**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY FACULTY OF SCIENCE AND HUMANITIES DEPARTMENT OF COMPUTER APPLICATIONS**



**PRACTICAL RECORD NOTE**

**STUDENT NAME :**

**REGISTER :**

**NUMBER**

**CLASS : MCA SECTION : G**

**YEAR & : I YEAR & II SEM**

**SEMESTER**

**SUBJECT CODE : PCA20S02J**

**SUBJECT : DATA ANALYSIS USING R**

**TITLE**

# APRIL 2024



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY FACULTY OF SCIENCE AND HUMANITIES DEPARTMENT OF COMPUTER APPLICATIONS**

SRM Nagar, Kattankulathur – 603 203

CERTIFICATE

*Certified to be the bonafide record of practical work done by*

*Register No. of MCA Degree*

*course for PCA20S02J – DATA ANALYSIS USING R in the Computer lab in SRM*

*Institute of Science and Technology during the academic year2023-2024.*

*Staff In-charge Head of the Department*

*Submitted for Semester Practical Examination held on .*

*Internal Examiner External Examiner*

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **DATE** | **TITLE** | **PAGE. NO** | **STAFF SIGN** |
| **1** |  | **INSTALLATION OF R AND R-STUDIO** | **1** |  |
| **2** |  | **WORKING WITH R PACKAGES** | **5** |  |
| **3** |  | **BUILT – IN FUNCTIONS** | **9** |  |
| **4** |  | **USER - DEFINED FUNCTIONS** | **14** |  |
| **5** |  | **DESCRIPTIVE STATISTICS** | **17** |  |
| **6** |  | **VECTOR MANIPULATION** | **21** |  |
| **7** |  | **MATRIX MANIPULATION** | **25** |  |
| **8** |  | **LIST MANIPULATION** | **28** |  |
| **9** |  | **DATA FRAMES IN R** | **31** |  |
| **10** |  | **IMPORTING FILES IN R** | **34** |  |
| **11** |  | **IMPLEMENTING NAVIE BAYES USING IRIS DATASET** | **37** |  |
| **12** |  | **BREAST CANCER PREDICTION USING KNN** | **40** |  |
| **13** |  | **DIABETIES PREDICTION USING DECISION TREE** | **43** |  |
| **14** |  | **WINE QUALITY PREDICTION USING RANDOM FOREST** | **47** |  |
| **15** |  | **CLUSTERING USARREST DATASET USING KMEANS ALGORITHM** | **51** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **16** |  | **CLUSTERING USARRESTS USING AGGLOMERATIVE HIERARCHICAL CLUSTERING** | **55** |  |
| **17** |  | **CLUSTERING USARRESTS USING DIVISIVE HIERARCHICAL CLUSTERING** | **57** |  |
| **18** |  | **CLUSTERING IRIS DATASET USING DBSCAN** | **59** |  |
| **19** |  | **VISUALIZATIONS** | **62** |  |

**EX.NO : 01**

**DATE :**

# AIM :

**INSTALLATION OF R AND R-STUDIO**

To install R and R-Studio.

# PROCEDURE :

**INSTALLATION OF R**

**STEP 1 :** Go to [CRAN R project](https://cran.r-project.org/) website - <https://cran.r-project.org/>

**STEP 2 :** Click on the Download R for Windows link.

**STEP 3 :** Click on the base subdirectory link or install R for the first time link.

**STEP 4 :** Click Download R 4.2.1 for Windows and save the executable

.exe file.

**STEP 5 :** Run the .exe file and follow the installation instructions.

**STEP 6 :** Select the desired language and then click Next.

**STEP 7 :** Read the license agreement and click Next.

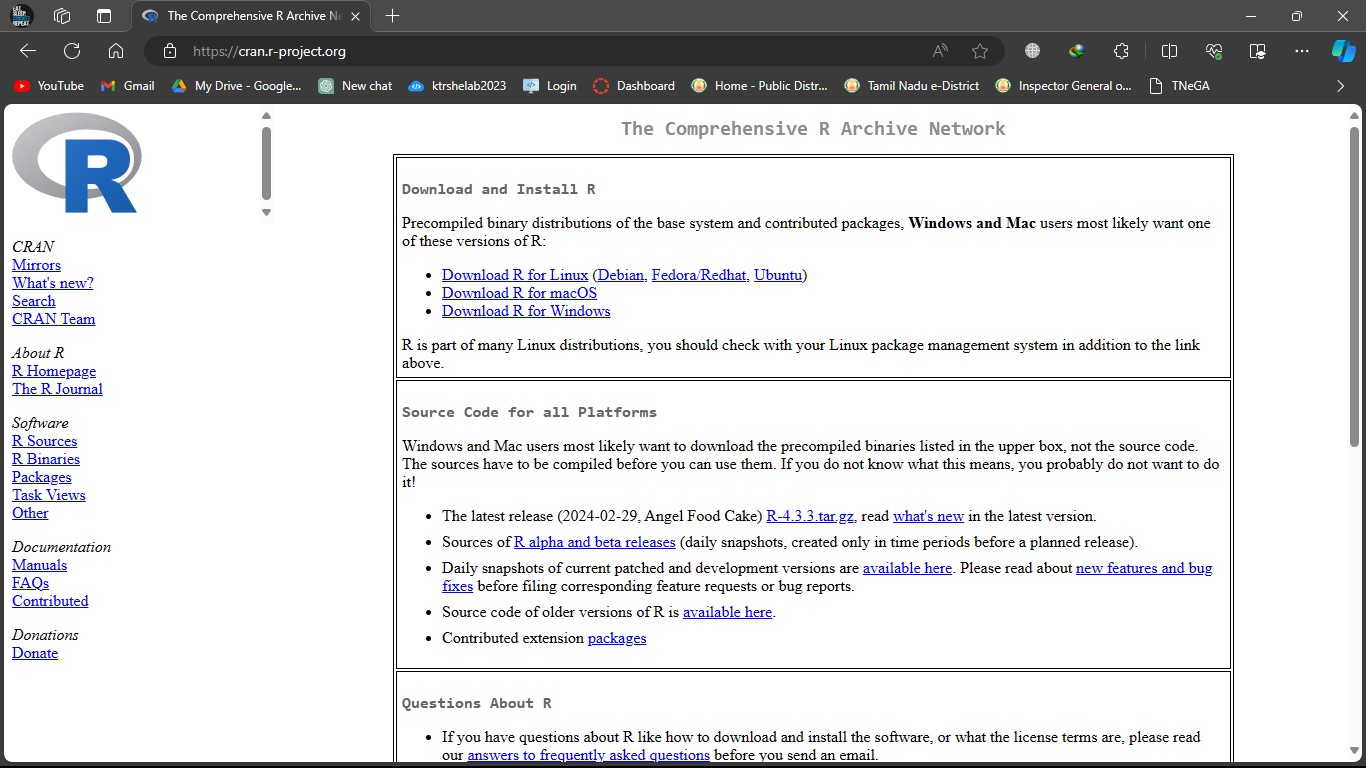
**STEP 8 :** Select the components you wish to install, Click Next.

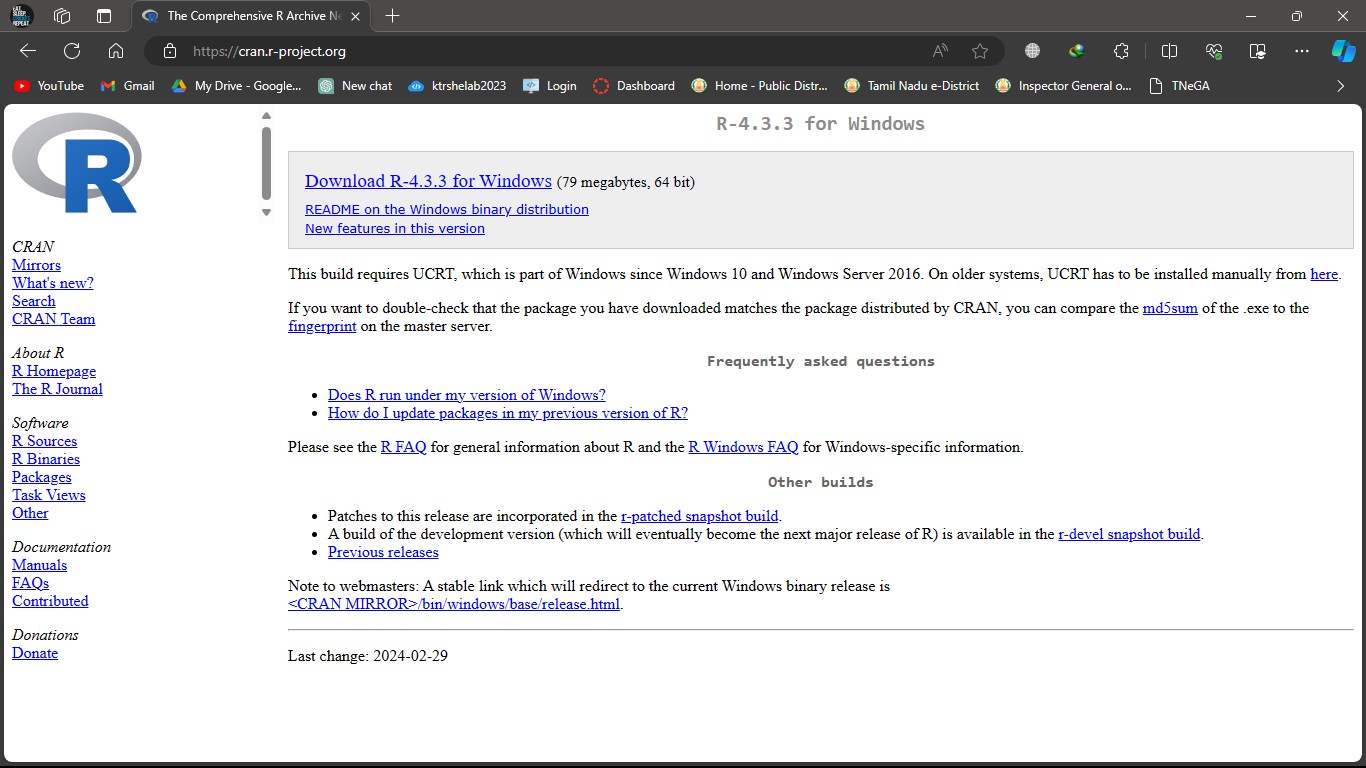
**STEP 9 :** Browse the folder/path you wish to install R into and then confirm by clicking Next.

**STEP 10 :** Select additional tasks like creating desktop shortcuts etc, then click Next.

**STEP 11 :** Wait for the installation process to complete.

# OUTPUT :





**INSTALLATION OF R – STUDIO**

**STEP 1 :** With R-base installed, let’s move on to installing R-Studio. To begin, go to download RStudio - <https://posit.co/> downloads and click on the download button for R-Studio desktop.

**STEP 2 :** Click on the link for the windows version of R-Studio and save the .exe file.

**STEP 3 :** Run the .exe and follow the installation instructions.

**STEP 4 :** Click Next on the welcome window.

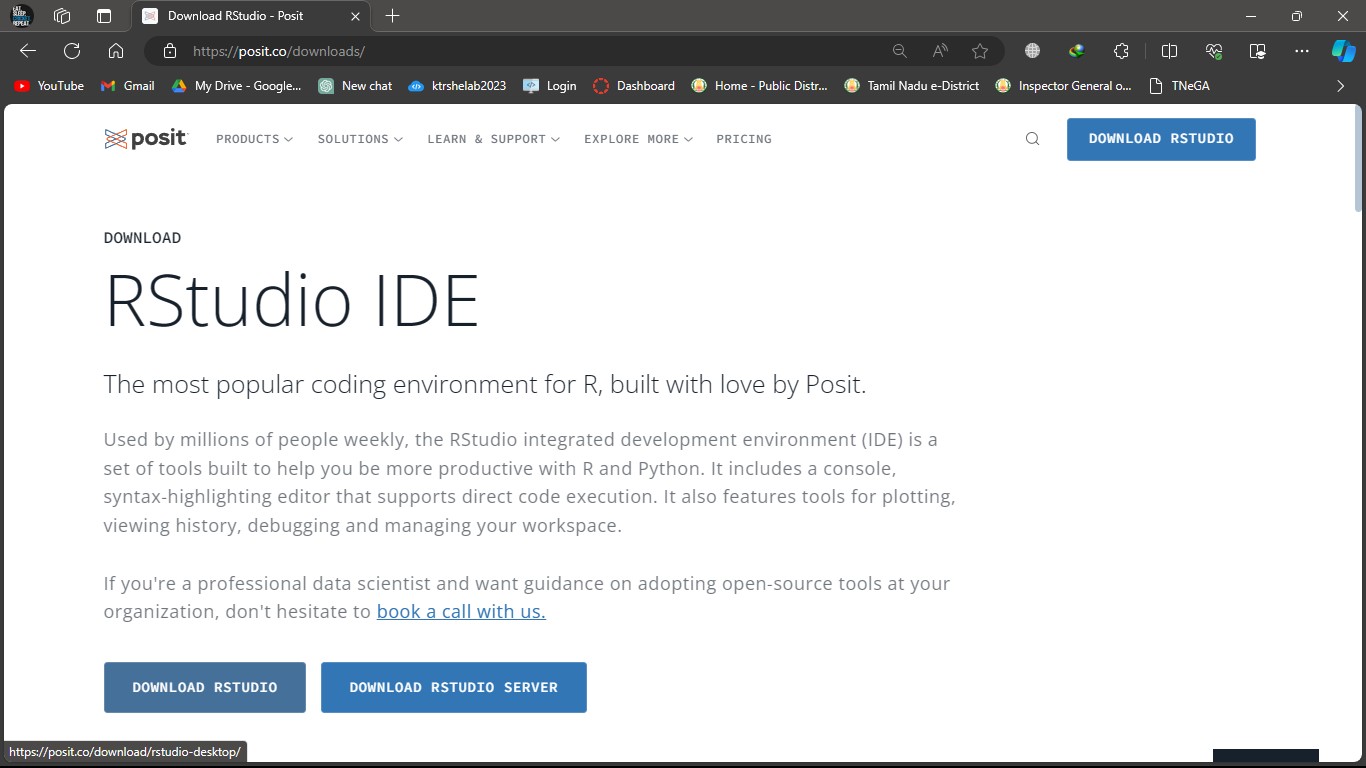
**STEP 5 :** Enter/browse the path to the installation folder and click Next to proceed.

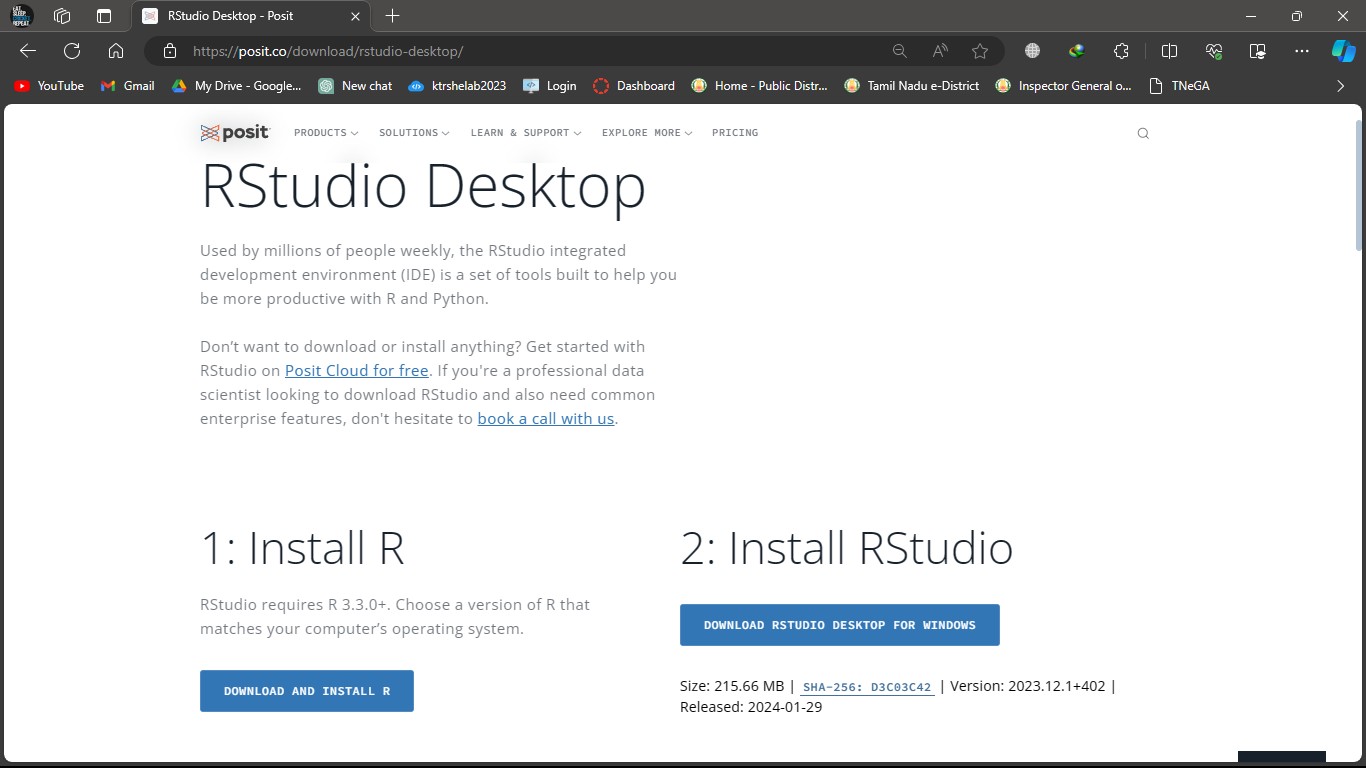
**STEP 6 :** Select the folder for the start menu shortcut and then click Next.

**STEP 7 :** Wait for the installation process to complete.

**STEP 8 :** Click on Finish to complete the installation.

# OUTPUT :





**EX.NO : 02**

**DATE :**

# AIM :

**WORKING WITH R PACKAGES**

To install, load and work with packages.

# PACKAGES :

**INSTALL R PACKAGES FROM CRAN :**

**install.packages ( )** - This function is used to install a required package in the R programming language.

**Example :**- To install ggplot2 package

# install.packages(“ggplot2”)

**UNINSTALL R PACKAGES :**

**remove.packages ( )** - This function is used to uninstall apackage in the R programming language.

**Example :-** To uninstall ggplot2 package

# remove.packages(“ggplot2”)

**LOADING OF R PACKAGES :**

**library ( )** - It is used to load and list all the packages in the R Programming language.

**Example :-** To load ggplot2 package

# library ( ggplot2 )

To list all the packages installed

# library ( )

**UPDATING R PACKAGES :**

**old.packages ( )** - It is used to check which packages need an update in R.

**Example :-** To check an update

# old.packages( )

**update.packages ( )** - It is used to update all the packages in R .

**Example :-** To update Packages

# update.packages( )

**LISTING THE PACKAGES THAT ARE INSTALLED :**

**install.packages ( )** - It is used to list out all packages which are installed in computer .

**Example :-** To list out installed packages

# installed.packages( )

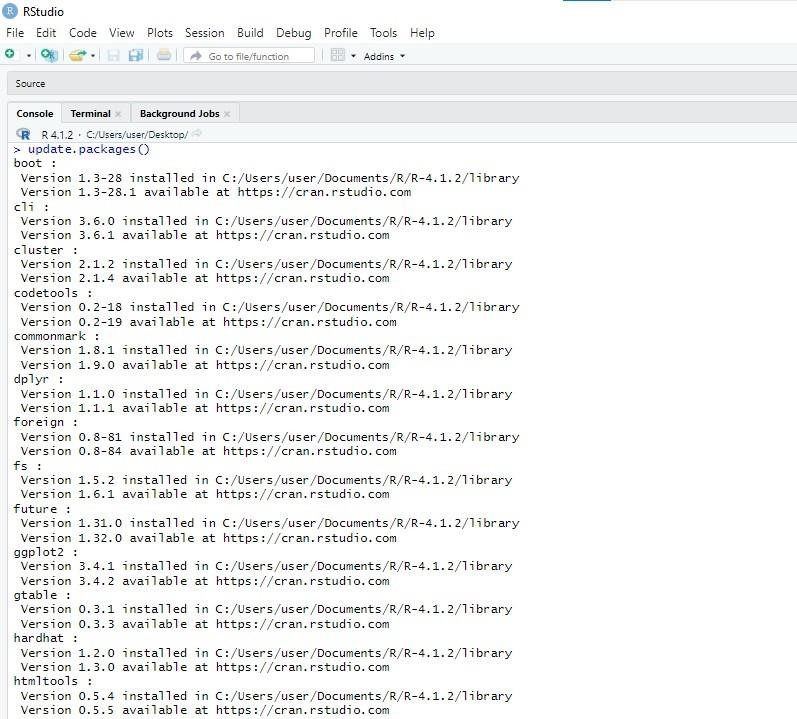
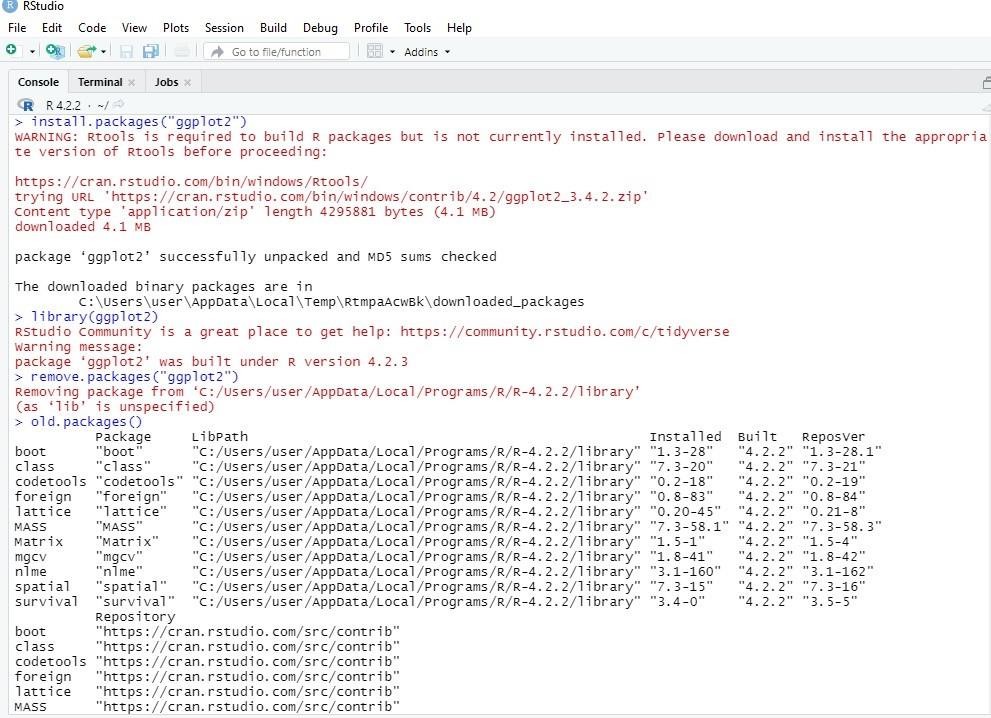
**GET HELP PAGES ABOUT PACKAGES :**

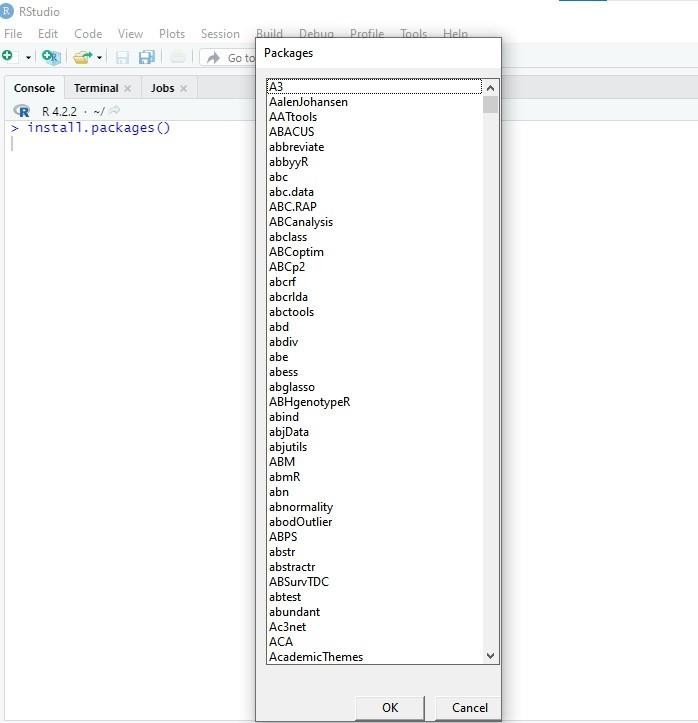
**Help ( ) and ( ? )** - They provide access to the documentation pages for R functions, data sets, and other objects, both for packages in the standard R distribution and for contributed packages.

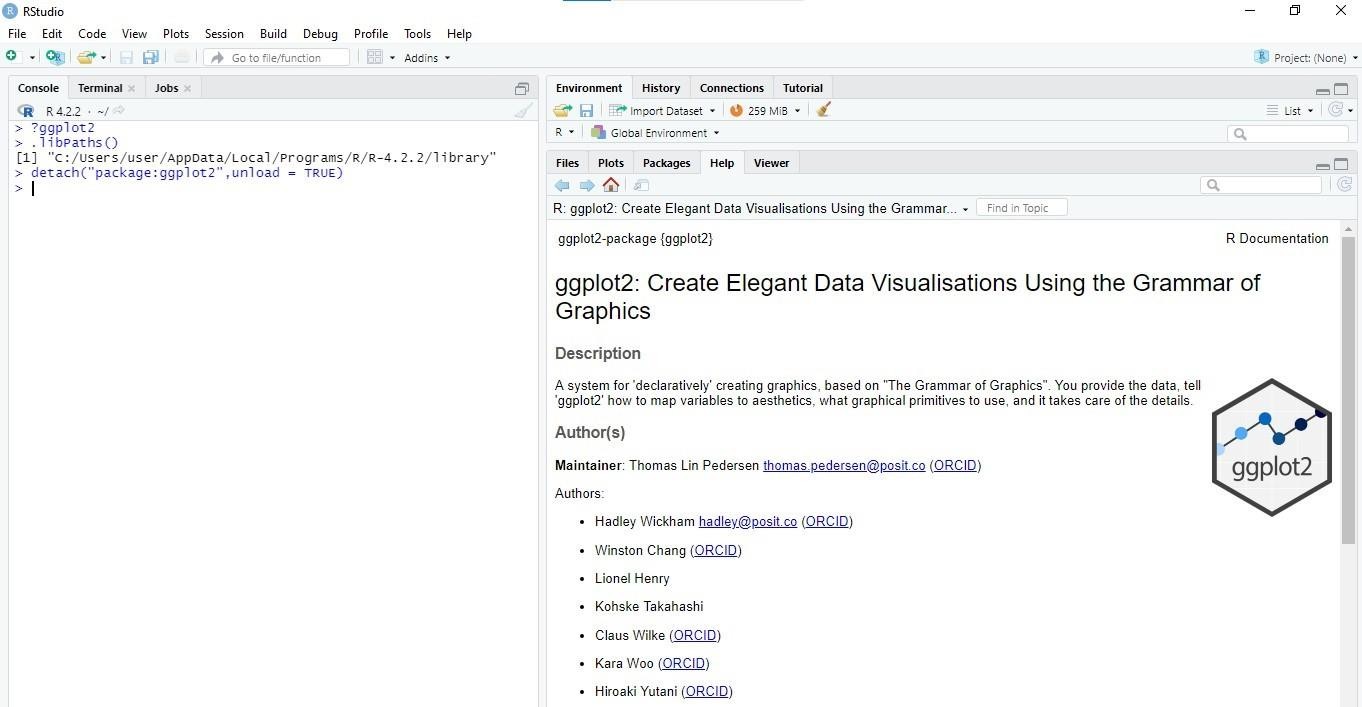
**Example :-** To get more description about ggplot2

# help(“ggplot2”) and ?ggplot2

**OUTPUT :**







**EX.NO : 03**

**DATE :**

# AIM :

**BUILT – IN FUNCTIONS**

To show the working of Built – In Functions.

# BUILT – IN FUNCTION’S :

|  |  |
| --- | --- |
| **FUNCTIONS** | **DESCRIPTIONS** |
| sum ( ) | It returns the sum of all the input vector. |
| prod ( ) | It returns the multiplication result of all the input vector. |
| max ( ) | It returns the maximum value of input vector. |
| min ( ) | It returns the minimum value of input vector. |
| unique ( ) | It returns the unique value. |
| sort ( ) | It returns sorted value from input vector. |
| rev ( ) | It returns reversed order of input vector. |
| rbind ( ) | It combines vector and matrix row-wise. |
| cbind ( ) | It combines vector and matrix column-wise. |
| setdiff ( ) | It returns differences between two vectors. |
| cumsum ( ) | It returns the cumulative sum of two vector. |

|  |  |
| --- | --- |
| abs ( ) | It returns the absolute value of input vector. |
| sqrt ( ) | It returns the square root of input vector. |
| ceiling ( ) | It returns the smallest integer which is larger than or equal to input vector. |
| floor ( ) | It returns the largest integer, which is smaller than or equal to input vector. |
| trunc ( ) | It returns the truncate value of input vector. |
| round ( ) | It returns round value of input vector. |
| cos( ), sin(), tan( ) | It returns cos( ), sin( ), tan( ) value of input vector. |
| log ( ) | It returns natural logarithm of input vector. |
| log10 ( ) | It returns common logarithm of input vector. |
| exp ( ) | It returns exponent of input vector. |

**PROGRAM :**

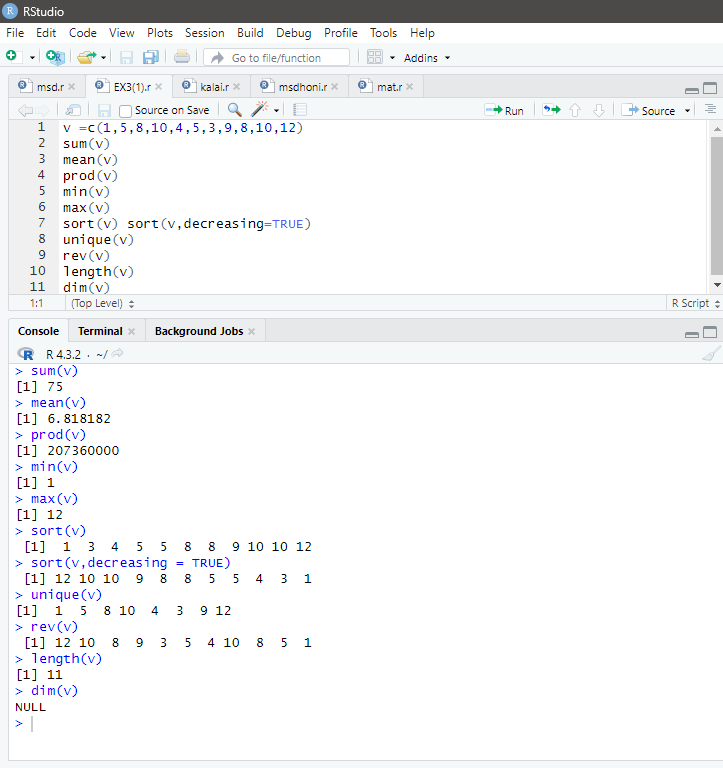
# Create a Vector ‘v’ with the values 1,5,8,10,4,5,3,9,8,10,12 and do the following :-

* 1. Find Sum, Mean and Product of the vector v =c(1,5,8,10,4,5,3,9,8,10,12)

sum(v) mean(v) prod(v)

* 1. Find the Minimum and the Maximum of the vector. min(v) max(v)
  2. Sort the vector in Ascending and Descending order. sort(v) sort(v,decreasing=TRUE)
  3. List the Distinct values in the vector. unique(v)
  4. Reverse the order of the vector. rev(v)
  5. Find the Length and the Dimension of the vector. length(v) dim(v)

# OUTPUT :



1. **Create two vectors as below :-**

A = 0,2,4,15

B = 3,12,4,11

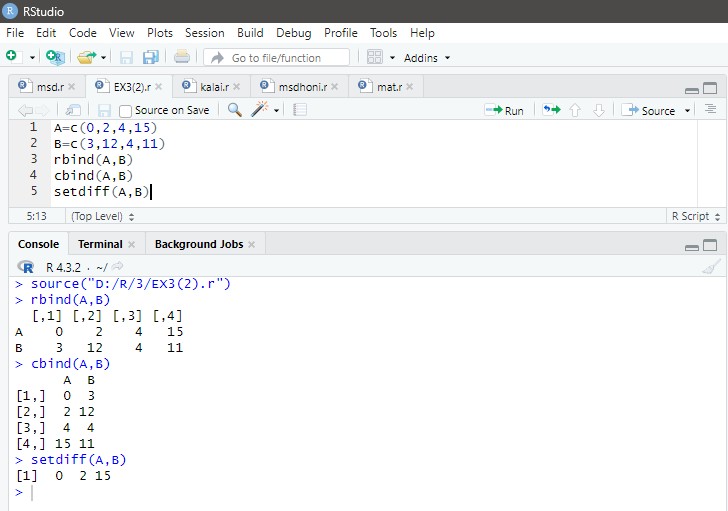
* 1. Combines these two vectors by column wise and row wise. A = c(0,2,4,15)

B = c(3,12,4,11)

rbind(A,B) cbind(A,B)

* 1. Find the elements of a ‘A’ vector that are not in ‘B’ vector. setdiff(A,B)

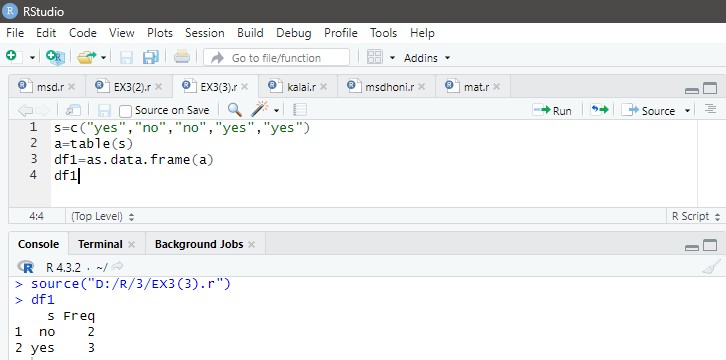
# OUTPUT :



1. A survey asks people if they smoke or not.The data is yes, no, no, yes, yes.
   1. Represent the above information in a vector. s=c(“yes”,”no”,”no”,”yes”,”yes”)
   2. Display frequency table for above information. a=table(s)

df1=as.data.frame(a) df1

# OUTPUT :



**EX.NO : 04**

**DATE :**

# AIM :

**USER - DEFINED FUNCTIONS**

To create and use User Defined function.

# USER DEFINED FUNCTION’S :

To declare a user-defined function in R, we use the keyword function.

# Syntax :

function\_name <- function(parameters)

{

function body

}

# PROGRAM :

1. Write a function "tables" in R with argument for table number and times to display the respective multiplication table. Use default value 15 for times.

table <- function(n,t=15)

{

for(i in 1:t)

{

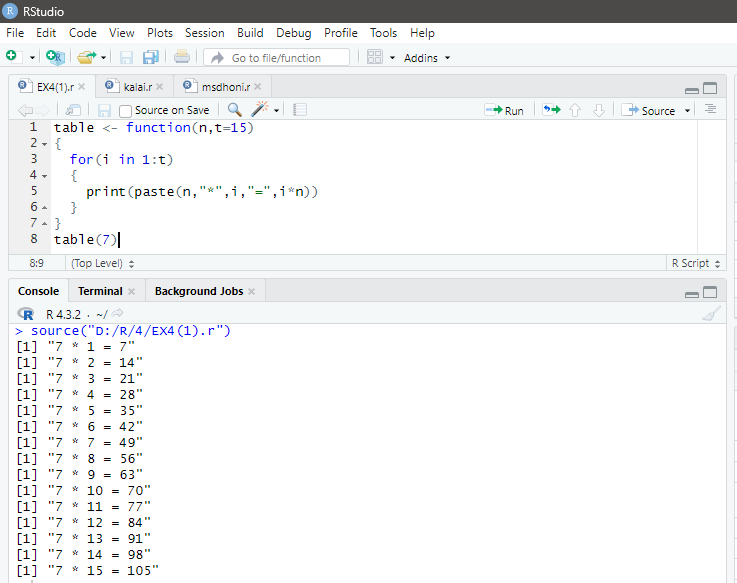
print(paste(n,"\*",i,"=",i\*n))

}

}

table(7)

# OUTPUT :



1. Write a function ‘fact’ in R to display the factorial value of the passed argument.

fact <- function(n)

{

f = 1

for (i in 1:n)

{

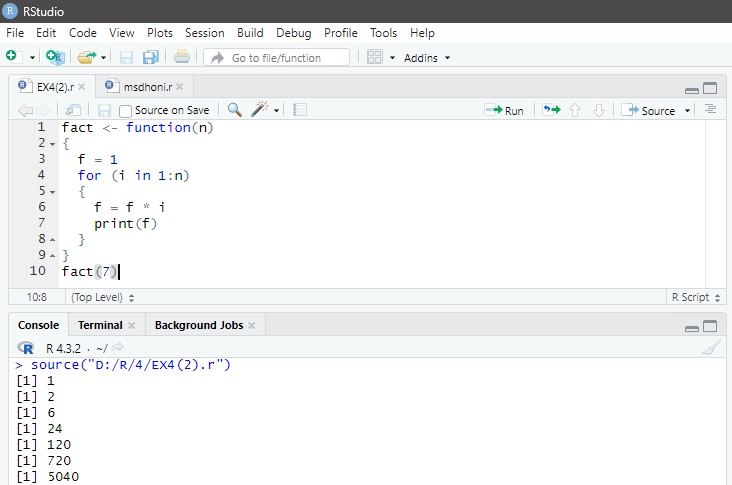
f = f \* i print(f)

}

}

fact(7)

# OUTPUT :



**EX.NO : 05**

**DATE :**

# AIM :

**DESCRIPTIVE STATISTICS**

To explore the commands that will give an overview of data to be used.

# DESCRIPTIVE STATISTICS:

Descriptive statistics is the branch of statistics that focuses on describing and gaining more insight into the data in its present state.

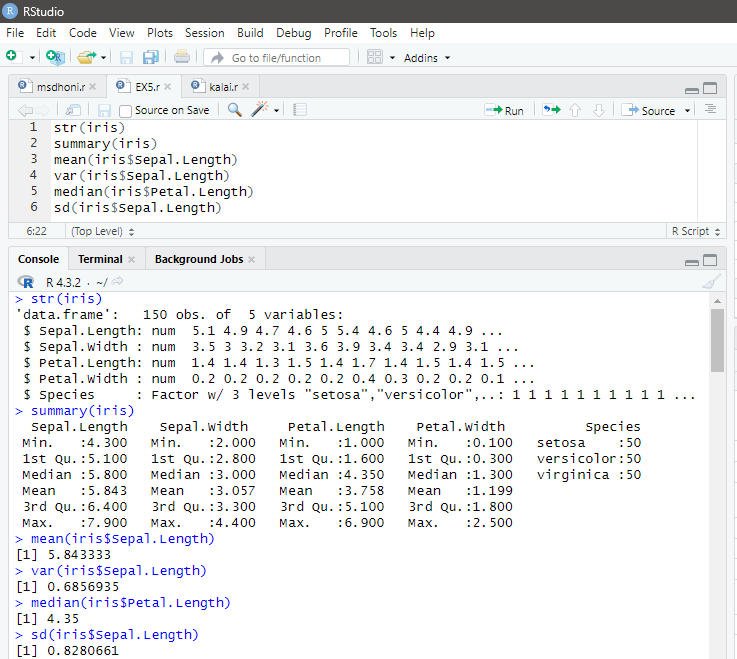
|  |  |
| --- | --- |
| **FUNCTIONS** | **DESCRIPTIONS** |
| mean() | It returns arithmetic average of a numeric input vector. |
| median() | It returns median of a numeric input vector. |
| var() | It returns variance of a numeric input vector. |
| sd() | It returns standard deviation of a numeric input vector. |
| range() | It returns the maximum and minimum value of a numeric input vector. |
| diff() | It returns the difference between pairs of consecutive elements of a numeric vector. |
| IQR() | It returns the interquartile range of a numeric input vector. |
| quantile() | It returns the sample quantile of a numeric input vector. |
| summary() | It returns summary statistics such as mean, median,  minimum, maximum, 1st quantile, 3rd quantile, etc. for each component in the object. |

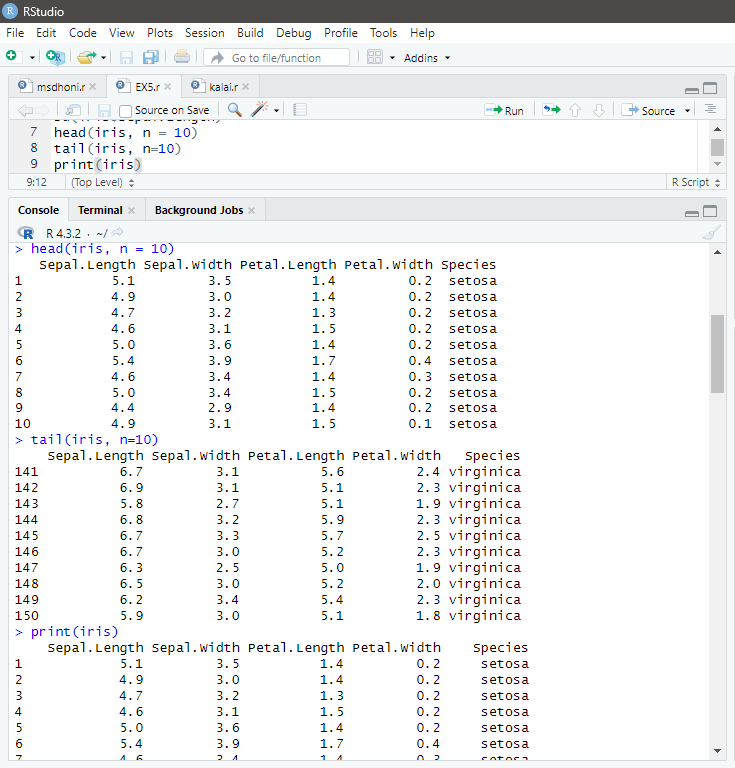
|  |  |
| --- | --- |
| str() | It displays the internal structure of an R object. |
| table() | It performs a tabulation of categorical variable and gives its frequency as output. |
| prop.table() | It calculates the proportions of a table, with the result presented as a table with proportions. |

# PROGRAM :

* 1. Display the structure and summary statistics of the iris dataset. str(iris) summary(iris)
  2. Find the mean of Sepal.Length of iris dataset. mean(iris$Sepal.Length)
  3. Display variance of Sepal.Length of iris dataset. var(iris$Sepal.Length)
  4. Display median of Petal.Length of iris dataset. median(iris$Petal.Length)
  5. Find Standard Deviation of Setal.Length of iris dataset. sd(iris$Sepal.Length)
  6. Display top 10 records from the dataset iris. head(iris, n = 10)
  7. Display bottom 10 records from the dataset iris. tail(iris, n=10)
  8. Print contents of the dataframe iris. print(iris)

# OUTPUT :





**EX.NO : 06**

**DATE :**

# AIM :

**VECTOR MANIPULATION**

To create and manipulate vector in R.

# VECTOR :

A vector is asequence of data elements of the same basic type. Data types can be numeric, integer,character, complex or logical. It is created using the c() function. Since, a vector must have elements of thesame type, this function will try and coerce elements to the same type, if they are different.Coercion is from lower to higher types from logical to integer to double to character.

# PROGRAM :

1. Create the following vectors.

**Note : use : operator, seq() and rep() a)** (1, 2, 3, . . . , 19, 20).

# seq(1,20)

**b)** (20, 19, . . . , 2, 1).

# seq(20,1)

**c)** (1, 2, 3, . . . , 19, 20, 19, 18, . . . , 2, 1).

# c(1:20,19:)

**d)** (5,10,15, … , 100).

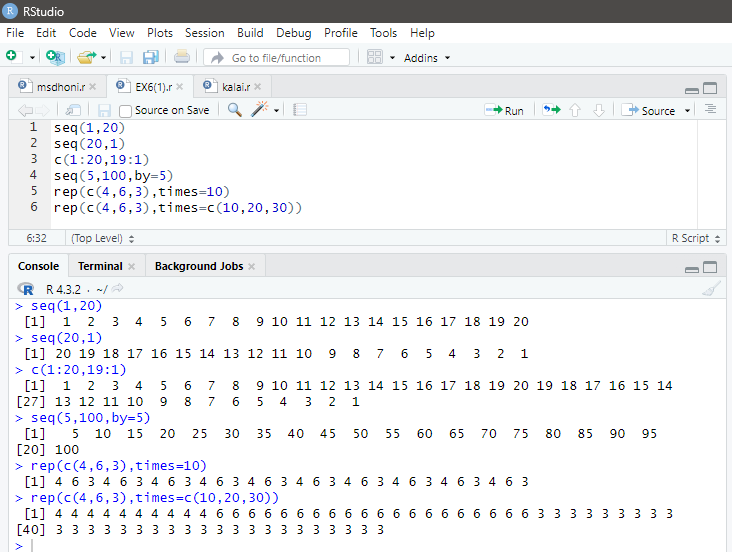
# seq(5,100,by=5)

**e)** (4, 6, 3, 4, 6, 3, . . . , 4, 6, 3) where there are 10 occurrences of 4,6,3. **rep(c(4,6,3),times=10)**

**f)** (4, 4, . . . , 4, 6, 6, . . . , 6, 3, 3, . . . , 3) where there are 10

occurrences of 4, 20 occurrences of 6 and 30 occurrences of 3.

# rep(c(4,6,3),times=c(10,20,30)) OUTPUT :

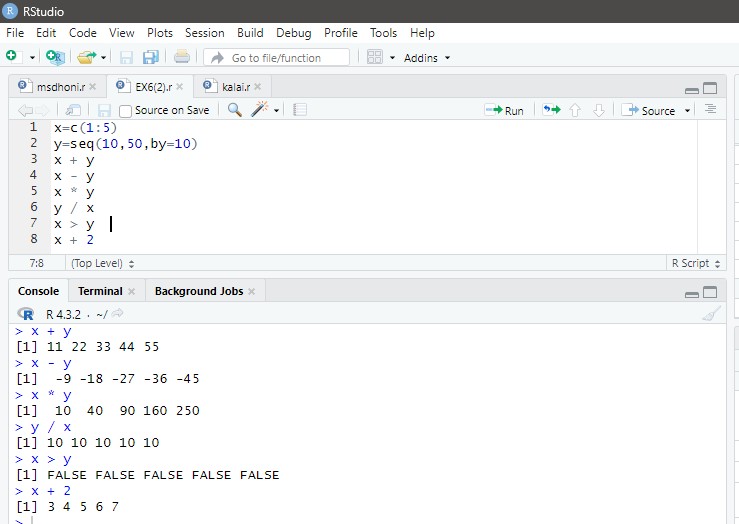


1. Create a vector with elements 1, 2, 3, 4, 5 and call it x.Create another vector with elements 10 20 30 40 50 and call it y What is the value of
   1. x + y **b)** x – y **c)** x \* y **d)** y /x **e)** x > y **f)** x+2 x=c(1:5)

y=seq(10,50,by=10)

|  |  |  |
| --- | --- | --- |
| x + y | x – y | x \* y |
| y / x | x > y | x + 2 |

# OUTPUT :



1. Create a Vector ‘v’ with the values 10, 20, 30, 10, 40, 50, 30, 40, 80, 90, 100 and do the following exercises.

# v = c(10, 20, 30, 10, 40, 50, 30, 40, 80, 90, 100)

* 1. List elements of the vector that are greater than 10 and less than 80. **v[v>10 & v<80]**
  2. List elements of the vector that are multiplies of 4.

# v[v %% 4 ==0]

* 1. How many times the element 40 occurred in the vector.

# sum (v == 40)

* 1. List the last value in the given vector. **v[ length(v) ]**
  2. Find second highest value in the given vector.

# sort(v , decreasing = TRUE) [2]

* 1. Find nth highest value in the given vector.

# n = 6

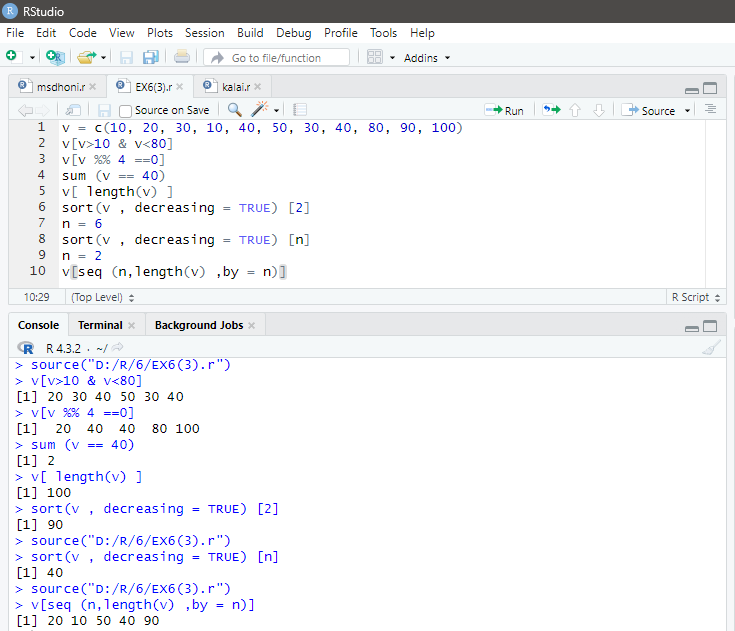
**sort(v , decreasing = TRUE) [n]**

* 1. Extract every nth element of a given vector.

# n = 2

**v[seq (n,length(v) ,by = n)]**

# OUTPUT :



**EX.NO : 07**

**DATE :**

# AIM :

**MATRIX MANIPULATION**

To create and manipulate matrix in R.

# MATRIX :

Matrix is a two dimensional data structure in R programming. It can be created using the matrix() function. Dimension of the matrix can be defined by passing appropriate value for arguments nrow and ncol

# PROGRAM :

1. Create the following matrix. 1 4

2 5

3 6

matrix1 <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3) print(matrix1)

1. Create a vector v of size 12 with random value between 1 to 100. v = sample(1:100,12) print(v)
2. Convert the vector v to a 4\*3 matrix1. vec <- c(16, 5, 4, 3, 2, 1)

matrix1 <- matrix(vec,nrow = 2, ncol = 3) print(matrix1)

1. Change the column names of matrix1 to x, y, z and row names to a, b, c, d.

rownames(matrix1) = c("a","b") colnames(matrix1) = c("x","y","z") print(matrix1)

1. Compute A+3, A-3, A\*3 and A/3.

matrix1+3 matrix1-3 matrix1\*3 matrix1/3

1. Obtain the transpose matrix of matrix1. t (matrix1)
2. Display 2nd row of matrix. Matrix1 [2, ]
3. Display the entire matrix leaving 2nd column. Matrix1 [ , -2]
4. Display only first three rows of matrix.

Matrix1 [ c(1:3), ]

1. Change the value of element at 2nd row and 3rd column to 300.

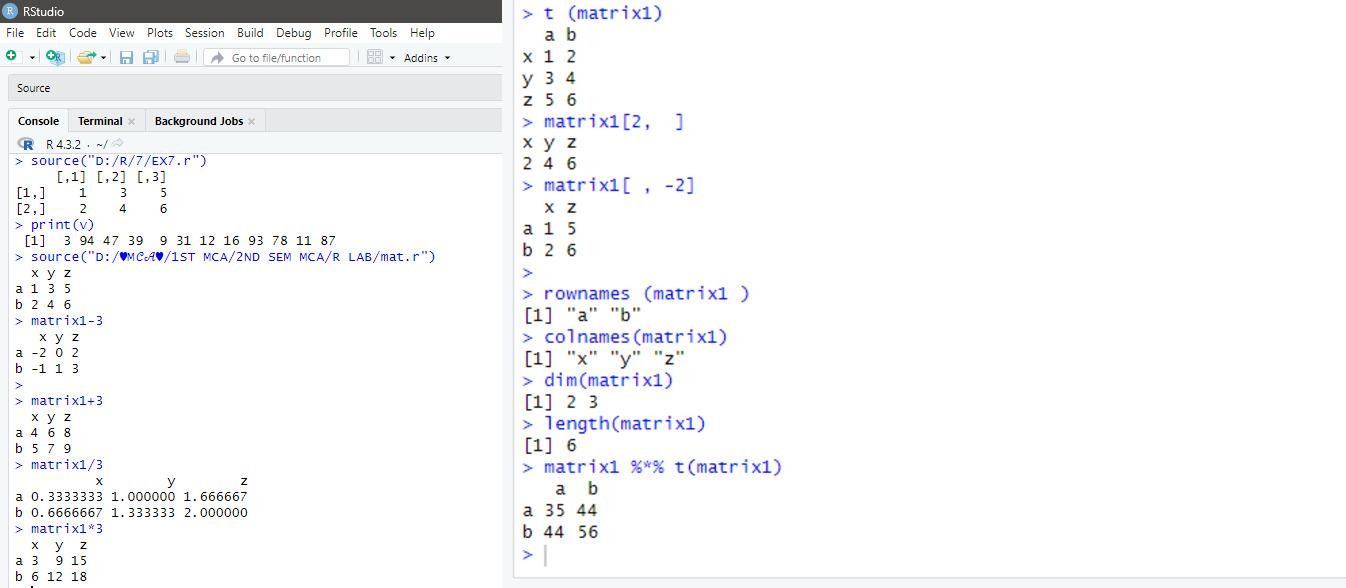
Matrix1 [2:3] = 300

1. Replace all elements of matrix1 that are greater than 50 with 200.

Matrix1 [ matrix1>50]=200

1. Display column and row names of matrix1. rownames (matrix1 ) colnames(matrix1)
2. Display dimension and the no of elements in matrix1. dim(matrix1) length(matrix1)
3. Multiply the matrix1 with its transpose. Matrix1 %\*% t(matrix1)

# OUTPUT :



**EX.NO : 08**

**DATE :**

# AIM :

**LIST MANIPULATION**

To create and manipulate list in R.

# LIST :

List is a data structure having components of mixed data types. A vector having all elements of the same type is called atomic vector but a vector havingelements of different type is called list. It can be created using the list() function.

# PROGRAM :

1. Create a list called my\_list from the 3 vectors provided below and do the following.

a <- "My First List"

b <- c("R", "is", "Fun!")

c <- matrix(1:9,3,3)

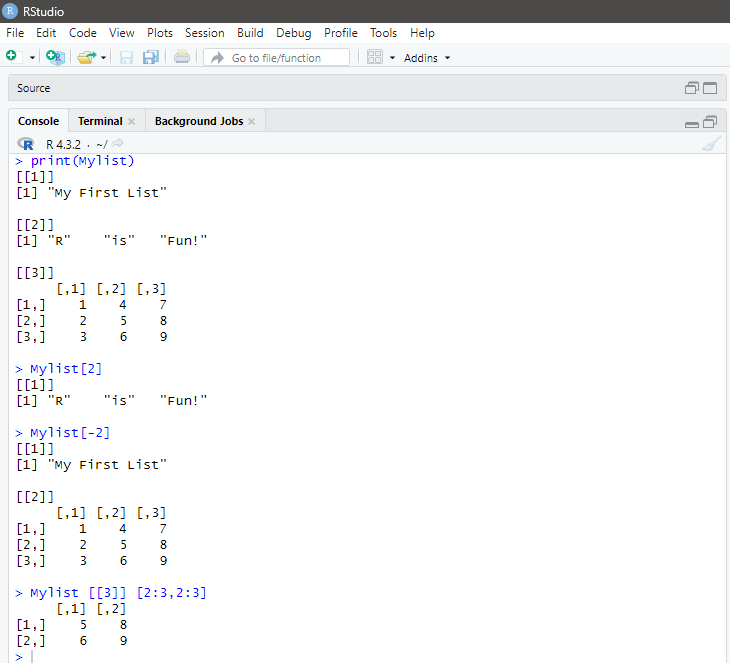
Mylist = list(a,b,c) print(Mylist)

* 1. Access second component of my\_list. Mylist[2]
  2. Display all components of my\_list leaving second component.

Mylist[-2]

* 1. Remove the first and third items from my\_list. Mylist [[3]] [2:3,2:3]

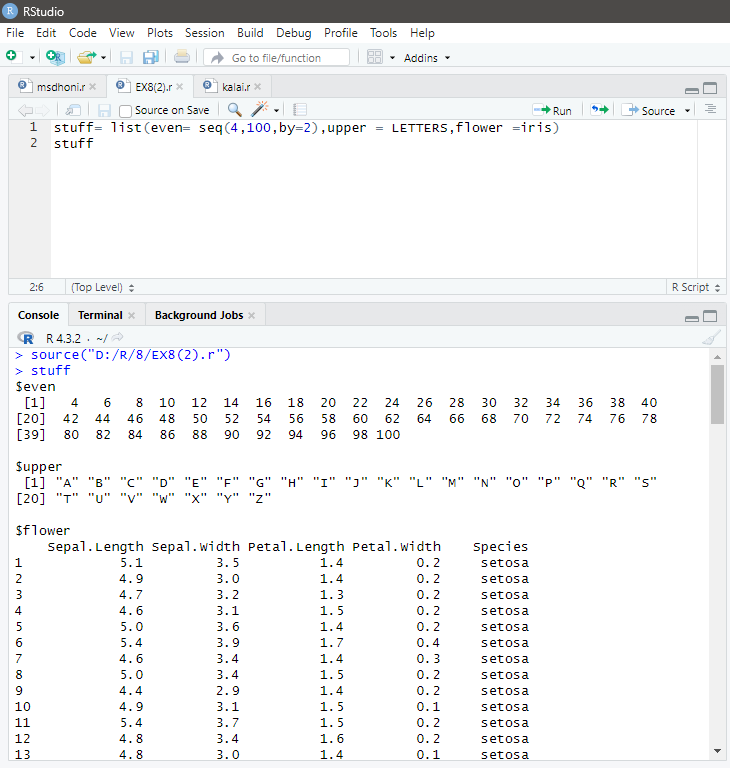
# OUTPUT :



1. Make a list called Stuff and display its structure. The list should contain three elements.
   1. The sequence of even numbers from 4 to 100. Its name should be even.
   2. The uppercase letters of the alphabet. Its name should be upper.
   3. The data frame iris. Its name should be flower.

# stuff= list(even= seq(4,100,by=2),upper = LETTERS,flower =iris) stuff

**OUTPUT :**



**EX.NO : 09**

**DATE :**

# AIM :

**DATA FRAMES IN R**

To create data frame and perform various operations on data frame in R.

# DATA FRAMES :

Data frame is a two dimensional data structure in R. It is a special case of a list which has each component of equal length. Each component forms the column and contents of the component form the rows.

# PROGRAM :

1. Create data frame to display voting details.

dataframe1 <- data.frame ( Name = c("Kalai", "Divya", "Pavai"), Age = c(22, 15, 9), Vote = c(TRUE, FALSE, TRUE))

print(dataframe1)

1. Access the name column from data frame.

# # pass index number inside [ ].

print(dataframe1[1])

# # pass column name inside [[ ]].

print(dataframe1[["Name"]])

# # use $ operator and column name.

print(dataframe1$Name)

1. Use rbind() in data frame.

dataframe2 <- data.frame (Name = c("Basheer", "Jothi"), Age = c(46, 89), Vote = c(TRUE,TRUE))

dataframe1 <- data.frame ( Name = c("Kalai", "Divya", "Pavai"), Age = c(22, 15, 9), Vote = c(TRUE, FALSE, TRUE))

updated <- rbind( dataframe1, dataframe2) print(updated)

1. Use cbind() in data frame.

# # create a data frame.

dataframe1 <- data.frame ( Name = c("Kalai", "Divya"), Age = c(22, 15))

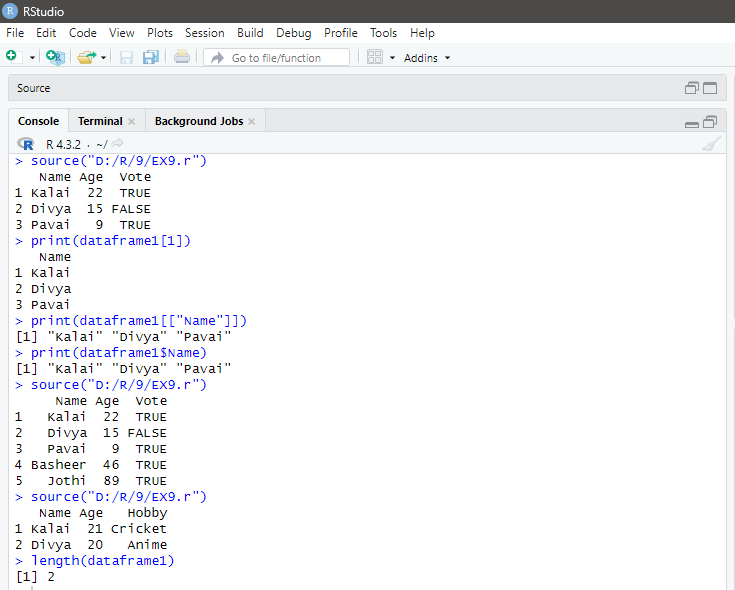
# # create another data frame.

dataframe2 <- data.frame ( Hobby = c("Cricket", "Anime"))

**# combine two data frames horizontally.** updated <- cbind(dataframe1, dataframe2) print(updated)

1. Find the length of Data frame. length(dataframe1)

# OUTPUT :



**EX.NO : 10**

**DATE :**

# AIM :

**IMPORTING FILES IN R**

To import different types of file such as csv,xml, excel and text files in R.

# PROGRAM :

1. **Importing a CSV File :-**

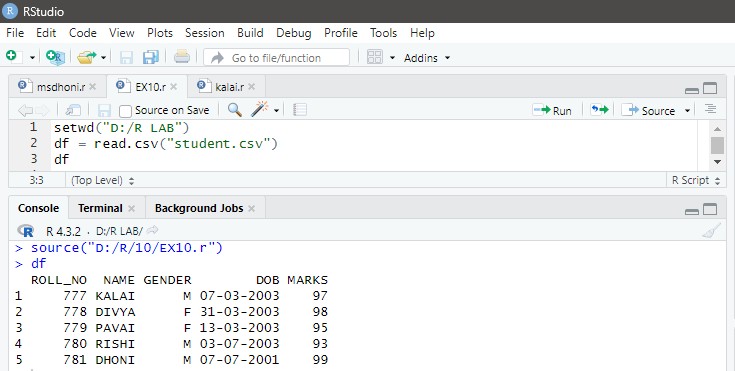
Create a csv file with the following student data and save it as student.csv, Import this file in r studio.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ROLL\_NO** | **NAME** | **GENDER** | **DOB** | **MARKS** |
| 777 | KALAI | M | 07-03-2003 | 97 |
| 778 | DIVYA | F | 31-03-2003 | 98 |
| 779 | PAVAI | F | 13-03-2003 | 95 |
| 780 | RISHI | M | 03-07-2003 | 93 |
| 781 | DHONI | M | 07-07-2001 | 99 |

setwd(“D:/R LAB”)

df = read.csv(“student.csv”) df

# OUTPUT :



1. **Importing a Excel File.**

Create a excel file which contains the following data and save it as ipl.xlsx.

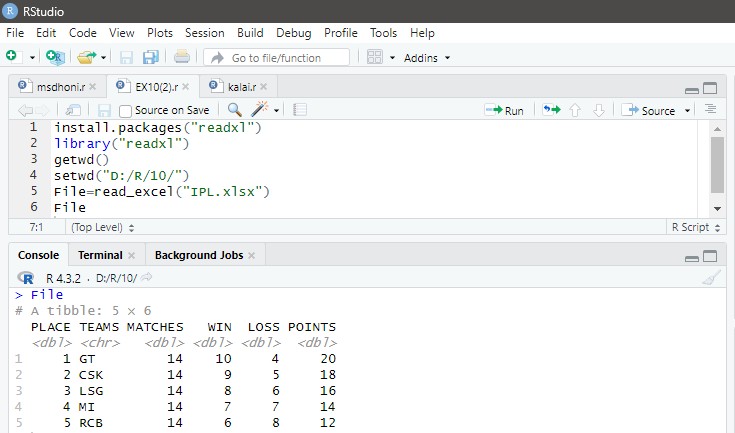
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PLACE** | **TEAMS** | **MATCHES** | **WIN** | **LOSS** | **POINTS** |
| 1 | GT | 14 | 10 | 4 | 20 |
| 2 | CSK | 14 | 9 | 5 | 18 |
| 3 | LSG | 14 | 8 | 6 | 16 |
| 4 | MI | 14 | 7 | 7 | 14 |
| 5 | RCB | 14 | 6 | 8 | 12 |

install.packages("readxl") library("readxl")

getwd() setwd("D:/R/10/")

File=read\_excel("IPL.xlsx") File

# OUTPUT :

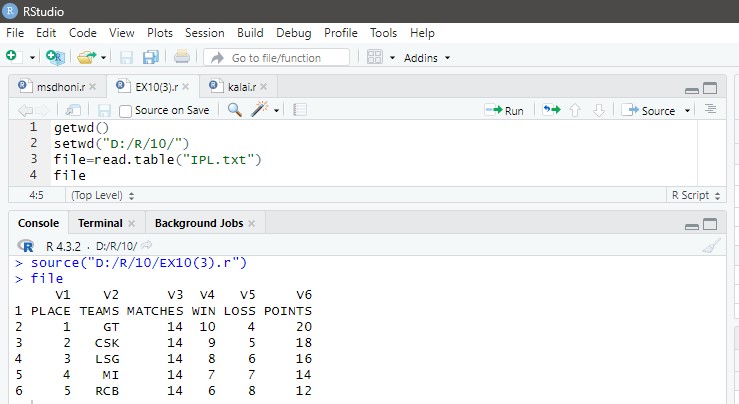


1. **Importing a Text File.**

Create a text file using notepad with the following data and save it as ‘IPL.txt’.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PLACE** | **TEAMS** | **MATCHES** | **WIN** | **LOSS** | **POINTS** |
| 1 | GT | 14 | 10 | 4 | 20 |
| 2 | CSK | 14 | 9 | 5 | 18 |
| 3 | LSG | 14 | 8 | 6 | 16 |
| 4 | MI | 14 | 7 | 7 | 14 |
| 5 | RCB | 14 | 6 | 8 | 12 |

# OUTPUT :



**EX.NO : 11 DATE :**

# IMPLEMENTING NAVIE BAYES USING IRIS DATASET AIM :

To show the implementation of Navie Bayes algorithm using Iris dataset.

# NAVIE BAYES ALGORITHM :

Naive Bayes is a Supervised Machine Learning algorithm based on the Bayes Theorem that is used to solve classification problems by following a probabilistic approach. It is based on the idea that the predictor variables in a Machine Learning model are independent of each other. Meaning that the outcome of a model depends on a set of independent variables(Predictor) that have nothing to do with each other.

# PROCEDURE :

* 1. Load the dataset .
  2. Create train/test set.
     1. id <- sample(2,nrow(df),replace=TRUE,prob=c(0.80,0.20))
     2. df\_train <- df [ id==1, ]
     3. df\_test <- df [ id==2, ]
  3. Build the model.
     1. Install Package e1071
     2. Load the package
     3. Model <- naviesBayes(df\_train[ ,5] ,df\_train$species)
  4. Make prediction.
     1. p= predict(model,df\_test[ , -5])
  5. Measure performance by confusion matrix.

# PROGRAM :

data("iris") str(iris) df=iris

summary(df) set.seed(123)

id = sample(1:2,nrow(df),replace=TRUE,prob=c(0.08,0.20)) df\_train = df[id == 1,]

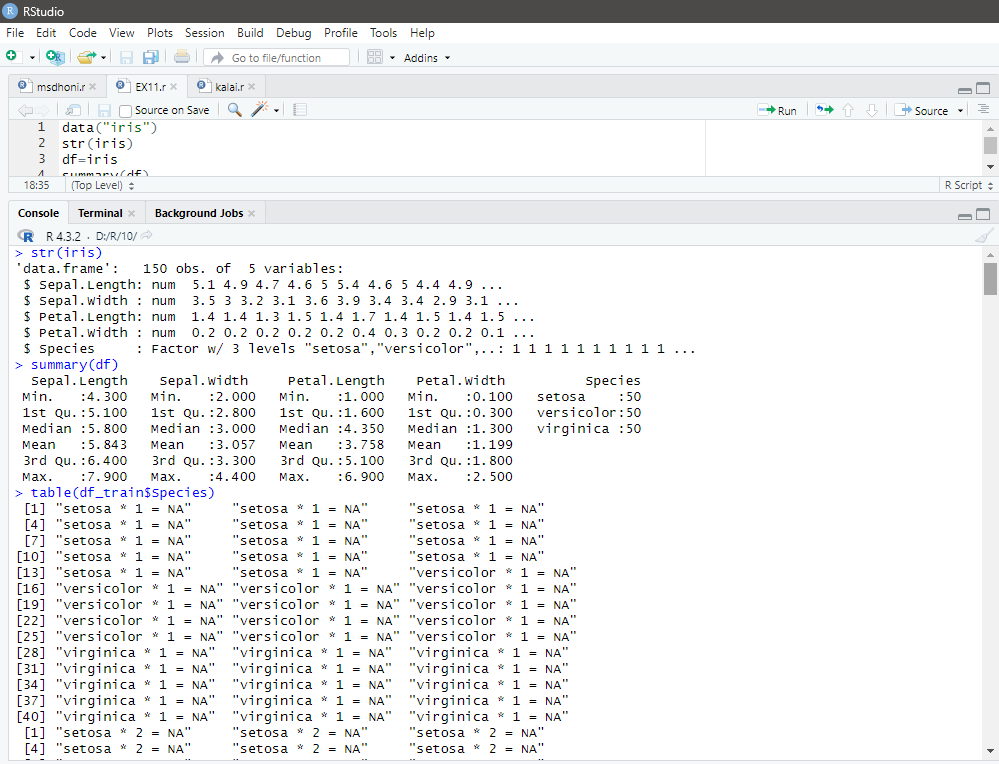
df\_test = df[id == 2,] table(df\_train$Species) table(df\_test$Species) install.packages("e1071") library(e1071)

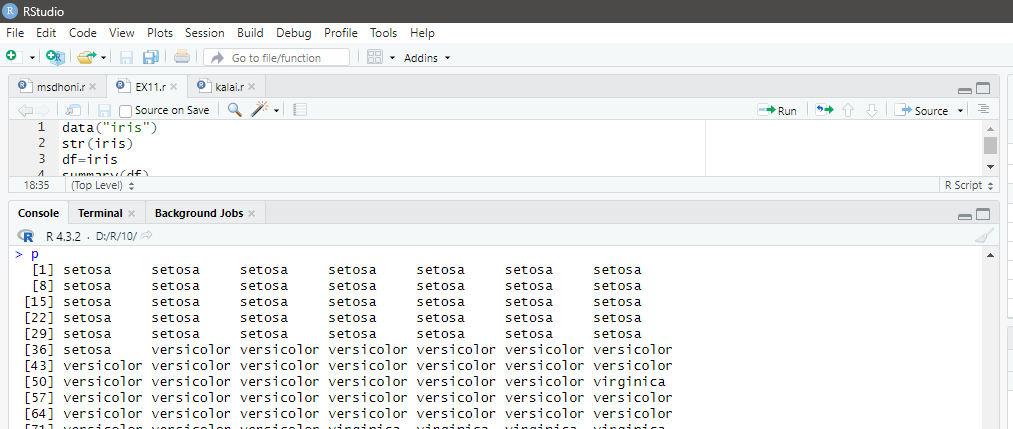
model=naiveBayes(df\_train[,-5],df\_train$Species) p<- predict(model,df\_test[,-5])

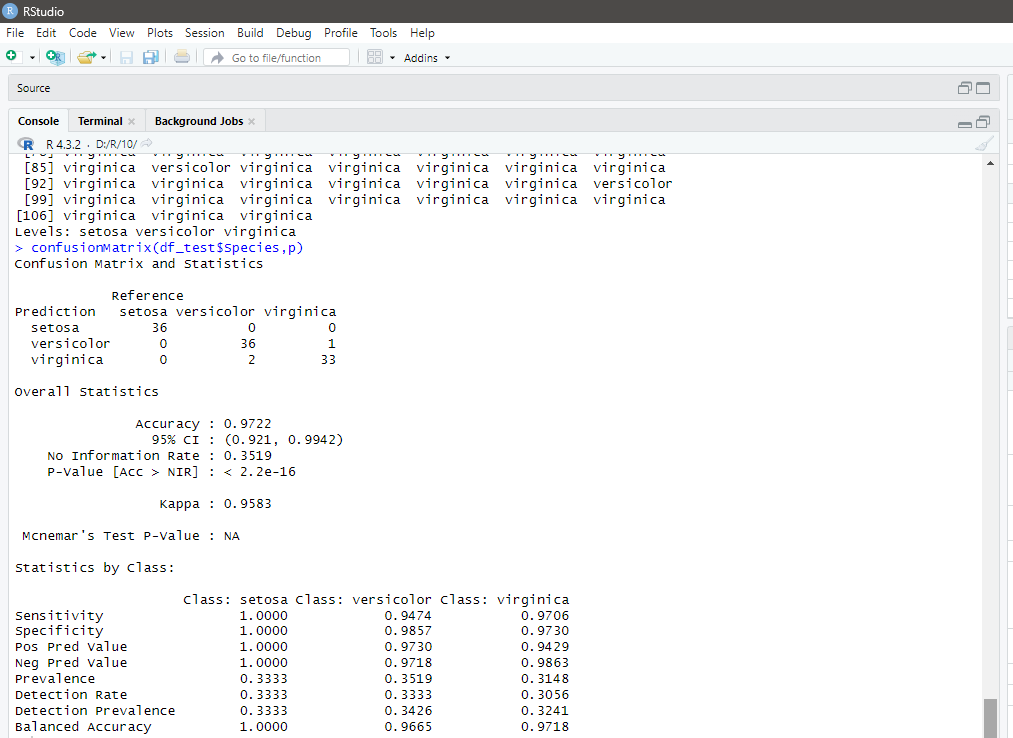
p install.packages("caret") library(caret)

confusionMatrix(df\_test$Species,p)

# OUTPUT :







**EX.NO : 12**

**DATE :**

# AIM :

**BREAST CANCER PREDICTION USING KNN**

To show Implementation of KNN using Breast Cancer dataset.

# KNN ALGORITHM :

K Nearest Neighbors or KNN Algorithm is a simple algorithm which uses the entire dataset in its training phase. Whenever a prediction is required for an unseen data instance, it searches through the entire training dataset for k-most similar instances and the data with the most similar instance is finally returned as the prediction.

# PROCEDURE :

1. Load the dataset.
2. Initialize the value of k.
3. For each sample in the training dataset.
   1. Calculate the distance between test data and each sample in the training data.
4. Based on the distance value, sort them in ascending order.
5. Get top k rows from the sorted array.
6. Return the most frequent class of these rows.

# PROGRAM :

library(class) library(caret) a=read.csv("diabetes.csv") head(a,10)

str(a) a$Outcome=factor(a$Outcome)

str(a)

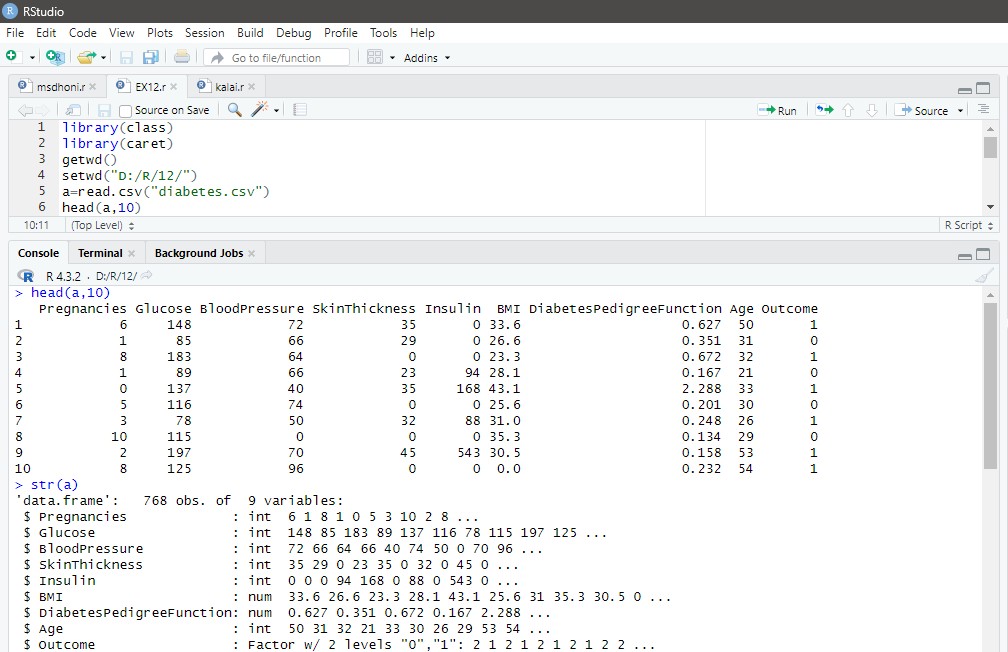
summary(a) #a[r,c] is.na(a) colSums(is.na(a)) train=a[1:500,] test=a[501:768,]

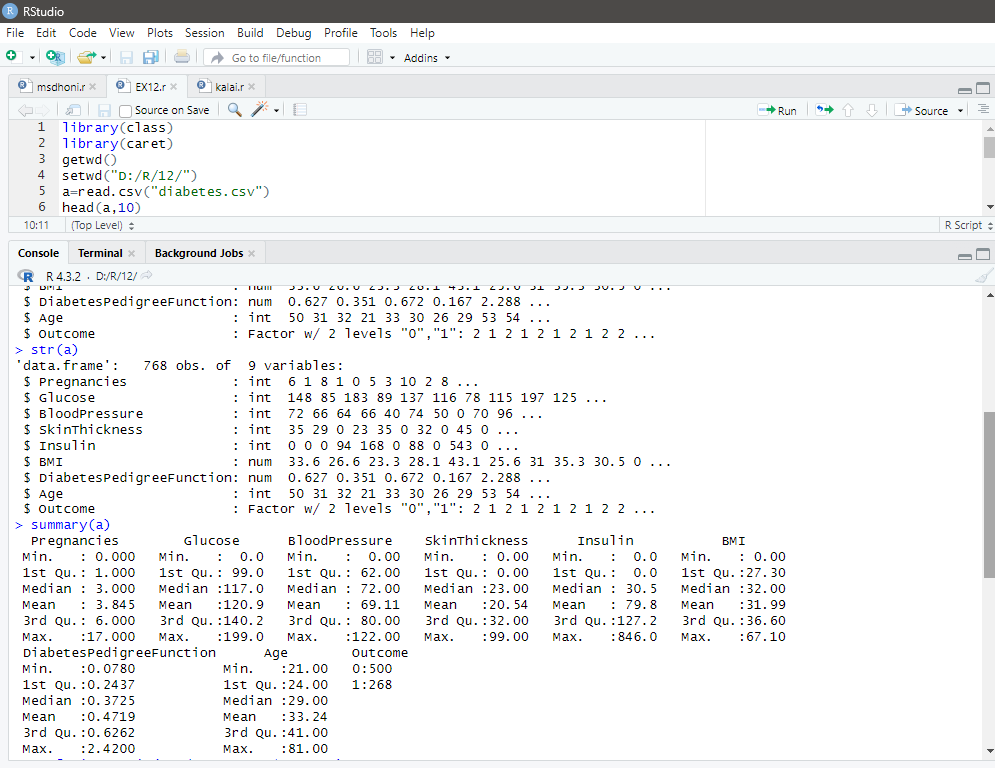
# knn(training,test)

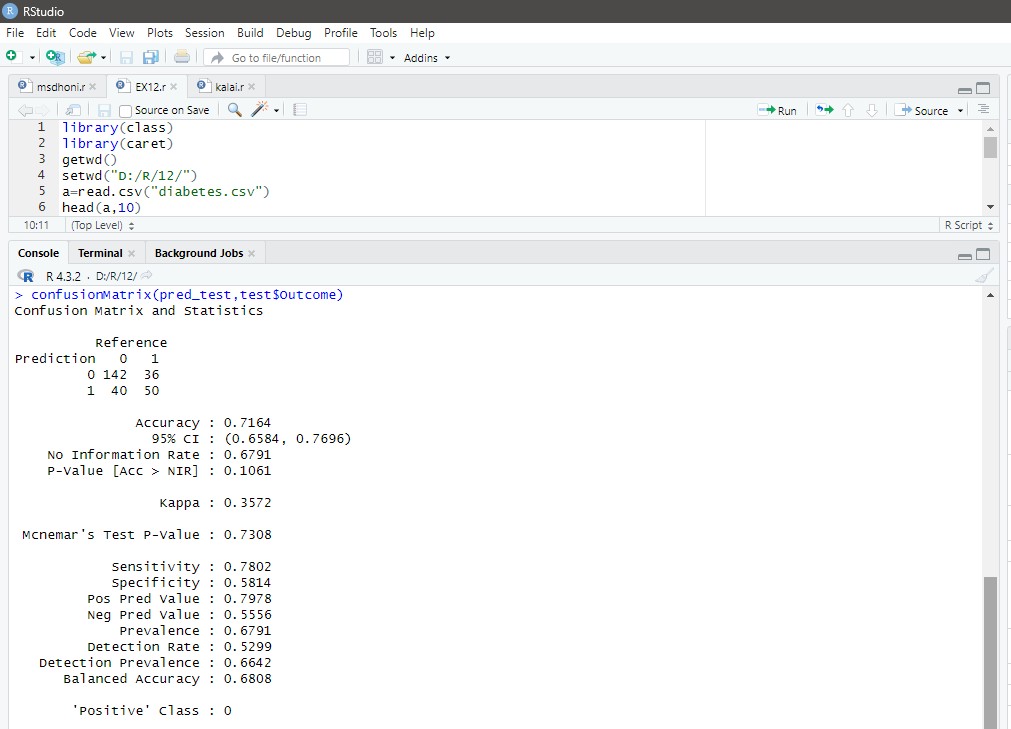
pred\_test=knn(train[,-9],test[,-9],train$Outcome,k=7) cm=table(pred\_test,test$Outcome)

cm confusionMatrix(pred\_test,test$Outcome)

# OUTPUT :







**EX.NO : 13 DATE :**

# DIABETIES PREDICTION USING DECISION TREE AIM :

To show the implementation of Decision Tree using Diabeties dataset.

# DECISION TREE ALGORITHM :

Decision Tree is one of the most widely used and practical methods for supervised learning.They can be used to solve both regression and classification problems. It is robust to noisy data and capable of learning disjunctive expressions. The decision tree algorithms such as ID3, CART, C4.5 & ASSISTANT are very popular inductive inference algorithms, and they are successfully applied to a broad range of tasks from learning to diagnose medical cases to learning to assess credit risk of loan applicants.

# PROCEDURE :

1. Load the dataset
2. Create train/test set.
   1. train <- sample(n,trunc(0.70\*n))
   2. df\_train <- df [ train, ]
   3. df\_test <- df [ - train, ]
3. Build the model.
   1. Install Package rpart
   2. Load the package .
   3. Model <- rpart(Outcome~. ,data =df\_train)
4. Make prediction.
   1. predict (model ,df\_test,type=”class”)
5. Measure performance by confusion matrix.

# PROGRAM :

df = read.csv(file.choose()) head(df)

str(df)

df$Outcome<- factor(df$Outcome,levels=c(0,1),labels=c("No","Yes"))

summary(df)

sapply(df,function(x) sum(is.na(x))) library(corrplot) set.seed(123)

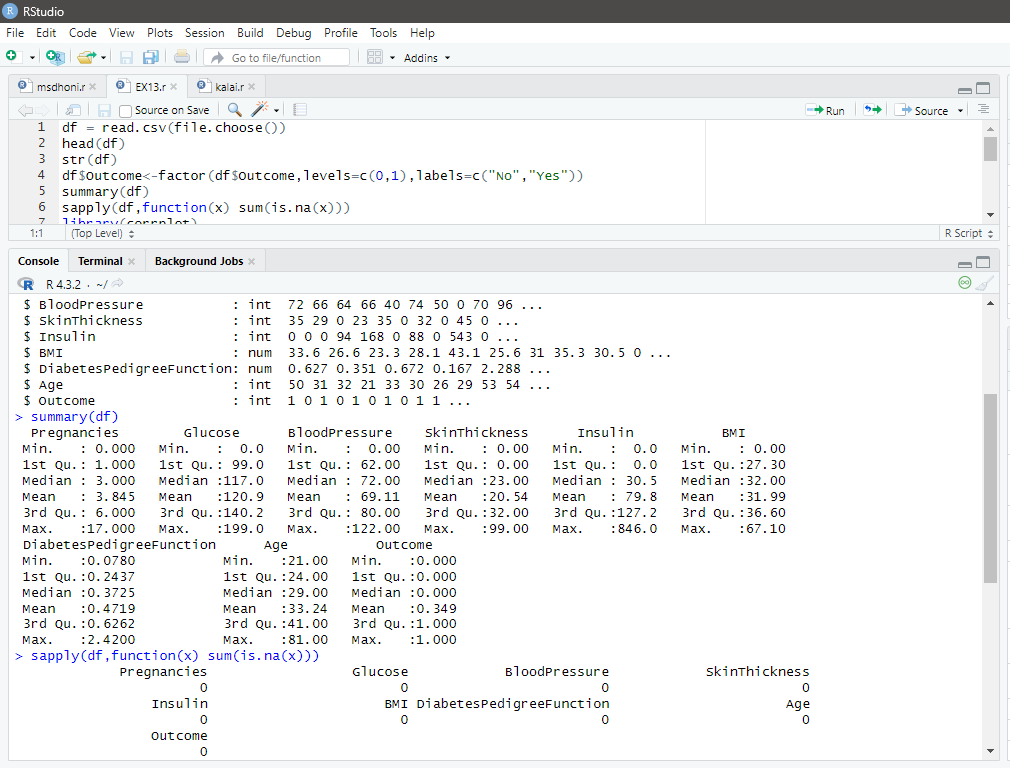
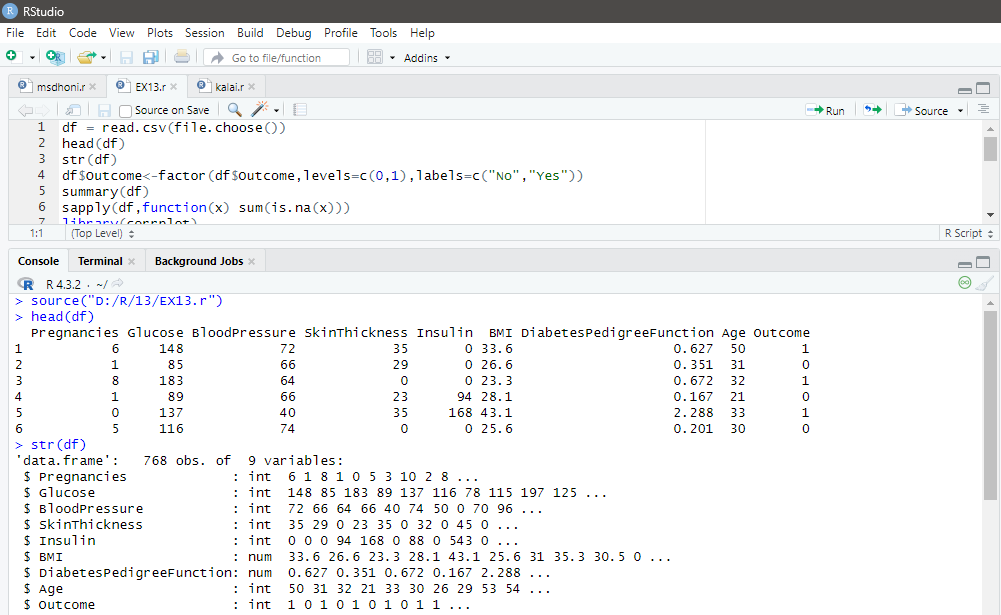
n<-nrow(df)

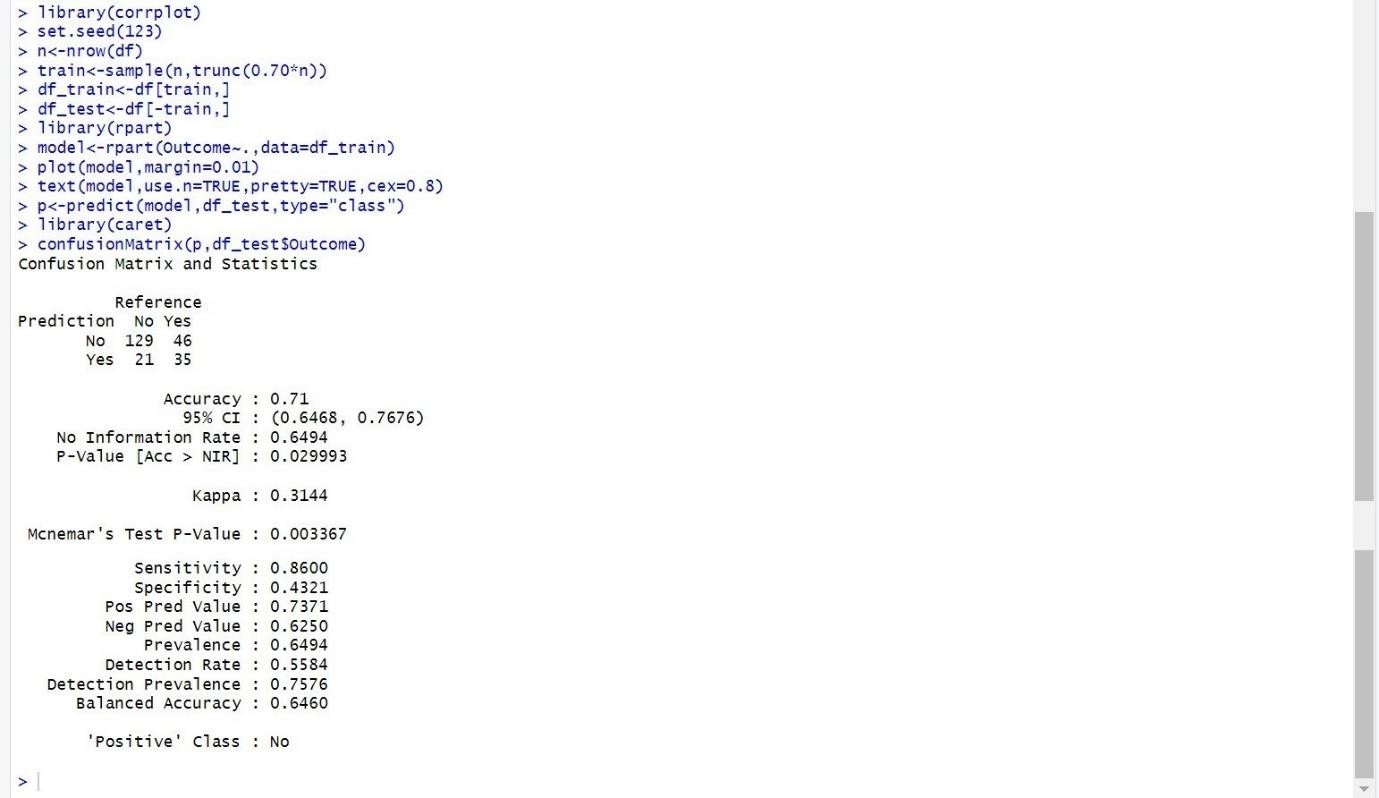
train<-sample(n,trunc(0.70\*n)) df\_train<-df[train,]

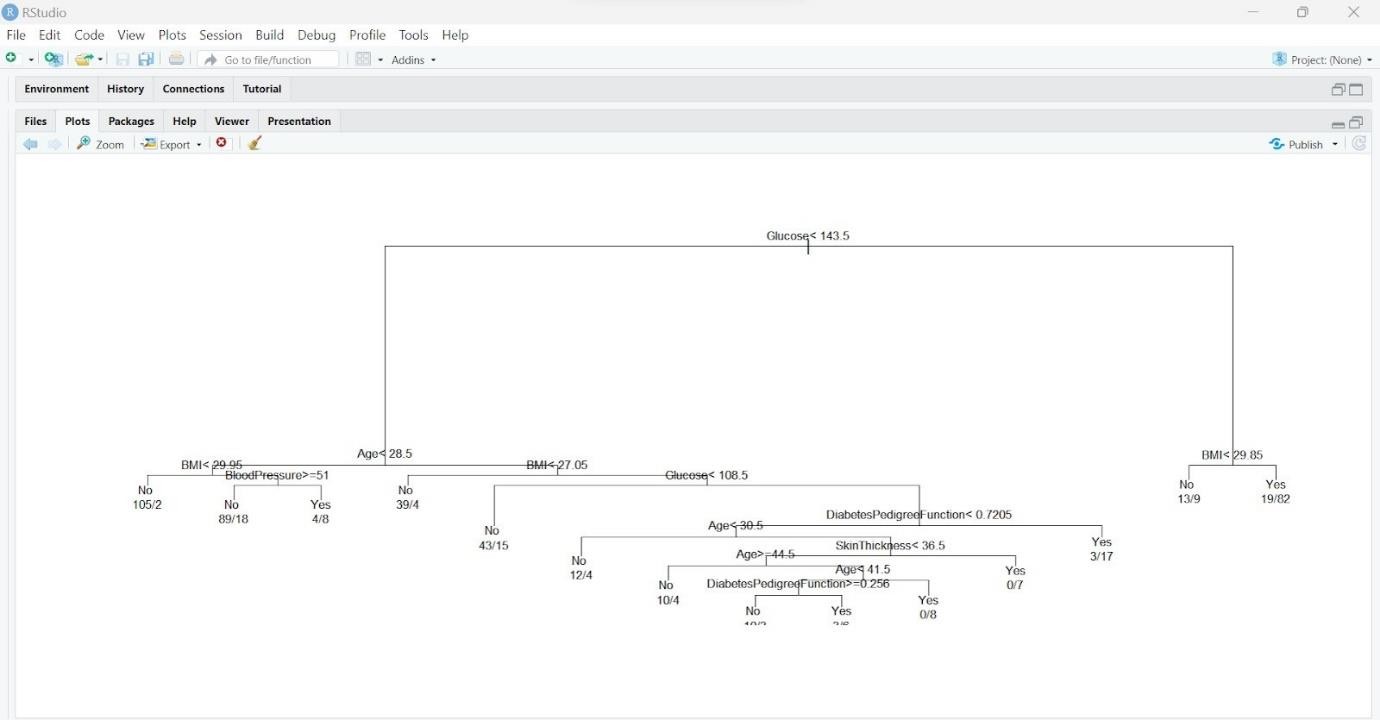
df\_test<-df[-train,] library(rpart)

model<-rpart(Outcome~.,data=df\_train) plot(model,margin=0.01) text(model,use.n=TRUE,pretty=TRUE,cex=0.8) p<-predict(model,df\_test,type="class") library(caret) confusionMatrix(p,df\_test$Outcome)

# OUTPUT :







**EX.NO : 14 DATE :**

# WINE QUALITY PREDICTION USING RANDOM FOREST AIM :

To show the Implementation of Random Forest using Wine Quality Prediction dataset.

# RANDOM FOREST ALGORITHM :

Random forest is a popular supervised machine learning algorithm—used for both classification and regression problems. It is based on the concept of ensemble learning, which enables users to combine multiple classifiers to solve a complex problem and to also improve the performance of the model. The random forest algorithm relies on multiple decision trees and accepts the results of the predictions from each tree. Based on the majority votes of predictions, it determines the final result.

# PROCEDURE :

1. Load the dataset.
2. Create train/test set.
   1. id <- sample(2,nrow(df),replace=TRUE,prob=c(0.70,0.30))
   2. df\_train <- df [ id==1, ]
   3. df\_test <- df [ id==2, ]
3. Build the model.
   1. Install Package randomForest.
   2. Load the package .
   3. Model<- randomForest(taste~.,data=df\_train,ntree=1000,ntry=5)
4. Make prediction.

p= predict(model,df\_test)

1. Calculate the votes for each of the predicted targets.
2. The most highly voted predicted target is the final prediction.

# PROGRAM :

df <- read.csv(file.choose()) str(df)

summary(df) table(df$quality)

df$taste=ifelse(df$quality<5,"Bad","Good") df$taste[df$quality == 5] <- "Normal" df$taste[df$quality == 6] <- "Normal" table(df$taste)

df=df[,-12]

df$taste = as.factor(df$taste) set.seed(123)

id <- sample(2,nrow(df),replace = TRUE,prob=c(0.7,0.3)) df\_train <- df[id==1,]

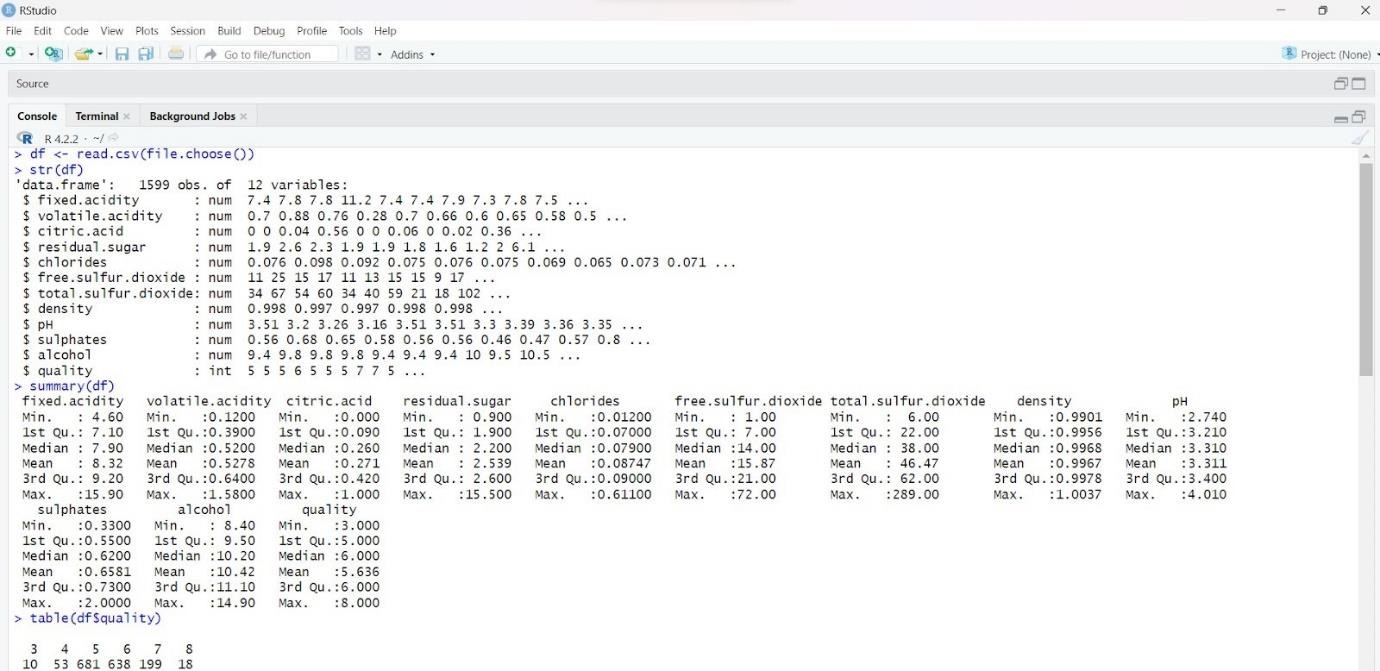
df\_test <- df[id==2,] install.packages("randomForest") library(randomForest)

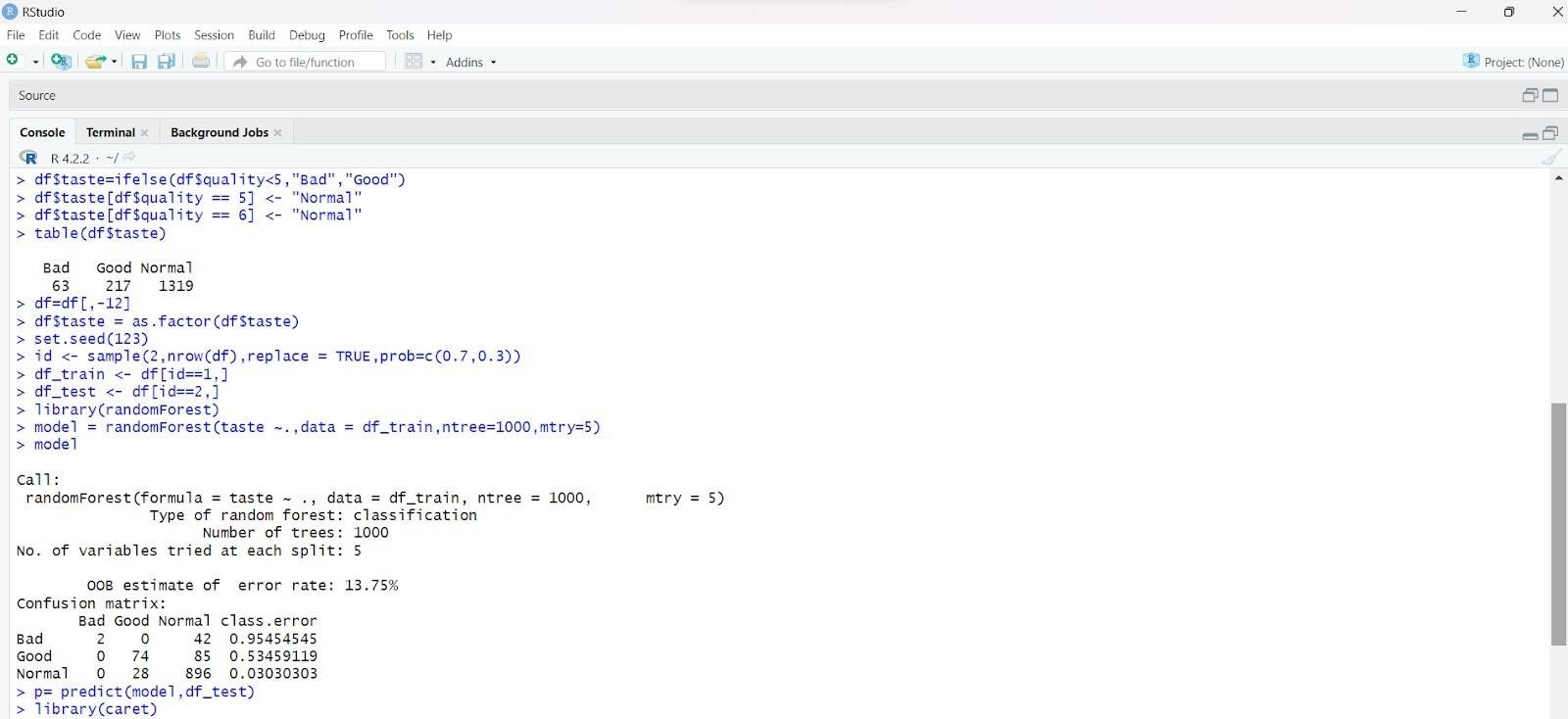
model = randomForest(taste ~.,data = df\_train,ntree=1000,mtry=5)

model

p= predict(model,df\_test) library(caret) confusionMatrix(p,df\_test$taste)

# OUTPUT :







**EX.NO : 15 DATE :**

# CLUSTERING USARREST DATASET USING KMEANS ALGORITHM

**AIM :**

To show the implementation of KMeans Algorithm using USArrest Dataset.

# KMEANS ALGORITHM :

Kmeans clustering is one of the simplest and popular unsupervised machine learning algorithms. A cluster refers to a collection of data points aggregated together because of certain similarities. The number k refers to the number of centroids. A centroid is the imaginary or real location representing the center of the cluster. Every data point is allocated to the nearest cluster.

# PROCEDURE :

1. Load the dataset.
2. Choose the number K clusters.
3. Select at random K points.
4. Assign each data point to closest centroid that forms K clusters.
5. Compute and place the new centroid of each centroid.
6. Reassign each data point to new cluster.

# PROGRAM :

install.packages("factoextra") library(factoextra)

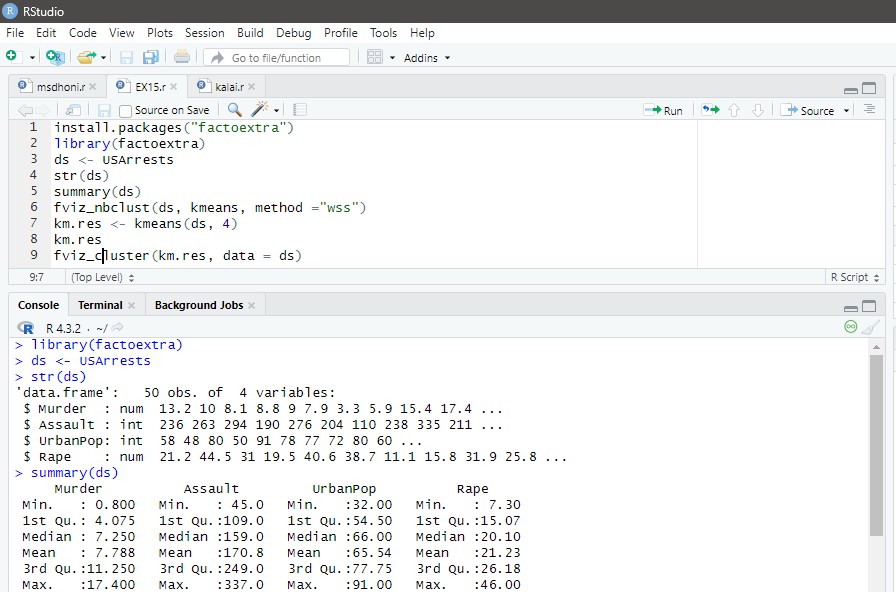
ds <- USArrests str(ds) summary(ds)

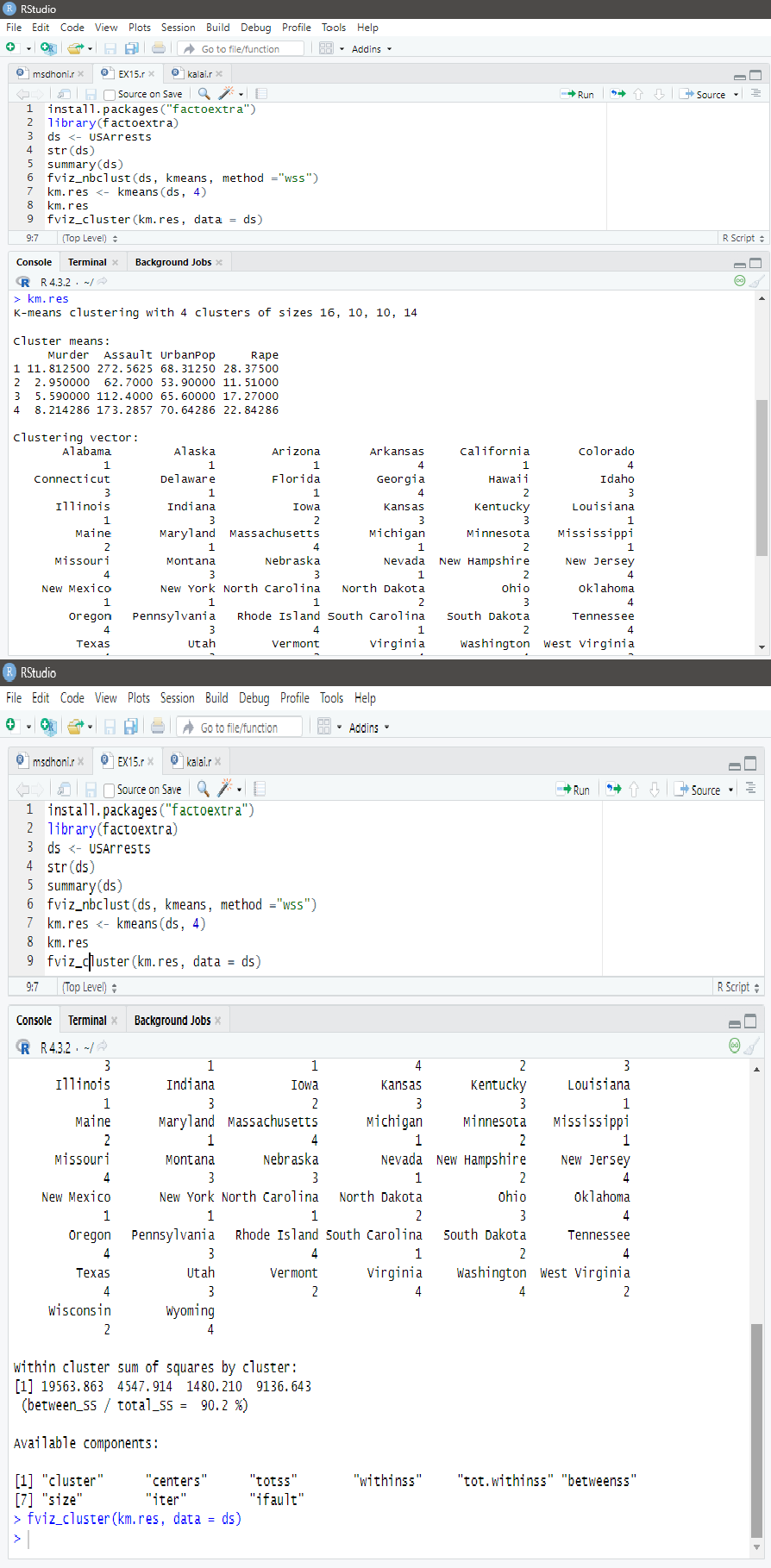
fviz\_nbclust(ds, kmeans, method ="wss")

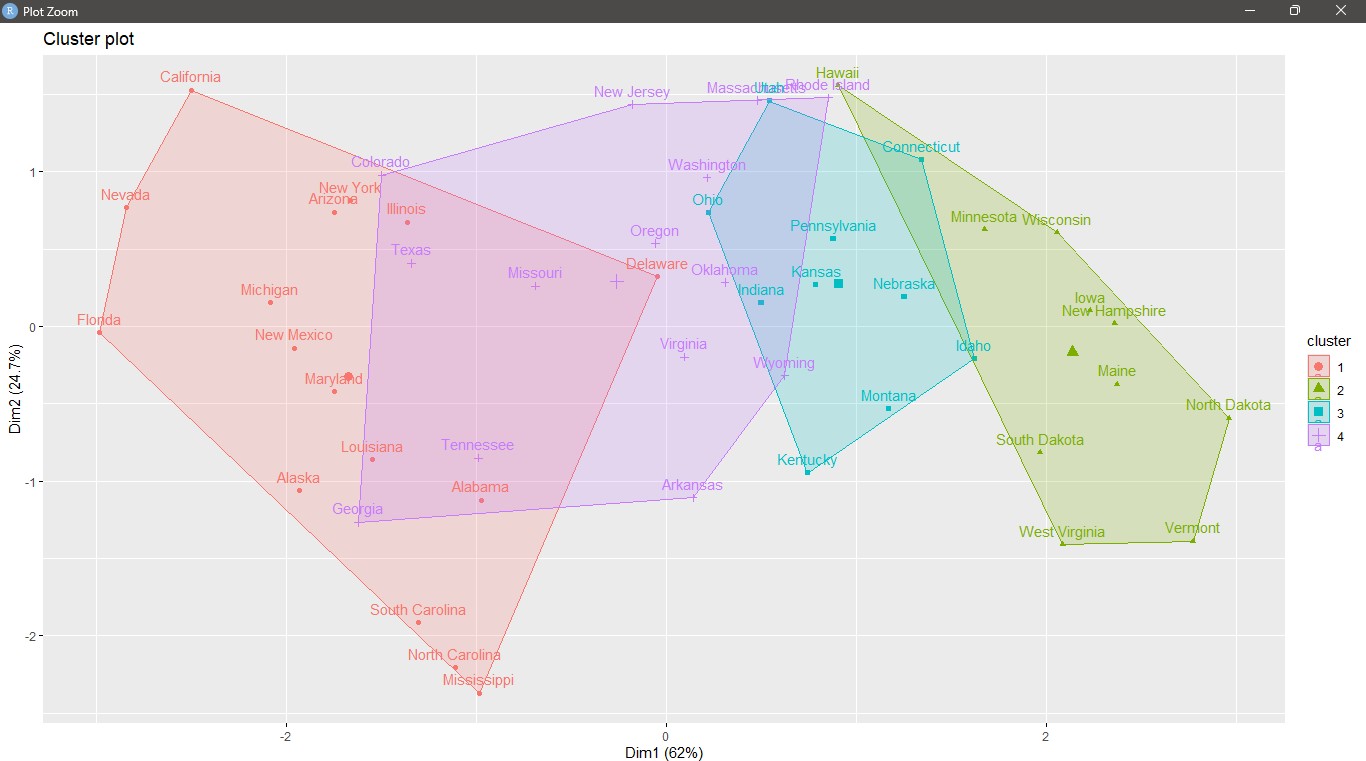
km.res <- kmeans(ds, 4) km.res

fviz\_cluster(km.res, data = ds)

# OUTPUT :







**EX.NO : 16 DATE :**

# CLUSTERING USARRESTS USING AGGLOMERATIVE HIERARCHICAL CLUSTERING

**AIM :**

To show the implementation of Agglomerative Hierarchical Clustering using USArrest Dataset.

# AGGLOMERATIVE HIERARCHICAL CLUSTERING :

The Agglomerative Hierarchical Clustering is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. It's a “bottom-up” approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

# PROCEDURE :

1. Make each data point a single-point cluster → forms N clusters.
2. Take the two closest clusters and make them one cluster → Forms N-1 clusters.
3. Repeat step-3 until you are left with only one cluster.

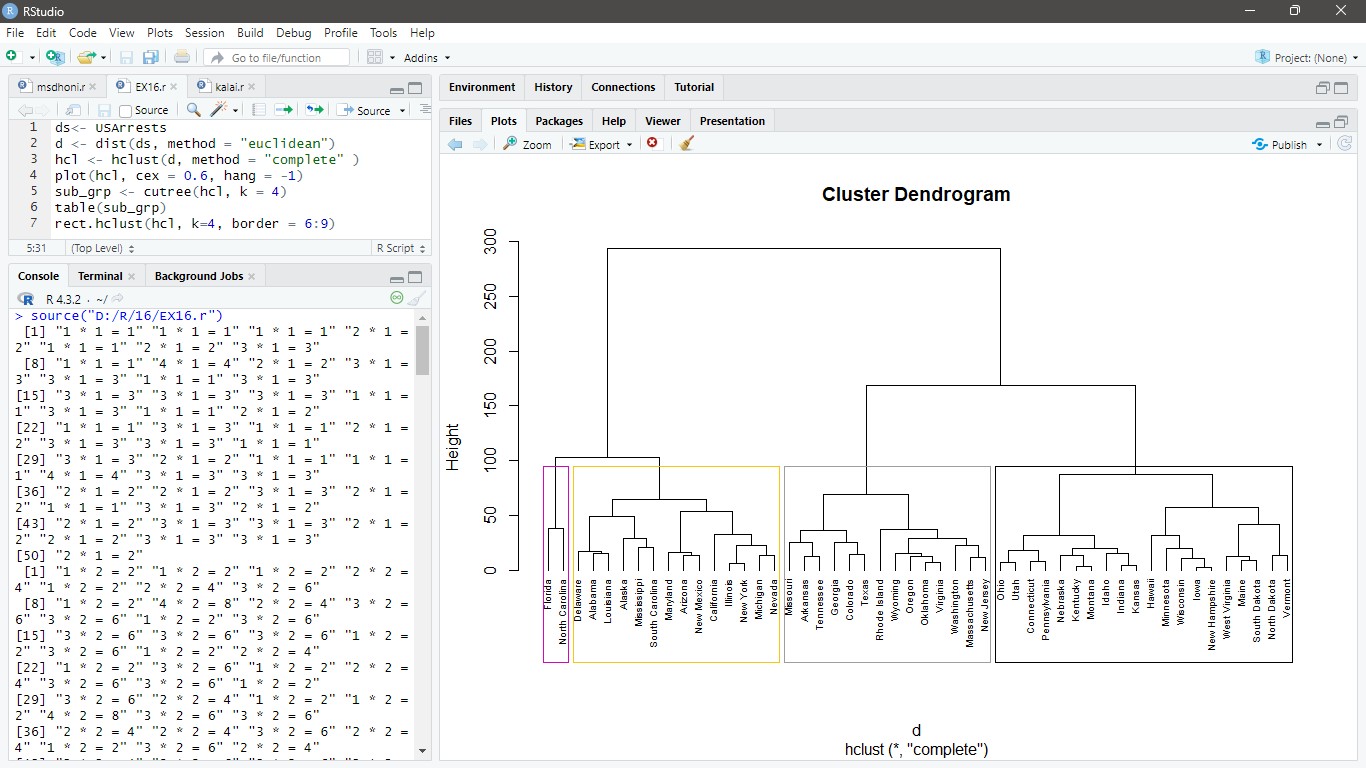
# PROGRAM :

ds<- USArrests

d <- dist(ds, method = "euclidean") hcl <- hclust(d, method = "complete" ) plot(hcl, cex = 0.6, hang = -1) sub\_grp <- cutree(hcl, k = 4) table(sub\_grp)

rect.hclust(hcl, k=4, border = 6:9)

# OUTPUT :



**EX.NO : 17**

**DATE :**

# AIM :

**CLUSTERING USARRESTS USING DIVISIVE HIERARCHICAL CLUSTERING**

To show the implementation of Divisive Hierarchical Clustering using USArrest Dataset.

# DIVISIVE HIERARCHICAL CLUSTERING :

Divisive Hierarchical Clustering is a top-down clustering method where we assign all of the observations to a single cluster and then partition the cluster to two least similar clusters.This is exactly opposite to Agglomerative clustering.Finally, we proceed recursively on each cluster until there is one cluster for each observation.

# PROCEDURE :

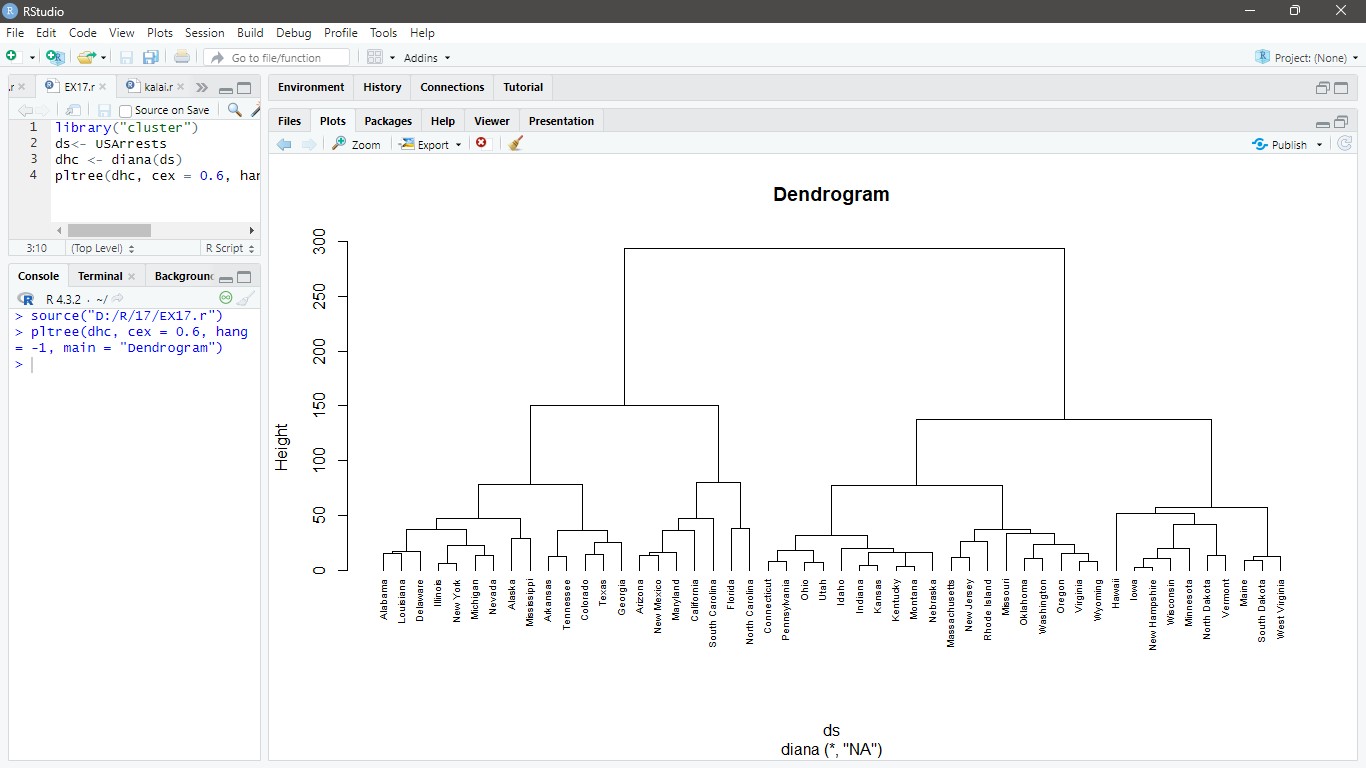
1. Make each data point a single-point cluster → forms N clusters.
2. Take the two closest clusters and make them one cluster → Forms N-1 clusters.
3. Repeat step-3 until you are left with only one cluster.

# PROGRAM :

library("cluster") ds<- USArrests dhc <- diana(ds)

pltree(dhc, cex = 0.6, hang = -1, main = "Dendrogram")

# OUTPUT :



**EX.NO : 18**

**DATE :**

# AIM :

**CLUSTERING IRIS DATASET USING DBSCAN**

To show the implementation of DBSCAN using Iris dataset.

# DBSCAN :

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. It is a density-based clustering algorithm that works on the assumption that clusters are dense regions in space separated by regions of lower density.It groups ‘densely grouped’ data points into a single cluster. It can identify clusters in large spatial datasets by looking at the local density of the data points. DBSCAN requires only two parameters: epsilon and minPoints. Epsilon is the radius of the circle to be created around each data point to check the density and minPoints is the minimum number of data points required inside that circle for that data point to be classified as a Core point.

# PROCEDURE :

1. Load the dataset.
2. Randomly select a point p.
3. Retrieve all the points that are density reachable from p with regard to maximum radius of the neighbourhood(eps) and minimum number of points within eps neighborhood(Min pts).
4. If the number of points in the neighborhood is more than Min pts then p is a core point.
5. For p core points, a cluster is formed. If p is not a core point, then mark it as a noise/outlier and move to the next point.
6. Continue the process until all the points have been processed.

# PROGRAM :

install.packages("fpc") library(fpc)

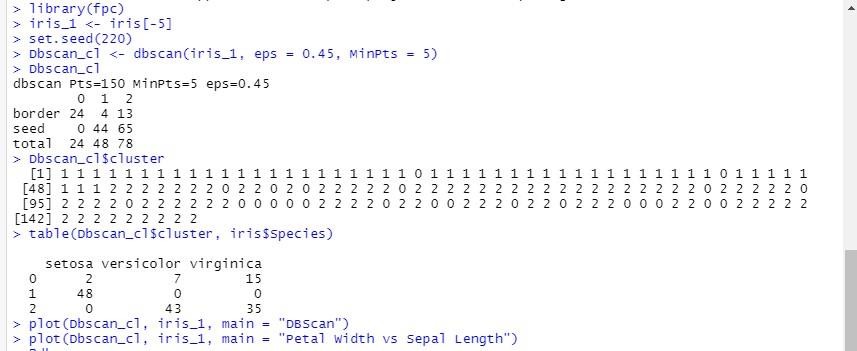
iris\_1 <- iris[-5] set.seed(220)

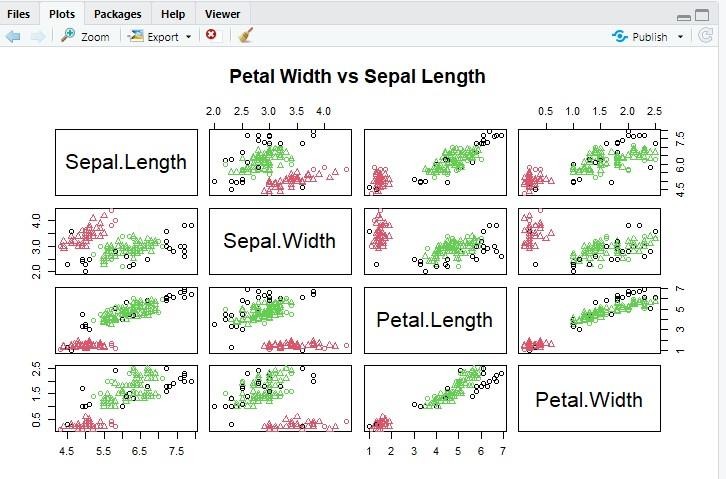
Dbscan\_cl <- dbscan(iris\_1, eps = 0.45, MinPts = 5) Dbscan\_cl

Dbscan\_cl$cluster table(Dbscan\_cl$cluster, iris$Species) plot(Dbscan\_cl, iris\_1, main = "DBScan")

plot(Dbscan\_cl, iris\_1, main = "Petal Width vs Sepal Length")

# OUTPUT :





**EX.NO : 19**

**DATE :**

# AIM :

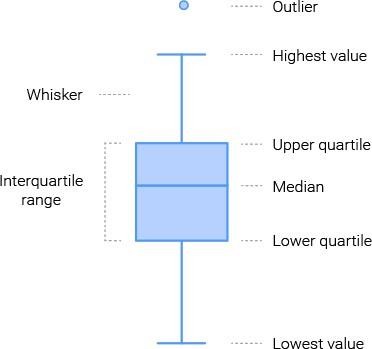
**VISUALIZATIONS**

To illustrate the concept of Visualization in R.

# VISUALIZATIONS :

Data visualization is the technique used to deliver insights in data using visual cues such as graphs, charts, maps, and many others. This is useful as it helps in intuitive and easy understanding of the large quantities of data and thereby make better decisions regarding it.

* **Pie Chart :** A Pie Chart is a special chart that shows relative sizes of data using pie slices.
* **Bar Plot :** A bar plot represents data in rectangular bars with length of the bar proportional to the value of the variable.
* **Scatter Plot** : Scatter plots show many points plotted in the Cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.
* **Histogram :** A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges.
* **Box Plot :** The box-whisker plot (or a boxplot) is a quick and easy way to visualize complex data where you have multiple samples.



1. Create a pie chart which shows the monthly expenditure?

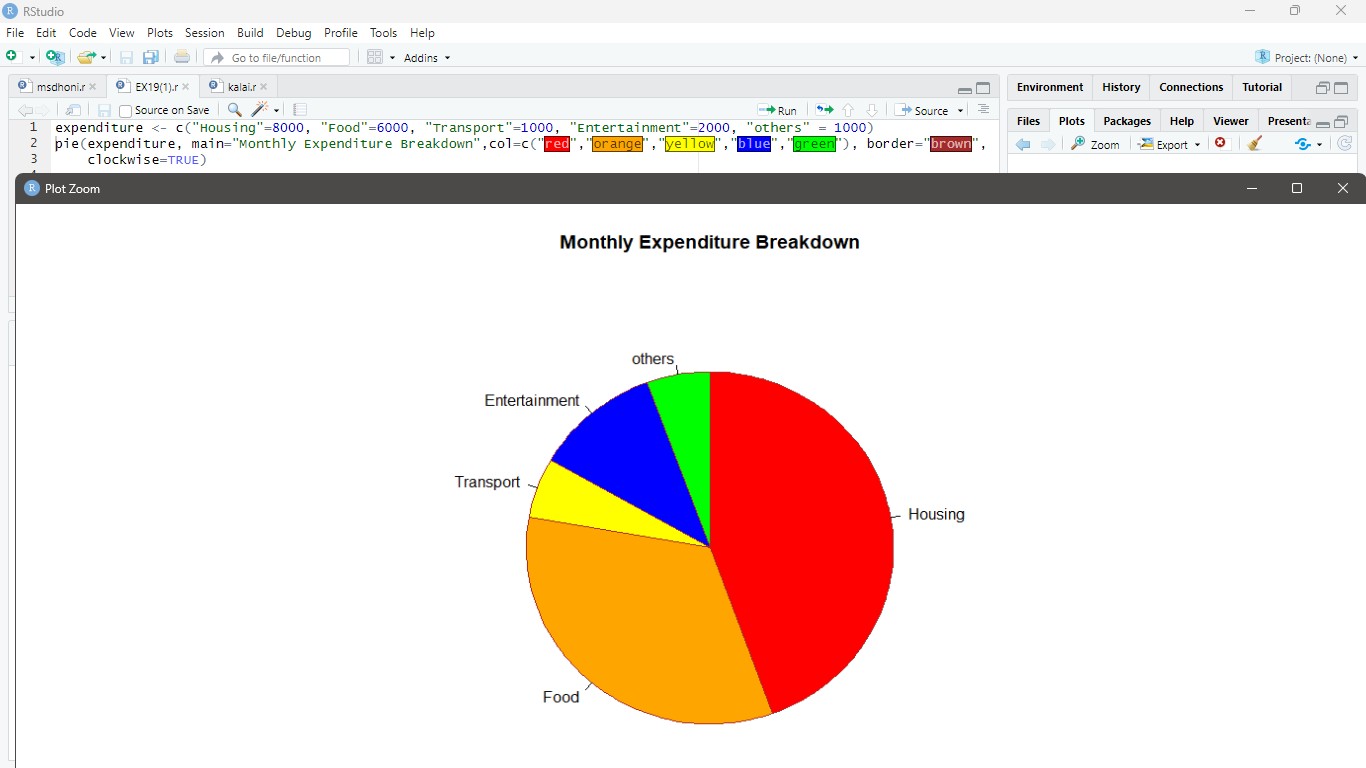
expenditure<-c("Housing"=8000, "Food"=6000, "Transport"=1000, "Entertainment"=2000, "others" = 1000)

# #Draw Pie Chart

pie(expenditure, main="Monthly Expenditure Breakdown", col=c("red","orange","yellow","blue","green"), border="brown",

clockwise=TRUE)

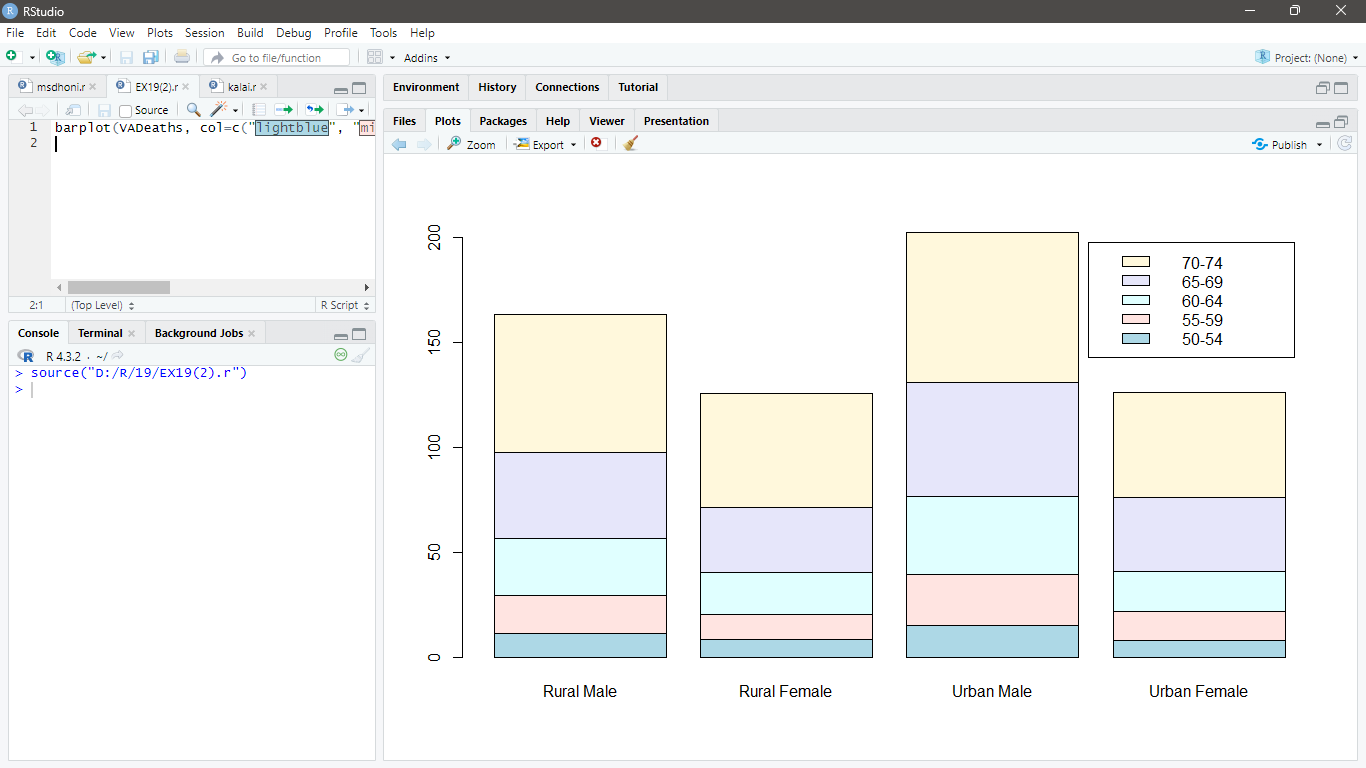
# OUTPUT :



1. Create a Bar Plot which will plot the data present in VADeaths dataset.

barplot(VADeaths, col=c("lightblue", "mistyrose", "lightcyan", "lavender", "cornsilk"), legend=rownames(VADeaths))

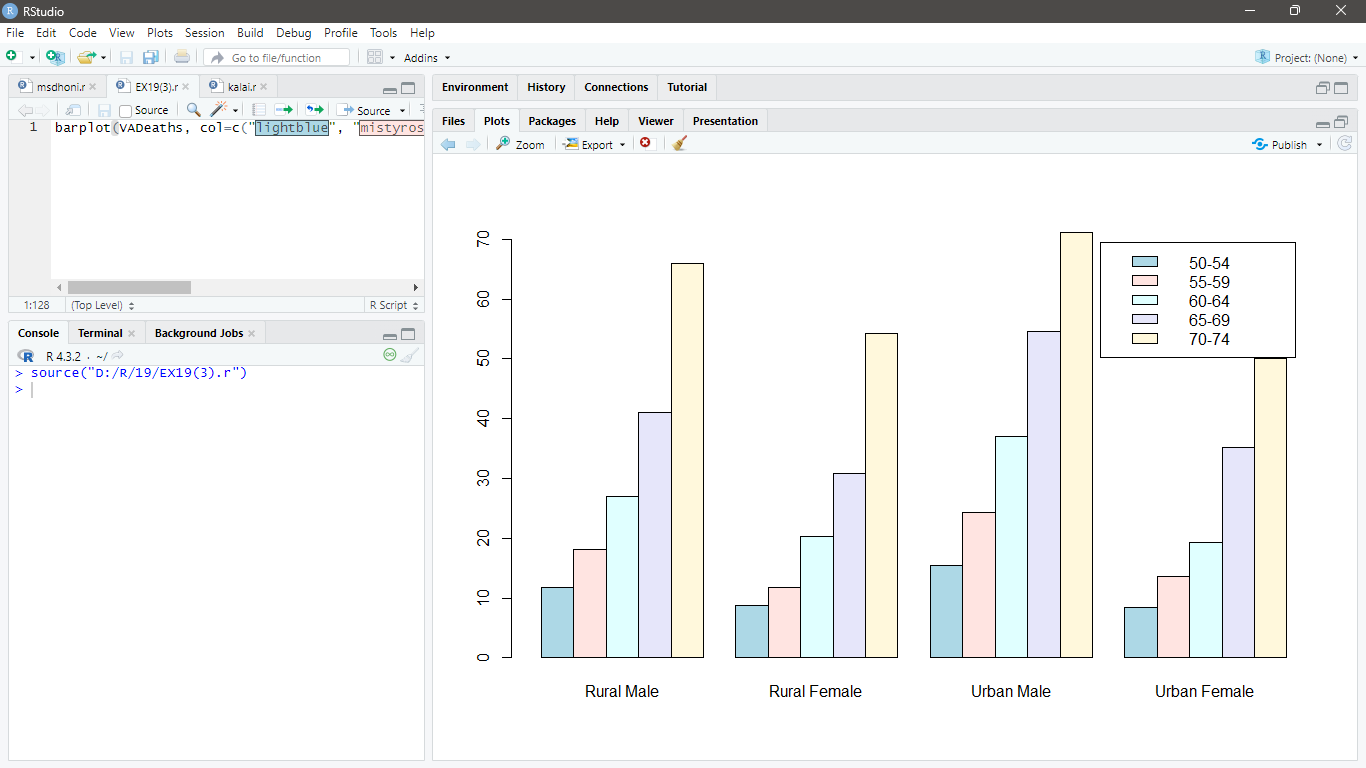
# OUTPUT :



1. Create a Bar Plot which will plot the data present in VADeaths dataset.

barplot(VADeaths, col=c("lightblue", "mistyrose", "lightcyan", "lavender", "cornsilk"), legend=rownames(VADeaths), beside=TRUE)

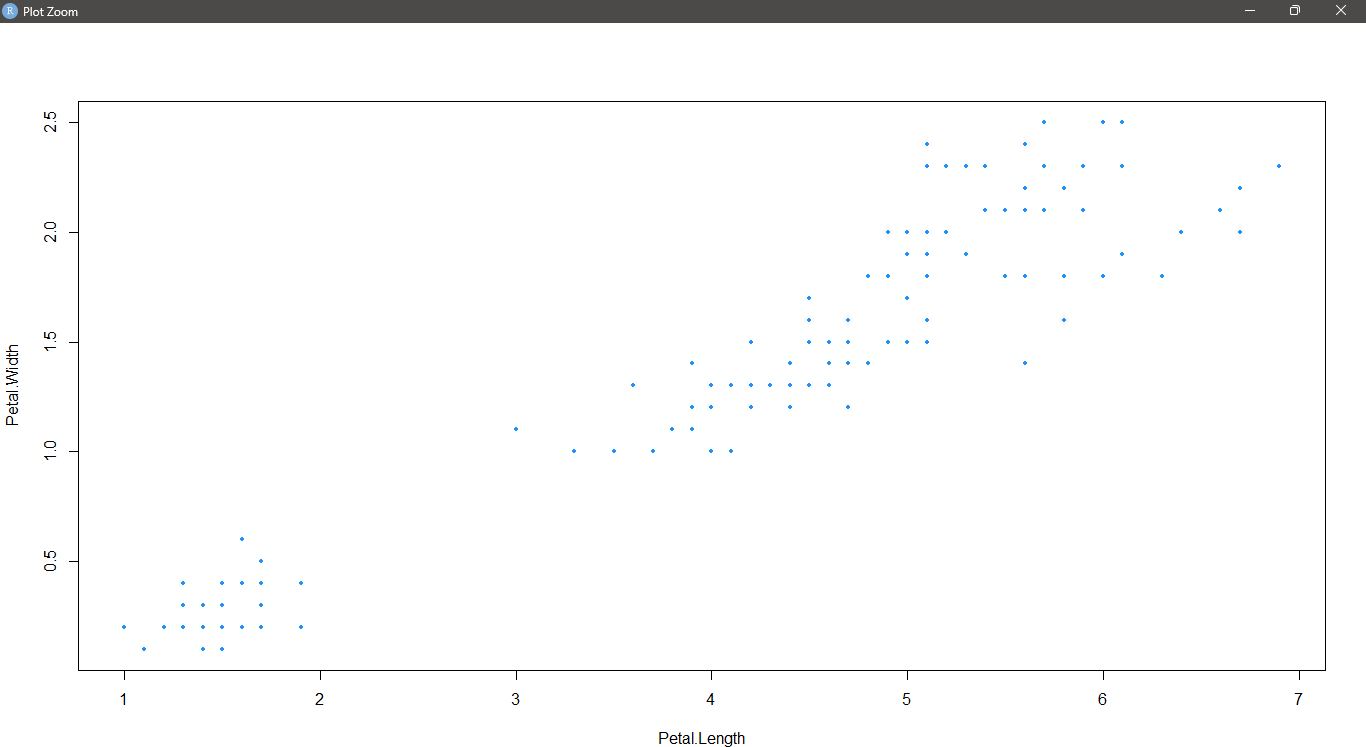
# OUTPUT :



1. Create a Scatter Plot which will plot the data present in iris dataset.

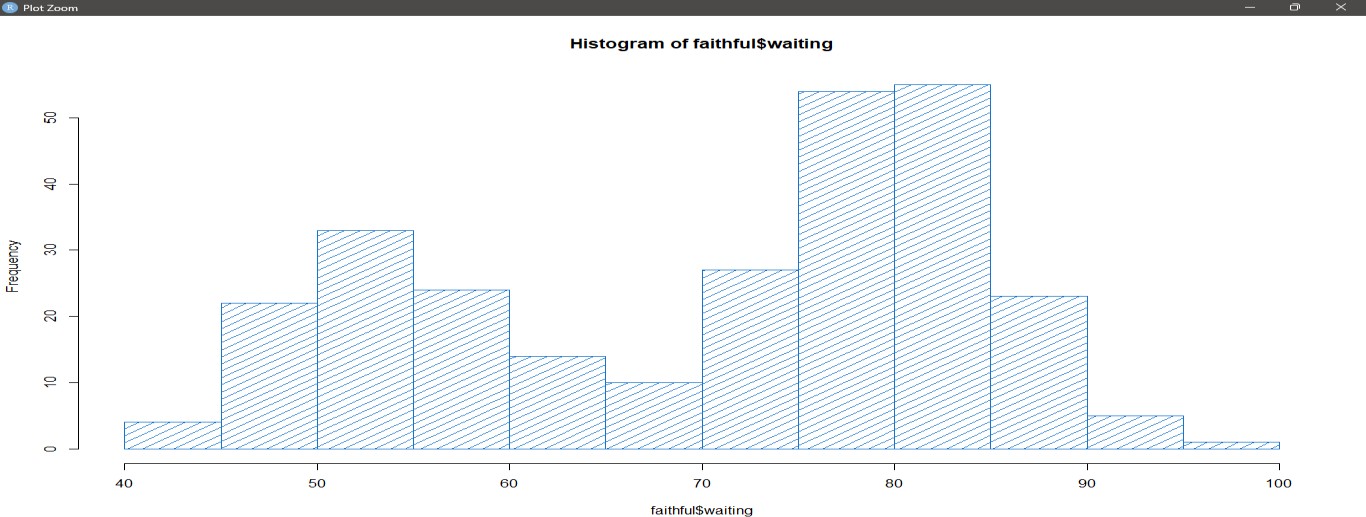
plot(Petal.Width ~ Petal.Length, data=iris, pch=20, cex=0.8, col="dodgerblue1")

# OUTPUT :



1. Create a histogram which will plot the data present in faithful dataset. hist(faithful$waiting, col="dodgerblue3", density=25, angle=60)

# OUTPUT :

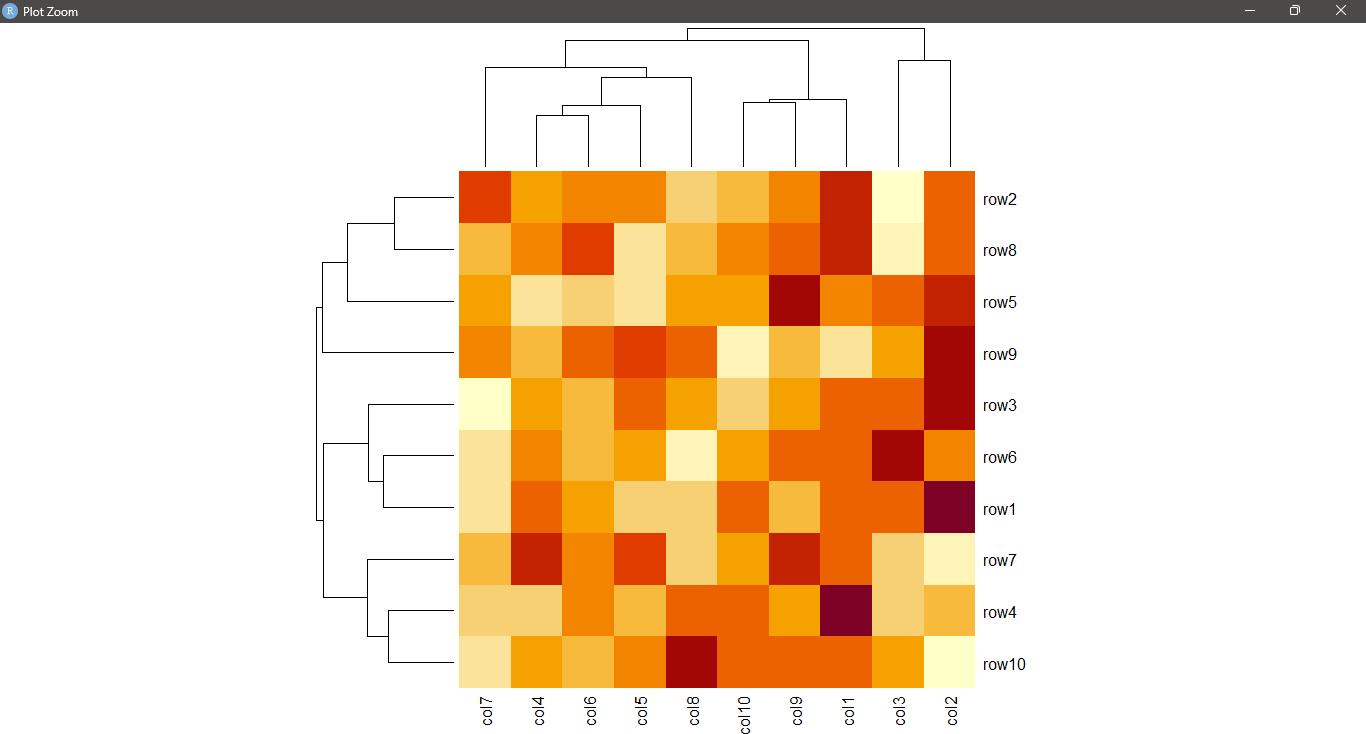


1. Create a heatmap in R programming language. set.seed(110)

data <- matrix(rnorm(100, 0, 5), nrow = 10, ncol = 10) colnames(data) <- paste0("col", 1:10)

rownames(data) <- paste0("row", 1:10) heatmap(data)

# OUTPUT :



1. Create a box plot using R programming language with ToothGrowth dataset.

boxplot(len ~ dose, data = ToothGrowth)

# OUTPUT :

