow=TRUE,ncol=2) #Creating a matrix

> y
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6

> class(y)
[1] "matrix" "array"

> dim(y)
[1] 3 2

> nrow(y)                      #no of rows
[1] 3

> ncol(y)                      #no of column
[1] 2

> rownames(y)<-c("a","b","c")

> colnames(y)<-c("col1","col2")

> y
  col1 col2
a     1    2
b     3    4
c     5    6

> y["a",]
col1 col2
1     1    2

```

```

> y[c(1,2),]
  col1 col2
a     1    2
b     3    4

> #Creating a list
> x = list(name="Arun Patel", nationality="Indian", height=5.5, grades=c(95,45,80))

> x
$name
[1] "Arun Patel"

$nationality
[1] "Indian"

$height
[1] 5.5

$grades
[1] 95 45 80

> class(y)
[1] "matrix" "array"

> x$name
[1] "Arun Patel"

> x$hei
[1] 5.5

> #Creating a Data frame
> z<-data.frame(var1=1:9,var2=letters[1:9])

> z
  var1 var2
1     1    a
2     2    b
3     3    c
4     4    d
5     5    e
6     6    f
7     7    g
8     8    h
9     9    i

> View(z)

> #Reading the Data from csv file
> data=read.csv("d:\\sample.csv",header=T,sep=",")

```

## RESULT:

Thus, the basic mathematical operations in R programming is successfully learned and executed.

<b>EX.NO: 2 A</b>	<b>MATRIX OPERATIONS</b>
<b>DATE :</b>	

### **AIM:**

To write a R program to perform the matrix operations such as Addition, Subtraction, Multiplication, Transpose, Inverse and Diagonal of a matrix using matrix, rbind and cbind commands.

### **CONCEPT:**

Matrix is simply several vectors stored together. The size of a matrix is specified by a number of rows and a number of columns.

### **PROGRAM:**

```
#Creating A and B Matrices using matrix command
A <- matrix(data=c(3,2,9,5), nrow=2, ncol=2, byrow=TRUE)
B <- matrix(data=c(7,1,6,4), nrow=2, ncol=2, byrow=FALSE)
#Creating C and D Matrices using rbind and cbind commands
C <- rbind(c(1,2,3),c(6,5,4)) D <- cbind(c(9,5,1),c(3,2,7)) print("Matrix of A") A
print("Matrix of B") B
print("Matrix of C") C
print("Matrix of D") D
print("Resultant Matrices")
print("Addition of Two Matrices=") A+B
print("Subtraction of Two Matrices=") A-B
print("Scalar Multiplication of A Matrix=")
3*A
print("Multiplication of Two Matrices C & D=")
C%*%D
print("Diagonal Matrix of A")
diag(A) cat("Transpose of
C") t(C)
cat("Inverse of B") solve(B)
```

## OUTPUT:

```
[workspace loaded from ~/.RData]
> source("~/active-rstudio-document", echo=TRUE)

> #Creating A and B Matrices using matrix command
> A <- matrix(data=c(3,2,9,5), nrow=2, ncol=2, byrow=TRUE)
> B <- matrix(data=c(7,1,6,4), nrow=2, ncol=2, byrow=FALSE)
> #Creating C and D Matrices using rbind and cbind commands
> C <- rbind(c(1,2,3),c(6,5,4))
> D <- cbind(c(9,5,1),c(3,2,7))
> print("Matrix of A")
[1] "Matrix of A"

> A
  [,1] [,2]
[1,]   3   2
[2,]   9   5

> print("Matrix of B")
[1] "Matrix of B"

> B
  [,1] [,2]
[1,]   7   6
[2,]   1   4
```

```
> print("Matrix of C")
[1] "Matrix of C"

> C
  [,1] [,2] [,3]
[1,]   1   2   3
[2,]   6   5   4

> print("Matrix of D")
[1] "Matrix of D"

> D
  [,1] [,2]
[1,]   9   3
[2,]   5   2
[3,]   1   7

> print("Resultant Matrices")
[1] "Resultant Matrices"

> print("Addition of Two Matrices=")
[1] "Addition of Two Matrices="

> A+B
  [,1] [,2]
[1,]  10   8
[2,]  10   9
```

```

> print("Subtraction of Two Matrices=")
[1] "Subtraction of Two Matrices="

> A-B
      [,1] [,2]
[1,]   -4  -4
[2,]    8   1

> print("Scalar Multiplication of A Matrix=")
[1] "Scalar Multiplication of A Matrix="

> 3*A
      [,1] [,2]
[1,]    9   6
[2,]   27  15

> print("Multiplication of Two Matrices C & D=")
[1] "Multiplication of Two Matrices C & D="

> C%%D
      [,1] [,2]
[1,]   22  28
[2,]   83  56

> print("Diagonal Matrix of A")
[1] "Diagonal Matrix of A"

> diag(A)
[1] 3 5

> cat("Transpose of C")
Transpose of C
> t(C)
      [,1] [,2]
[1,]    1   6
[2,]    2   5
[3,]    3   4

> cat("Inverse of B")
Inverse of B
> solve(B)
      [,1] [,2]
[1,] 0.18181818 -0.2727273
[2,] -0.04545455 0.3181818
>

```

## RESULT:

Thus , R program to perform the matrix operation is successfully executed.

<b>EX.NO:2 B</b>	<b>MATRIX OPERATIONS</b>
<b>DATE :</b>	

**AIM:**

To write a R program to perform the Matrix Operations such as Addition, Subtraction , Multiplication (%\*%), Transpose (t), Inverse (solve()) and Diagonal of a Matrix (diag) using vector and matrix. (Get the input matrix from Keyboard).

**PROGRAM:**

```

print("Enter the size of A Matrix")
m = as.integer(readline(prompt ='m='))
n = as.integer(readline(prompt ='n='))
asize<- m*n
avec=vector(mode="integer", length=0)
for(i in 1:asize)
{
  mv1=as.integer(readline())
  avec <- c(avec, mv1)
}
A<- matrix(data=avec, nrow =m,ncol =n,byrow=TRUE)
print("Enter the size of B Matrix")
p=as.integer(readline(prompt ='p='))
s=as.integer(readline(prompt ='s='))
bsize<- p*s
bvec=vector(mode="integer", length=0)
for(i in 1:bsize)
{
  mv2=as.integer(readline())
  bvec <- c(bvec, mv2)
}
A<- matrix(data=bvec, nrow =p,ncol =s,byrow=TRUE)
print("Matrix of A")

```

A

print("Matrix of B") B

cat("Resultant Matrices") print("Addition  
of Two Matrices=") A+B

print("Subtraction of Two Matrices=") A-B

print("Scalar Multiplication of a Matrix=")

3\*A

print("Multiplication of Two Matrices=")

A%\*%B

print("Diagonal Matrix")

diag(A)

cat("Transpose of A")

t(A)

cat("Inverse of A") solve(A)

## OUTPUT:

```
> source("~/active-rstudio-document", echo=TRUE)

> print("Enter the size of A Matrix")
[1] "Enter the size of A Matrix"

> m = as.integer(readline(prompt = 'm='))
m=2

> n = as.integer(readline(prompt = 'n='))
n=2

> asize<- m*n

> avec=vector(mode="integer", length=0)

> for(i in 1:asize)
+ {
+   mv1=as.integer(readline())
+   avec <- c(avec, mv1)
+ }
1
2
3
4

> A <- matrix(data=avec, nrow =m,ncol =n,byrow=TRUE)

> print("Enter the size of B Matrix")
[1] "Enter the size of B Matrix"

> p=as.integer(readline(prompt = 'p='))
p=2
```



```

> s=as.integer(readline(prompt = 's='))
s=2

> bsize<- p*s

> bvec=vector(mode="integer", length=0)

> for(i in 1:bsize)
+ {
+   mv2=as.integer(readline())
+   bvec <- c(bvec, mv2)
+ }
1
2
3
4

> B <- matrix(data=bvec, nrow =p,ncol =s,byrow=TRUE)

> print("Matrix of A")
[1] "Matrix of A"

> A
      [,1] [,2]
[1,]    1    2
[2,]    3    4

> print("Matrix of B")
[1] "Matrix of B"

```

```

> B
      [,1] [,2]
[1,]    1    2
[2,]    3    4

> cat("Resultant Matrices")
Resultant Matrices
> print("Addition of Two Matrices=")
[1] "Addition of Two Matrices="

> A+B
      [,1] [,2]
[1,]    2    4
[2,]    6    8

> print("Subtraction of Two Matrices=")
[1] "Subtraction of Two Matrices="

> A-B
      [,1] [,2]
[1,]    0    0
[2,]    0    0

> print("Scalar Multiplication of a Matrix=")
[1] "Scalar Multiplication of a Matrix="

> 3*A
      [,1] [,2]
[1,]    3    6
[2,]    9   12

> print("Multiplication of Two Matrices=")
[1] "Multiplication of Two Matrices="

> A%%B
      [,1] [,2]
[1,]    7   10
[2,]   15   22

> print("Diagonal Matrix")
[1] "Diagonal Matrix"

> diag(A)
[1] 1 4

> cat("Transpose of A")
Transpose of A
> t(A)
      [,1] [,2]
[1,]    1    3
[2,]    2    4

> cat("Inverse of A")
Inverse of A
> solve(A)
      [,1] [,2]
[1,] -2.0  1.0
[2,]  1.5 -0.5
> |

```

## RESULT:

Thus, the R program to perform the matrix operation using vector and matrix is successfully executed.

<b>EX.NO:3A</b>	<b>CREATE A DATA FRAME USING FOR STUDENT DATA</b>
<b>DATE :</b>	

### **AIM:**

To create a data frame called student data and explore the structure of the data and process it using data frame.

### **CONCEPT:**

A data frame is R's most natural way of presenting a data set with a collection of recorded observations for one or more variables. Data frame is one of the most important and frequently used tools in R for statistical data analysis.

### **PROGRAM:**

```
sdata <- data.frame(sname=c("Raja","somu","Roja"),
srollno=c(101,103,102),
sage=c(19,20,18),
ssex=c("male","male","female"),
sbranch=c("CSE","MECH","EEE"),
m1=c(90,79,88),          m2=c(95,85,90),
m3=c(85,25,85),          m4=c(70,40,60),
m5=c(67,67,89))
head(sdata)
nrow(sdata)
ncol(sdata)
result = vector(mode="character",length=0)
for(i in 1:nrow(sdata))
{
  if((sdata$m1[i] > 50) && sdata$m2[i] > 50 && sdata$m3[i] > 50 &&
sdata$m4[i] > 50 && sdata$m5[i] > 50)
{
    status<-"Pass"
  }
  else
```

```

{
  status<-"Fail"
}
result = append(result,status)
}
Total = sdata$m1+sdata$m2+sdata$m3+sdata$m4+sdata$m5
ptge = Total/5
tdata = cbind(sdata,Total,ptge)
tdata

```

## OUTPUT:

```

> head(sdata)
  sname srollno sage  ssex sbranch m1 m2 m3 m4 m5
1 Raja      101  19   male    CSE 90 95 85 70 67
2 somu      103  20   male    MECH 79 85 25 40 67
3 Roja      102  18 female    EEE 88 90 85 60 89

> nrow(sdata)
[1] 3

> ncol(sdata)
[1] 10

> result = vector(mode="character",length=0)

> for(i in 1:nrow(sdata))
+ {
+   if((sdata$m1[i] > 50) && sdata$m2[i] > 50 && sdata$m3[i] > 50 &&
+     sdata$m4[i] > 50 && sdata$m5[i] > 50)
+   {
+     .... [TRUNCATED]
+   }
+ }

> Total = sdata$m1+sdata$m2+sdata$m3+sdata$m4+sdata$m5

> ptge = Total/5

> tdata = cbind(sdata,Total,ptge)

> tdata
  sname srollno sage  ssex sbranch m1 m2 m3 m4 m5 Total ptge
1 Raja      101  19   male    CSE 90 95 85 70 67  407 81.4
2 somu      103  20   male    MECH 79 85 25 40 67  296 59.2
3 Roja      102  18 female    EEE 88 90 85 60 89  412 82.4

```

---

## RESULT:

Thus , the R program for loading the student data set is successfully executed.

<b>EX.NO:3B</b>	<b>LOADING THE STUDENT DATASET USING READ TABLE</b>
<b>DATE :</b>	

**AIM:**

To load the Student dataset from folder, explore the structure of the dataset and process it using read.table.

**PROGRAM:**

```
sdata <- read.table(file.choose(), sep=",", header=TRUE)
head(sdata) nrow(sdata) ncol(sdata)
result = vector(mode="character",length=0)
for(i in 1:nrow(sdata))
{
  if((sdata$m1[i] > 50) && sdata$m2[i] > 50 && sdata$m3[i] > 50 &&
sdata$m4[i] > 50 && sdata$m5[i] > 50)
  {
    status<-"Pass"
  }
else
  {
    status<-"Fail"
  }
  result = append(result,status)
}
Total = sdata$m1+sdata$m2+sdata$m3+sdata$m4+sdata$m5
ptge = Total/5
sdata = cbind(sdata,result>Total,ptge) sdata
```

## OUTPUT:

```
> head(sdata)
  sname sroll age gender branch m1 m2 m3 m4 m5
1  ragu   12  20  male    ds  90 98 90 97 89
2   vel    5  20  male    ds  90 90 90 92 98
3  deva    4  20  male    ai  90 98 87 98 98
4 naveen   47  21  male    ds  89 87 86 85 84
5 navith   25  20  male    ds  91 92 93 94 95
6 vicky   42  19  male    ds  81 82 83 84 85

> nrow(sdata)
[1] 6

> ncol(sdata)
[1] 10

> result = vector(mode="character",length=0)

> for(i in 1:nrow(sdata))
+ {
+   if((sdata$m1[i] > 50) && sdata$m2[i] > 50 && sdata$m3[i] > 50 &&
+     sdata$m4[i] > 50 && sdata$m5[i] > 50)
+     {
+       .... [TRUNCATED]
+     }
+ }

> Total = sdata$m1+sdata$m2+sdata$m3+sdata$m4+sdata$m5

> ptge = Total/5
```

## RESULT:

Thus, the R program for loading the student dataset is successfully executed.

<b>EX.NO: 4A</b>	<b>INTERQUARTILE RANGE(IQR)</b>
<b>DATE :</b>	

### **AIM:**

To write a R program to compute Interquartile Range (IQR) for a given data .

### **CONCEPT:**

A Quantile is a value computed from a collection of numeric measurements that indicate an observation's rank when compared to all the other present observations. IQR is computed as the difference between the upper and lower quartiles of your data.

### **PROGRAM:**

```
xdata = c(5,10,12,15,20,25,27,30,35)

#Compute Minimum, First, Second or Median, Third and Maximum Quartiles

MinQ= quantile(xdata,0)
FQ = quantile(xdata,0.25)
SQ = quantile(xdata,0.5)
TQ = quantile(xdata,0.75)
MaxQ= quantile(xdata,1)

#Print the Quartile One by One
cat("Minimum=",MinQ)
cat("Lower Quartile=",FQ)
cat("Median=",SQ) cat("Upper
Quartile=",TQ)
cat("Maximum=",MaxQ)

#Compute All the Quartiles (Min,First, Second or Median, Third and Max
Quartiles)
AQ = quantile(xdata,prob=c(0,0.25,0.5,0.75,1))

#Print All the Quartiles cat("All
the Quartiles",AQ)

#Summary provides the Statistics Information of xdata.
summary(xdata)

#Draw a box plot for xdata
```

```
boxplot(xdata,main="Interquartile Range (IQR)",ylab="xdata")
```

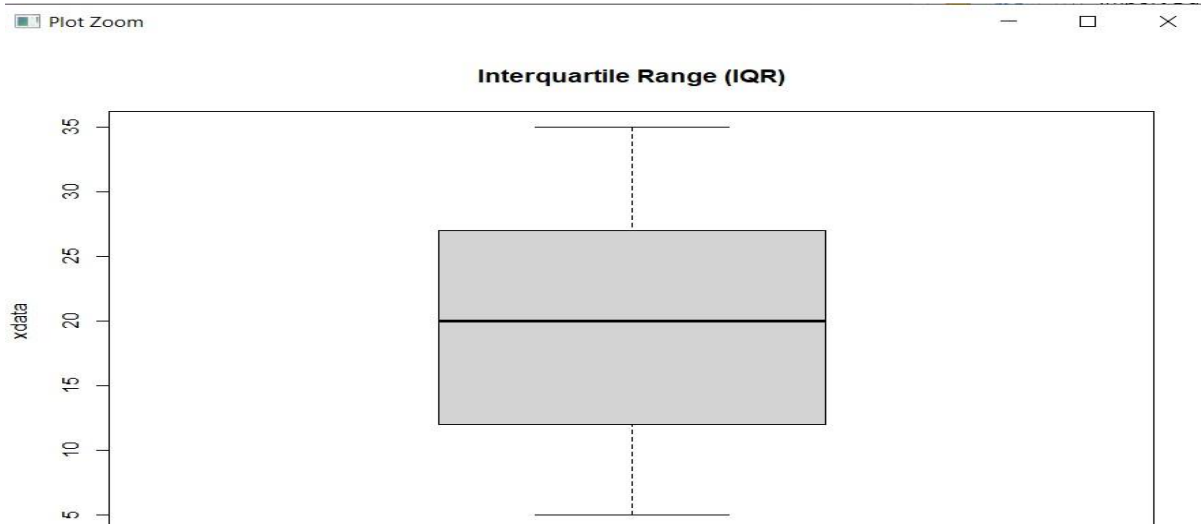
#IQR is computed as the difference between the upper and lower quartiles of your data

as.numeric(quantile(xdata,0.75)-quantile(xdata,0.25)) #IQR is computed using IQR function.

```
IQR(xdata)
```

## OUTPUT:

```
Minimum= 5
> cat("Lower Quartile=",FQ)
Lower Quartile= 12
> cat("Median=",SQ)
Median= 20
> cat("Upper Quartile=",TQ)
Upper Quartile= 27
> cat("Maximum=",MaxQ)
Maximum= 35
> #Compute All the Quartiles (Min,First, Second or Median, Third and Max Quartiles)
> AQ = quantile(xdata,prob=c(0,0.25,0.5,0.75,1))
> #Print All the Quartiles
> cat("All the Quartiles",AQ)
All the Quartiles 5 12 20 27 35
> #Summary provides the Statistics Information of xdata.
> summary(xdata)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00  12.00   20.00   19.89   27.00   35.00
> #Draw a box plot for xdata
> boxplot(xdata,main="Interquartile Range (IQR)",ylab="xdata")
> #IQR is computed as the difference between the upper and lower quartiles of
> #your data
> as.numeric(quantile(xdata,0.75)-quantile(xdata,0.25))
[1] 15
> #IQR is computed using IQR function.
> IQR(xdata)
[1] 15
> |
```



## RESULT:

Thus, R program to compute Interquartile Range(IQR) for a given data is successfully executed.

<b>EX.NO: 4B</b>	<b>INTERQUARTILE RANGE FOR SEPAL LENGTH</b>
<b>DATE :</b>	

**AIM:**

To write a R program to compute Interquartile Range (IQR) for Sepal Length of Iris data.

**PROGRAM:**

```
head(iris,150)

# Loading the sepal length of Iris data xdata
= iris$Sepal.Length

# Compute Minimum, First, Second or Median, Third and Maximum Quartiles
MinQ= quantile(xdata,0)
FQ = quantile(xdata,0.25)
SQ = quantile(xdata,0.5)
TQ = quantile(xdata,0.75)
MaxQ= quantile(xdata,1)

# Print the Quartiles One by One
cat("Minimum=",MinQ)
cat("Lower Quartile=",FQ)
cat("Median=",SQ) cat("Upper
Quartile=",TQ)
cat("Maximum=",MaxQ)

# Compute All the Quartiles (Min,First, Second or Median, Third and Max Quartiles)
AQ = quantile(xdata,prob=c(0,0.25,0.5,0.75,1)) cat("All the Quartiles",AQ)

# Summary provides the Statistics Information of xdata.
summary(xdata)

# Draw a box plot for xdata
boxplot(xdata,main="Interquartile Range for Sepal Length",ylab="Centimetres")

# IQR is computed as the difference between the upper and lower quartiles of your data
as.numeric(quantile(xdata,0.75)-quantile(xdata,0.25))

IQR(xdata)
```

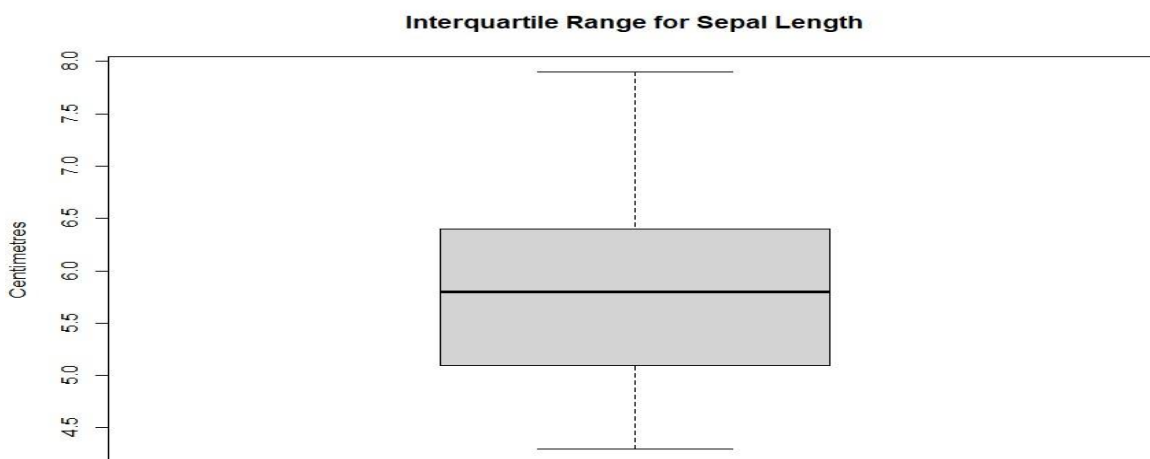


## OUTPUT:

```
> head(iris,150)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4          0.2   setosa
2           4.9         3.0          1.4          0.2   setosa
3           4.7         3.2          1.3          0.2   setosa
4           4.6         3.1          1.5          0.2   setosa
5           5.0         3.6          1.4          0.2   setosa
6           5.4         3.9          1.7          0.4   setosa
7           4.6         3.4          1.4          0.3   setosa
8           5.0         3.4          1.5          0.2   setosa
9           4.4         2.9          1.4          0.2   setosa
10          4.9         3.1          1.5          0.1   setosa
..          ..          ..          ..          ..      ..

> MinQ= quantile(xdata,0)
> FQ = quantile(xdata,0.25)
> SQ = quantile(xdata,0.5)
> TQ = quantile(xdata,0.75)
> MaxQ= quantile(xdata,1)
> # Print the Quartiles One by One
> cat("Minimum=",MinQ)
Minimum= 4.3
> cat("Lower Quartile=",FQ)
Lower Quartile= 5.1
> cat("Median=",SQ)
Median= 5.8
> cat("Upper Quartile=",TQ)
Upper Quartile= 6.4
> cat("Maximum=",MaxQ)
Maximum= 7.9
> # Compute All the Quartiles (Min,First, Second or Median, Third and Max Quartiles)
> AQ = quantile(xdata,prob=c(0,0.25,0.5,0.75,1))
> cat("All the Quartiles",AQ)
All the Quartiles 4.3 5.1 5.8 6.4 7.9
> # Summary provides the Statistics Information of xdata.
> summary(xdata)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
4.300  5.100   5.800   5.843  6.400   7.900
> # Draw a box plot for xdata
> boxplot(xdata,main="Interquartile Range for Sepal Length",ylab="Centimetres")
> # IQR is computed as the difference between the upper and lower quartiles of your data
> as.numeric(quantile(xdata,0.75)-quantile(xdata,0.25))
[1] 1.3
> # IQR is computed using IQR function.
> IQR(xdata)
[1] 1.3
```

Plot Zoom



## RESULT:

Thus, R program to compute Interquartile Range for sepal length to Iris data is executed successfully.

<b>EX.NO: 5</b>	<b>FREQUENCY DISTRIBUTIONS OF MTCARS</b>
<b>DATE :</b>	

### **AIM:**

To write a R program to compute data of frequency distributions of MTcars.

### **PROGRAM:**

```

packages="datasets"

#Loading the MTCars Datasets head(mtcars)

#Finding the unique value of carburetors
u1<-unique(mtcars$carb)

cat("Carburetors :", u1)

#Build a contingency table of the counts/frequencies at each values/levels.
t1=table(mtcars$carb)

#Frequency Distribution of MT Car's Carburetors

barplot(t1,xlab="Air Temperatures", ylab="Frequencies",main="Frequency Distribution of
MT Car's Carburetors")

#Loading the Air Quality Datasets head(airquality)

#Finding the unique value of Temperatures
u2<-unique(airquality$Temp)

cat("Air Equality's Temperature", u2)

#Build a contingency table of the counts/frequencies at each values/levels.
t2=table(airquality$Temp)

#Frequency Distribution

barplot(t2,xlab="Air Temperatures", ylab="Frequencies",main="Frequency Distribution of Air
Temperatures")

#Build a contingency table for range of temperatures and their counts/frequencies.
t3=table(cut(airquality$Temp,9))

#Frequency Distribution of range of temperaturess

barplot(t3,xlab="Range of Air Temperatures", ylab="Frequencies",main="Frequency
Distribution of Range of Air Temperatures")

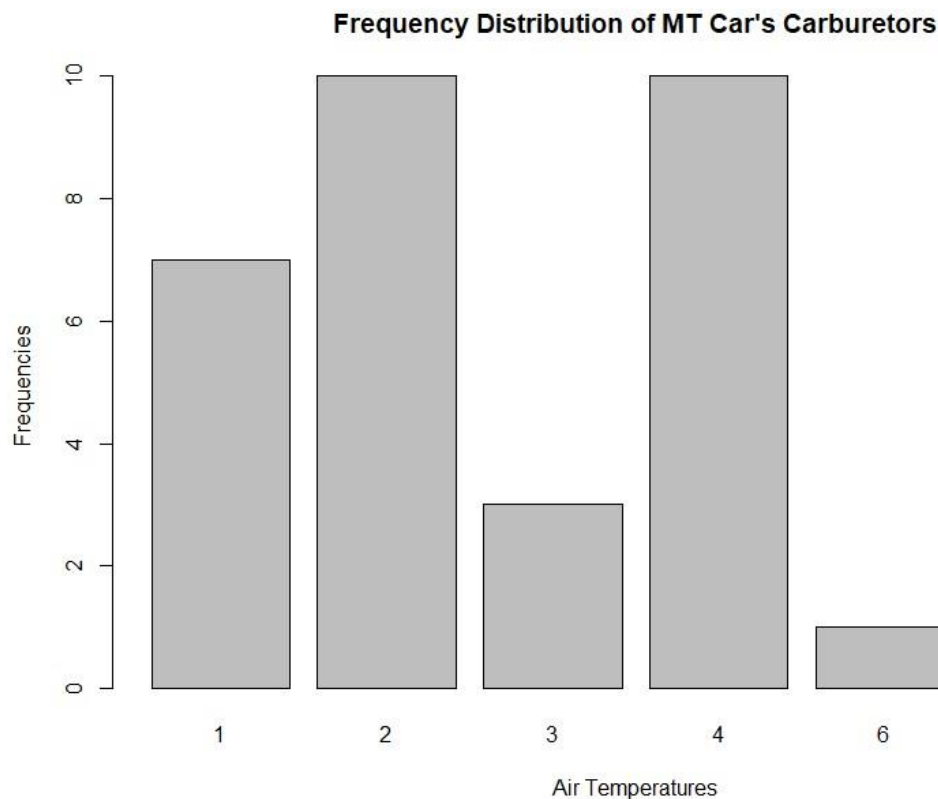
```

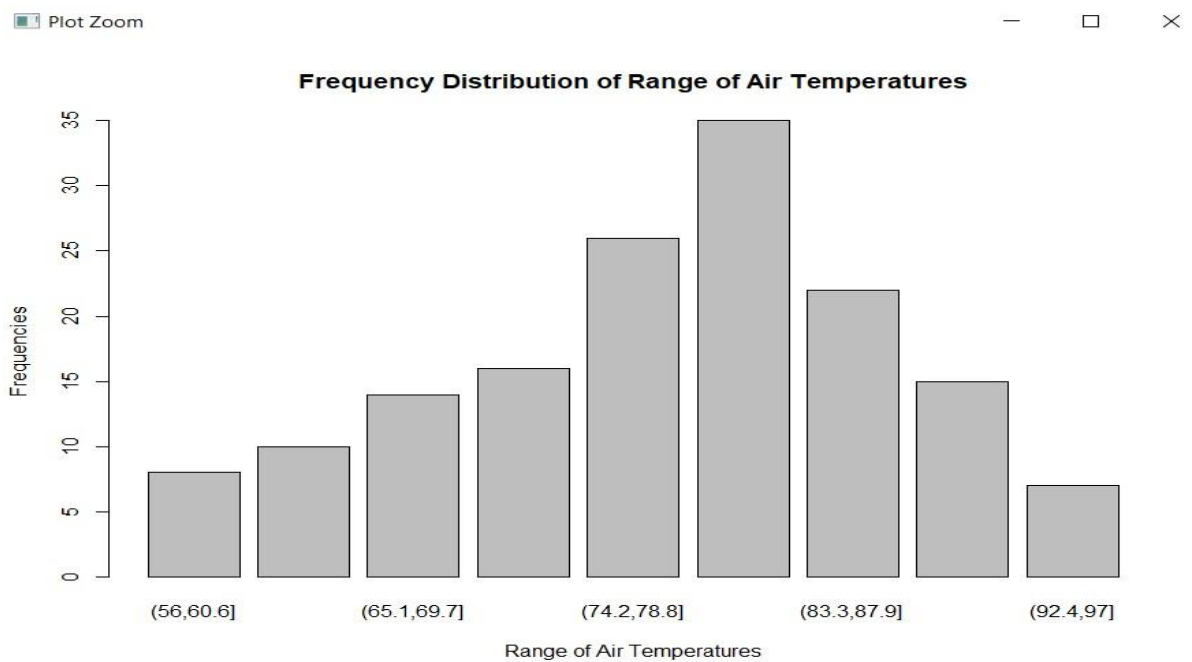
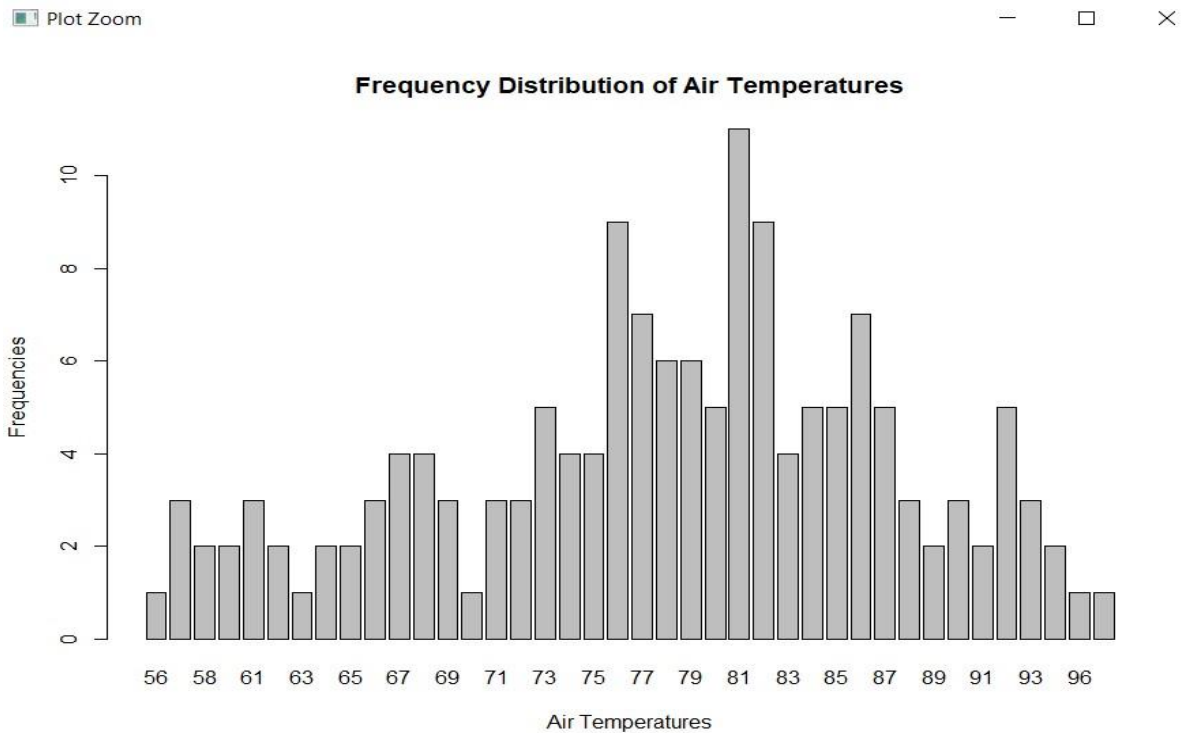
## OUTPUT

```
> head(mtcars)
      mpg  cyl  disp  hp drat   wt  qsec vs  am  gear  carb
Mazda RX4     21.0   6  160  110 3.90 2.620 16.46 0   1    4    4
Mazda RX4 Wag 21.0   6  160  110 3.90 2.875 17.02 0   1    4    4
Datsun 710     22.8   4  108   93 3.85 2.320 18.61 1   1    4    1
Hornet 4 Drive 21.4   6  258  110 3.08 3.215 19.44 1   0    3    1
Hornet Sportabout 18.7   8  360  175 3.15 3.440 17.02 0   0    3    2
Valiant        18.1   6  225  105 2.76 3.460 20.22 1   0    3    1
> #Finding the unique value of carburetors
> u1<-unique(mtcars$carb)
> cat("Carburetors :", u1)
Carburetors : 4 1 2 3 6 8
> #Build a contingency table of the counts/frequencies at each values/levels.
> t1=table(mtcars$carb)
> #Frequency Distribution of MT Car's Carburetors
> barplot(t1,xlab="Air Temperatures", ylab="Frequencies",main="Frequency Distribution of MT Ca
r's Carburetors")
> #Loading the Air Quality Datasets
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1   41    190  7.4   67     5   1
2   36    118  8.0   72     5   2
3   12    149 12.6   74     5   3
4   18    313 11.5   62     5   4
5   NA     NA 14.3   56     5   5
6   28     NA 14.9   66     5   6
> #Finding the unique value of Temperatures
> u2<-unique(airquality$Temp)
> cat("Air Equality's Temperature", u2)
Air Equality's Temperature 67 72 74 62 56 66 65 59 61 69 68 58 64 57 73 81 79 76 78 84 85 82 8
7 90 93 92 80 77 75 83 88 89 91 86 97 94 96 71 63 70
```

Plot Zoom

— □ ×





## RESULT:

Thus, the frequency distribution of MT cars carburators and air quality temperature is successfully executed.

<b>EX.NO: 6A</b>	<b>UNIVARIATE NORMAL DENSITY</b>
<b>DATE :</b>	

### AIM:

To write a R program to construct Univariate Normal Density and to predict whether A Person is Adult or not Based on Height Univariate Training function.

### CONCEPT:

It involves single variable or one dimension.

### PROGRAM:

```

uvtrain <- function(hdata)
{
  xv=vector(mode="numeric", length=0)
  pv=vector(mode="numeric", length=0)
  hmin = min(hdata)-15
  hmax = max(hdata)+15
  m = mean(hdata);
  v = var(hdata);
  cat("Mean of Height", m ,"\n")
  cat("Variance of Height",v)
  for(x in hmin:hmax)
  {
    r = (x-m)^2/v
    p = (1/(sqrt(2*pi*v)))*exp(-0.5*r);
    xv <- c(xv, x)   pv <- c(pv, p)
  }

  plot(xv,pv,xlab="Height of Person",ylab="p(x)",main="Univariate Normal Density",col =
"blue")
  return(list(m,v))
}

# Univariate Testing Function
uvtest <- function(m,v,ht)

```

```

{
  r = (ht-m)^2/v
  pt = (1/(sqrt(2*pi*v)))*exp(-0.5*r)
  if (pt >= 0.00005)
    print("The given height of person is an adult")
  else
    print("The given height of person is not an adult")
}

# Univariate Training Code

hdata <- c(165, 170, 160, 154, 175, 155, 167, 177, 158, 178)

mv = uvtrain(hdata) # Univariate Testing Code

ht = as.numeric(readline(prompt = 'Enter the height of person ='))

m = as.numeric(mv[1]) v = as.numeric(mv[2]) uvtest(m,v,ht)

```

OUTPUT:

```

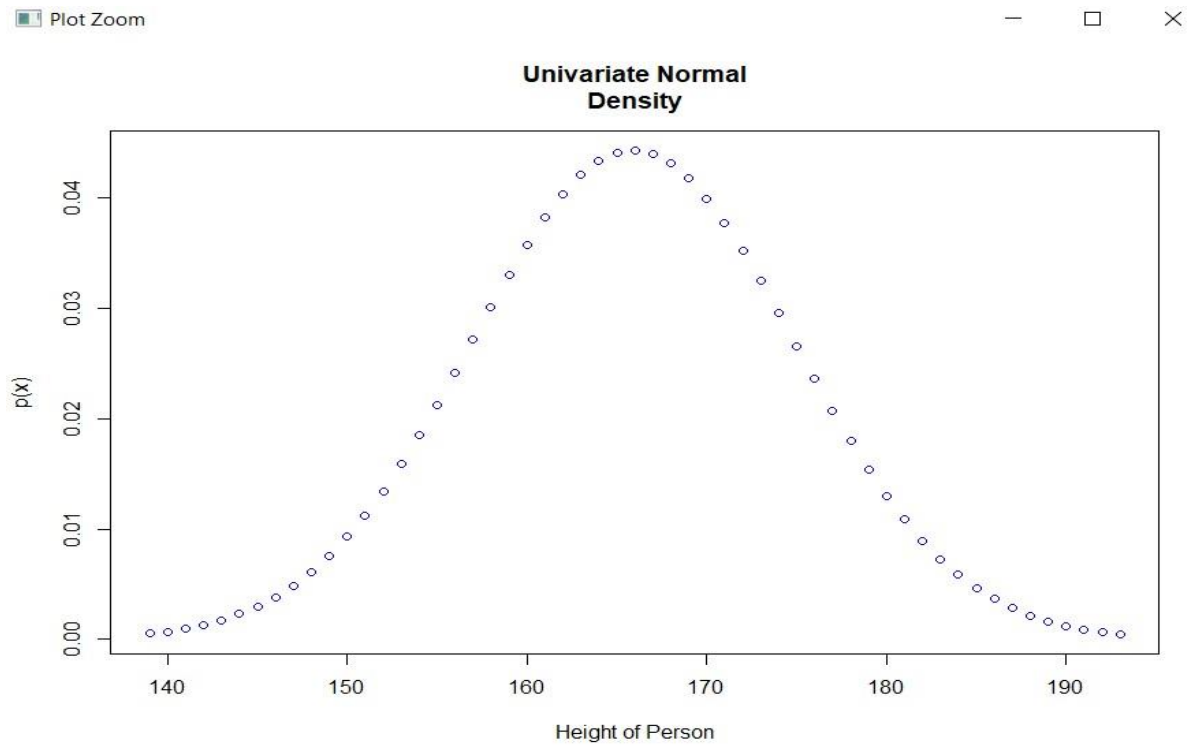
> uvtrain <- function(hdata)
+ {
+   xv=vector(mode="numeric", length=0)
+   pv=vector(mode="numeric", length=0)
+   hmin = min(hdata)-15
+   hmax = max(hdata)+15
+   m = mean(hdata);
+   v = var(hdata);
+   cat("Mean of Height", m ,"\n")
+   cat("Variance of Height",v)
+
+   for(x in hmin:hmax)
+   {
+     r = (x-m)^2/v
+     p = (1/(sqrt(2*pi*v)))*exp(-0.5*r);
+     xv <- c(xv, x)
+     pv <- c(pv, p)
+   }
+   plot(xv,pv,xlab="Height of Person",ylab="p(x)",main="Univariate Normal
+ Density",col = "blue")
+   return(list(m,v))
+ }
> uvtest <- function(m,v,ht)
+ {
+   r = (ht-m)^2/v
+   pt = (1/(sqrt(2*pi*v)))*exp(-0.5*r)
+   if (pt >= 0.00005)
+     print("The given height of person is an adult")
+   else
+     print("The given height of person is not an adult")
+ }
> # Univariate Training Code
> hdata <- c(165, 170, 160, 154, 175, 155, 167, 177, 158, 178)

```

```

> mv = uvtrain(hdata)
Mean of Height 165.9
Variance of Height 80.98889
> # Univariate Testing Code
> ht = as.numeric(readline(prompt = 'Enter the height of person ='))
Enter the height of person =173
> m = as.numeric(mv[1])
> v = as.numeric(mv[2])
> uvtest(m,v,ht)
[1] "The given height of person is an adult"

```



## RESULT:

Thus, R program to construct Univariate density to predict whether a person is adult or not based on Height is executed successfully.

<b>EX.NO: 6B</b>	<b>MULTIVARIATE NORMAL DENSITY</b>
<b>DATE :</b>	

### **AIM:**

To write a R program to construct multivariate Normal Density and to predict whether A Person is Adult or not Based on Height and Weight Multivariate Training Function.

### **CONCEPT:**

A multivariate normal distribution is a vector in multiple normally distributed variables, such that any linear combination of the variables is also normally distributed.

### **PROGRAM:**

```
mvtrain <- function(hwdata)
{
  nd=2
  hv=vector(mode="numeric", length=0)
  wv=vector(mode="numeric", length=0)
  pv=vector(mode="numeric", length=0)
  hmin = min(hwdata[,1])-15
  hmax = max(hwdata[,1])+15
  wmin = min(hwdata[,2])-15
  wmax = max(hwdata[,2])+15
  mv = colMeans(hwdata);
  cv = cov(hwdata);
  cat("Mean Vector", mv ,"\n")
  cat("Covariance of Height",cv)
  for(h in hmin:hmax)
  {
    for(w in wmin:wmax)
    {
      d = c(h,w)-mv
```



```

    r = ((t(d) %*% solve(cv)) %*% (d))
p = 1/(2*pi*sqrt(det(cv)))*exp(-0.5*r)
hv <- c(hv, h)
wv <- c(wv, w)
pv <- c(pv, p)
    }
}
# install.packages("rgl", dependencies = TRUE)
# library(rgl)
plot3d(x = hv, y = wv, z = pv, col = "blue", xlab = "Height", ylab = "Weight", zlab = "p(h,w)")
return (mvdata=data.frame(mv=mv,cv=cv))
}
# Multivariate Testing Function mvtest
<- function(mvdata,hwdata)
{
  mv = mvdata$mv
  cv = cbind(mvdata$cv.1, mvdata$cv.2)
  d = hwdata - mv
  r = ((t(d) %*% solve(cv)) %*% (d))
  pt = 1/(2*pi*sqrt(det(cv)))*exp(-0.5*r)
  if (pt >= 0.00005)
    print("person is an adult based on H & W")
  else
    print("person is not an adult based on H & W")
}
# Multivariate Training Code
hwdata <- cbind(c(165, 170, 160, 154, 175, 155, 167, 177, 158, 178),
c(78, 71, 60, 53, 72, 51, 64, 65, 55, 69))
mvdata = mvtrain(hwdata)
#Multivariate Testing Code
ht= as.numeric(readline(prompt = 'Enter the Height of person ='))

```

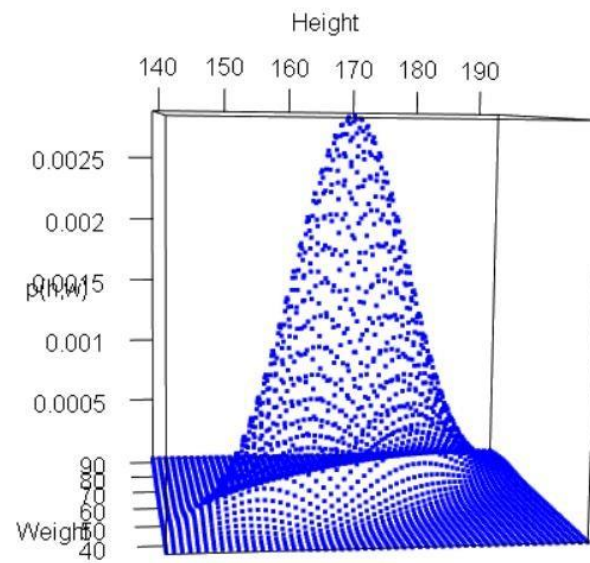
```
wt= as.numeric(readline(prompt ='Enter the Weight of person ='))
```

```
hwdata = c(ht,wt) mvtest(mvdata,hwdata)
```

## OUTPUT

```
> mvtrain <- function(hwdata)
+ {
+   nd=2
+   hv=vector(mode="numeric", length=0)
+   wv=vector(mode="numeric", length=0)
+   pv=vector(mode="numeric", length=0)
+   hmin = min(hwdata[,1])-15
+   hmax = max(hwdata[,1])+15
+   wmin = min(hwdata[,2])-15
+   wmax = max(hwdata[,2])+15
+   mv = colMeans(hwdata);
+   cv = cov(hwdata);
+   cat("Mean Vector", mv ,"\n")
+   cat("Covariance of Height",cv)
+   for(h in hmin:hmax)
+   {
+     for(w in wmin:wmax)
+     {
+       d = c(h,w)-mv
+       r = ((t(d) %*% solve(cv)) %*% (d))
+       p = 1/(2*pi*sqrt(det(cv)))*exp(-0.5*r)
+       hv <- c(hv, h)
+       wv <- c(wv, w)
+       pv <- c(pv, p)
+     }
+   }
+ }
+
+ #install.packages("rgl", dependencies = TRUE)
+ library(rgl)
+
+ plot3d(x = hv, y = wv, z = pv, col = "blue", xlab="Height", ylab="Weight",zlab="p(h,w)")
+ return (mvdata=data.frame(mv=mv,cv=cv))
+ }

> # Multivariate Testing Function
> mvtest <- function(mvdata,hwdata)
+ {
+   mv = mvdata$mv
+   cv = cbind(mvdata$cv.1, mvdata$cv.2)
+   d = hwdata - mv
+   r = ((t(d) %*% solve(cv)) %*% (d))
+   pt = 1/(2*pi*sqrt(det(cv)))*exp(-0.5*r)
+   if (pt >= 0.00005)
+     print("person is an adult based on H & W")
+   else
+     print("person is not an adult based on H & W")
+ }
> # Multivariate Training Code
> hwdata <- cbind(c(165, 170, 160, 154, 175, 155, 167, 177, 158, 178),
+               c(78, 71, 60, 53, 72, 51, 64, 65, 55, 69))
> mvdata = mvtrain(hwdata)
Mean Vector 165.9 63.8
Covariance of Height 80.98889 58.64444 58.64444 80.17778> #Multivariate Testing Code
> ht= as.numeric(readline(prompt ='Enter the Height of person ='))
Enter the Height of person =172
> wt= as.numeric(readline(prompt ='Enter the weight of person ='))
Enter the weight of person =67
> hwdata = c(ht,wt)
> mvtest(mvdata,hwdata)
[1] "person is an adult based on H & W"
```

**RESULT:**

Thus, R program to construct multivariate density is executed successfully.

<b>EX.NO: 07</b>	<b>LINEAR AND NON LINEAR ANALYSIS</b>
<b>DATE :</b>	

**AIM:**

To write a R program to analyze the Linear and Nonlinear variables.

**PROGRAM:**

**1. Analysis of the Positive Relationship between Height and Weight of Women Using Correlation Coefficients.**

```
# loading the Women's Data sets head(women,15)
#Scatter Plot library(ggplot2)
scatter.smooth(women$height,women$weight,main="ScatterPlot",xlab="Height",ylab="Weight")
#Finding the covariance between Height and Weight of Women.
c11 = cov(women$height, women$height)
c12 = cov(women$height, women$weight)
c21 = cov(women$weight, women$height)
c22 = cov(women$weight, women$weight)
#Constructing the Covariance Matrix
cm1 = matrix(data = c(c11,c12,c21,c22), nrow = 2, byrow = TRUE)
print("Covariance Matrix") print(cm1)
#Constructing the full Covariance Matrix at a time
cm2 = cov(women) print("Covariance Matrix")
print(cm1)
#Finding the Correlation Coefficients between Height and Weight of Women.
cc11 = cor(women$height,women$height)
cc12 = cor(women$height,women$weight)
cc21 = cor(women$weight,women$height)
cc22 = cor(women$weight,women$weight)
#Constructing the Correlation Coefficients
```

```

cc1 = matrix(data = c(cc11,cc12,cc21,cc22), nrow = 2, byrow = TRUE)

print("Pearson's Correlation Coefficients")

print(cc1)

#Constructing the Correlation Coefficients at a time

cc2 = cor(women)

print("Pearson's Correlation Coefficients")

print(cc2)

cc3 = cor(women,method = "spearman")

print("Spearman's Correlation Coefficients")

print(cc3) if(cc11 > 0)
{
  print("Relationship b/w Women's Weight and Height is Positive")
} else {
  print("Relationship b/w Women's Weight and Height is Negative")}

```

## OUTPUT:

```

> # loading the Women's Data sets
> head(women,15)
  height weight
1     58    115
2     59    117
3     60    120
4     61    123
5     62    126
6     63    129
7     64    132
8     65    135
9     66    139
10    67    142
11    68    146
12    69    150
13    70    154
14    71    159
15    72    164
> #Scatter Plot library(ggplot2)
> scatter.smooth(women$height,women$weight,main="ScatterPlot",xlab="Height",ylab="weight")
> #Finding the covariance between Height and weight of Women.
> c11 = cov(women$height, women$height)
> c12 = cov(women$height, women$weight)
> c21 = cov(women$weight, women$height)
> c22 = cov(women$weight, women$weight)
> #Constructing the Covariance Matrix
> cm1 = matrix(data = c(c11,c12,c21,c22), nrow = 2, byrow = TRUE)
> print("Covariance Matrix")
[1] "Covariance Matrix"
> print(cm1)
      [,1] [,2]
[1,] 20 69.0000
[2,] 69 240.2095

```

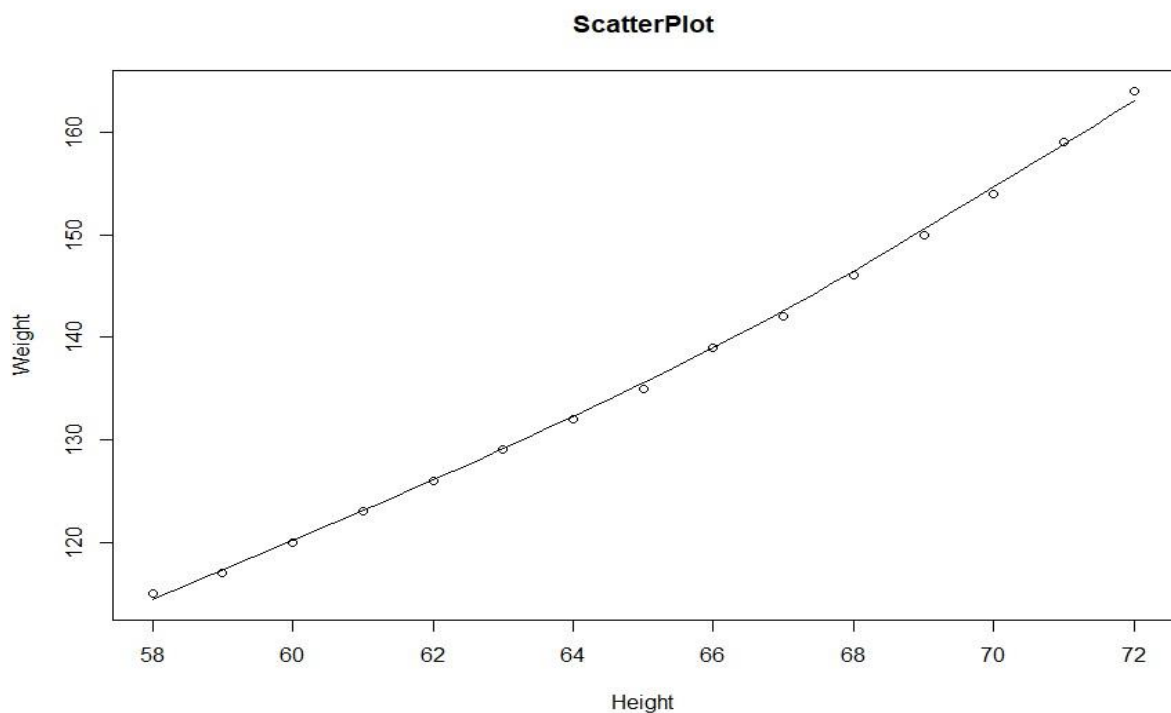
```

> #Constructing the full Covariance Matrix at a time
> cm2 = cov(women)
> print("Covariance Matrix")
[1] "Covariance Matrix"
> print(cm1)
      [,1] [,2]
[1,]  20 69.0000
[2,]  69 240.2095
> #Finding the Correlation Coefficients between Height and Weight of Women.
> cc11 = cor(women$height,women$height)
> cc12 = cor(women$height,women$weight)
> cc21 = cor(women$weight,women$height)
> cc22 = cor(women$weight,women$weight)
> #Constructing the Correlation Coefficients
> cc1 = matrix(data = c(cc11,cc12,cc21,cc22), nrow = 2, byrow = TRUE)
> print("Pearson's Correlation Coefficients")
[1] "Pearson's Correlation Coefficients"
> print(cc1)
      [,1] [,2]
[1,] 1.0000000 0.9954948
[2,] 0.9954948 1.0000000
> #Constructing the Correlation Coefficients at a time
> cc2 = cor(women)
> print("Pearson's Correlation Coefficients")
[1] "Pearson's Correlation Coefficients"
> print(cc2)
      height weight
height 1.0000000 0.9954948
weight 0.9954948 1.0000000
> cc3 = cor(women,method = "spearman")
> print("Spearman's Correlation Coefficients")
[1] "Spearman's Correlation Coefficients"
> print(cc3)
      height weight
height      1      1
weight      1      1
> if(cc11 > 0){
+   print("Relationship b/w Women's Weight and Height is Positive")
+ } else {
+   print("Relationship b/w Women's Weight and Height is Negative")}
[1] "Relationship b/w Women's Weight and Height is Positive"

```

Plot Zoom

— □ ×



## 2. Analysis of the Negative Relationship Between Weight of Cars and Mileage Using

**Correlation coefficients.** #loading the mtcars Data sets head(mtcars,32)

#Finding the Correlation Coeff. b/w Weight of Cars and Mileage.

```
co = cov(mtcars$wt, mtcars$mpg)
```

```
print("Covariance")
```

```
print(co)
```

#Finding the Pearson Correlation Coeff. b/w Weight of Cars and Mileage.

```
cc = cor(mtcars$wt, mtcars$mpg)
```

```
print("Pearson's Correlation Coefficient")
```

```
print(cc)
```

#Finding the Spearman Correlation Coeff. b/w Weight of Cars and Mileage.

```
ccs = cor(mtcars$wt, mtcars$mpg,method = "spearman")
```

```
print("Spearman's Correlation Coefficient")
```

```
print(ccs)
```

#Scatter Plot library(ggplot2)

```
scatter.smooth(mtcars$wt, mtcars$mpg, main="Scatter Plot", xlab="CarWeight",  
ylab="Mileage")
```

```
if(cc > 0){
```

```
print("Relationship b/w Car Weight and Mileage is Positive")
```

```
} else
```

```
{print("Relationship b/w Car Weight and Mileage is Negative")}
```

## OUTPUT

```
> head(mtcars,32)
```

	mpg	cyl	displacement	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4



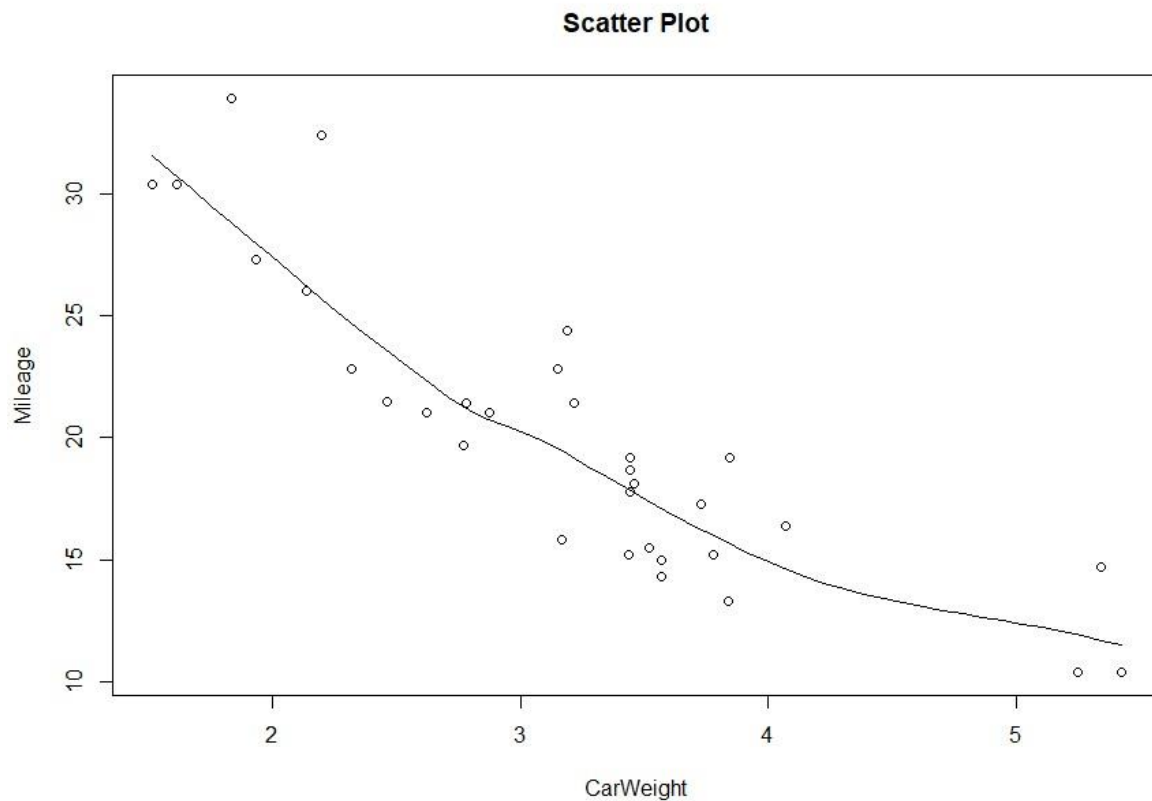
```

[1] "Covariance"
> print(co)
[1] -5.116685
> #Finding the Pearson Correlation Coeff. b/w Weight of Cars and Mileage.
> cc = cor(mtcars$wt, mtcars$mpg)
> print("Pearson's Correlation Coefficient")
[1] "Pearson's Correlation Coefficient"
> print(cc)
[1] -0.8676594
> #Finding the Spearman Correlation Coeff. b/w Weight of Cars and Mileage.
> ccs = cor(mtcars$wt, mtcars$mpg, method = "spearman")
> print("Spearman's Correlation Coefficient")
[1] "Spearman's Correlation Coefficient"
> print(ccs)
[1] -0.886422
> #Scatter Plot library(ggplot2)
> scatter.smooth(mtcars$wt, mtcars$mpg, main="Scatter Plot", xlab="CarWeight", ylab="Mileage")
> if(cc > 0){
+   print("Relationship b/w Car Weight and Mileage is Positive")
+ } else
+ {print("Relationship b/w Car weight and Mileage is Negative")}
[1] "Relationship b/w Car Weight and Mileage is Negative"

```

Plot Zoom

— □ ×



## RESULT:

Thus, R program to analysis of the linear and non linear variables is successfully executed.



<b>EX.NO: 08</b>	<b>MULTIPLE CORRELATION COEFFICIENTS</b>
<b>DATE :</b>	

**AIM:**

To write a R program to Analyze the Linear and Nonlinear variables using multiple correlations.

**CONCEPT:**

A multiple correlation coefficient (R) yields the maximum degree of liner relationship that can be obtained between two or more independent variables and a single dependent variable.

**PROGRAM:**

```
#loading the Iris Data sets
head(iris[1:5,])
head(iris[51:55,])
head(iris[101:105,])
iris.nospecies<- iris[,-5]
#Constructing the Covariance Matrix
coi = cov(iris.nospecies)
print("Covariance Matrix") print(coi)
#Finding the Multiple Pearson's Correlation Coefficients
cci = cor(iris.nospecies)
print("Multiple Pearson's Correlation Coefficients")
print(cci)
#Finding the Multiple Spearman Correlation Coefficients
ccs = cor(iris.nospecies, method = "spearman")
print("Multiple Spearman's Correlation Coefficients")
print(ccs)
#Analysis of Iris Data Using Box Plot
qplot(Species, Petal.Length, data=iris, geom="boxplot", fill=Species)
#Analysis of Iris Data Using Normal Density
qplot (Petal.Length, data=iris, geom="density", alpha=I(.7),fill=Species)
```

```

if(cci[4,1] > 0)
{ print("Relationship b/w Petal Width and Sepal Length is Positive")
}
else
{
print("Relationship b/w Petal Width and Sepal Length is Negative")}
if(cci[2,1] > 0)
{
print("Relationship b/w Sepal Width and Sepal Length is Positive")
}
else
{
print("Relationship b/w Sepal Width and Sepal Length is Negative")}
#Relationship between the petal lengths of the different iris species
install.packages("corrgram")
library(corrgram)
corrgram(iris, lower.panel=panel.conf, upper.panel=panel.pts)
# Overlapping Density Plot for Three Species
corrgram(iris, lower.panel=panel.pie, upper.panel=panel.pts,
diag.panel=panel.density, main=paste0("corrgram of petal and sepal", "measurements in iris
data set"))

```

OUTPUT:

```

> #loading the Iris Data sets
> head(iris[1:5,])
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5         1.4         0.2   setosa
2          4.9         3.0         1.4         0.2   setosa
3          4.7         3.2         1.3         0.2   setosa
4          4.6         3.1         1.5         0.2   setosa
5          5.0         3.6         1.4         0.2   setosa
> head(iris[51:55,])
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
51          7.0         3.2         4.7         1.4 versicolor
52          6.4         3.2         4.5         1.5 versicolor
53          6.9         3.1         4.9         1.5 versicolor
54          5.5         2.3         4.0         1.3 versicolor
55          6.5         2.8         4.6         1.5 versicolor
> head(iris[101:105,])
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
101          6.3         3.3         6.0         2.5 virginica
102          5.8         2.7         5.1         1.9 virginica
103          7.1         3.0         5.9         2.1 virginica
104          6.3         2.9         5.6         1.8 virginica
105          6.5         3.0         5.8         2.2 virginica

```

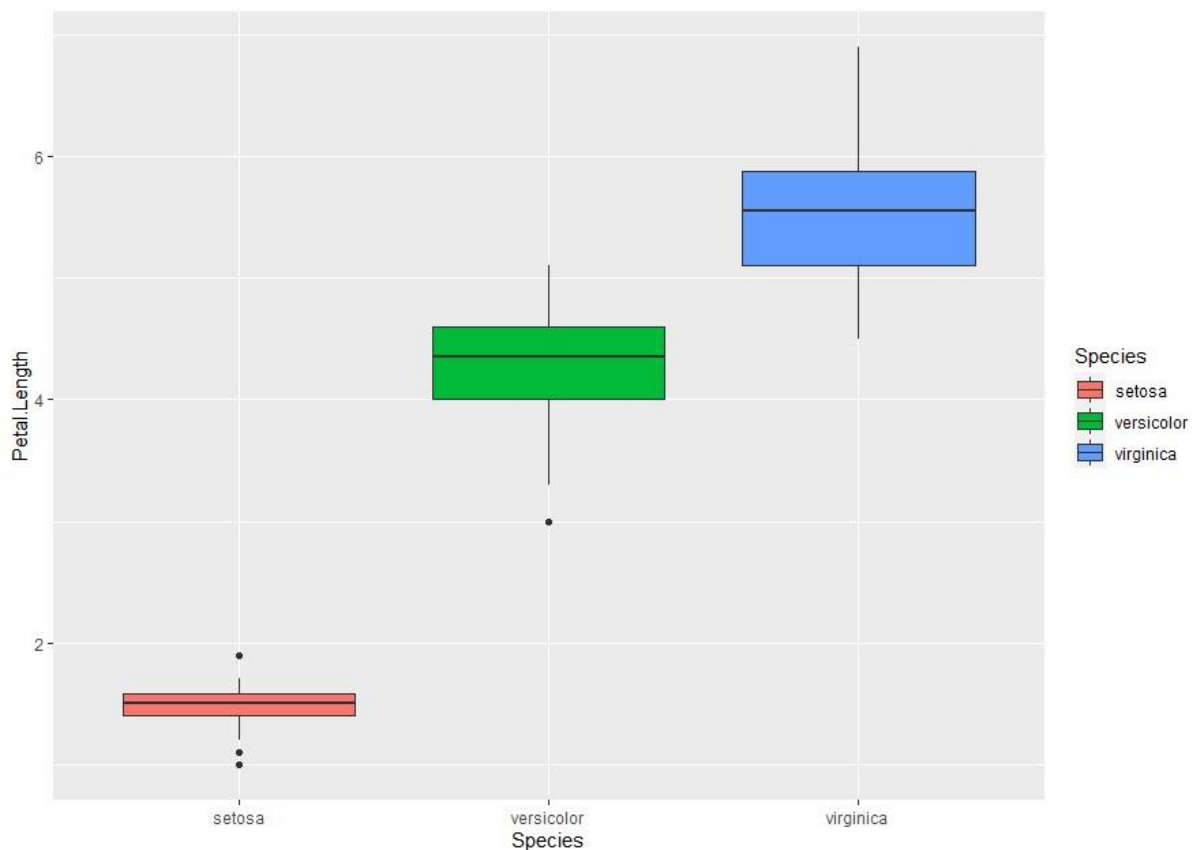
```

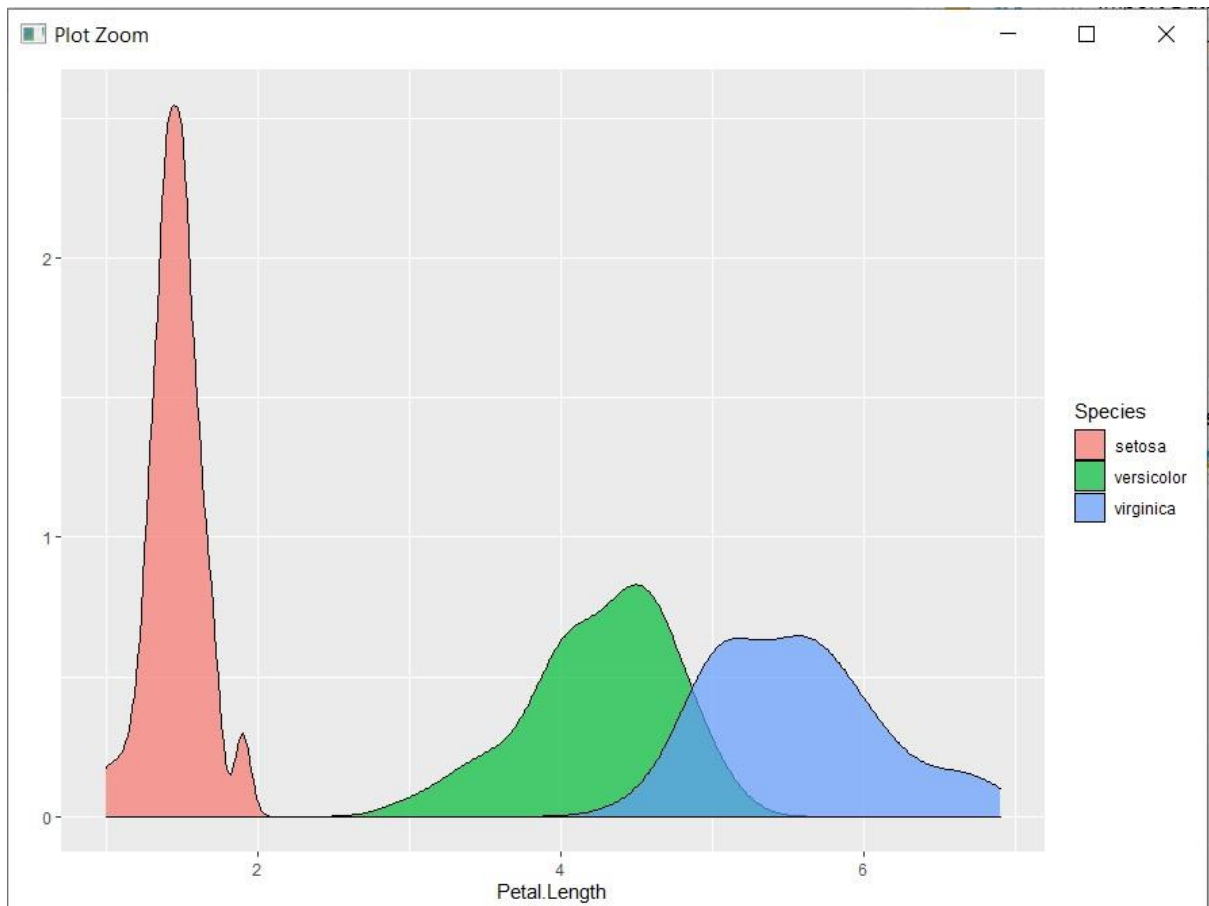
> print("Covariance Matrix")
[1] "Covariance Matrix"
> print(coi)
      Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length  0.6856935 -0.0424340  1.2743154  0.5162707
Sepal.Width   -0.0424340  0.1899794 -0.3296564 -0.1216394
Petal.Length   1.2743154 -0.3296564  3.1162779  1.2956094
Petal.Width    0.5162707 -0.1216394  1.2956094  0.5810063
> #Finding the Multiple Pearson's Correlation Coefficients
> cci = cor(iris.nospecies)
> print("Multiple Pearson's Correlation Coefficients")
[1] "Multiple Pearson's Correlation Coefficients"
> print(coi)
      Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length  0.6856935 -0.0424340  1.2743154  0.5162707
Sepal.Width   -0.0424340  0.1899794 -0.3296564 -0.1216394
Petal.Length   1.2743154 -0.3296564  3.1162779  1.2956094
Petal.Width    0.5162707 -0.1216394  1.2956094  0.5810063
> #Finding the Multiple Spearman Correlation Coefficients
> ccs = cor(iris.nospecies, method = "spearman")
> print("Multiple Spearman's Correlation Coefficients")
[1] "Multiple Spearman's Correlation Coefficients"
> print(ccs)
      Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length  1.0000000 -0.1667777  0.8818981  0.8342888
Sepal.Width   -0.1667777  1.0000000 -0.3096351 -0.2890317
Petal.Length   0.8818981 -0.3096351  1.0000000  0.9376668
Petal.Width    0.8342888 -0.2890317  0.9376668  1.0000000

+ print("Relationship b/w Petal Width and Sepal Length is Negative"))
[1] "Relationship b/w Petal Width and Sepal Length is Positive"
> if(cci[2,1] > 0){ print("Relationship b/w Sepal Width and Sepal Length is Positive")
+ } else {
+ print("Relationship b/w Sepal Width and Sepal Length is Negative"))
[1] "Relationship b/w Sepal Width and Sepal Length is Negative"

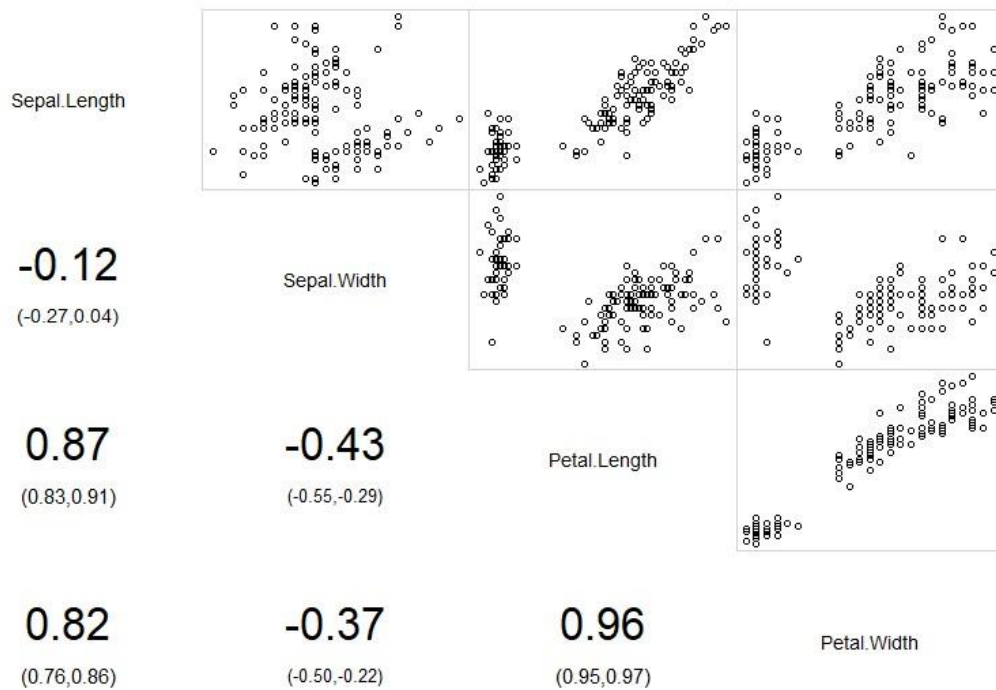
```

Plot Zoom

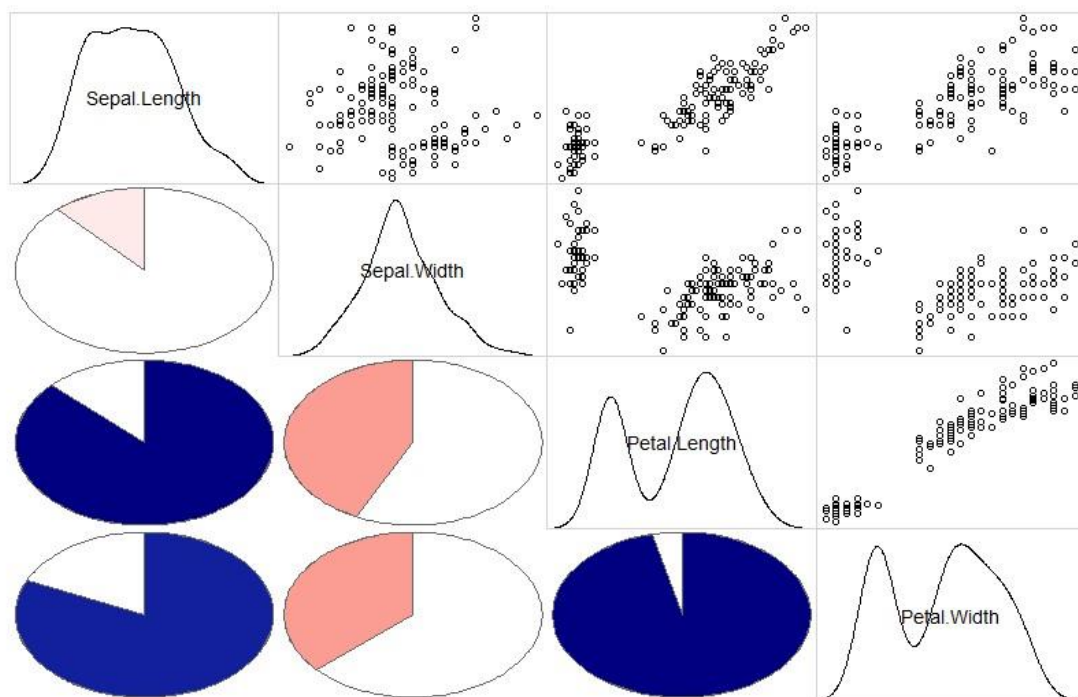




Plot Zoom



corrgram of petal and sepal measurements in iris data set

**RESULT:**

Thus, R program to Analysis of the Linear and Nonlinear variables using multiple correlations is successfully executed.

<b>EX.NO:09</b>	<b>BAYE'S RULE</b>
<b>DATE:</b>	

**AIM:**

To write a R program to predict whether A person is male or female based on height.

**CONCEPT:**

Bayes' Theorem states that the conditional probability of an event, based on the occurrence of another event, is equal to the likelihood of the second event given the first event multiplied by the probability of the first event.

**PROGRAM:**

```
uvtrain <- function(hm,hf)
{
  hmm = vector(mode="numeric",length=0)
  pmh = vector(mode="numeric",length=0)
  hmin = min(hm)-15  hmax = max(hm)+15
  mm = mean(hm);  vm= var(hm);
  cat("Mean of Male Height",mm,"\n")
  cat("variance of Male Height ",vm)
  for(h in hmin:hmax)
  {
    r = (h-mm)^2/vm
    p = (1/(sqrt(2*pi*vm)))*exp(-0.5*r);
    hmm <- c(hmm,h)
    pmh <- c(pmh,p)
  }
  hff= vector (mode="numeric",
length=0)
  pff=vector(mode="numeric",length=0)

  hmin = min(hf)-15
```

```

hmax = max(hf)+15
mf = mean(hf)
vf = var(hf)
cat("Mean of Female Height",mf,"\n")
cat("variance of Female Height",vf)
for(h in hmin:hmax)
{
  r = (h-mf)^2/vf
  p = (1/(sqrt(2*pi*vf)))*exp(-0.5*r);
  hff <- c(hff,h)   pff <- c(pff,p)
}
plot(hmm,pmh,type="l",col="red",pch=9,xlim=c(min(hff),max(hmm)),
xlab="Height  of Person(Male and Female)",
ylab="p(male | hm) and p(female | hf)",main="Normal Density")
lines(hff,pff,col = "blue")
return(list(mm,vm,mf,vf))
}
#Bayes Rule Testing Function Using Normal Density
uvtest <- function(mm,vm,mf,vf,ht)
{
  #finding probability of Male wrt Height
  rm = (ht-mm)^2/vm
  pm=(1/(sqrt(2*pi*vm)))*exp(-0.5*rm)
  #finding probability of Female wrt Height
  rf = (ht-mf)^2/vf
  pf=(1/(sqrt(2*pi*vf)))*exp(-0.5*rf)
  if(pm>pf)
    print("The given Height of Person is Male")
  else
    print("The given Height of Person is Female")
}

```

```

#clear the console screen
cat("\014")

#Train Function Call
hm<-c(165,170,160,154,175,155,167,177,158,178)
hf<-c(140,145,149,152,157,135,139,160,155,163)
mv = uvtrain(hm,hf)

#Testing Function Call
ht=as.numeric(readline(prompt='Enter the height of person for prediction='))
mm=as.numeric(mv[1])
vm=as.numeric(mv[2])
mf=as.numeric(mv[3])
vf=as.numeric(mv[4])
uvtest(mm,vm,mf,vf,ht)

```

## OUTPUT:

```

> #Train Function Call
> hm<-c(165,170,160,154,175,155,167,177,158,178)
> hf<-c(140,145,149,152,157,135,139,160,155,163)

> mv = uvtrain(hm,hf)
Mean of Male Height 165.9
variance of Male Height 80.98889Mean of Female Height 149.5
variance of Female Height 90.72222
> #Testing Function Call
> ht=as.numeric(readline(prompt='Enter the height of person for prediction='))
Enter the height of person for prediction=170

> mm=as.numeric(mv[1])
> vm=as.numeric(mv[2])
> mf=as.numeric(mv[3])
> vf=as.numeric(mv[4])

> uvtest(mm,vm,mf,vf,ht)
[1] "The given Height of Person is Male"

```



```

> #Train Function Call
> hm<-c(165,170,160,154,175,155,167,177,158,178)

> hf<-c(140,145,149,152,157,135,139,160,155,163)

> mv = uvtrain(hm,hf)
Mean of Male Height 165.9
variance of Male Height 80.98889Mean of Female Height 149.5
variance of Female Height 90.72222
> #Testing Function Call
> ht=as.numeric(readline(prompt='Enter the height of person for prediction='))
Enter the height of person for prediction=140

> mm=as.numeric(mv[1])

> vm=as.numeric(mv[2])

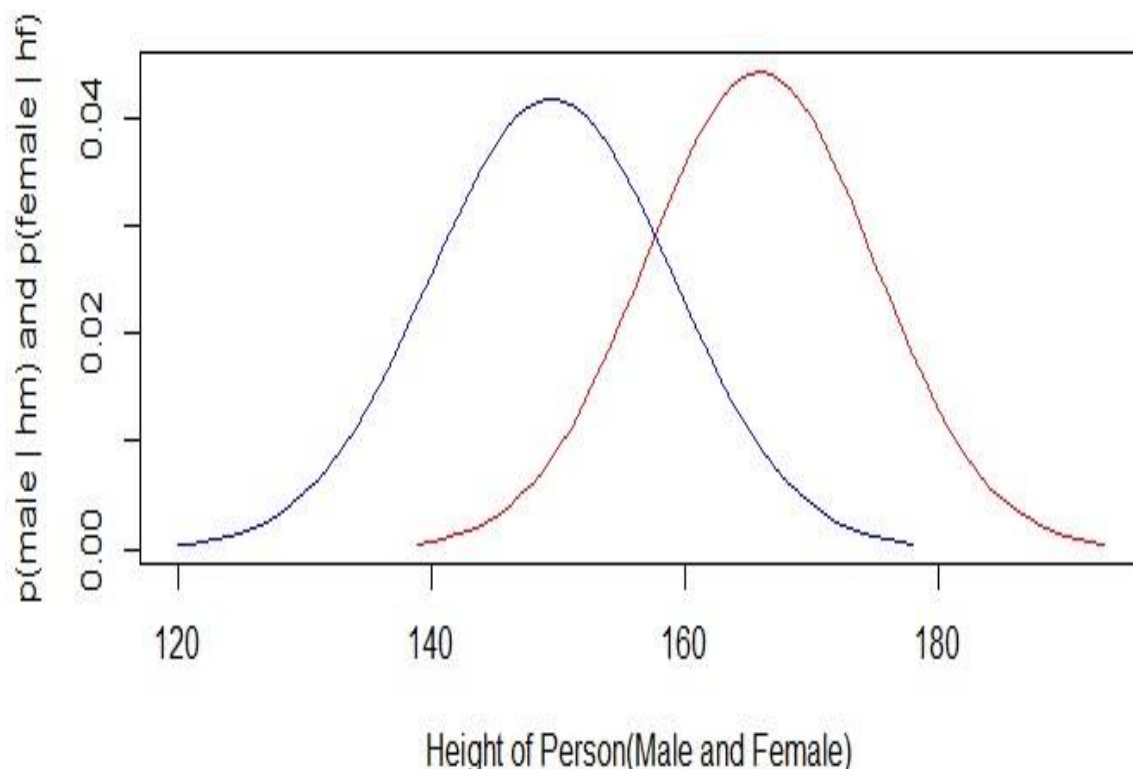
> mf=as.numeric(mv[3])

> vf=as.numeric(mv[4])

> uvtest(mm,vm,mf,vf,ht)
[1] "The given Height of Person is Female"
>

```

## Bayes Rule with Normal Density



### RESULT:

Thus, the R program to predict whether A person is male or female based on height.

<b>EX.NO: 10</b>	<b>RAINFALL PREDICTION</b>
<b>DATE :</b>	

**AIM:**

To write a R program to find prediction of rainfall using sample mean and population with US precipitation cities data.

**PROGRAM:**

```

head(precip)
is.vector(precip)
mean(precip)
t.statistic <- function(thesample, thepopulation)
{
  numerator <- mean(thesample) - mean(thepopulation)
  denominator <- sd(thesample) / sqrt(length(thesample))
  t.stat <- numerator / denominator
  return(t.stat)
}
population.precipitation <- rnorm(100000, mean=38)
t.stats <- numeric(10000)
for(i in 1:10000)
{
  a.sample <- sample(population.precipitation, 70)
  t.stats[i] <- t.statistic(a.sample, population.precipitation)
}
library(ggplot2)
tmpdata <- data.frame(vals=t.stats)
qplot(vals, data=tmpdata, geom="histogram",color=I("white"),xlab="sampling distribution of
t-statistic",ylab="frequency")
t.statistic(precip, population.precipitation)
qt(.025, df=69)
t.test(precip, mu=38)

```

```
t.test(precip, mu=38, alternative="less")
```

## OUTPUT:

```
> head(precip)
      Mobile      Juneau      Phoenix Little Rock Los Angeles Sacramento
      67.0       54.7       7.0       48.5      14.0       17.2
> is.vector(precip)
[1] TRUE
> mean(precip)
[1] 34.88571
> t.statistic <- function(thesample, thepopulation)
+ {
+   numerator <- mean(thesample) - mean(thepopulation)
+   denominator <- sd(thesample) / sqrt(length(thesample))
+   t.stat <- numerator / denominator
+   return(t.stat)
+ }
> population.precipitation <- rnorm(100000, mean=38)
> t.stats <- numeric(10000)
> for(i in 1:10000)
+ {
+   a.sample <- sample(population.precipitation, 70)
+   t.stats[i] <- t.statistic(a.sample, population.precipitation)
+ }
> library(ggplot2)

> t.statistic(precip, population.precipitation)
[1] -1.900501
> qt(.025, df=69)
[1] -1.994945
> t.test(precip, mu=38)
```

One Sample t-test

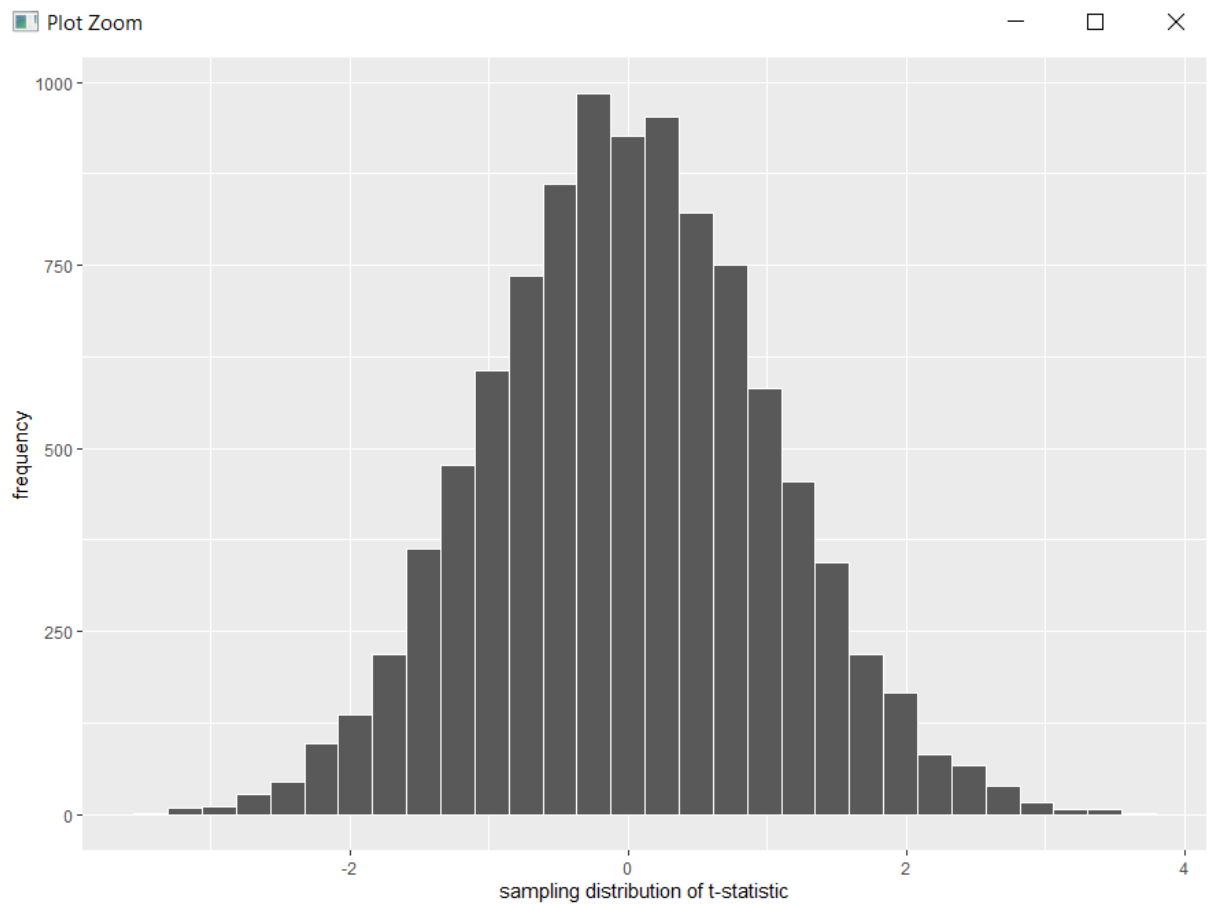
```
data: precip
t = -1.901, df = 69, p-value = 0.06148
alternative hypothesis: true mean is not equal to 38
95 percent confidence interval:
 31.61748 38.15395
sample estimates:
mean of x
 34.88571
```

```
> t.test(precip, mu=38, alternative="less")
```

One Sample t-test

```
data: precip
t = -1.901, df = 69, p-value = 0.03074
alternative hypothesis: true mean is less than 38

95 percent confidence interval:
 -Inf 37.61708
sample estimates:
mean of x
 34.88571
```



## RESULT:

Thus , R program to find prediction of rainfall using sample mean and population with US precipitation cities data is successfully executed.

<b>EX.NO:11</b>	<b>TESTING TWO MEANS</b>
<b>DATE :</b>	

**AIM:**

To write a R program to test the hypothesis which proves the mileage is better for manual cars than cars with automatic transmission using two means from MTcars dataset.

**PROGRAM:**

```
library(car)
head(WeightLoss)
table(WeightLoss$group)
qplot(group, wl2, data=WeightLoss, geom="boxplot", fill=group)
the.anova <- aov(wl2 ~ group, data=WeightLoss)
summary(the.anova)
pairwise.t.test(WeightLoss$wl2, as.vector(WeightLoss$group))
mean(mtcars$mpg[mtcars$am==0])
mean(mtcars$mpg[mtcars$am==1])
mtcars.copy <- mtcars
mtcars.copy$transmission <- ifelse(mtcars$am==0,"auto", "manual")
mtcars.copy$transmission <- factor(mtcars.copy$transmission)
qplot(transmission, mpg, data=mtcars.copy,geom="boxplot", fill=transmission)
automatic.mpgs <- mtcars$mpg[mtcars$am==0]
manual.mpgs <- mtcars$mpg[mtcars$am==1]
t.test(automatic.mpgs, manual.mpgs, alternative="less")
t.test(mpg ~ am, data=mtcars, alternative="less")
```

## OUTPUT:

```
> mean(mtcars$mpg[mtcars$am==0])
[1] 17.14737
> mean(mtcars$mpg[mtcars$am==1])
[1] 24.39231
> mtcars.copy <- mtcars
> mtcars.copy$transmission <- ifelse(mtcars$am==0,"auto", "manual")
> mtcars.copy$transmission <- factor(mtcars.copy$transmission)
> qplot(transmission, mpg, data=mtcars.copy,geom="boxplot", fill=transmission)
> automatic.mpgs <- mtcars$mpg[mtcars$am==0]
> manual.mpgs <- mtcars$mpg[mtcars$am==1]
> t.test(automatic.mpgs, manual.mpgs, alternative="less")
```

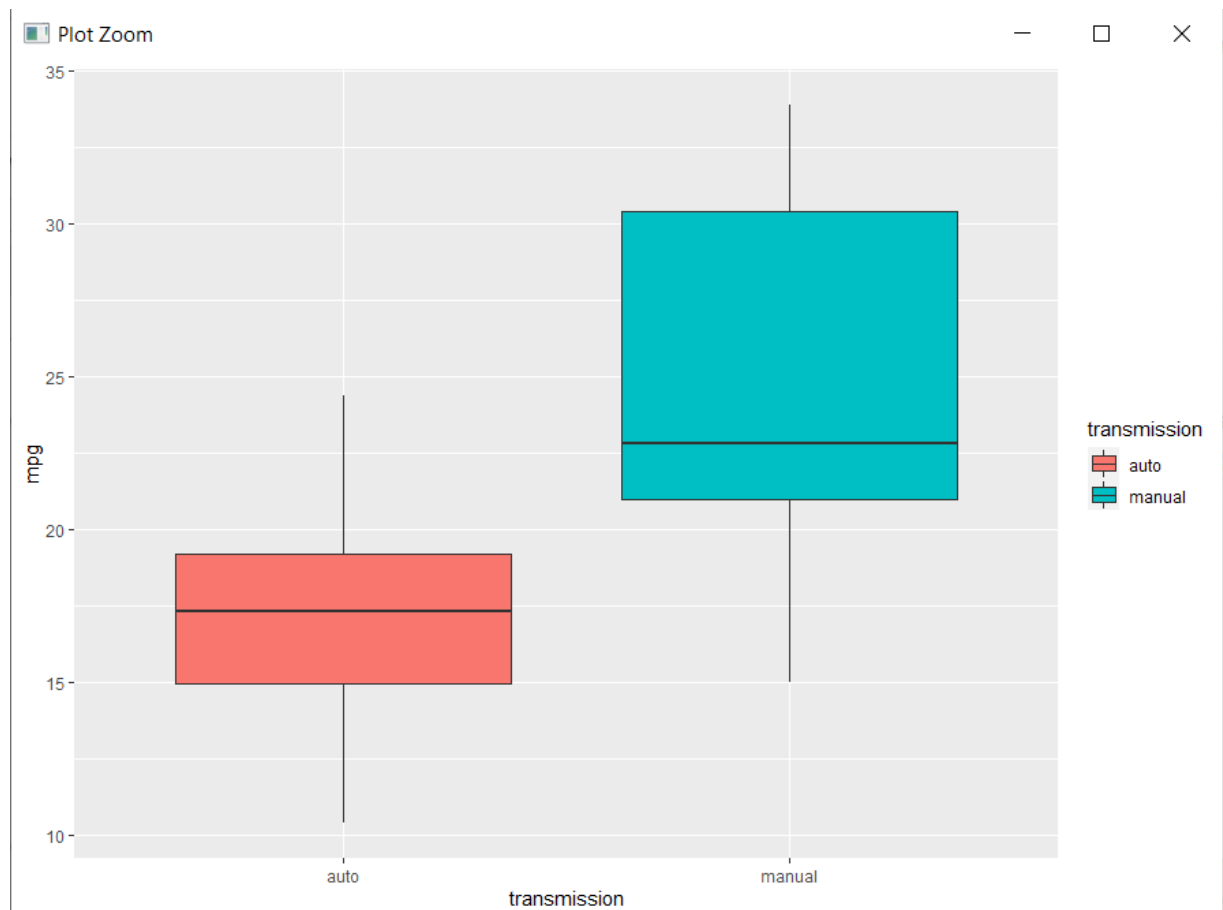
Welch Two Sample t-test

```
data: automatic.mpgs and manual.mpgs
t = -3.7671, df = 18.332, p-value = 0.0006868
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
 -Inf -3.913256
sample estimates:
mean of x mean of y
 17.14737  24.39231
```

```
> t.test(mpg ~ am, data=mtcars, alternative="less")
```

Welch Two Sample t-test

```
data: mpg by am
t = -3.7671, df = 18.332, p-value = 0.0006868
alternative hypothesis: true difference in means between group 0 and group 1 is less than 0
95 percent confidence interval:
 -Inf -3.913256
sample estimates:
mean in group 0 mean in group 1
 17.14737      24.39231
```



## RESULT:

Thus, R program to test the hypothesis which proves the mileage is better for manual cars than cars with automatic transmission using two means from MTcars dataset.

<b>EX.NO:12</b>	<b>SIMPLE LINEAR REGRESSION</b>
<b>DATE :</b>	

### AIM:

To write a R program to predict the mileage of car based on weight of car using simple linear regression.

### PROGRAM:

```
# Clear the Console Screen
cat("\014")
packages="datasets"
#Loading the MTCars Datasets
print("Training Data:\n")
head(mtcars)
# Fit a simple linear regression model using lm()
model <- lm(mpg ~ wt, data=mtcars)
# Plot of the linear model
plot(mtcars$wt,mtcars$mpg,main = "Simple Linear Regression (Mileage
and Weight)")
#Straight Line Equation:  $y = 37.285 - 5.345x$ 
abline(model,col = "red")
# Summary of the linear model
summary(model)
print("Testing Data:")
# Predicted car mileage based car weight (6,000 Pounds)
pred_mpg=predict(model, newdata=data.frame(wt=6))
cat("Mileage per Gallons (Predicted):", pred_mpg)
# Model coefficients
coeff=model$coefficients
# y-Intercept
cat('y-Intercept (b0) :',coeff[1])
# Coefficients
```



```
cat('Coefficients (b1) :',coeff[2])
```

### #Multiple regression

```
model <- lm(mpg ~ wt + hp, data=mtcars)
```

```
summary(model)
```

```
coef(lm(mpg ~ wt + hp, data=mtcars))
```

```
coef(lm(mpg ~ wt, data=mtcars))
```

```
coef(lm(mpg ~ hp, data=mtcars))
```

```
x<-predict(model, newdata = data.frame(wt=2.5, hp=275))
```

x

### OUTPUT:

```
> packages="datasets"
> #Loading the MTCars Datasets
> print("Training Data:\n")
[1] "Training Data:\n"
> head(mtcars)
      mpg  cyl  disp  hp drat    wt  qsec vs  am  gear carb
Mazda RX4     21.0   6  160 110 3.90 2.620 16.46 0   1    4    4
Mazda RX4 Wag  21.0   6  160 110 3.90 2.875 17.02 0   1    4    4
Datsun 710     22.8   4  108  93 3.85 2.320 18.61 1   1    4    1
Hornet 4 Drive  21.4   6  258 110 3.08 3.215 19.44 1   0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02 0   0    3    2
Valiant        18.1   6  225 105 2.76 3.460 20.22 1   0    3    1
> # Fit a simple linear regression model using lm()
> model <- lm(mpg ~ wt, data=mtcars)
> # Plot of the linear model
> plot(mtcars$wt,mtcars$mpg,main = "Simple Linear Regression (Mileage
+ and Weight)")
> #Straight Line Equation: y = 37.285 - 5.345x
> abline(model,col = "red")
> # Summary of the linear model
> summary(model)

Call:
lm(formula = mpg ~ wt, data = mtcars)
```

```

Residuals:
    Min       1Q   Median       3Q      Max
-4.5432 -2.3647 -0.1252  1.4096  6.8727

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.2851     1.8776   19.858 < 2e-16 ***
wt           -5.3445     0.5591   -9.559 1.29e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.046 on 30 degrees of freedom
Multiple R-squared:  0.7528,    Adjusted R-squared:  0.7446
F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10

> print("Testing Data:")
[1] "Testing Data:"
> # Predicted car mileage based car weight (6,000 Pounds)
> pred_mpg=predict(model, newdata=data.frame(wt=6))
> cat("Mileage per Gallons (Predicted):", pred_mpg)
Mileage per Gallons (Predicted): 5.218297> # Model coefficients
> coeff=model$coefficients
> # y-Intercept
> cat('y-Intercept (b0) :',coeff[1])
y-Intercept (b0) : 37.28513> # Coefficients

Call:
lm(formula = mpg ~ wt + hp, data = mtcars)

Residuals:
    Min       1Q   Median       3Q      Max
-3.941 -1.600 -0.182  1.050  5.854

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.22727     1.59879   23.285 < 2e-16 ***
wt           -3.87783     0.63273   -6.129 1.12e-06 ***
hp           -0.03177     0.00903   -3.519 0.00145 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.593 on 29 degrees of freedom
Multiple R-squared:  0.8268,    Adjusted R-squared:  0.8148
F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12

> coef(lm(mpg ~ wt + hp, data=mtcars))
            (Intercept)           wt           hp
37.22727012 -3.87783074 -0.03177295
> coef(lm(mpg ~ wt, data=mtcars))
            (Intercept)           wt
37.285126 -5.344472
> coef(lm(mpg ~ hp, data=mtcars))
            (Intercept)           hp
30.09886054 -0.06822828
> x<-predict(model, newdata = data.frame(wt=2.5, hp=275))
> x
      1
18.79513

```

