**WORKING WITH R**

**Aim**

To work with R IDE and learn about its windows and menus.

**Introduction**

**"R is an interpreted computer programming language which was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand."** The ***R Development Core Team*** currently develops R. It is also a software environment used to

analyze **statistical information**, **graphical representation**, **reporting**, and **data modeling**. R is the implementation of the **S programming** language, which is combined with **lexical scoping semantics**.

**Installing R**

Official website for installing R: [**http://cloud.r-project.org**](http://cloud.r-project.org/) **RStudio IDE**

RStudio is an integrated development environment which allows us to interact with R more readily.

RStudio is similar to the standard RGui, but it is considered more user-friendly. This IDE has various drop-down menus, Windows with multiple tabs, and so many customization processes.

The first time when we open RStudio, we will see three Windows. The fourth Window will be hidden by default. We can open this hidden Window by clicking the **File** drop-down menu, then **New File** and then **R Script**.

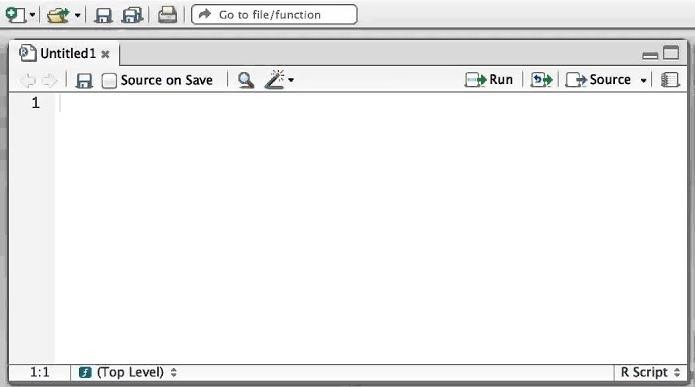
**RStudio Windows and Description**

|  |  |  |
| --- | --- | --- |
| **RStudio Windows/Tabs** | **Location** | **Description** |
| Console Window | Lower-left | The location where commands are entered and output is printed. |
| Source Tabs | Upper-left | Built-in test editor |
| Environment Tab | Upper-left | An interactive list of loaded R objects. |
| History Tab | Upper-left | List of keystrokes entered into the console. |
| Files Tab | Lower- right | File explorer to navigate C drive folders. |
| Plots Tab | Lower- right | Output location for plots. |
| Packages Tab | Lower- right | List of installed packages. |
| Help Tab | Lower- right | Output location for help commands and help search Window. |
| Viewer Tab | Lower- right | Advanced tab for local web content. |

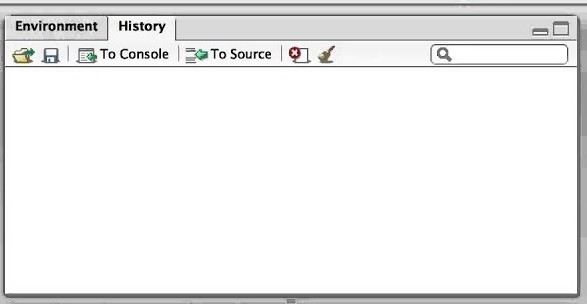
**To Run file in R IDE**

We should run a file using run option, which is present in the upper left panel. Place the cursor on the line where you want to run.

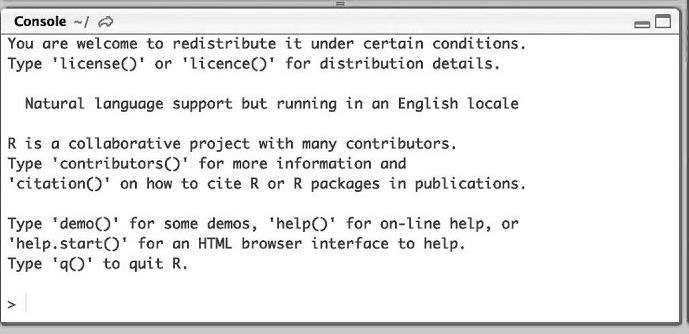
**Upper-left (Source Tab)**



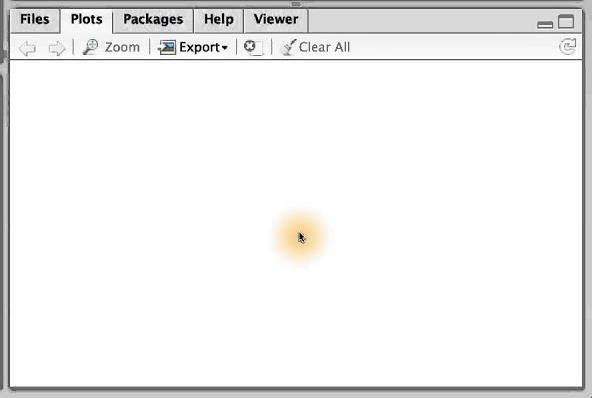
**Upper-right**



**Bottom-left**



**Bottom-right**



**Result**

Thus, we successfully learnt about R IDE.

**Aim**

# FIBONACCI SERIES

To write a program to find the Fibonacci sequence of the given number using R.

**Algorithm**

1. Take the input from the user.
2. Initialize, n1=0 and n2=1 and count=1.
3. If the input is less than or equal to zero then print “Please enter a positive integer”.
4. Else print Fibonacci sequence for the given input.
5. Use a while to find next term in the sequence.
6. Inside the while loop, print the first two terms n1 and n2.
7. Then calculate the next term nth by adding the last two term and print it.

**Source Code**

nterms <- as.integer(readline(prompt="How many terms? ")) n1 = 0

n2 = 1

count = 2 if(nterms <= 0) {

print("Please enter a positive integer")

} else {

if(nterms == 1) { print("Fibonacci sequence:")

print(n1)

} else {

print("Fibonacci sequence:") print(n1)

print(n2)

while(count < nterms) { nth = n1 + n2

print(nth) n1 = n2 n2 = nth

count = count + 1

}

}

}

**Output**

How many terms? 7 "Fibonacci sequence:"

[1] 0

[1] 1

[1] 1

[1] 2

[1] 3

[1] 5

[1] 8

How many terms? 0

[1] "Please enter a positive integer"

How many terms? 1

1. "Fibonacci sequence:"

[1] 0

**Result:**

Thus, the above program was executed successfully and the output was verified.

**Aim**

# FACTORIAL

To write a program to find the factorial of the given number using R.

**Algorithm**

* 1. Get the input from the user.
  2. Initialize factorial=1.
  3. If the input is less than 0, then print “sorry, factorial does not exist for negative numbers”
  4. Else if the input is equal to zero, then print factorial value for input.
  5. Use for loop to calculate the factorial
  6. Factorial =factorial \* i

**Source Code**

num = as.integer(readline(prompt="Enter a number: ")) factorial = 1

if(num < 0) {

print("Sorry, factorial does not exist for negative numbers")

} else if(num == 0) { print("The factorial of 0 is 1")

} else {

for(i in 1:num) { factorial = factorial \* i

}

print(paste("The factorial of", num ,"is",factorial))

}

**Output**

Enter a number: 6

[1] "The factorial of 6 is 720"

Enter a number: 0

[1] "The factorial of 0 is 1"

Enter a number: -3

1. "Sorry, factorial does not exist for negative numbers"

**Result:**

Thus, the above program was executed successfully and the output was verified.

**Aim**

# SUM OF SERIES

To write a program to find the sum of series using R.

**Algorithm**

* 1. Get the input from the user.
  2. If the input is less than zero, print “Enter a positive number”.
  3. Else Calculate sum=(num\*(num+1))/2
  4. Display the sum of series up to the given input.

**Source Code**

num = as.integer(readline(prompt = "Enter a number: ")) if(num < 0) {

print("Enter a positive number")

} else {

sum = (num \* (num + 1)) / 2; print(paste("The sum is", sum))

}

**OUTPUT**

Enter a number: -4

[1] "Enter a positive number"

Enter a number: 6

1. "The sum is 21"

**Result:**

Thus, the above program was executed successfully and the output was verified.

# PALINDROME

**Aim**

To write a program to check whether the given number is palindrome or not.

**Algorithm**

* 1. Get the input from the user.
  2. Initialize rev=0, num=n.
  3. Use while loop to reverse the given input and print
  4. If reverse equal to the given input, then print “palindrome”.
  5. Else print “Not palindrome”.

**Source Code**

n <- as.integer(readline(prompt=”Enter a four digit number please: “)) num<-n

rev<-0 while(n!=0){

rem<-n%%10

rev<-rem+(rev\*10) n<-as.integer(n/10)

}

print(rev) if(rev==num){

cat(num,”is a palindrome number”)

}else{

cat(num,”is not a palindrome number”)

}

**OUTPUT**

Enter a four digit number please: 8008 8008 is a palindrome number

Enter a four digit number please: 5403 5403 is not a palindrome number

**Result:**

Thus, the above program was executed successfully and the output was verified.

Ex.No. **Working with R Data types** Page No: Date:

**Aim**

To write a program to working with R data types in R studio.

**Algorithm**

* + 1. There are several datatypes in R.
    2. We have use Logical, Numeric, Real, Integer, Complex**,** Character, Raw, List, Vector, Matrix, Array, Factor, Data frame.
    3. Using these data types, we had implemented some examples .

**Source Code #Logical data type**

bool1 <- TRUE print(bool1) print(class(bool1)) bool2 <- FALSE print(bool2) print(class(bool2)) **#Numeric data type**

**# floating point values** weight <- 63.5 print(weight) print(class(weight))

**# Real numbers** height <- 182 print(height) print(class(height)) **#Integer datatype**

integer\_variable<- 186L print(class(integer\_variable)) **# Complex data type** complex\_value<- 3 + 2i

**# print class of complex\_value**

print(class(complex\_value)) complex\_value

**#Character data type**

**# Create a string variable**

fruit <- "Apple" print(class(fruit))

**# Create a character variable** my\_char<- 'A' print(class(my\_char))

**#Raw data type**

**# convert character to raw**

raw\_variable<- charToRaw("Welcome to r program") print(raw\_variable)

print(class(raw\_variable))

**# convert raw to character** char\_variable<- rawToChar(raw\_variable) print(char\_variable) print(class(char\_variable))

**# Create a vector.**

apple <- c('red','green',"yellow") print(apple)

**# Get the class of the vector.**

print(class(apple))

**# Create a list.**

list1 <- list(c(2,5,3),21.3,sin)

**# Print the list.**

print(list1)

**# Create a matrix.**

M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE) print(M)

**# Create an array.**

a <- array(c('green','yellow'),dim = c(3,3,2)) print(a)

**# Create a vector.**

apple\_colors<- c('green','green','yellow','red','red','red','green')

**# Create a factor object.**

factor\_apple<- factor(apple\_colors)

**# Print the factor.** print(factor\_apple) print(nlevels(factor\_apple))

**# Create the data frame.**

BMI < - data.frame(

gender = c("Male", "Male","Female"), height = c(152, 171.5, 165),

weight = c(81,93, 78), Age = c(42,38,26)

)

print(BMI)

**OUTPUT:**

[1] TRUE

[1] "logical"

[1] FALSE

[1] "logical" [1] 63.5

[1] "numeric" [1] 182

[1] "numeric"

[1] "integer"

[1] "complex" [1]"character" [1]"character"

[1] 57 65 6c 63 6f 6d 65 20 74 6f 20 72 20 70 72 6f 67 72 61 6d

[1] "raw"

[1] "Welcome to r program"

[1] "character"

[1] "red" "green" "yellow"

[1] "character" [[1]]

[1] 2 5 3

[[2]]

[1] 21.3

[[3]]

function (x) .Primitive("sin") [,1] [,2] [,3]

[1,] "a" "a""b"

[2,] "c" "b""a"

, , 1

[,1] [,2] [,3]

[1,] "green" "yellow" "green"

[2,] "yellow" "green" "yellow"

[3,] "green" "yellow" "green"

, , 2

[,1] [,2] [,3]

[1,] "yellow" "green" "yellow"

[2,] "green" "yellow" "green"

[3,] "yellow" "green" "yellow"

[1] green greenyellowred red red green Levels: green redyellow

[1] 3

gender height weight Age 1 Male152.0 81 42

2 Male171.5 93 38

3 Female165.0 78 26

**Result:**

Thus, the above program was executed successfully and the output was verified.

Ex. No. **Data import and export** Page No: Date:

**Aim**

To write a program to data import and export in R studio.

**Algorithm**

1. Install the “readxl” package using **install.packages()** command.
2. Then load the package using library(“readxl”)
3. Store the excel value to excel variable using read\_xlsx() methed.
4. Print the variable excel.
5. Import the csv file to data variable using read.csv() method.
6. Print the variable data.
7. Install the “openxlsx” package using **install.packages()** command.
8. Then load the package using library(“openxlsx”)
9. Create s.no,fruits,shop1,shop2 value using c() method.
10. Frame the all data using data.frame() methed and store in rate object.
11. Then write the dataset in specified location using write.xlsx(rate,file=”d://”)

**Source Code #Data import**

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

excel<-read\_xlsx("student.xlsx",sheet=1) excel

**#csv file**

data<-read.csv("stud\_1.csv") data

**OUTPUT:**

idname englishmathsscience

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *<dbl><chr>* | | *<dbl><dbl>* | | *<dbl>* |
| 1 | 1anu | 90 | 98 | 89 |
| 2 | 2banu | 87 | 99 | 92 |
| 3 | 3mani | 89 | 100 | 90 |
| 4 | 4kavi | 90 | 89 | 98 |
| 5 | 5hari | 91 | 90 | 96 |
| 6 | 6janu | 93 | 92 | 98 |

idname englishmathsscience

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *<dbl><chr>* | | *<dbl><dbl>* | | *<dbl>* |
| 1 | 1venu | 90 | 98 | 89 |
| 2 | 2guru | 87 | 99 | 92 |
| 3 | 3jemi | 89 | 100 | 90 |

**#Data export** install.packages("openxlsx") library(openxlsx)

s.no <- seq(1,3,by=1)

fruits <- c("apple","mango","orange") shop1 <-c(12,14,16)

shop2 <- c(22,15,24) shop3 <- c(12,15,14) shop4 <-c(66,26,33)

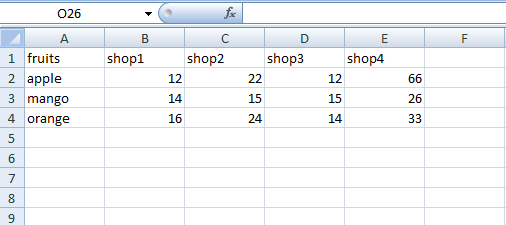
rate <- data.frame(fruits,shop1,shop2,shop3,shop4) rate

write.xlsx(rate,file="G:/R/fruits.xlsx")

**OUTPUT:**

fruits shop1 shop2 shop3 shop4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1apple | 12 | 22 | 12 | 66 |
| 2mango | 14 | 15 | 15 | 26 |
| 3orange | 16 | 24 | 14 | 33 |



**Result:**

Thus, the above program was executed successfully and the output was verified.

Ex. No. **Plots and Graphs** Page No: Date:

**Aim**

To write a program for working with plots and graphs in R studio.

**Algorithm**

1. Initialize x variable and store the numbers using c() function.
2. Using barplot(x) function to display the barplot.
3. Initialize y variable and store the numbers using c() function.
4. Put the name for particular value using name() method.
5. Display the pie chart using pie(y) method.
6. Set the z value that take from Orange predefined dataset.
7. Create scatter plot using the plot() method.

**Source Code**

**# Bar Plot or Bar Chart**

x <- c(7, 15, 23, 12, 44, 56, 32)

barplot(x, xlab = "Student", ylab = "Mark", col = "white", col.axis = "darkgreen", col.lab = "darkgreen")

**# Pie Diagram or Pie Chart**

y <- c(210, 450, 250, 100, 50, 90)

names(y) <- c("HTML", "DS", "Java", "C", "C++", "Python") pie(y, labels = names(y), col = "white",

main = "Student interest area", radius = -1, col.main = "darkblue")

**#Scatter Plot**

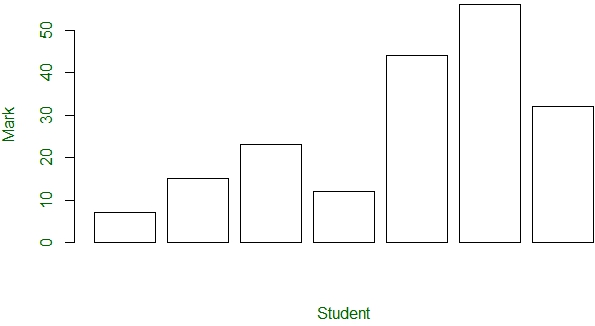
z <- Orange[, c('age', 'circumference')]

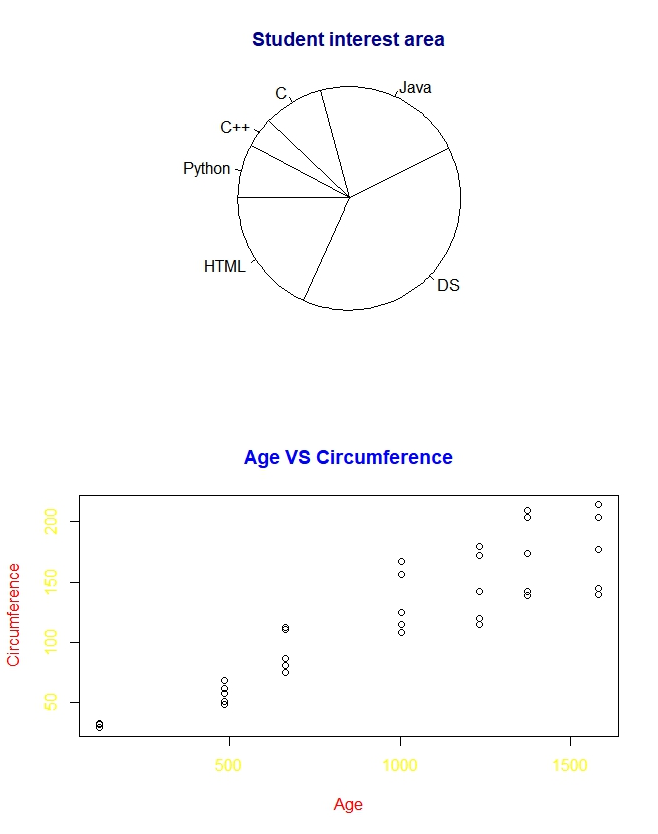
plot(x = z$age, y = z$circumference, xlab = "Age",

ylab = "Circumference", main = "Age VS Circumference", col.lab = "red", col.main = "blue",

col.axis = "yellow")

**Output**





**Result:**

Thus, the above program was executed successfully and the output was verified.

Ex.No: Date:

# CLUSTERING TECHNIQUES- FINDING NUMBER OF CLUSTERS

Page No:

## AIM:

To write a R program for finding the optimal number of clusters by using the elbow method (WSS).

## ALGORITHM:

* 1. Load the required packages - factoextra and ggplot2.
  2. Import the dataset.
  3. Scale each variable in the dataset.
  4. Find the Optimal Number of Clusters k , by using the elbow method**.**
  5. To perform k-means clustering in R , Use the built-in kmeans() function.
  6. Using the fviz\_nbclust() function, plot the number of clusters vs. the total within sum of squares by using the method = “wss”.
  7. Performing k-means clustering on the dataset, the optimal value for *k* will be displayed.

## CODING:

data1.s<-scale(data1[-1:-3]) data1.s

?dist

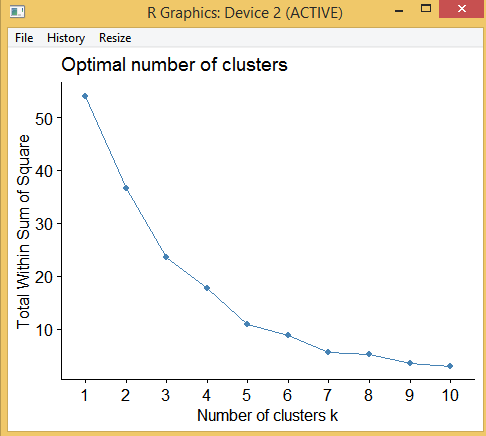
d<-dist(data1.s) options(max.print=1000000) d

k1<-kmeans(data1.s,1) k1

k1$cluster library(ggplot2) library(factoextra)

fviz\_nbclust(data1.s,kmeans,method="wss")

## OUTPUT:



**RESULT:**

Thus the above program is executed successfully and the output is verified.

Ex.No: Date:

# CLUSTERING TECHNIQUES- KMEANS ALGORITHM

Page No:

## AIM:

To write a R program for determining appropriate cluster value by using K-means Algorithm

## ALGORITHM:

1. Install the packages using this command install.packages (“packages\_name”).
2. Load the R libraries and use the predefined dataset.
3. Create a function wssplot using function().
4. Calculate the maximum number of clusters using wss plot.
5. Find the Cluster value using kmeans() function.
6. Auto plot the cluster value.

install.packages('stats') install.packages('dplyr') install.packages('ggplot2') install.packages('readxl')

library('stats') library('dplyr') library('ggplot2') library('ggfortify') library('readxl')

data<-read\_xlsx("student\_1.xlsx",sheet = 1) data

mydata=select(data,c(3,4,5)) mydata

wss<-numeric(15)

for(k in 1:15) wss[k]<-sum(kmeans(mydata,centers = k,nstart=25)$withinss)

plot(1:15,wss,type="b",xlab="Number of clusters",ylab="within groups sum of squares")

km=kmeans(mydata,3) autoplot(km,mydata,frame=TRUE)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **data** |  | | | |
| sno | name | english | math | science |
| 1 1 | a | 90 | 95 | 100 |
| 2 2 | b | 91 | 96 | 81 |
| 3 3 | c | 92 | 97 | 82 |
| 4 4 | d | 93 | 98 | 83 |
| 5 5 | e | 94 | 99 | 94 |
| 6 6 | f | 95 | 100 | 87 |
| 7 7 | g | 96 | 81 | 98 |
| 8 8 | h | 97 | 82 | 98 |
| 9 9 | i | 98 | 83 | 95 |
| 10 10 | j | 99 | 94 | 96 |
| 11 11 | k | 100 | 87 | 97 |
| 12 12 | l | 81 | 98 | 98 |
| 13 13 | m | 82 | 98 | 99 |
| 14 14 | n | 83 | 56 | 100 |
| 15 15 | o | 94 | 87 | 87 |
| 16 16 | p | 87 | 90 | 90 |
| 17 17 | q | 98 | 91 | 91 |
| 18 18 | r | 98 | 92 | 92 |
| 19 19 | s | 56 | 93 | 93 |
| 20 20 | t | 87 | 94 | 94 |

## mydata

english math science 1 90 95 100

2 91 96 81

3 92 97 82

4 93 98 83

5 94 99 94

6 95 100 87

7 96 81 98

8 97 82 98

9 98 83 95

10 99 94 96

11 100 87 97

12 81 98 98

13 82 98 99

14 83 56 100

15 94 87 87

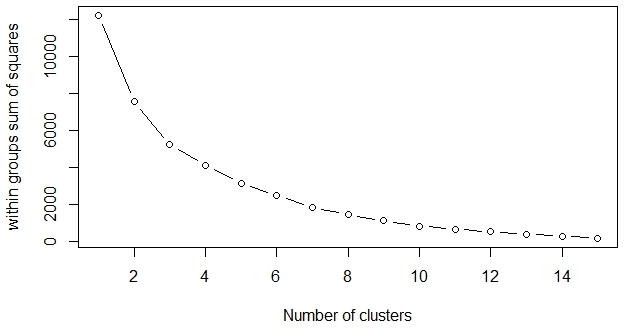
16 87 90 90

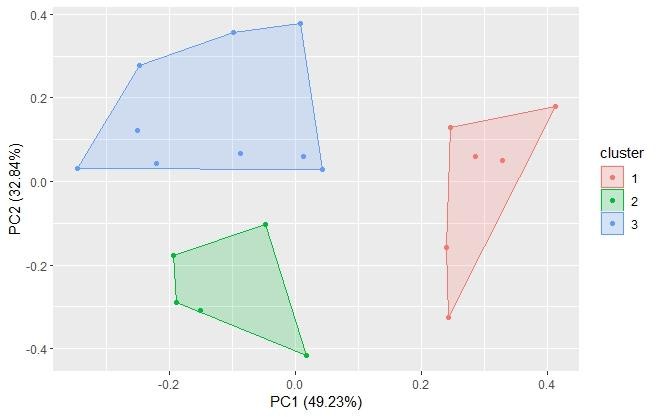
17 98 91 91

18 98 92 92

19 56 93 93

20 87 94 94





## RESULT:

Thus the program is executed and the output is verified successfully.

Ex.No: Date:

# ASSOCIATION RULE – APRIORI ALGORITHM Page No:

## AIM:

To write a R program to generate association rule using apriori algorithm.

## ALGORITHM:

1. Install the required packages – arules and arulesViz.
2. Load the default dataset “Groceries”.
3. Apply apriori algorithm with input parameter dataset\_name ,support and confidence value.
4. Plot the graph for rule1,rule2,rule3 using the method = “grouped”.
5. Display the graphical representation of all rules.

install.packages("arules") install.packages('arulesViz') library(arules) library(arulesViz) data("Groceries") summary(Groceries)

apriori(Groceries,parameter = list(support=0.002,confidence=0.5)) ->rule1 inspect(head(rule1,10))

inspect(head(sort(rule1,by="lift"),5)) plot(rule1)

plot(rule1,method = "grouped")

apriori(Groceries,parameter = list(support=0.002,confidence=0.5,minlen=3)) -

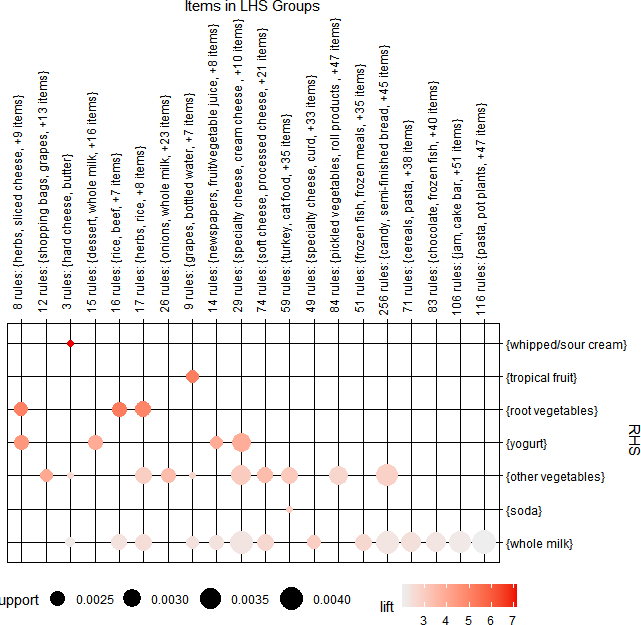
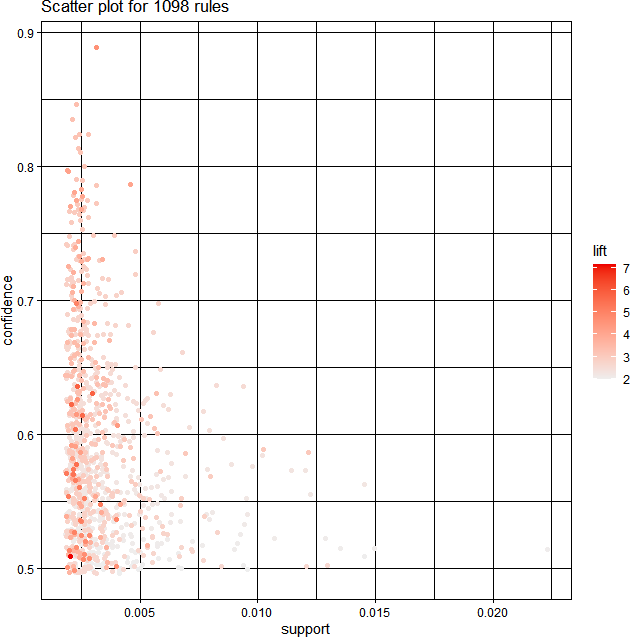
>rule2 inspect(head(rule2,7)) plot(rule2)

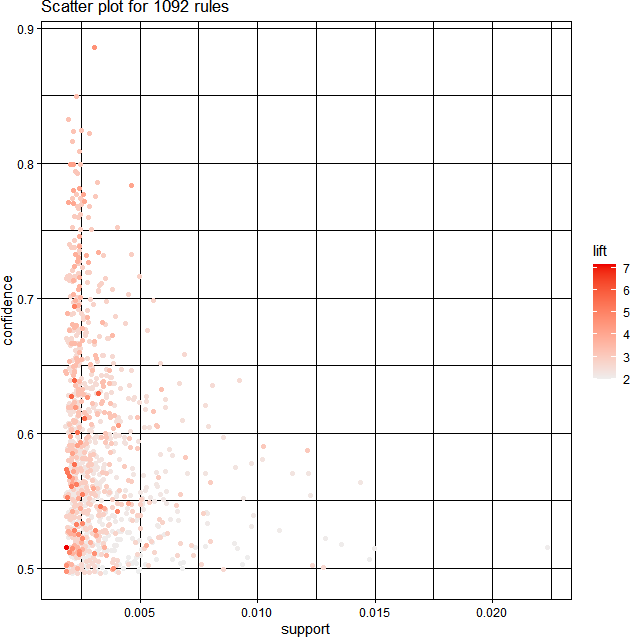
plot(rule2,method = "grouped")

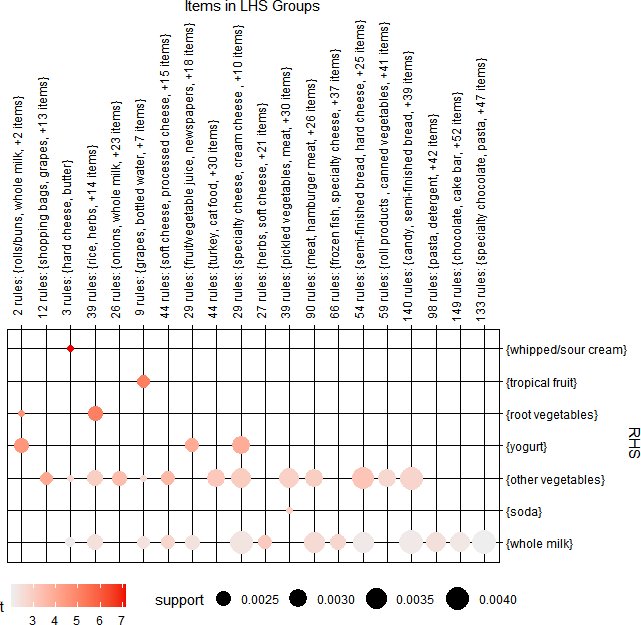
apriori(Groceries,parameter = list(support=0.007,confidence=0.6)) ->rule3 inspect(head(rule3,4))

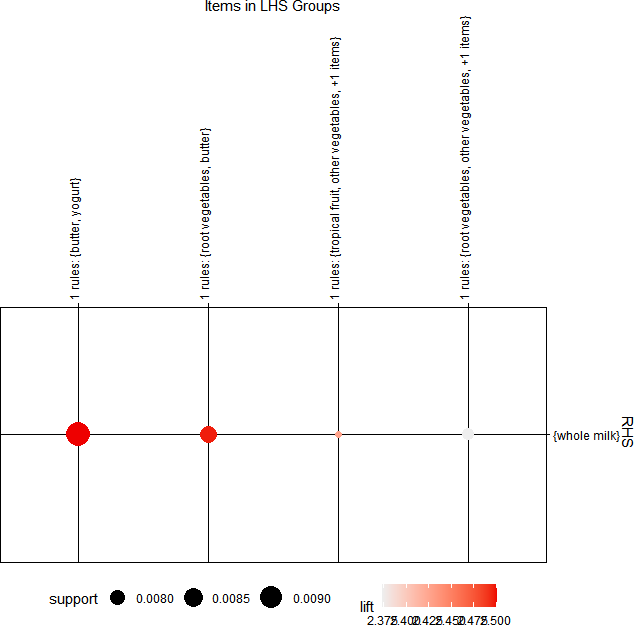
plot(rule3,method = "grouped")

## Rule1:









**RESULT:**

Thus the program is executed and the output is verified successfully.

Ex No: Date:

## AIM:

**REGRESSION** Page No:

1. **LINEAR REGRESSION**

To write a R program for Linear Regression method to analysis the statistical tool to establish model between two variables.

## ALGORITHM:

* 1. Create two variables, to store a list of values using c() function.
  2. Store the variable relation, lm() function is used to fit linear models to data frames.
  3. Use predict() function to predict values based on the input data.
  4. Print the result value.
  5. Install the “readxl” package using install.packages() command.
  6. Then load the package using library(“readxl”)
  7. View the variable ads, nrow(), ncol(), colnames() function used.
  8. Plot the Tv and Sales values using plot() function.
  9. Use lm() function for Tv and Sales.
  10. Variable model view summary(), attributes(), coefficients and abline() function is used them.

## CODING:

a<-c(151,174,138,186,128,136,179,163,152,131) b<-c(63,81,56,91,47,57,76,72,62,48)

relation<-lm(b~a)

a<-data.frame(a=170) result<-predict(relation,a) print(result)

install.packages('readxl') library('readxl')

ads<-read.csv('d:/advertising.csv') View(ads)

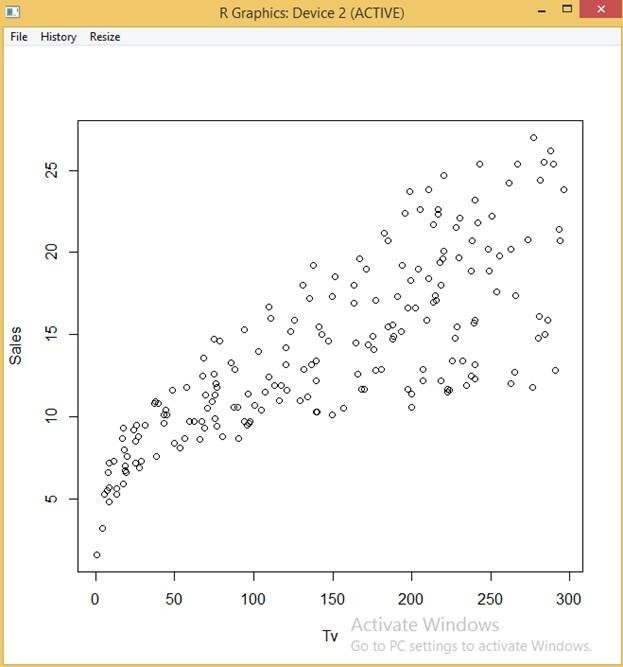
nrow(ads) ncol(ads) colnames(ads) Tv<-ads$TV Sales<-ads$Sales plot(Tv,Sales)

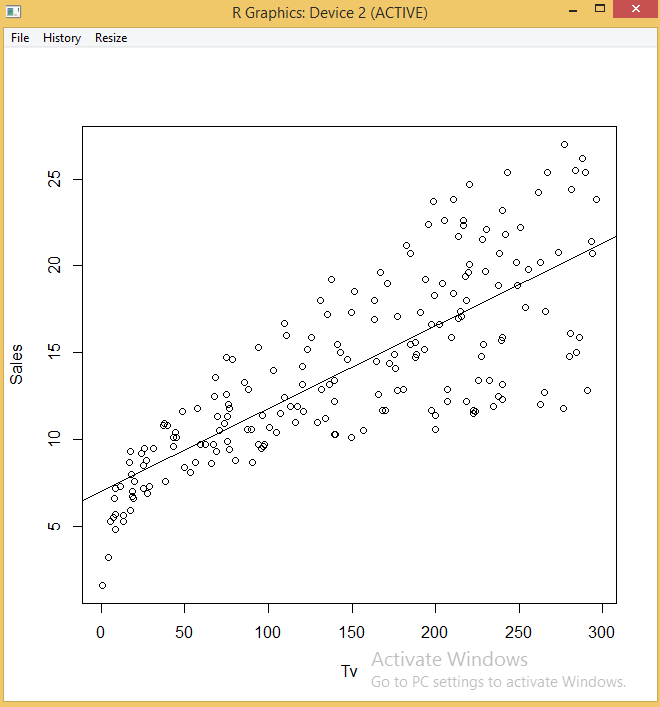
model<-lm(Sales~Tv) summary(model) attributes(model) coefficients(model) coef(model) abline(model)

## OUTPUT:

1

76.22869





## RESULT:

Thus the above R Program is executed and the output is verified successfully.

1. **LOGISTIC REGRESSION**

**AIM:**

To write a R program for Logistic Regression Method with the predictor variables.

## ALGORITHM:

1. Import the csv file “puffinbill”.
2. Extract the sex and curlen column data from the dataset “puffinbill”.
3. Create sexcode variable for translating the variables male and female into binary responses 0 and 1.
4. Plot the curlen values, using jitter() function to separate the individual bill length of thesame sex.
5. Use glm() function to summarize the model.
6. Create xv and yv variables for making a sigmoidal curve using seq() and predict() function and plot the graph using lines() function.
7. Install the package “popbio” for construction and analysis of matrix population model.
8. Plot the binary logistic regression for frequency distribution using logi.hist.plot() function.

## CODING:

sex<-puffinbill$sex curlen<-puffinbill$curlen

sexcode<-ifelse(sex == "F",1,0)

plot(curlen, jitter(sexcode, 0.15), pch = 19, xlab = "Bill length (mm)", ylab = "Sex (0 - male, 1 - female)")

model<- glm(sexcode~curlen, binomial) summary(model)

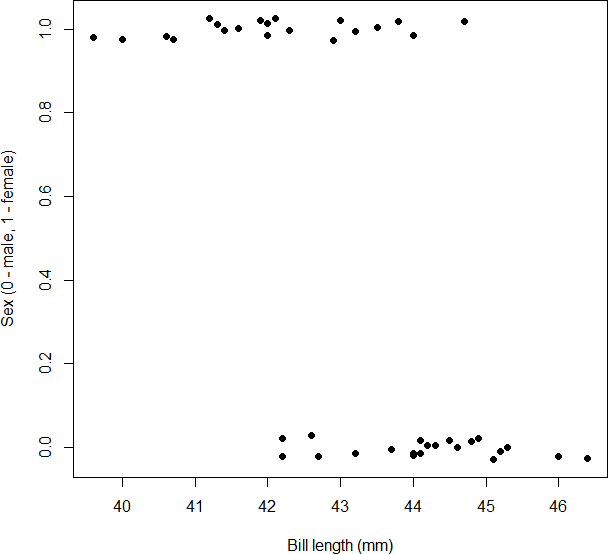
xv<- seq(min(curlen),max(curlen),0.01)

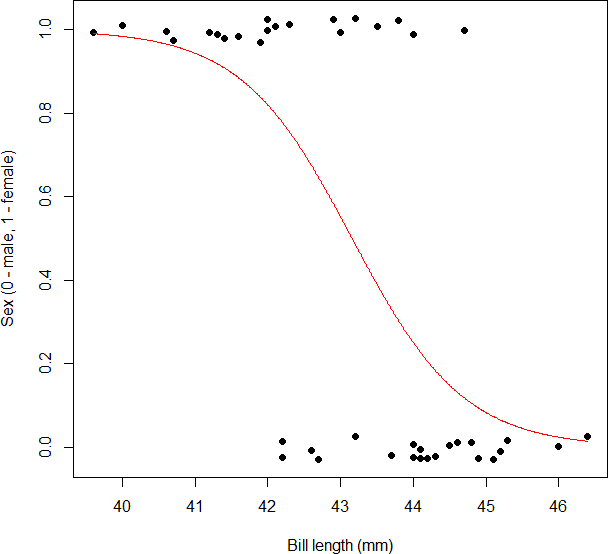
yv<- predict(model,list(curlen=xv),type="response") lines(xv, yv, col = "red")

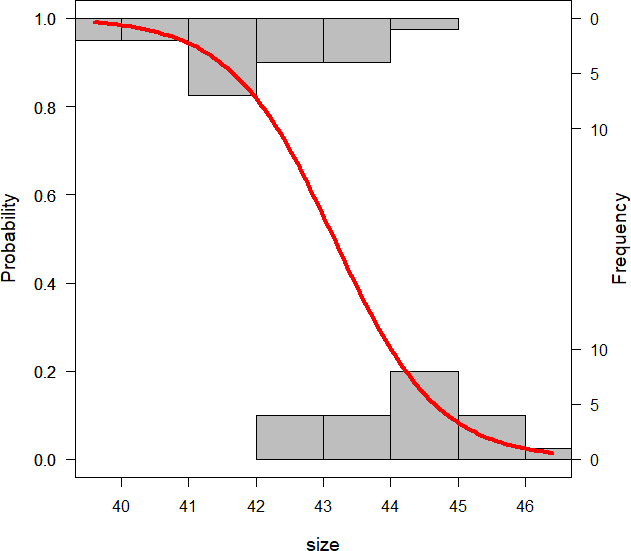
library(popbio)

logi.hist.plot(curlen,sexcode,boxp = FALSE,type= "count",col="gray",xlabel = "size")

## OUTPUT:







**RESULT:**

Thus the above R program is executed and the output is verified successfully.

Ex.No: Date:

# CLASSIFIERS

1. **NAIVE BAYES**

Page No:

## AIM:

To write a R program to illustrate the Naive Bayes Classifier.

## ALGORITHM:

* 1. Load the data.
  2. Install the required packages “e1071”,”caTools”,”caret” using install.packages() command.
  3. Then load the packages using library() command.
  4. Split the data into train and test data.
  5. Scale the feature.
  6. Fit the Naïve Bayes model to training dataset.
  7. Predict the test data.
  8. Print the confusion matrix.

## CODING:

install.packages("e1071") install.packages("caTools") install.packages("caret") library(e1071) library(caTools) library(caret) library(ggplot2)

split <- sample.split(iris, SplitRatio = 0.7) train\_cl <- subset(iris, split == "TRUE") test\_cl <- subset(iris, split == "FALSE") train\_scale <- scale(train\_cl[, 1:4]) test\_scale <- scale(test\_cl[, 1:4]) set.seed(120) # Setting Seed

classifier\_cl <- naiveBayes(Species ~ ., data = train\_cl) classifier\_cl

y\_pred <- predict(classifier\_cl, newdata = test\_cl) cm <- table(test\_cl$Species, y\_pred)

cm confusionMatrix(cm)

## OUTPUT:

Naive Bayes Classifier for Discrete Predictors Call:

naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities: Y

setosa versicolor virginica 0.3333333 0.3333333 0.3333333

Conditional probabilities: Sepal.Length

Y [,1] [,2]

setosa 4.986667 0.3812261

versicolor 5.950000 0.5556916

virginica 6.720000 0.5903827

Sepal.Width

Y [,1] [,2]

setosa 3.430000 0.3816028

versicolor 2.720000 0.3517444

virginica 2.983333 0.2730206

Petal.Length

Y [,1] [,2]

setosa 1.440000 0.1693802

versicolor 4.283333 0.5239659

virginica 5.670000 0.5784343

Petal.Width

Y [,1] [,2]

setosa 0.2366667 0.1129032

versicolor 1.3100000 0.2171127

virginica 2.0500000 0.2403302

y\_pred

setosa versicolor virginica setosa 20 0 0

|  |  |  |  |
| --- | --- | --- | --- |
| versicolor | 0 | 20 | 0 |
| virginica | 0 | 3 | 17 |

Confusion Matrix and Statistics y\_pred

setosa versicolor virginica

|  |  |  |  |
| --- | --- | --- | --- |
| setosa | 20 | 0 | 0 |
| versicolor | 0 | 20 | 0 |
| virginica | 0 | 3 | 17 |

Overall Statistics

Accuracy : 0.95

95% CI : (0.8608, 0.9896)

No Information Rate : 0.3833

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.925 Mcnemar's Test P-Value : NA Statistics by Class:

Class: setosa Class: versicolor Class: virginica

|  |  |  |  |
| --- | --- | --- | --- |
| Sensitivity | 1.0000 | 0.8696 | 1.0000 |
| Specificity | 1.0000 | 1.0000 | 0.9302 |
| Pos Pred Value | 1.0000 | 1.0000 | 0.8500 |
| Neg Pred Value | 1.0000 | 0.9250 | 1.0000 |
| Prevalence | 0.3333 | 0.3833 | 0.2833 |
| Detection Rate | 0.3333 | 0.3333 | 0.2833 |
| Detection Prevalence | 0.3333 | 0.3333 | 0.3333 |
| Balanced Accuracy | 1.0000 | 0.9348 | 0.9651 |

## RESULT:

Thus the above R program is executed and the output is verified successfully.

1. **DECISION TREE**

**AIM:**

To write a R program to illustrate the Decision Tree Classifier.

## ALGORITHM:

1. Install the required packages “rpart”,”rpart.plot”using install.packages() command.
2. Then load the packages using library() command.
3. Import the csv file to data variable using read.csv() method.
4. Classify the tree using rpart function and store it in tree variable.
5. Frame the data using data.frame () method and store in a object.
6. Predict the result and print the result.
7. Plot the tree.

install.packages(“rpart”) install.packages(“rpart.plot”) library(rpart) library(rpart.plot)

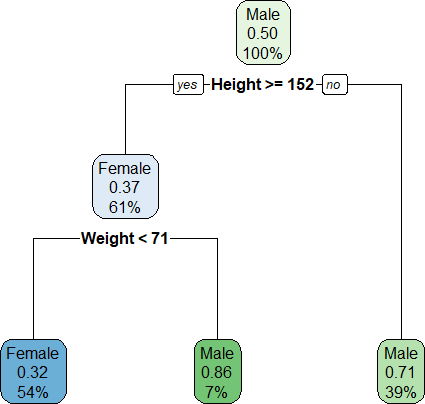
data = read.csv("D:\\R\\Gender.csv")

tree <- rpart(Gender ~ Height+Weight,data)

a <- data.frame(Height=c(149),Weight=c(75)) result <- predict(tree,a)

print(result) rpart.plot(tree)

Female Male 1 0.2926829 0.7073171



## RESULT:

Thus the above R program is executed and the output is verified successfully.

Ex.No: Date:

# TIME SERIES ANALYSIS-ARIMA MODEL

Page No:

## AIM:

To write a R program to illustrate Time Series Analysis using ARIMA

model.

## ALGORITHM:

* 1. Install the required packages “forecast” using install.packages() command.
  2. Then load the packages using library() command.
  3. Frame the data using data.frame() method and read the csv file using read.csv() method.
  4. Visualize the time series.
  5. Stationarize the series.
  6. Plot ACF/PACF chart and find optional parameters.
  7. Build the ARIMA model.
  8. Make predictions for next 12 months.

## CODING:

install.packages('forecast') library(forecast)

weather\_prod\_input <- as.data.frame( read.csv("weather.csv") ) weather\_prod <- ts(weather\_prod\_input[,3]) plot(weather\_prod, xlab = "Time (months)",

ylab = "weather between(1901-2017)") plot(diff(weather\_prod))

abline(a=0, b=0)

acf(diff(weather\_prod), xaxp = c(0, 48, 4), lag.max=48, main="") pacf(diff(weather\_prod), xaxp = c(0, 48, 4), lag.max=48, main="") arima\_1 <- arima (weather\_prod,

order=c(0,1,0),

seasonal = list(order=c(1,0,0),period=12))

arima\_1

acf(arima\_1$residuals, xaxp = c(0, 48, 4), lag.max=48, main="") pacf(arima\_1$residuals, xaxp = c(0, 48, 4), lag.max=48, main="") arima\_2 <- arima (weather\_prod,

order=c(0,1,1),

seasonal = list(order=c(1,0,0),period=12))

arima\_2

acf(arima\_2$residuals, xaxp = c(0, 48, 4), lag.max=48, main="") pacf(arima\_2$residuals, xaxp = c(0, 48,4), lag.max=48, main="") #Normality and Constant Variance

plot(arima\_2$residuals, ylab = "Residuals") abline(a=0, b=0)

hist(arima\_2$residuals, xlab="Residuals", xlim=c(-20,20))

qqnorm(arima\_2$residuals, main="") qqline(arima\_2$residuals) #Forecasting

#predict the next 12 months

?predict()

arima\_2.predict <- predict(arima\_2,n.ahead=12)

?matrix()

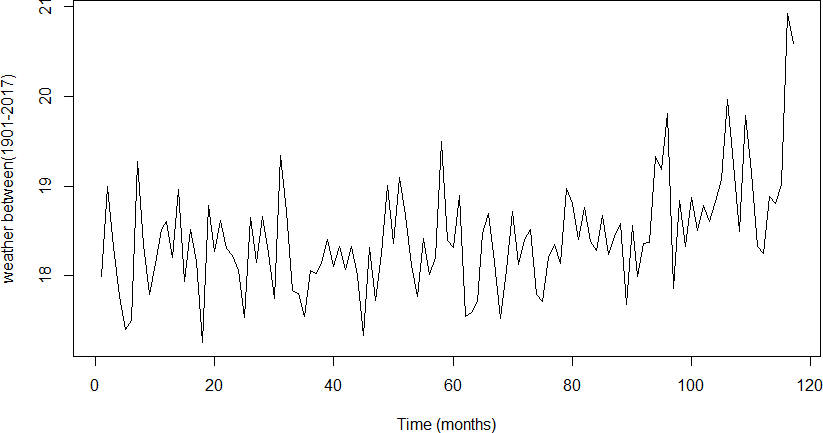
matrix(c(arima\_2.predict$pred-1.96\*arima\_2.predict$se, arima\_2.predict$pred, arima\_2.predict$pred+1.96\*arima\_2.predict$se), 12,1, dimnames=list( c(117:128) ,c("Pred")) )

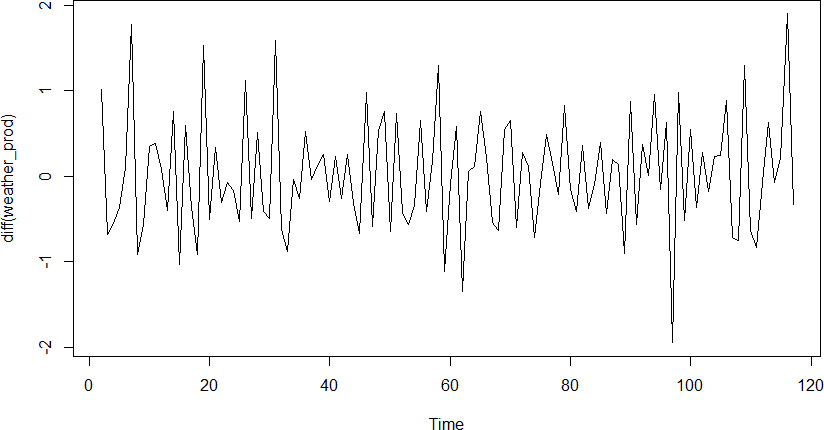
plot(weather\_prod) lines(arima\_2.predict$pred)

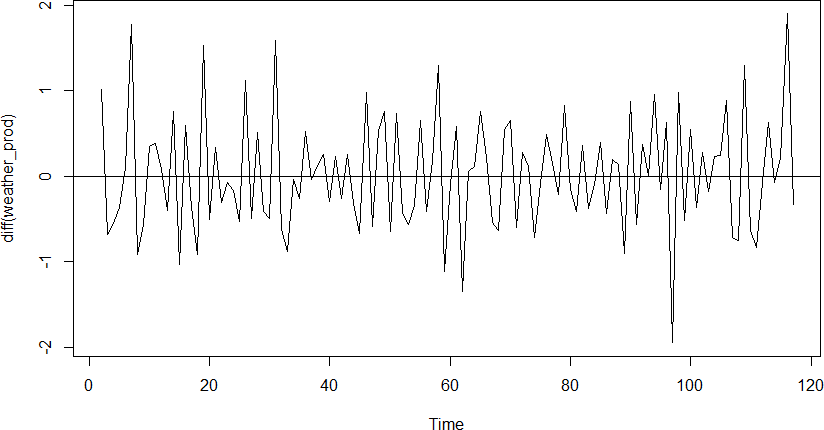
lines(arima\_2.predict$pred+1.96\*arima\_2.predict$se, col=4, lty=2) lines(arima\_2.predict$pred-1.96\*arima\_2.predict$se, col=4, lty=2)

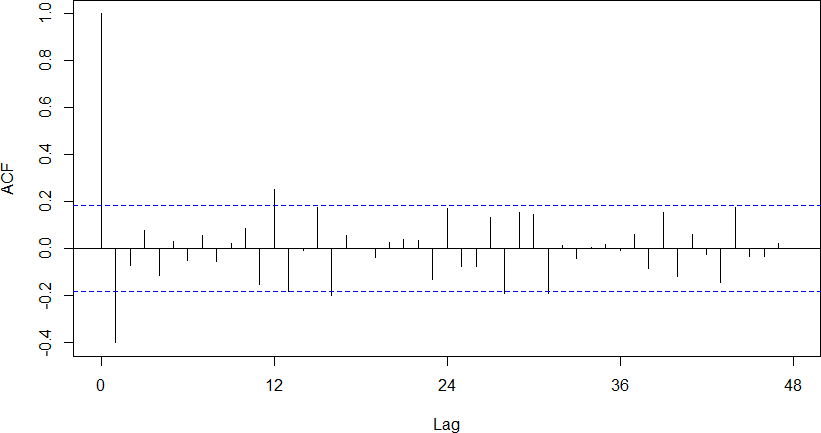
?arima()

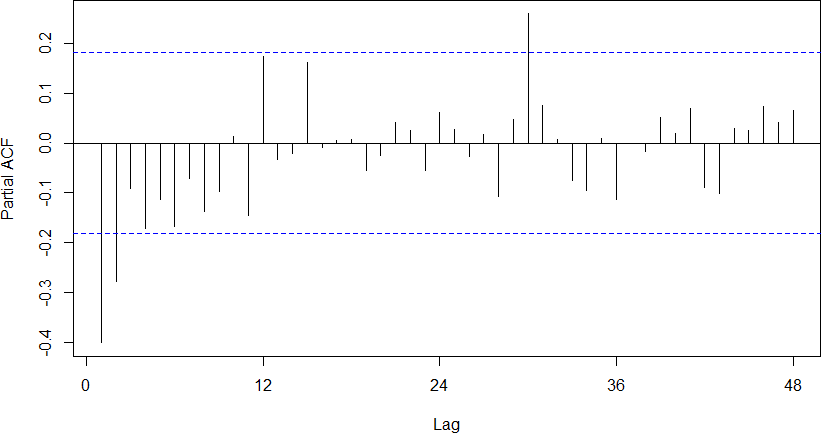
## OUTPUT:

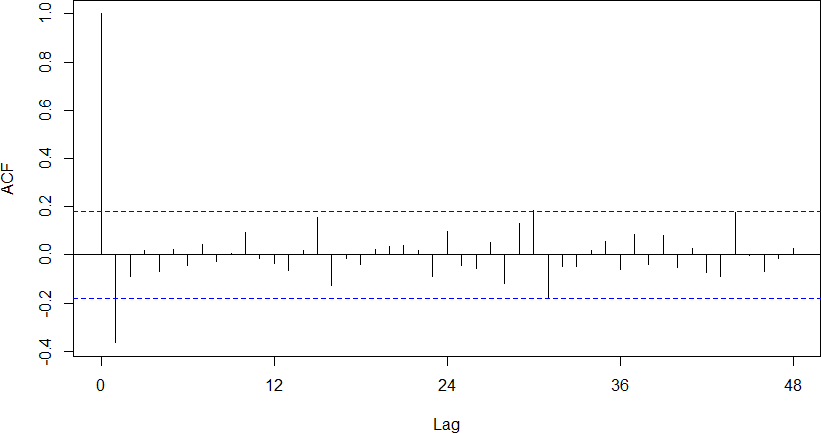


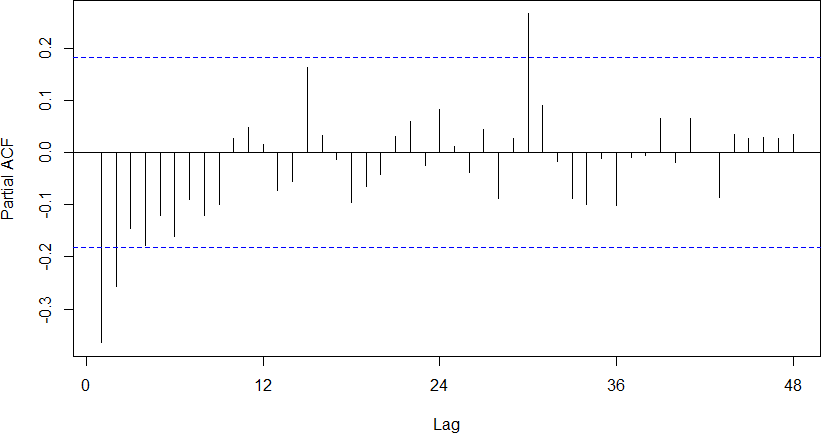


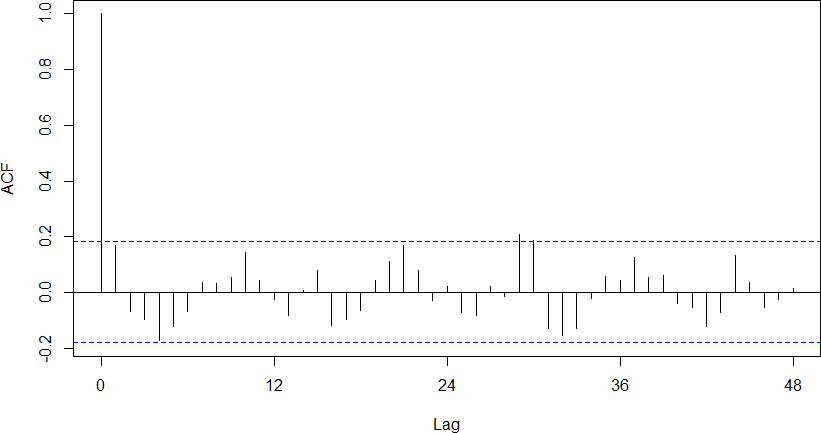


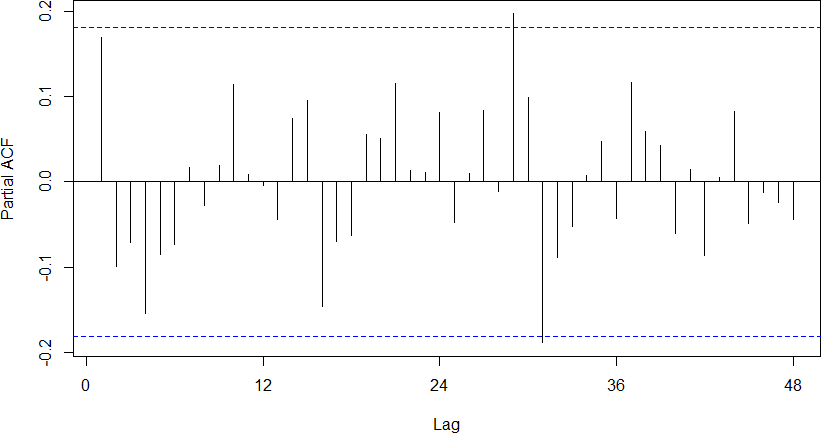


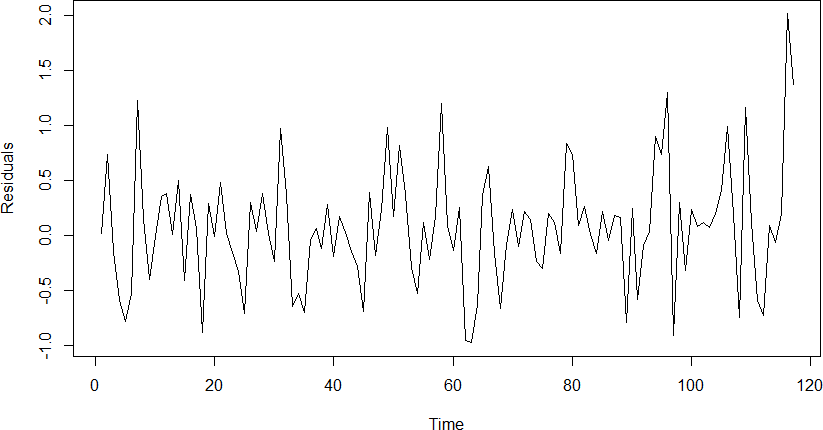


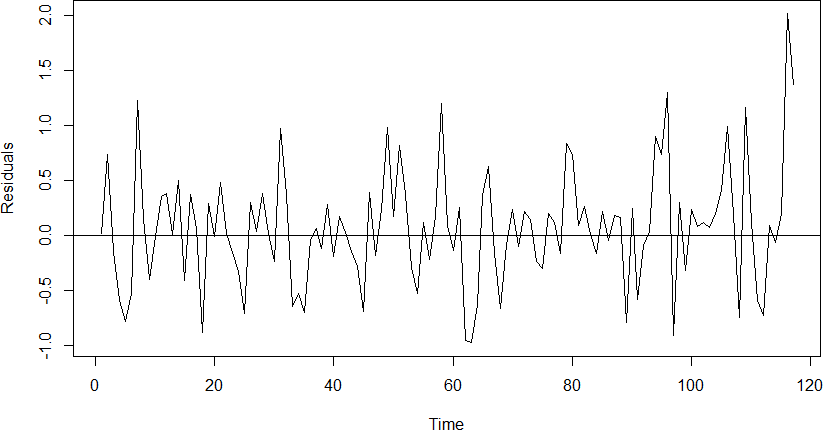


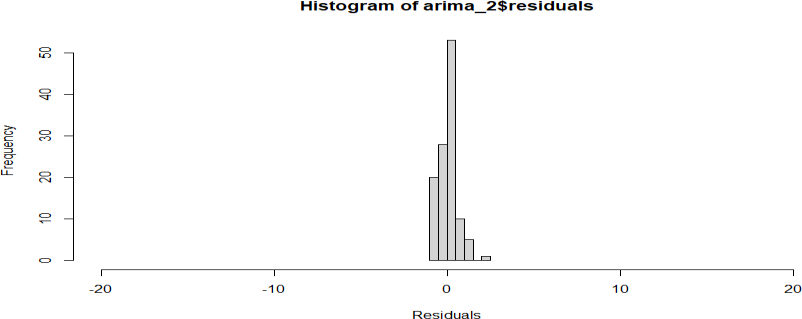


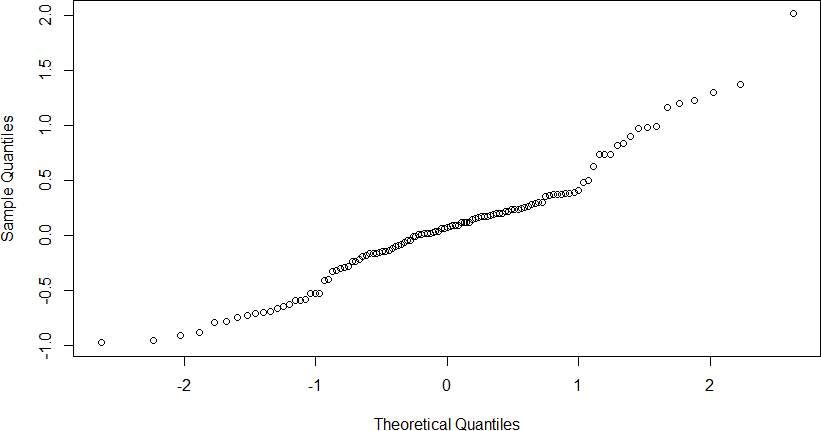


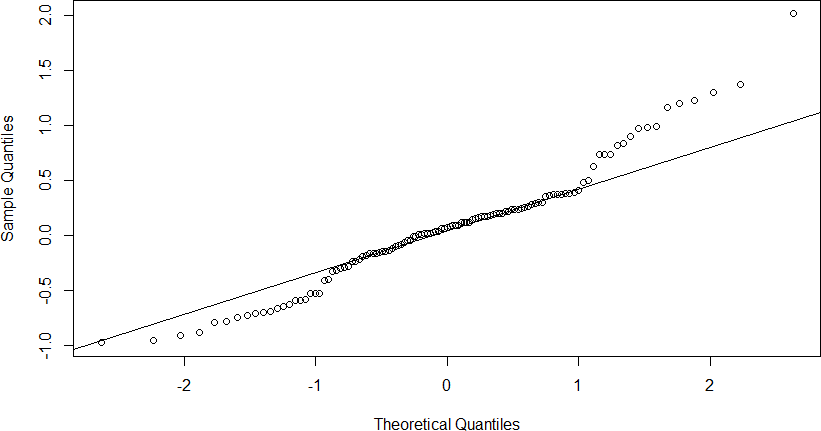


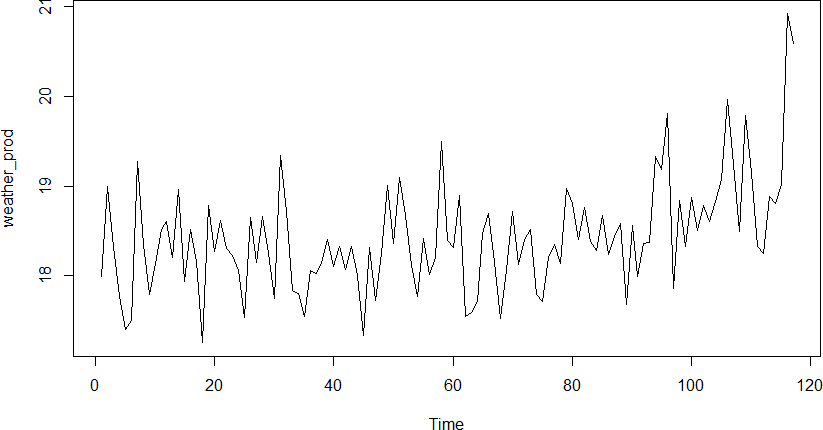


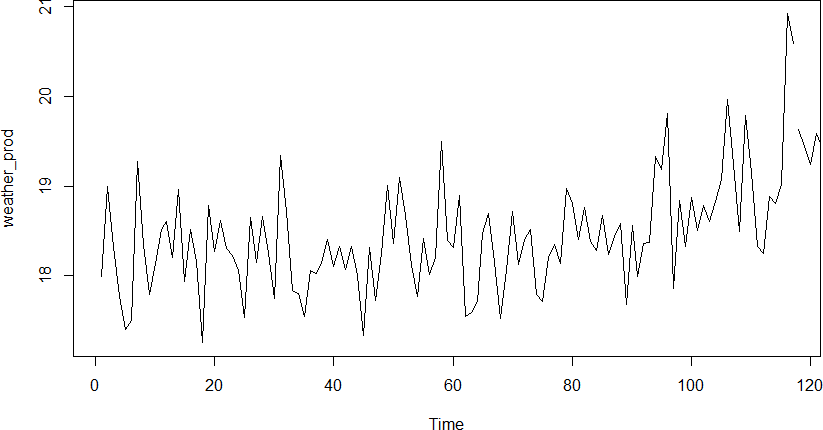


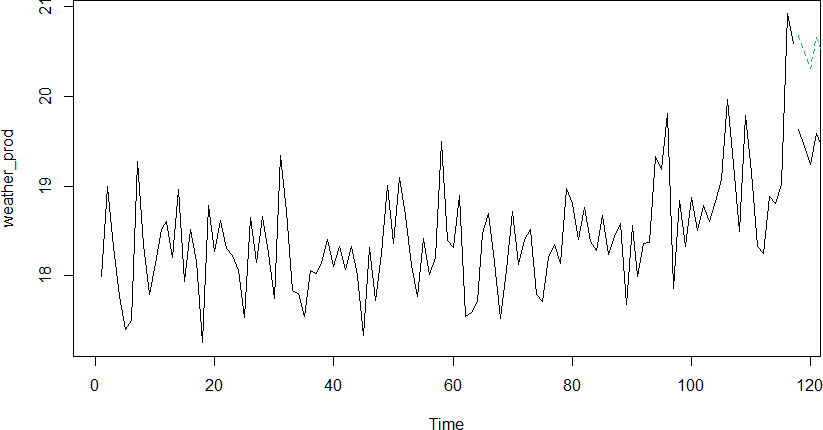


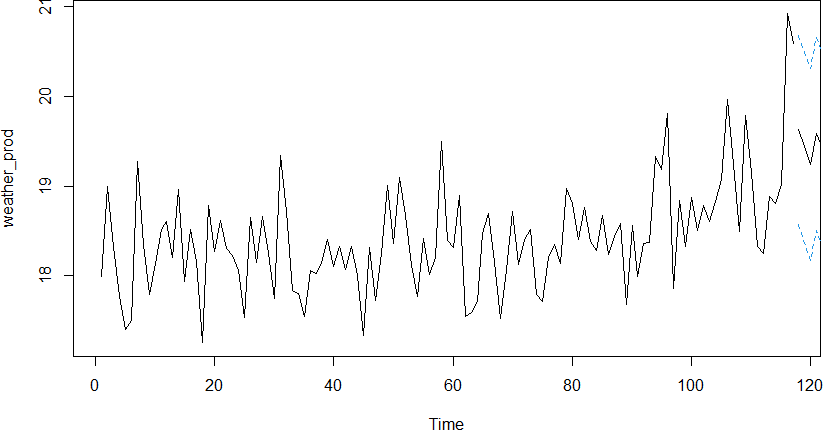












**RESULT:**

Thus the above R program is created and the output is executed successfully.

Ex.No: Date:

**TEXT ANALYSIS** Page No:

## AIM:

To write a R program to illustrate Text Analysis.

## ALGORITHM:

1. Find TF for words.
2. Find IDF for words.
3. Calculate TF-IDF and generate a matrix, TF-IDF=tf\*idf.
4. Install the required package “tm” using install.packages() command.
5. Then load the packages using library() command.
6. Create a text variable, give the input.
7. Create a corpus.
8. Tokenize the words.
9. Call the tfidf function.

## CODING:

#Term Frequency(TF) tf<-function(row){ return(row/sum(row))

}

#Inverse Document Frequency(IDF) idf<-function(col){

return(log10(length(col)/length(which(col>0))))

}

#Term Frequency - Inverse Document Frequency(TF-IDF) tfidf<-function(token.matrix){

tf<-t(apply(token.matrix,1,tf)) idf<-apply(token.matrix,2,idf) return(tf\*idf)

}

#Testing the Function install.packages("tm") library(tm)

text<-c("A dog and a cat","The dog is barking","The cat is on the wall") corp<-Corpus(VectorSource(text))

#Finding the tokens

doc.tokens<-as.matrix(DocumentTermMatrix(corp)) tfidf(doc.tokens)

## OUTPUT:

Terms

Docs and cat dog barking the wall

1 0.1590404 0.1590404 0.15904042 0.00000000 0.00000000 0.0000000

2 0.0000000 0.0000000 0.05869709 0.05869709 0.05869709 0.0000000

3 0.0000000 0.1192803 0.00000000 0.00000000 0.08804563 0.1192803

## RESULT:

Thus the above program was created and the output was executed successfully.

Ex.No: Date:

# SQL ESSENTIALS IN R

## AIM:

To write a R program to illustrate Set Operations, Joins, Grouping Extensions in SQL.

## ALGORITHM:

* 1. Install the required package “sqldf”,”readr” using install.packages() command.
  2. Then load the packages using library() command.
  3. Read the csv file using read.csv() method.
  4. Use SQL Set Operations UNION,UNION ALL,INTERSECT,EXCEPT.
  5. Use SQL Joins INNER JOIN,FULL OUTER JOIN,LEFT OUTER JOIN,RIGHT OUTER JOIN.
  6. Use SQL Group Extensions using GROUP BY method and HAVING,SUM clause.

## CODING:

install.packages("sqldf") install.packages("readr") library(sqldf) library(readr)

Book1<-read\_csv("Book1.csv") Book2<-read\_csv("Book2.csv") Book3<-read\_csv("Book3.csv") View(Book1)

View(Book2) View(Book3) **#Set\_Operations**

sqldf("SELECT \* FROM Book1 UNION SELECT \* FROM Book2") sqldf("SELECT \* FROM Book1 UNION ALL SELECT \* FROM Book2") sqldf("SELECT \* FROM Book1 INTERSECT SELECT \* FROM Book2") sqldf("SELECT \* FROM Book1 EXCEPT SELECT \* FROM Book2")

**#Join\_Operations**

sqldf("SELECT Book1.Name,Book3.College FROM Book1 INNER JOIN Book3 ON Book1.Id=Book3.Id")

sqldf("SELECT Book1.Name,Book3.College FROM Book1 FULL OUTER JOIN Book3 ON Book1.Id=Book3.Id")

sqldf("SELECT Book1.Name,Book3.College FROM Book1 LEFT OUTER JOIN Book3 ON Book1.Id=Book3.Id")

sqldf("SELECT Book1.Name,Book3.College FROM Book1 RIGHT OUTER JOIN Book3 ON Book1.Id=Book3.Id")

**#Grouping\_Extensions** Book4<- read\_csv("Book4.csv") View(Book4)

sqldf("SELECT Department,SUM(Salary) as Salary FROM Book4 GROUP BY Department")

sqldf("SELECT Department,Category,SUM(Salary) as Salary FROM Book4 GROUP BY Department, Category")

sqldf("SELECT Department,SUM(Salary) as Salary FROM Book4 GROUP BY Department HAVING SUM(Salary) = 25000")

sqldf("SELECT Department,Category,SUM(Salary) as Salary FROM Book4 GROUP BY Department, Category HAVING SUM(salary) = 50000")

## OUTPUT:

* sqldf("SELECT \* FROM Book1 UNION SELECT \* FROM Book2")

ID Name Course Percentage

|  |  |  |  |
| --- | --- | --- | --- |
| 1 1 | Ram | MCA | 90 |
| 2 2 | Dharsh | MBA | 85 |
| 3 2 | Raju | MBA | 87 |
| 4 3 | Priya | ME | 78 |
| 5 3 | Vijay | ME | 76 |
| 6 4 | Joseph | MSC | 85 |
| 7 5 | Akash | MA | 83 |
| 8 5 | Anu | MA | 90 |
| > |  |  |  |

* sqldf("SELECT \* FROM Book1 UNION ALL SELECT \* FROM Book2")

ID Name Course Percentage

|  |  |  |  |
| --- | --- | --- | --- |
| 1 1 | Ram | MCA | 90 |
| 2 2 | Raju | MBA | 87 |
| 3 3 | Priya | ME | 78 |
| 4 4 | Joseph | MSC | 85 |
| 5 5 | Anu | MA | 90 |
| 6 1 | Ram | MCA | 90 |
| 7 2 Dharsh | | MBA | 85 |
| 8 3 Vijay | | ME | 76 |
| 9 4 Joseph | | MSC | 85 |
| 10 5 Akash | | MA | 83 |
| > | |  |  |

* sqldf("SELECT \* FROM Book1 INTERSECT SELECT \* FROM Book2")

ID Name Course Percentage

|  |  |  |  |
| --- | --- | --- | --- |
| 1 1 | Ram | MCA | 90 |
| 2 4 | Joseph | MSC | 85 |
| > |  |  |  |

* sqldf("SELECT \* FROM Book1 EXCEPT SELECT \* FROM Book2")

ID Name Course Percentage

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 Raju | MBA | 87 |
| 2 | 3 Priya | ME | 78 |
| 3 | 5 Anu | MA | 90 |
| > |  |  |  |

* #Join\_Operations
* sqldf("SELECT Book1.Name,Book3.College FROM Book1 INNER JOIN Book3 ON Book1.Id=Book3.Id")

Name College

1. Ram MSU
2. Raju SAC 3 Joseph SXC

* > sqldf("SELECT Book1.Name,Book3.College FROM Book1 FULL OUTER JOIN Book3 ON Book1.Id=Book3.Id")

Name College

1. Ram MSU
2. Raju SAC 3 Priya <NA> 4 Joseph SXC
3. Anu <NA>
4. <NA> SJC
5. <NA> FXEC

>

* sqldf("SELECT Book1.Name,Book3.College FROM Book1 LEFT OUTER JOIN Book3 ON Book1.Id=Book3.Id")

Name College

1. Ram MSU
2. Raju SAC
3. Priya <NA>
4. Joseph SXC
5. Anu <NA>

>

* sqldf("SELECT Book1.Name,Book3.College FROM Book1 RIGHT OUTER JOIN Book3 ON Book1.Id=Book3.Id")

Name College

1. Ram MSU
2. Raju SAC 3 Joseph SXC
3. <NA> SJC
4. <NA> FXEC

>

#Grouping\_Extensions

* sqldf("SELECT Department,SUM(Salary) as Salary FROM Book4 GROUP BY Department")

Department Salary 1 IT 95000

2 Sales 45000

>

* sqldf("SELECT Department,Category,SUM(Salary) as Salary FROM Book4 GROUP BY Department, Category")

Department Category Salary

|  |  |  |
| --- | --- | --- |
| 1 | IT | A 95000 |
| 2 | Sales | B 20000 |
| 3 | Sales | C 25000 |
| > |  |  |

* sqldf("SELECT Department,SUM(Salary) as Salary FROM Book4 GROUP BY Department HAVING SUM(Salary) = 95000")

Department Salary

1 IT 95000

>

* sqldf("SELECT Department,Category,SUM(Salary) as Salary FROM Book4 GROUP BY Department, Category HAVING SUM(salary) = 25000")

Department Category Salary

1 Sales C 25000

>

## RESULT:

Thus the above program is created and the output is verified successfully.

Ex No: Date:

**SENTIMENTAL ANALYSIS** Page No:

## AIM:

To write a program to illustrate Sentimental Analysis.

## ALGORITHM:

1. Install the required packages “tm”,”wordcloud”,”syuzhet” using install.packages() command.
2. Then load the packages using library() command.
3. Read the csv file using read.csv() method.
4. Wordcloud() method is used to frequency table of words and plot the word frequencies.
5. Print(review\_score) method is used to find out how many times words are occurs.
6. In Barplot the review\_score values are plotted.

## CODING:

install.packages("tm") install.packages("wordcloud") install.packages("syuzhet") library(tm) library(wordcloud) library(syuzhet)

reviews<-read.csv(file.choose(),header=1) str(reviews)

corpus<-iconv(reviews$text)

corpus<-Corpus(VectorSource(corpus)) inspect(corpus[1:10])

corpus <- tm\_map(corpus,tolower)

corpus <- tm\_map(corpus,removePunctuation) corpus <- tm\_map(corpus,removeNumbers)

corpus <- tm\_map(corpus,removeWords,stopwords("english")) #corpus <- tm\_map(corpus,removeWords,c("book","read","life")) corpus <- tm\_map(corpus,stripWhitespace)

inspect(corpus[1:10]) reviews\_final <- corpus

tdm <- TermDocumentMatrix(reviews\_final) tdm <- as.matrix(tdm)

tdm[1:10,1:10]

w <- sort(rowSums(tdm),decreasing = T) set.seed(2000)

wordcloud(words = names(w), freq = w, max.words=50, random.order = T,

min.freq=5, colors=brewer.pal(25,"Dark2"), scale = c(5,0.5))

sentiment\_data <- iconv(reviews$text) s <- get\_nrc\_sentiment(sentiment\_data) s[1:10,]

s$score <- s$positive - s$negative s[1:10,]

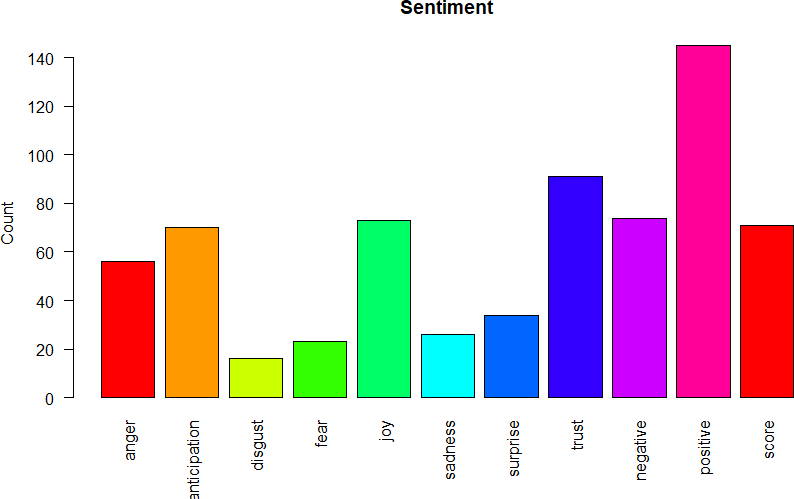
write.csv(x=s,file="reviews.csv") review\_score <- colSums(s[,]) print(review\_score) barplot(colSums(s),

las =2, col=rainbow(10), ylab='Count', main='Sentiment')

## OUTPUT:



|  |  |  |  |
| --- | --- | --- | --- |
| anger anticipation | disgust | fear | joy sadness |
| 56 70 | 16 | 23 | 73 26 |
| surprise trust | negative | positive | score |
| 34 91 | 74 | 145 | 71 |



**RESULT:**

Thus the above program is created and the output is verified successfully.

Ex No:

Date: **ADVANCED SQL COMMANDS**

Page No:

## AIM:

To write a SQL Queries using windows function.

## ALGORITHM:

1. Create a tables Employees and Students then, insert the values.
2. Use count() command to find out the number of times column values occurs in selected column.
3. Use avg(),min(),max() commands for find out the average value,minimum value,maximum value for selected column.
4. Use where command to find out the values for specified column.
5. Use sum() command for find out the total for the selected column.
6. Use rank(),dense rank,row number() commands to find out the rank for the list of students in Students table.
7. Using with cte as() command we find out the maximum,average,minimum values list in all the rows in the table.

## CODING:

create table employees1(id int,name varchar(30),salary int,age int,location varchar(30),mobile int);

insert into employees1 values(101,'Suka',50000,22,'Palayamkottai',8667897130); insert into employees1 values(102,'Saran',40000,25,'Tuticorin',9807654321); insert into employees1 values(103,'Priya',30000,29,'Tirunelveli',8765432190); insert into employees1 values(104,'Hari',20000,24,'Chennai',9678543210);

insert into employees1 values(105,'Pavi',36000,23,'Tenkasi',9390876543); insert into employees1 values(106,'Ram',55000,25,'Kanyakumari',9345678902); select \* from employees1;

select count(age) from employees1; select AVG(salary) from employees1; select SUM(salary) from employees1;

select salary from employees1 where id=104; select min(salary) from employees1;

select max(salary) from employees1;

create table students11(id int,name varchar(20),subject\_name varchar(20),Marks\_scored int,Total\_marks int);

insert into students11 values(1,'Joseph','DIP',99,100); insert into students11 values(2,'Ravi','IOT',90,100); insert into students11 values(3,'Jack','DS',87,100); insert into students11 values(4,'Steve','SC',85,100);

insert into students11 values(5,'Karthi','Python',95,100); insert into students11 values(6,'Devi','JAVA',80,100); select \* from students11;

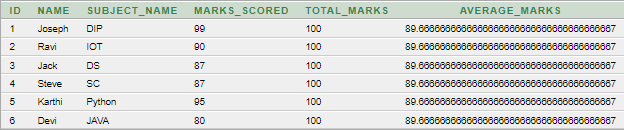
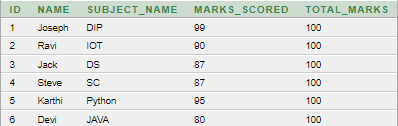
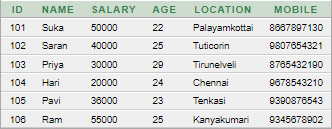
select Marks\_scored,id,name,rank() over(order by Marks\_scored desc),dense\_rank() over(order by Marks\_scored desc),row\_number() over(order by Marks\_scored desc)from students11;

with cte as(select AVG(Marks\_scored) as average\_marks from students11) select students11.\*,cte.average\_marks from students11,cte

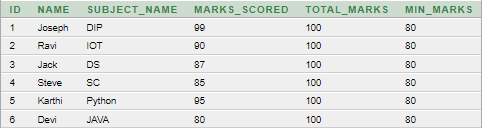
with cte as(select MIN(Marks\_scored) as Min\_marks from students11) select students11.\*,cte.Min\_marks from students11,cte

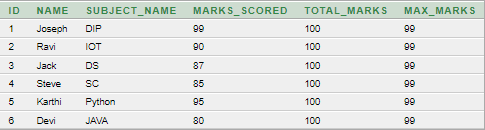
with cte as(select MAX(Marks\_scored) as Max\_marks from students11) select students11.\*,cte.Max\_marks from students11,cte

## OUTPUT:









**RESULT:**

Thus the above SQL Queries are created and the output is verified successfully.