**SRI RAMAKRISHNA MISSION VIDYALAYA COLLEGE OF ARTS AND SCIENCE**

*(An Autonomous Institution Affiliated to Bharathiar University,*

*Re-Accredited by NAAC with A grade)*

**COIMBATORE-641 020**

**DEPARTMENT OF MASTER OF APPLICATION**

****

**RECORD NOTE**

**Core Practical –I:**Data science using R

**Subject Code:** 20PCA3CP8

This is certified that this is a bonafide record of work done by

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

**Staff-In-Charge Head of the Department**

Submitted for the Practical Examination held at Sri Ramakrishna Mission Vidyalaya College of Arts and Science on \_\_\_\_\_\_\_\_\_\_\_\_\_\_ during the year 2021.

**Examiners**

**Internal External**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **SIGN** |
| **1** | **Merge two vector in single vector** |  |
| **2** | **Create two data CSV having name, age, DoJ, DoR, empcode,salary** |  |
| 1. How will you merge these two tables to create a single table |  |
| 1. Print those who are receiving salary greater than 5000 |  |
| 1. Print those who are receiving salary inbetween 1000 and 10000 |  |
| 1. Print those employees whose age is greater than 50 |  |
| 1. Print those employees who have joined the company in less than one year |  |
| 3 | **Mean, Median, Mode and Standard Deviation** |  |
| 4 | **Correlation** |  |
| 5 | **Linear Regression with mtcars Dataset** |  |
| 6 | **Logistic Regression** |  |
| 7 | **Multiple Linear Regression** |  |
| 8 | **Poisson Regression** |  |
| 9 | **Write data throw XLS file and find the Mean, Median, and Mode** |  |
| 10 | **Charts and Graphs** |  |
| 1. Bar Chart |
| B) Box Plot |
| C) Pie Chart |
| D) Histogram |
| E) Line Plot |
| 11 | **Distributions** |  |
| 1. Binomial Distribution |
| 1. Poisson Distribution |
| 1. Continuous Uniform Distribution |
| 1. Exponential Distribution |
| 1. Normal Distribution |
| 1. Chi-squared Distribution |
| 12 | **Hierarchical Clustering** |  |
| 13 | **K – Means Clustering** |  |
| 14 | **Association Rule** |  |
| 15 | **Case study on Guna’s Theory** |  |
| 16 | **Analysis of Variance (ANOVA)** |  |
| 17 | **Wilcox on signed-rank test** |  |
| 18 | **Time Series Analysis** |  |
| 1. Moving Average |  |
| 2.Auto Regression |  |
| 3. ARIMA Model |  |
| 4. Time series analysis using Stock Data |  |
| 5. Time series analysis using Weather Temperature data |  |
| 19 | **Decision Trees on any dataset** |  |
| 20 | **Random Forest using Titatnic dataset** |  |
| 21 | **Survival analysis** |  |
| 22 | **Mathematical functions using Numpy** |  |
| 23 | **Data analytics using Pandas** |  |
| 24 | **Visual representations using Matplotlib** |  |

**1. Merge two vector in single vector**

**Aim:**

Write a program to accept two vectors of string. Merge two vectors of strings into single vector. Convert each string to reverse using conditional looping.

**Algorithm:**

Step 1 : Open the R Studio

Step 2 : Type five strings and store it as vector variable a and another set as b

Step 3 : Merge the a and b vector in vector c

Step 4: Write a function with reversing logic of string

Step 5 : By using conditional(for) loop print the reversed string one by one

Step 6 : Save and run the program.

Step 7: Verify the output in output area.

**Program:**

a<-c("Berlin","Moscow","Tokyo","Nairobi","Denver");

b<-c("Dubai","Rio","Oslo","Helsinki","lisbon");

c<-c(a,b);

reverse\_chars<- function(string)

{

 string\_split = strsplit(string, split = "")

  rev\_order = nchar(string):1

  reversed\_chars = string\_split[[1]][rev\_order]

  paste(reversed\_chars, collapse = "")

}

for (val in c)

{

  print(reverse\_chars(val))

}

**Output:**

|  |
| --- |
| [1] "nilreB"  [1] "wocsoM"  [1] "oykoT"  [1] "iboriaN"  [1] "revneD"  [1] "iabuD"  [1] "oiR"  [1] "olsO"  [1] "iknisleH"  [1] "nobsil" |
|  |
| |  | | --- | |  | |

**2.Create two data tables having name, age, DoJ, DoR, empcode,salary**

**Aim:**

Write a program to create two data tables having name, age, DoJ, DoR, empcode, salary. First 10 employees belong to first table “employee-table1” and next 10 employees belongs to second table “employee-table2”. If there is any redundant row, it should be removed. Create program by referring the below questions

a) How will you merge these two tables to create a single table that does not have any redundant column in it?

b) Print those who are receiving salary greater than 5000.

c) Print those who are receiving salary in between 1000 and 10000

d) Print those employees whose age is greater than 50

e) Print those employees who have joined the company in less than one year

**Algorithm:**

Step 1: Open the R studio

Step 2: Create 2 tables using data.table with 10 records in each table containing given columns name, age, doj, dor, empcode and salary.

Step 3: Print both the tables and check the data displayed was correct.

Step 4: Merge the table without redundant column present.

Step 5: Get the salary greater than 5000 and print from merged table

Step 6: Get the salary in between 1000 and 10000

inbetween = MergedTable[salary > 1000& salary<10000]

Step 7: Get the employees age greater than 50 and print from merged table.

Step 8: Get the employees who have joined company in less than one year

A<-(Sys.Date()>as.Date("2020-10-2"))

Step 9: Save the program and run the conditions one by one to get the output.

**Program:**

install.packages(data.table)

library(data.table)

#creating Table one with ten records

Table1 <- data.table(

name<-c("Roshan","varun","Bagavathi","Keerthi","hari","Mouli","bala","Keerthi","hasan","vijay"),

age<-c(21,21,22,21,24,25,27,28,26,25),

DOJ<-c("2001-20-01","2021-12-21","2021-21-10","2010-15-10","2002-16-01","2010-28-02","2020-28-02","2000-15-10","2000-29-02","2010-24-12"),

DOR<-c("2051-06-09","2060-12-12","2024-24-10","2026-31-02","2027-27-10","2080-29-20","3000-30-03","2090-10-02","2020-29-10","2040-25-10"),

empcode<-c(1,2,3,4,5,6,7,8,9,10),

salary<-c(1000,2000,10000,4000,10000,1500,12000,1100,1900,2000))

print(Table1)

#creating Table two with ten records

Table2 <- data.table(

name<-c("ragavan","shankar","Lokesh","Keerthi","joseph","Pranesh","Mani","hasan","Chandru","aravindh"),

age<-c(25,16,21,23,24,25,26,27,29,30),

DOJ<-c("2001-20-01","202112-12-","2021-12-10","2010-15-10","2002-16-01","2010-02-28","2020-10-20","2000-10-12","2000-02-10","2010-12-10"),

DOR<-c("06-09-2051","12-12-2060","24-10-2024","31-02-2026","27-10-2027","29-20-2080","30-03-3000","10-02-2090","29-10-2020","25-10-2040"),

empcode<-c(11,12,13,14,15,15,17,18,19,20),

salary<-c(1000,2000,1000,4000,1000,1500,12000,1100,1900,2000))

print(Table2)

#filtering duplicate rows

Table1<-unique(Table1,by ="V2")

Table2<-unique(Table2,by ="V2")

#Print salary above 5000

con1 = Table2[salary > 5000]

print(con1)

# Print those who are receiving salary in between 1000 and 10000

con2 = Table2[salary > 1000& salary<10000]

print(con2)

# Print those employees whose age is greater than 50

con3 = Table2[age>50]

print(con3)

#Print those employees who have joined the company in less than one year

Oneyear<-Sys.Date()-365

print(Oneyear)

con3 = Table1[as.Date(DOJ)<as.Date("2020-11-13")]

print(con3)

A<-(Sys.Date()>as.Date("2020-10-2"))

print(a)

**Output:**

**3. Mean, Median, Mode and Standard Deviation**

**Aim :**

To Find the mean, median, mode and Standard Deviation following list of values:

13, 18, 13, 14, 13, 16, 14, 21, 13

**Algorithm**

Step 1: Start the R language

Step 2: To Formula for find Mean is

    1-∑n
x¯= n    xi
      i=1


Step 3: Assign the values to x.

Step4: Use following function to find means **mean(x)**

Step 5: We apply the median function to compute the median value **median(x)**

Step 6: To finds mode use **names(sort(-table(x)))[1]**

Step 7: To find Standard deviation **sqrt(var(x))**

Step 8: Print the result

Step 9: Stop the program

**Program:**

x <- c(12,7,3,4.2,18,2,54,-21,8,-5)

result.mean<- mean(x)

print(result.mean)

median.result<- median(x)

print(median.result)

result <- getmode(x)

print(result)

**Output:**

|  |
| --- |
| > x <- c(12,7,3,4.2,18,2,54,-21,8,-5)  >result.mean<- mean(x)  > print(result.mean)  [1] 8.22  >median.result<- median(x)  > print(median.result)  [1] 5.6  > result <- getmode(x)  [1] 12.0 7.0 3.0 4.2 18.0 2.0 54.0 -21.0 8.0 -5.0  [1] 1 1 1 1 1 1 1 1 1 1  > print(result)  [1] 12 |
|  |
| |  | | --- | |  | |

**4. correlation**

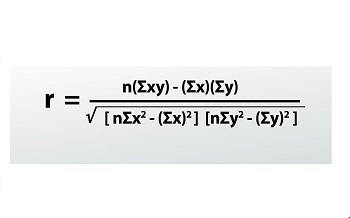
**Aim:**

Write a program of a Correlation. x = 11, 35, 23, 12, 16, 18, 28, 32, 46, 39

y = 46, 35, 33, 14, 18, 39, 43, 11, 10, 23

**Algorithm:**

Step 1: To open the R-Studio software in the system. And use the Correlation formula.



Step 2: Enter the x and y values, then multiple the sum values **sum (x\*y)**

Step 3: Then enter the sum of **n(x\*y), a(x), b(y)** values, Using the formula and the **sqrt** values of **(x1 & y1),** then print the (x1 and y1).

Step 4: Enter the **r** value **r<-(10\*n-a\*b)/(x1\*y1)** and **print(r)**

Step 5: Then finally execute and Save the program.

**Program:**

x<-c(11, 35, 23, 12, 16, 18, 28, 32, 46, 39)

y<-c(46, 35, 33, 14, 18, 39, 43, 11, 10, 23)

sum (x\*y)

n<-sum(x\*y)

a<-sum(x)

b<-sum(y)

x1<-sqrt((10\*sum(x\*x))-a\*a)

y1<-sqrt((10\*sum(y\*y))-b\*b)

print(x1)

print(y1)

r<-(10\*n-a\*b)/(x1\*y1)

print(r)

**Output:**

>x<-c(11, 35, 23, 12, 16, 18, 28, 32, 46, 39)

>y<-c(46, 35, 33, 14, 18, 39, 43, 11, 10, 23)

>sum (x\*y)

[1] 6561

>n<-sum(x\*y)

>a<-sum(x)

>b<-sum(y)

>x1<-sqrt((10\*sum(x\*x))-a\*a)

>y1<-sqrt((10\*sum(y\*y))-b\*b)

>print(x1)

[1] 113.3137

>print(y1)

[1] 129.2904

>r<-(10\*n-a\*b)/(x1\*y1)

>print(r)

[1] -0.3487966

**5. Linear Regression with mtcars Dataset**

**Aim :**

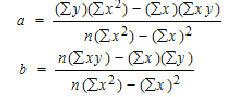
Write a program for Linear Regression with mtcars Dataset.

**Algorithm**

Step 1: Start the R language

Step 2: Get mtcars dataset details

Step 3: To Formula for find Linear Regressionis



Step4: Use following function to Linear Regression**lm(dataset)**

Step 5: To draw plot for Linear Regression

**Program:**

input <- mtcars[,c("mpg","disp")]

print(head(input))

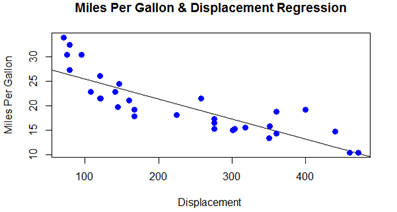
a<-lm(mpg~disp,data=input)

print(a)

plot(input$disp,input$mpg,col = "blue",main = "Miles Per Gallon & Displacement Regression",

abline(lm(input$mpg~input$disp)),cex = 1.3,pch = 16,xlab = "Displacement ",ylab = "Miles Per Gallon")

**Output:**



**6. Logistic Regression**

**Aim :**

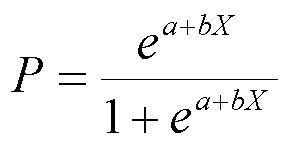
Write a program for Logistic Regression with mtcars Dataset.

**Algorithm**

Step 1: Start the R language

Step 2: Get mtcars dataset details

Step 3: To Formula for find Logistic Regressionis



Step4: Use following function to Logistic Regression**glm(dataset)**

Step 5: To print the summery of Logistic Regression.

**Program**

input <- mtcars[,c("am","cyl","hp","wt")]

print(head(input))

input <- mtcars[,c("am","cyl","hp","wt")]

am.data = glm(formula = am ~ cyl + hp + wt, data = input, family = binomial)

print(summary(am.data))

**Output:**

Deviance Residuals:

  Min        1Q Median    3Q       Max

-2.17272  -0.14907  -0.01464   0.14116   1.27641

Coefficients:

         Estimate Std. Error z value Pr(>|z|)

(Intercept) 19.70288 8.11637   2.428   0.0152 \*

cyl      0.48760 1.07162   0.455   0.6491

hp       0.03259 0.01886   1.728   0.0840 .

wt      -9.14947 4.15332  -2.203   0.0276 \*

---

Signif. codes:  0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 43.2297  on 31  degrees of freedom

Residual deviance:  9.8415  on 28  degrees of freedom

AIC: 17.841

Number of Fisher Scoring iterations: 8

**7. Multiple Linear Regression**

**Aim :**

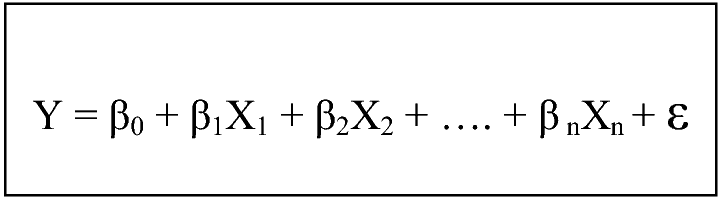
Write a program for MultipleLinear Regression with mtcars Dataset.

**Algorithm**

Step 1: Start the R language

Step 2: Get mtcars dataset details

Step 3: To Formula for find MultipleLinear Regressionis



Step4: Use following function to MultipleLinear Regression**lm(dataset),coef(model)**

Step 5: To draw plot for MultipleLinear Regression

**Program:**

input <- mtcars[,c("mpg","disp","hp","wt")]

print(head(input))

model <- lm(mpg~disp+hp+wt, data = input)

print(model)

cat("# # # # The Coefficient Values # # # ","\n")

a <- coef(model)[1]

print(a)

Xdisp<- coef(model)[2]

Xhp<- coef(model)[3]

Xwt<- coef(model)[4]

print(Xdisp)

print(Xhp)

print(Xwt)

x1 = 221

x2 = 102

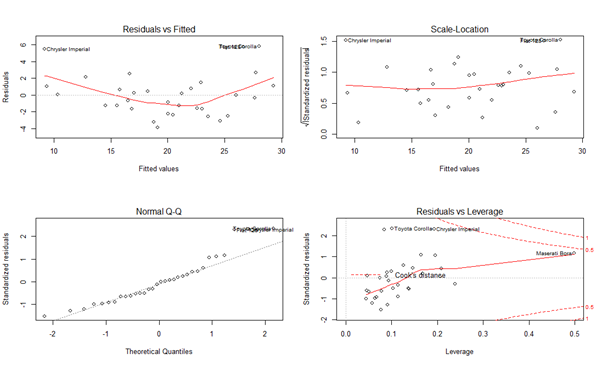
x3 = 2.91

Y = 37.15+(-0.000937)\*x1+(-0.0311)\*x2+(-3.8008)\*x3

layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page

plot(model)

**Output:**

**tttttttttttttttttttttttttrrrrrrrrr**

**8. Poisson Regression**

**Aim:**

To implement the concept of Poisson regression using R Language.

**Algorithm:**

Step1:Start the R studio

Step2: Poisson Regression involves regression models in which the response variable is in the form of counts and not fractional numbers.

Step3:The general mathematical equation for Poisson regression is

log(y) = a + b1x1 + b2x2 + bnxn.....

Step4:There are four types of Poisson functions.

There are dpois,qpois,rpois,ppois.

Step5:Create a Poisson regression model is the glm() function.

Step6:Create a Regressional model

Step7: The wool "type" and "tension" are taken as predictor variables in a Poisson regressions.

Step8: Print the Result

Step9:Stop the process.

**Program:**

input <- warpbreaks

print(head(input))

output <-glm(formula = breaks ~ wool+tension, data = warpbreaks,

family = poisson)

print(output)

**Output:**

|  |
| --- |
| > print(output)  Call: glm(formula = breaks ~ wool + tension, family = poisson, data = warpbreaks)  Coefficients:  (Intercept) woolBtensionMtensionH  3.6920 -0.2060 -0.3213 -0.5185  Degrees of Freedom: 53 Total (i.e. Null); 50 Residual  Null Deviance: 297.4  Residual Deviance: 210.4 AIC: 493.1 |
|  |
| |  | | --- | |  | |

**9 .Writethrow XLS file and find the Mean, Median, and Mode**

**Aim:**

To Write a program to input data through XLS file and find the Mean, Median, and Mode for salary and also write a program to connect web data into a CSV file. [Input data-> Name, Age, gender, Address, DOB and Salary].

**Algorithm:**

Step1:Start the R studio

Step2:Read the Xlxs file

Step3:Calculate Mean, Median and Mode for Salary and print the Values of Mean, Median and Mode.

Step4:Read the Library files in Xml,RCurl,stringr And plyr

Step5:Store the file names as a list and Gather the html links present in the webpage.

Step4:print the file names.

Step5:Create a function to download the files by passing the URL and filename list.

Step6:Download the CSV file.

Step6:Print the result

Step7:Stop the process.

**Program:**

library("readxl")  
my\_data<- read\_excel("/input.xlsx")  
print(my\_data)  
**4444444444444444444444444444444444444444444444444444444444444444444444r  
Output:**

# A tibble: 5 × 6  
  Name               Age Gender Address    DOB                 Salary  
  <chr>            <dbl><chr>  <chr>      <dttm>               <dbl>  
1 Steve Rogers        28 Male   Queens     1995-08-17 00:00:00  10000  
2 Tony Stark          34 Male   Los Angels 1977-08-29 00:00:00  20000  
3 Natasha Romanoff    28 Female Los Angels 1995-05-14 00:00:00  10000  
4 Bruce Banner        30 Male   German     1997-08-15 00:00:00  10000  
5 Thor                40 Male   Asgard     1970-03-12 00:00:00  20000  
  
  
result.mean<- mean(my\_data$Salary,trim = 0.3)  
print(result.mean)  
  
 **Output:**  
[1] 13333.33  
  
  
  
median.result<- median(my\_data$Salary)  
print(median.result)  
  
 **Output:**  
[1] 10000  
  
  
getmode<- function(v) {  
   uniqv<- unique(v)  
   uniqv[which.max(tabulate(match(v, uniqv)))]  
}  
result <- getmode(my\_data$Salary)  
print(result)  
  
 **Output:**  
[1] 10000  
  
0setwd("e:/")  
library(XML)  
library(RCurl)  
library(stringr)  
library(plyr)  
# Read the URL.  
url<- "[http://www.omegahat.net](http://www.omegahat.net/)"  
# Gather the html links present in the webpage.  
links <- getHTMLLinks(url)  
filenames <- links[str\_detect(links, "JCMB\_2015")]  
# Store the file names as a list.  
filenames\_list<- as.list(filenames)  
download.file("[http://www.omegahat.net](http://www.omegahat.net/)","webfile")  
try(getHTMLLinks("<http://www.geos.ed.ac.uk/~weather/jcmb_ws/>"))  
print(links)  
# Identify only the links which point to the JCMB 2015 files.  
filenames <- links[str\_detect(links, "JCMB\_2015")]  
print(filenames)  
# Store the file names as a list.  
filenames\_list<- as.list(filenames)  
print(filenames\_list)  
# Create a function to download the files by passing the URL and filename list.  
downloadcsv<- function (mainurl,filename) {  
  filedetails<- str\_c(mainurl,filename)  
  download.file(filedetails,filename)  
}  
print(downloadcsv())

**10 Charts and Graphs**

**Aim:**

To create a Pie Chart, Bar Chart, Box Plot, Line Plot and Histogram in R Language.

**Algorithm:**

Step1:Start the R studio

Step2:Create Data for the Charts in Pie Chart, Bar Chart and Box Plot

Step3:Plot the Charts for Pie Chart, Bar Chart, Box Plot.

Step4: Create the Data for Graphs in Histogram and Lion Plot

Step5: Create a graph Histogram using the method hist and create a graph for Line Plot using the method plot .

Step6:Save the R File.

Step7: Print the Various charts and Graphs.

Step8:Stop the process.

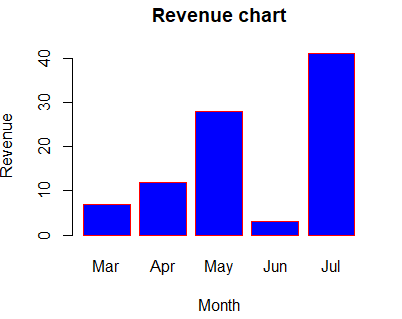
**A) Bar Chart**

H <- c(7,12,28,3,41)

M <- c("Mar","Apr","May","Jun","Jul")

barplot(H,names.arg=M,xlab="Month",ylab="Revenue",col="blue",main="Revenue chart",border="red")

**Output:**



**B) Box Plot**

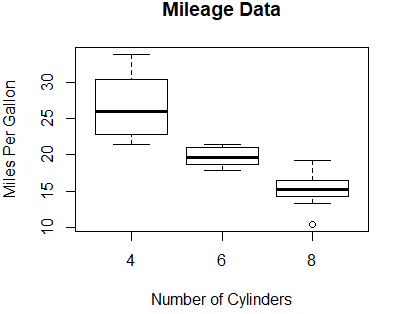
mtcars

input <- mtcars[,c('mpg','cyl')]

print(head(input))

boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders",ylab = "Miles Per Gallon", main = "Mileage Data")

**Output:**



**C) Pie Chart**

x <- c(21, 62, 10, 53,76)

labels <- c("London","NewYork","Singapore","Mumbai","Chennai")

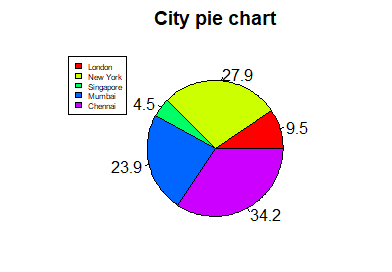
piepercent<- round(100\*x/sum(x), 1)

pie(x, labels = piepercent, main = "City pie chart",col = rainbow(length(x)))

legend("topleft", c("London","NewYork","Singapore","Mumbai","Chennai"), cex = 0.5,

fill = rainbow(length(x)))

**Output:**



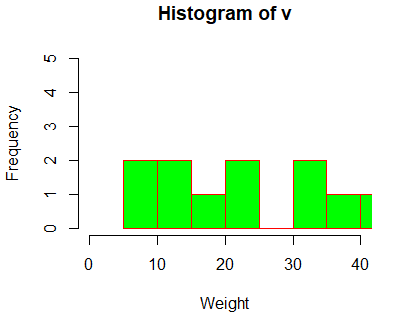
**D) Histogram**

v <- c(9,13,21,8,36,22,12,41,31,33,19)

hist(v,xlab = "Weight",col = "green",border = "red", xlim = c(0,40), ylim = c(0,5),

breaks = 5)

**Output:**



**E) Line Plot**

v <- c(7,12,28,3,41)

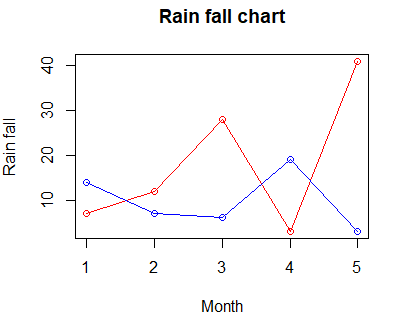
t <- c(14,7,6,19,3)

plot(v,type = "o",col = "red", xlab = "Month", ylab = "Rain fall",

main = "Rain fall chart")

lines(t, type = "o", col = "blue")

**Output:**



**11 Distributions**

**Aim:**

Write a R Program For Finding Distributions

a)Binomial Distribution

b)Poisson Uniform

c)Continuous Uniform Distribution

d)Exponential Distribution

e)Normal Distribution

f)Chi-Squared Distribution

**Algorithm:**

Step1: Start the Program

Step2: Create Vector x with Sequence of number from 0-50

Step3: Find the Binomial Distributions using dbinom(),qbinom(),rbinom(),pbinom() funtions

Step4: Find the Poisson Distributions using glm() function

Step5: Find the Continous uniform Distributions using runif() and dunif() function

Step6: Find the Exponenetial Distribution using dexp(),qexp(),rexp() and pexp() Function

Step7: Find the Normal Distribution using dnorm(),pnorm(),qnorm(),rnorm()

Step8: Find the Chi-Squared Distribution using Dchisq(),pchisq(),qchisq(),rchisq()

Step9: Stop the Program

**Program:**

**A) Binomial Distribution**

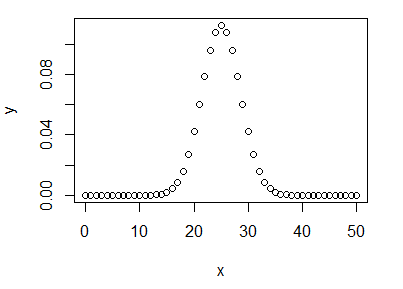
**dbinom()**

x <- seq(0,50,by = 1)

y <- dbinom(x,50,0.5)

plot(x,y)

**Output:**



## pbinom()

x <- pbinom(26,51,0.5)

print(x)

**Output**

0.610116

## qbinom()

x <-qbinom(0.25,51,1/2)

print(x)

**Output**

23

## rbinom()

x <-rbinom(8,150,.4)

print(x)

**Output**

54 64 66 61 53 63 58 65

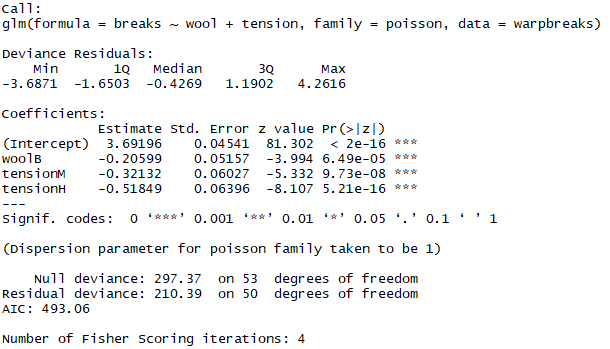
**B) Poisson Distribution**

output <-glm(formula = breaks ~ wool+tension, data = warpbreaks,

family = poisson)

print(summary(output))

**Output**



**C) Continuous Uniform Distribution**

## [runif()](https://www.geeksforgeeks.org/create-random-deviates-of-uniform-distribution-in-r-programming-runif-function/)

## runif(15, min=1, max=3)

## Output

## 

## [qunif()](https://www.geeksforgeeks.org/compute-the-value-of-quantile-function-over-uniform-distribution-in-r-programming-qunif-function/)

min <- 0

max <- 40

qunif(0.2, min = min, max = max)

**Output**

****

[**dunif()**](https://www.geeksforgeeks.org/compute-density-of-the-distribution-function-in-r-programming-dunif-function/)

x <- 5:10

dunif(x, min = 1, max = 20)

**Output**



[**punif()**](https://www.geeksforgeeks.org/compute-the-value-of-cdf-on-uniform-distribution-in-r-programming-punif-function/)

min <- 0

max <- 60

punif (15 , min =min , max = max)

**Output**



**D) Exponential Distribution**

**Program**

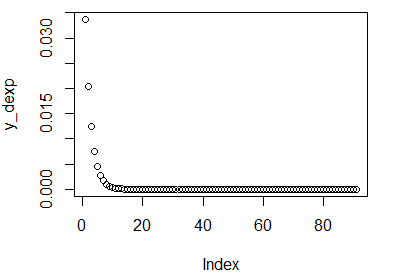
**dexp()**

x\_dexp<- seq(1, 10, by = 0.1)

y\_dexp<- dexp(x\_dexp, rate = 5)

plot(y\_dexp)

**Output**



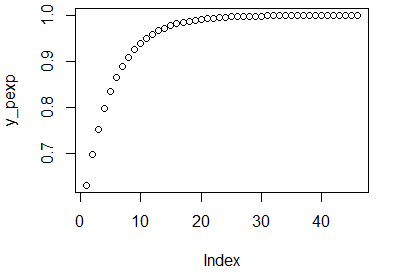
#### **pexp()**

x\_pexp<- seq(1, 10, by = 0.2)

y\_pexp<- pexp(x\_pexp, rate = 1)

plot(y\_pexp)

**Output**



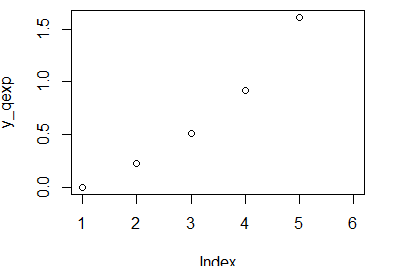
#### **qexp()**

x\_qexp<- seq(0, 1, by = 0.2)

y\_qexp<- qexp(x\_qexp, rate = 1)

plot(y\_qexp)

**Output**



#### **rexp()**

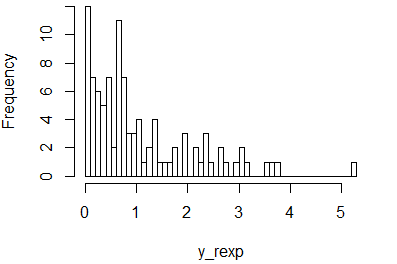
set.seed(500)

N <- 100

y\_rexp<- rexp(N, rate = 1)

hist(y\_rexp, breaks = 50, main = "")

**Output**



**E) Normal Distribution**

**Program**

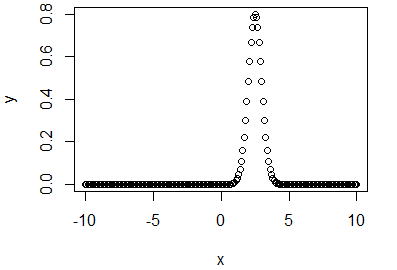
## dnorm()

## x <- seq(-10, 10, by = .1)

## y <- dnorm(x, mean = 2.5, sd = 0.5)

## plot(x,y)

**Output**



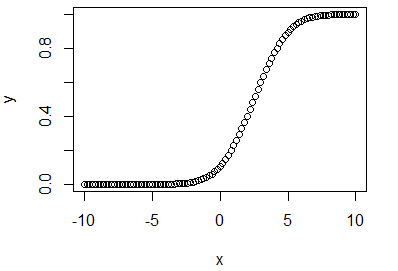
## pnorm()

## x <- seq(-10,10,by = .2)

## y <- pnorm(x, mean = 2.5, sd = 2)

## plot(x,y)

## Output:

****

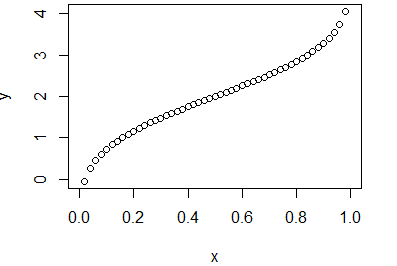
## qnorm()

## x <- seq(0, 1, by = 0.02)

## y <- qnorm(x, mean = 2, sd = 1)

## plot(x,y)

**Output**

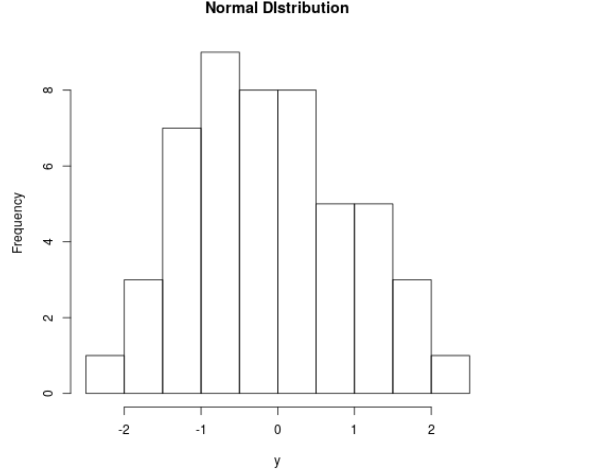
****

## rnorm()

## y <- rnorm(50)

## hist(y, main = "Normal DIstribution")

**Output**

****

**F) Chi-squared Distribution**

**Program**

## dchisq()

## dchisq(4:8, df = 7)

**Output**

****

## pchisq()

## pchisq(6, df = 7,lower.tail = TRUE)

**Output**

## 

## pchisq(6, df = 7,lower.tail = FALSE)

**Output**

## 

## qchisq()

## qchisq(0.999, df=7,lower.tail = TRUE)

**Output**

## 

## rchisq()

## set.seed(53535)

## N <- 10000

## y\_rchisq<- rchisq(N, df = 5)

## y\_rchisq

**Output**

## 

**12. Find hierarchical clustering**

**Aim:**  
 Write a R Program to Find Hierarchical Clustering  
  
 **Algorithm :**

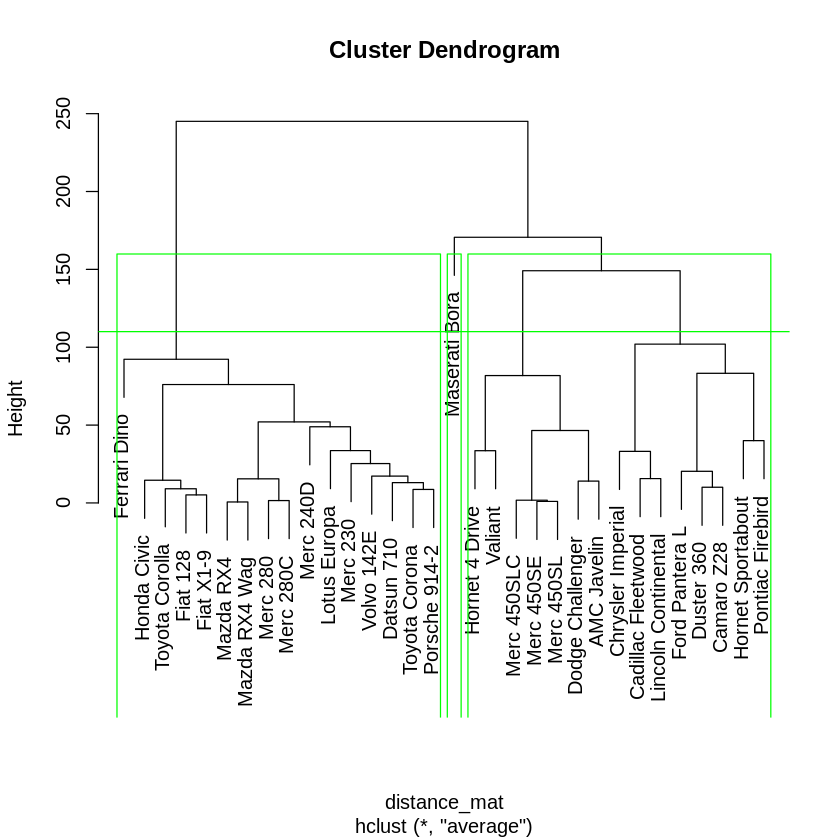
Step1: Start the program  
Step2: Install Package dplyr and Load it  
Step3: Load Mtcars in an Input Variable  
Step4: Find the HieraricalClustring using hclust() function  
Step5: Plot Graph  
Step6: Stop the program

**Program:**

install.packages("dplyr")  
library(dplyr)  
head(mtcars)

distance\_mat<- dist(mtcars, method = 'euclidean')  
distance\_mat  
set.seed(240) # Setting seed  
Hierar\_cl<- hclust(distance\_mat, method = "average")  
Hierar\_cl  
# Plotting dendrogram  
plot(Hierar\_cl)  
abline(h = 110, col = "green")  
# Cutting tree by no. of clusters  
fit <- cutree(Hierar\_cl, k = 3 )  
fit  
table(fit)  
rect.hclust(Hierar\_cl, k = 3, border = "green")

**Output:**



**13. K – Means clustering**

**Aim:**  
 Write a R program to Find K-Means Clustering  
  
  
 **Algorithm:**

Step1: Start the Program  
Step2: Load Data iris  
Step3: Satandarize the Data  
Step4: Check Mean and SD  
Step5: Create a Function Wssplot()  
Step6: Fitting the Cluster  
Step7: Cross Validate with Original Species available in data  
step8: Stop the program

**Program:**

**#**Loading iris dataset

data(iris)

#Viewing the head of iris

head(iris)

#Removing "Species column"

iris\_2<-iris[-5]

head(iris\_2)

#Standardize data

iris\_3<-as.data.frame(scale(iris\_2))

head(iris\_3)

#Checking Mean and SD of data before and after standardization

sapply(iris\_2,mean)

sapply(iris\_2,sd)

sapply(iris\_3,mean)

sapply(iris\_3,sd)

library(NbClust)

# creating function wssplot

wssplot<- function(data, nc=15, seed=1234){

wss<- (nrow(data)-1)\*sum(apply(data,2,var))

for (i in 2:nc){

set.seed(seed)

wss[i] <- sum(kmeans(data, centers=i)$withinss)}

plot(1:nc, wss, type="b", xlab="Number of Clusters",

ylab="Within groups sum of squares")}

# calling function wssplot()

wssplot(iris\_3,nc=30,seed=1234)

# fitting the clusters

iris\_kmeans<-kmeans(iris\_3,7)

iris\_kmeans$centers

iris\_kmeans$size

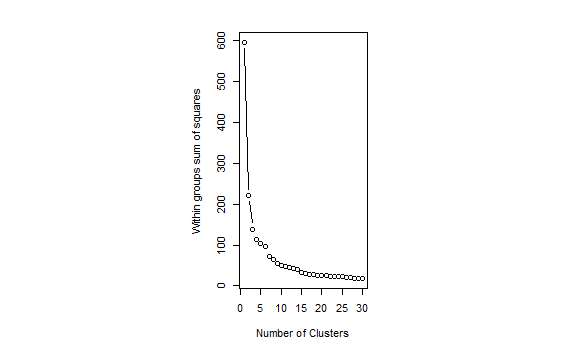
iris$clstr<-iris\_kmeans$cluster

# cross-validation with original species available in data

iris$clstr<-iris\_kmeans$cluster

table(iris$Species,iris$clstr

**Output:**



**14.Association rule**

**Aim:**

Write a R program to find Association Rule

**Algorithm:**

Step1: Start the Program

Step2: Load the Library arules

Step3: Find the Associate rule using apriori() Functions

Step4: Sort the Data

Step5: Stop the Program

**15 .Guna’s theory**

**Aim:**

Write a RProgram to Case Studyon Guna's Theory

**Algorithm:**

Step1 : Start the program

Step2 : Read the CSV Data From Guna1.Csv file and Store it in Guna variable

Step3 : Use barplot() function to plot the csv file

Step4 : Load the library lattice

Step5 : And Perform all the Plotting Function For Case Study

Step6 : Stop the Program

**Program:**

guna<-read.csv("guna1.csv")

head(guna)

barplot(table(guna$gunas),col="green")

xyplot(Apptitude ~ Attitude ,data=guna)

Library(lattice)

xyplot(Apptitude ~ Attitude | gunas,data=guna,layout=c(3,1))

Library(ggplot2)

qplot(Attitude,Apptitude,data=guna)

boxplot(Attitude ~ Apptitude, guna,xlab="Attitude",ylab="Apptitude")

Base plot for annotation

with(guna,plot(Attitude,Apptitude),main="Attitude Vs Apptitude")

with(subset(guna,gunas=="Rajasic"),points(Attitude,Apptitude,col="Blue"))

with(subset(guna,gunas=="Sattvic"),points(Attitude,Apptitude,col="Red"))

with(subset(guna,gunas=="Tamasic"),points(Attitude,Apptitude,col="Green"))

legend("bottomright",pch=1,col=c("green","blue","red"),legend=c("Tamasic","Rajasic","Sattvic"))

with(guna,plot(Attitude,Apptitude),main="Attitude Vs Apptitude")

model<-lm(Attitude ~ Apptitude,guna)

abline(model,lwd=4)

abline(model,lwd=2)

library(lattice)

xyplot(Attitude~Apptitude|Gender,data=guna,layout=c(3,1))

xyplot(Attitude ~ Apptitude | gunas, guna,

group = Gender,

grid = TRUE,

scales = list(x = list(log = 10, equispaced.log = FALSE)))

xyplot(Attitude ~ Apptitude | gunas, guna1,

group = Gender, xlab="Attitude", ylab="Apptitude",

pch=c(3,0), col=c("blue", "red"), cex=0.5,

main="Attitude vs Apptitude ",

key=list(text=list(c("Female", "Male")),

points=list(pch=c(3,0), col=c("blue", "red")), columns=2))

xyplot( Number + Letter + Verbal + Logical+Non.Verbal +analytical ~Attitude , groups = Gender , outer =TRUE , data =guna1 , auto.key = list ( columns =2) )

pdf(file="myplot1.pdf")

with(guna1,plot(Attitude,Apptitude))

title(main="Attitude Vs Apptitude")

dev.off()

set.seed(1234)

par(mar = c(0, 0, 0, 0))

x <- rnorm(12, mean = rep(1:3, each = 4), sd = 0.2)

y <- rnorm(12, mean = rep(c(1, 2, 1), each = 4), sd = 0.2)

plot(x, y, col = "blue", pch = 19, cex = 2)

text(x + 0.05, y + 0.05, labels = as.character(1:12))

dataFrame<- data.frame(x = x, y = y)

dist(dataFrame)

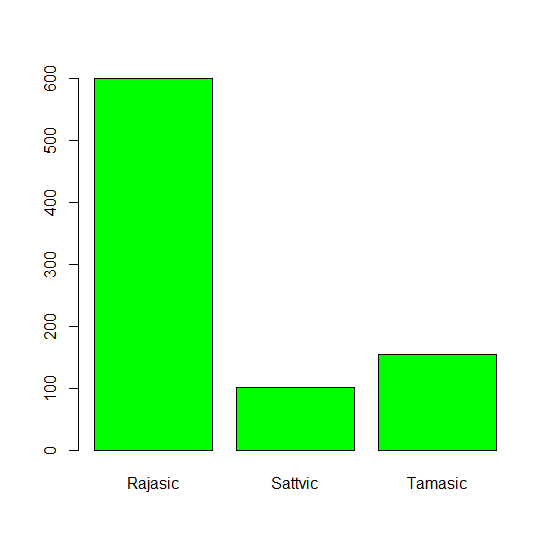
dataFrame<- data.frame(x = x, y = y)

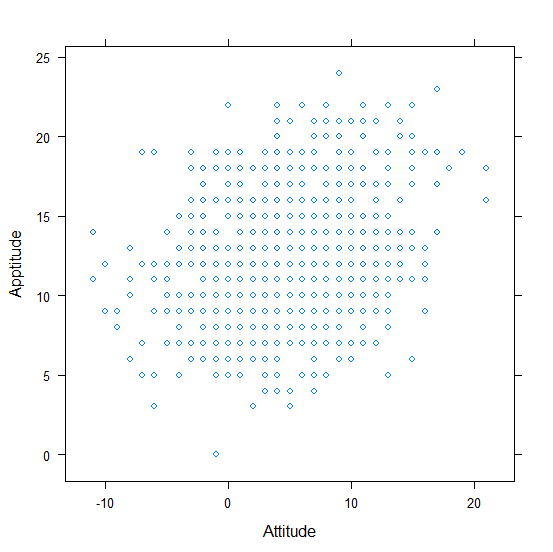
distxy<- dist(dataFrame)

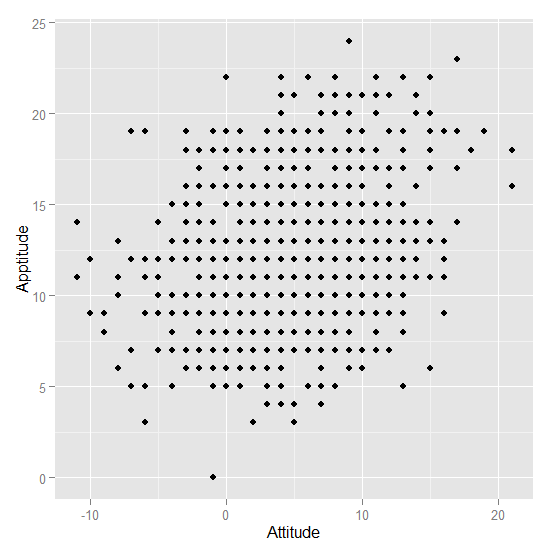
hClustering<- hclust(distxy)

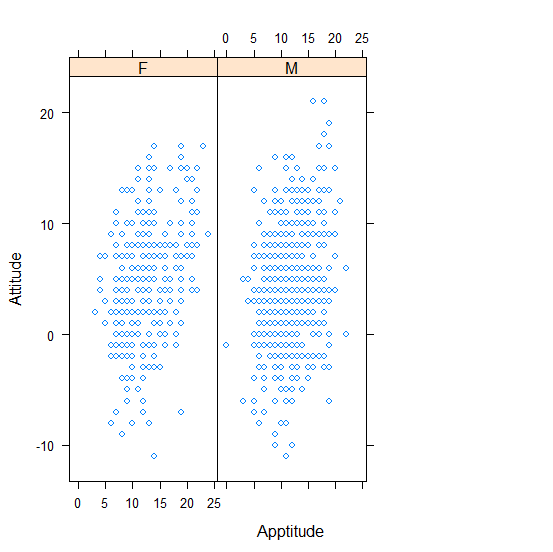
plot(hClustering)

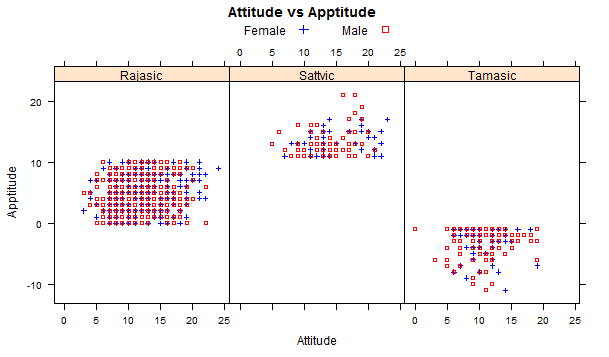
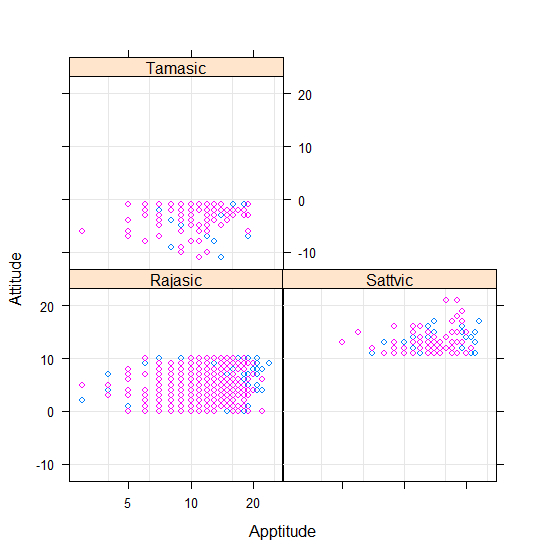
**Output:**

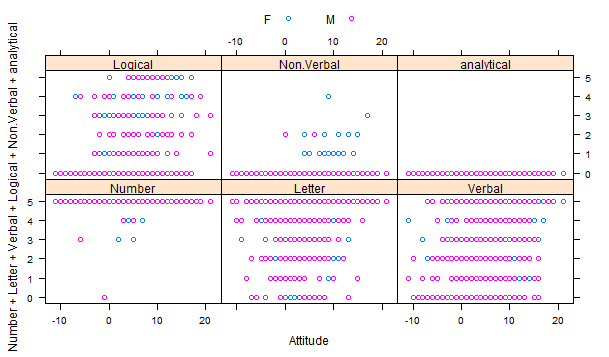


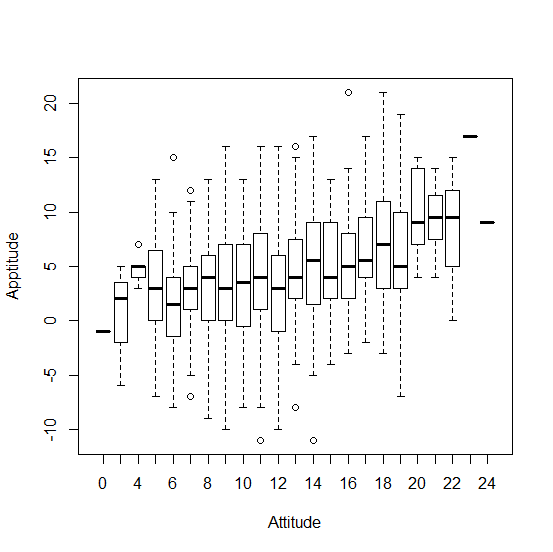


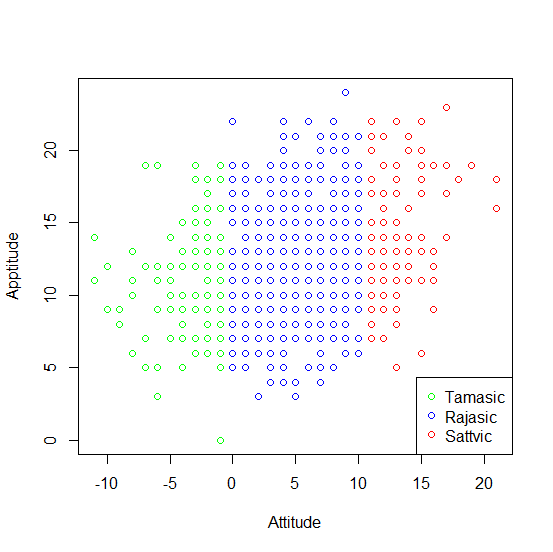
­­

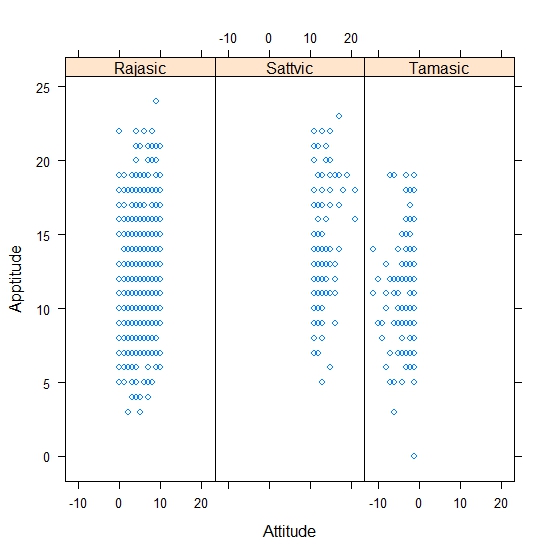












**16.Analysis of variance (ANOVA)**

**Aim:**

To write a program to analysis of variance using R.

**Algorithm:**

Step 1: Open R studio. Start the program.

Step 2: Type the interval estimation data.

Step 3: Set working directory.

Step 4: Process the data.

Tm=g1(k,1,n\*k,factor(f))

Step 5: Get the output.

Step 6: Save and exit the program

**Program:**

input <- mtcars[,c("am","mpg","hp")]  
print(head(input))  
input <- mtcars  
result <- aov(mpg~hp\*am,data = input)  
print(summary(result))

**Output:**

       Df Sum Sq Mean Sq F value   Pr(>F)      
hp           1  678.4   678.4  77.391 1.50e-09 \*\*\*  
am           1  202.2   202.2  23.072 4.75e-05 \*\*\*  
hp:am        1    0.0     0.0   0.001    0.981      
Residuals   28  245.4     8.8                      
---  
Signif. codes:  0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  
  
input <- mtcars  
result <- aov(mpg~hp+am,data = input)  
print(summary(result))  
  
**Output:**  
            Df Sum Sq Mean Sq F value   Pr(>F)      
hp           1  678.4   678.4   80.15 7.63e-10 \*\*\*  
am           1  202.2   202.2   23.89 3.46e-05 \*\*\*  
Residuals   29  245.4     8.5                      
---  
Signif. codes:  0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  
  
input <- mtcars  
result1 <- aov(mpg~hp\*am,data = input)  
result2 <- aov(mpg~hp+am,data = input)  
print(anova(result1,result2))  
  
**Output:**

Analysis of Variance Table  
  
Model 1: mpg ~ hp \* am  
Model 2: mpg ~ hp + am  
  Res.Df    RSS Df  Sum of Sq     F Pr(>F)  
1     28 245.43                            
2     29 245.44 -1 -0.0052515 6e-04 0.9806

print(input)  
result1 <- aov(mpg~hp\*cyl,data = input)  
result2 <- aov(mpg~hp+cyl,data = input)  
print(anova(result1,result2))

**Output:**  
  
  
                     mpg cyl  disp  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  
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3  
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3  
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3  
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4  
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4  
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4  
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1  
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2  
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1  
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1  
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2  
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2  
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4  
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2  
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1  
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2  
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2  
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4  
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6  
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8  
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2

print(anova(result1,result2))

**Output:**

Analysis of Variance Table  
  
Model 1: mpg ~ hp \* cyl  
Model 2: mpg ~ hp + cyl  
  Res.Df    RSS Df Sum of Sq      F Pr(>F)    
1     28 247.60                              
2     29 291.98 -1   -44.375 5.0181 0.0332 \*  
---  
Signif. codes:  0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

**17.**W**ilcox on signed-rank test**

**Aim:**

To write a program Wilcox on Signed-rank test R Studio

**Algorithm:**

Step1: To Open the R-Studio soft ware in the System.

Step2: Use the library Dataset integer named immer of the head of data frame columns y1 and y2.

Step3: Enter the R-Studio in local variables of barley yields (y1 and y2) immer values of the Wilcox.test

Step4: Then data values of the immer y1 and y2.

Step5: Then Finally execute and Save the program.

**Program:**

set.seed(1234)

myData = data.frame(

name = paste0(rep("R\_", 10), 1:10),

weight = round(rnorm(10, 30, 2), 1)

)

wilcox.test(myData$weight, mu = 25,

alternative = "less")

print(result)

**Output:**

Wilcoxon signed rank test with continuity correction

data:  myData$weight

V = 55, p-value = 0.9979

alternative hypothesis: true location is less than 25

**18.Time Series Analysis**

**19.Decision Trees on any dataset**

**Aim:**

To write a program on Decision tree with any dataset

**Algorithm:**

Step1: Open R Package

Step2: install.packages("party")

Step3: The package "party" has the function ctree() which is used to create and analyzedecison tree.

Step4: We will use the R in-built data set named readingSkills to create a decision tree.

Step5: Create the input data frame.

Step6: Give the chart file a name png(file = "decision\_tree.png")

Step7: Create a tree and plot a tree

Step8: Then Finally execute and Save the program.

**Program:**

install.packages("party")

library(party)

print(head(readingSkills))

library(party)

print(head(readingSkills))

input.dat <- readingSkills[c(1:105),]

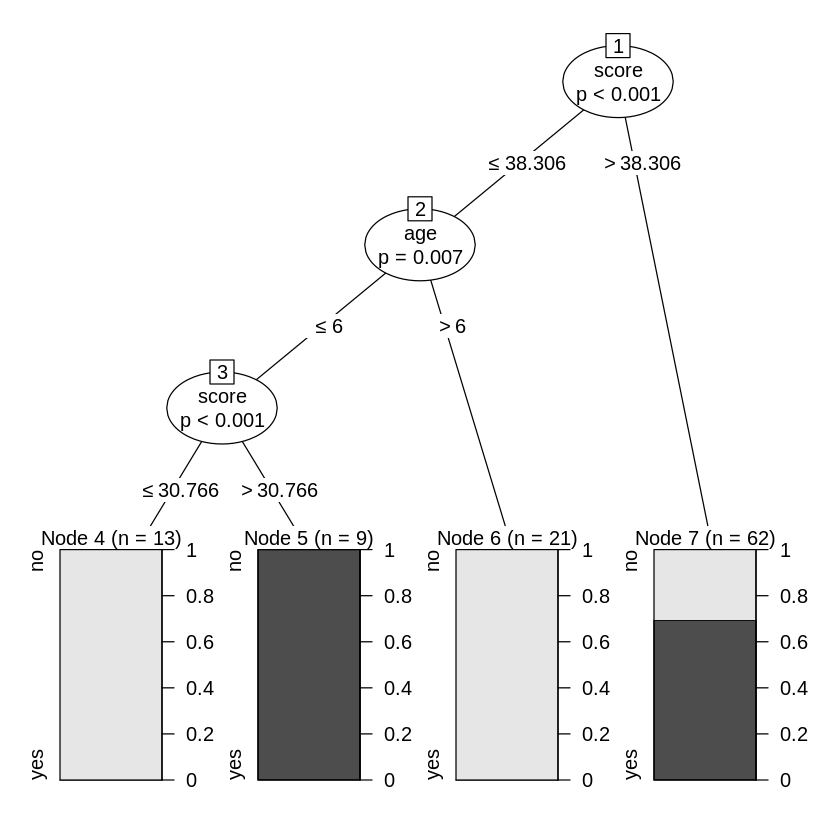
output.tree<- ctree(

nativeSpeaker ~ age + shoeSize + score,

data = input.dat)

plot(output.tree)

**Output:**

****

**20.Random Forest using Titatnic dataset**

**Aim:**

To write a program on Random forest using Titatnic dataset

**Algorithm:**

Step1: Open R Package

Step2: Use the below command in R console to install the package. You also have to install the dependent packages if any install.packages("randomForest")

Step3: We will use the randomForest() function to create the decision tree and see it's graph.

Step4: Load the Titatnic package. It will automatically load other required packages.

Step5: Give the chart file a name png(file = "titatnic.png")

Step6: Create a tree and plot a tree

Step7: Then Finally execute and Save the program.

**Program:**

install.packages("randomForest")

install.packages("titanic")

library(titanic)

print(head(Titanic))

library(randomForest)

output.forest<- randomForest(Class ~Sex, data = Titanic)

print(output.forest)

print(importance(output.forest,type = 2))

**Output:**

|  |
| --- |
| Call:  randomForest(formula = Class ~ Sex, data = Titanic)  Type of random forest: classification  Number of trees: 500  No. of variables tried at each split: 1  OOB estimate of error rate: 100%  Confusion matrix:  1st 2nd 3rd Crew class.error  1st 0 4 1 3 1  2nd 2 0 2 4 1  3rd 4 1 0 3 1  Crew 3 2 3 0 1  >  >print(importance(output.forest,type = 2))  MeanDecreaseGini  Sex 0.7419123 |
|  |
| |  | | --- | |  | |

**21 Survival analysis**

**Aim:**

Write a program to analyse the **Survival**.

**Algorithm:**

Step 1: Open the R Studio

Step 2: Install the package of survival.

Step 3: Create library survival

Step 4:print the first few rows.

Step 5: Create the survival object.

step 6: plot the graph.

Step 6: Save and run the program.

Step 7: Verify the output in output area.

**Program:**

install.packages("survival")

library("survival")

print(head(pbc))

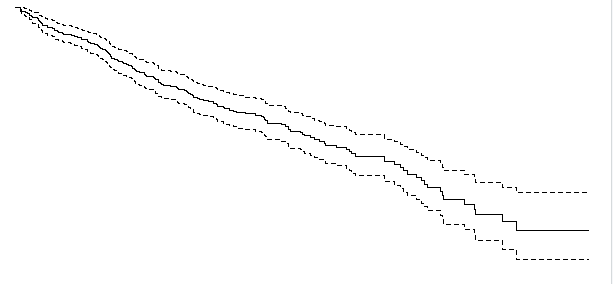
library("survival")

survfit(Surv(pbc$time,pbc$status == 2)~1)

plot(survfit(Surv(pbc$time,pbc$status == 2)~1))

**Output:**

|  |
| --- |
| Call: survfit(formula = Surv(pbc$time, pbc$status == 2) ~ 1)  n events median 0.95LCL 0.95UCL  418 161 3395 3090 3853 |
|  |



**22. Mathematical functions using Numpy**

**Aim:**

Write a mathematical functions using numpy in R Studio.

**Algorithm:**

Step 1: Open the R studio.

Step 2: Import numpy and math library

Step 3: Use the math functions like sin, cos, sinhetc..

Step 4: Declare the array and use the appropriate math functions and print the array values.

Step 5: By using numpy and math write mathematical functions one by one

Step 6: Save the program, run it one by one and get the output in output area.

**Program:**

**sin() function**

import numpy as np

import math

in\_array = [0, math.pi / 2, np.pi / 3, np.pi]

print ("Input array : \n", in\_array)

Sin\_Values = np.sin(in\_array)

print ("\nSinevalues : \n", Sin\_Values)

**Output:**

Input array :

[0, 1.5707963267948966, 1.0471975511965976, 3.141592653589793]

Sine values :

[ 0.00000000e+00 1.00000000e+00 8.66025404e-01 1.22464680e-16]

**cos() function**

import numpy as np

import math

in\_array = [0, math.pi / 2, np.pi / 3, np.pi]

print ("Input array : \n", in\_array)

cos\_Values = np.cos(in\_array)

print ("\nCosinevalues : \n", cos\_Values)

**Output:**

("\nCosinevalues : \n", cos\_Values)

Input array :

[0, 1.5707963267948966, 1.0471975511965976, 3.141592653589793]

Cosine values :

[ 1.000000e+00 6.123234e-17 5.000000e-01 -1.000000e+00]

**sinh() function**

import numpy as np

import math

in\_array = [0, math.pi / 2, np.pi / 3, np.pi]

print ("Input array : \n", in\_array)

Sinh\_Values = np.sinh(in\_array)

print ("\nSine Hyperbolic values : \n", Sinh\_Values)

**Output:**

Input array :

[0, 1.5707963267948966, 1.0471975511965976, 3.141592653589793]

Sine Hyperbolic values :

[ 0. 2.3012989 1.24936705 11.54873936]

**cosh() function**

import numpy as np

import math

in\_array = [0, math.pi / 2, np.pi / 3, np.pi]

print ("Input array : \n", in\_array)

cosh\_Values = np.cosh(in\_array)

print ("\ncosine Hyperbolic values : \n", cosh\_Values)

**Output:**

Input array :

[0, 1.5707963267948966, 1.0471975511965976, 3.141592653589793]

cosine Hyperbolic values :

[ 1. 2.50917848 1.60028686 11.59195328]

**around() function**

import numpy as np

in\_array = [.5, 1.5, 2.5, 3.5, 4.5, 10.1]

print ("Input array : \n", in\_array)

round\_off\_values = np.around(in\_array)

print ("\nRoundedvalues : \n", round\_off\_values)

in\_array = [.53, 1.54, .71]

print ("\nInputarray : \n", in\_array)

round\_off\_values = np.around(in\_array)

print ("\nRoundedvalues : \n", round\_off\_values)

in\_array = [.5538, 1.33354, .71445]

print ("\nInputarray : \n", in\_array)

round\_off\_values = np.around(in\_array, decimals = 3)

print ("\nRoundedvalues : \n", round\_off\_values)

**Output:**

Input array :

[0.5, 1.5, 2.5, 3.5, 4.5, 10.1]

Rounded values :

[ 0. 2. 2. 4. 4. 10.]

Input array :

[0.53, 1.54, 0.71]

Rounded values :

[1. 2. 1.]

Input array :

[0.5538, 1.33354, 0.71445]

Rounded values :

[0.554 1.334 0.714]

**round\_() function**

import numpy as np

in\_array = [.5, 1.5, 2.5, 3.5, 4.5, 10.1]

print ("Input array : \n", in\_array)

round\_off\_values = np.round\_(in\_array)

print ("\nRoundedvalues : \n", round\_off\_values)

in\_array = [.53, 1.54, .71]

print ("\nInputarray : \n", in\_array)

round\_off\_values = np.round\_(in\_array)

print ("\nRoundedvalues : \n", round\_off\_values)

in\_array = [.5538, 1.33354, .71445]

print ("\nInputarray : \n", in\_array)

round\_off\_values = np.round\_(in\_array, decimals = 3)

print ("\nRoundedvalues : \n", round\_off\_values)

**Output:**

Input array :

[0.5, 1.5, 2.5, 3.5, 4.5, 10.1]

Rounded values :

[ 0. 2. 2. 4. 4. 10.]

Input array :

[0.53, 1.54, 0.71]

Rounded values :

[1. 2. 1.]

Input array :

[0.5538, 1.33354, 0.71445]

Rounded values :

[0.554 1.334 0.714]

**exp() function**

import numpy as np

in\_array = [1, 3, 5]

print ("Input array : ", in\_array)

out\_array = np.exp(in\_array)

print ("Output array : ", out\_array)

import numpy as np

in\_array = [1, 3, 5]

print ("Input array : ", in\_array)

out\_array = np.exp(in\_array)

print ("Output array : ", out\_array)

**Output:**

Input array : [1, 3, 5]

Output array : [ 2.71828183 20.08553692 148.4131591 ]

**log() function**

import numpy as np

in\_array = [1, 3, 5, 2\*\*8]

print ("Input array : ", in\_array)

out\_array = np.log(in\_array)

print ("Output array : ", out\_array)

print("\nnp.log(4\*\*4) : ", np.log(4\*\*4))

print("np.log(2\*\*8) : ", np.log(2\*\*8))

**Output:**

Input array : [1, 3, 5, 256]

Output array : [0. 1.09861229 1.60943791 5.54517744]

np.log(4\*\*4) : 5.545177444479562

np.log(2\*\*8) : 5.545177444479562

**importing numpy**

import numpy as np

in\_num = 2.0

print ("Input number : ", in\_num)

out\_num = np.reciprocal(in\_num)

print ("Output number : ", out\_num)

**Output:**

Input number : 2.0

Output number : 0.5

**divide() function**

import numpy as np

# input\_array

arr1 = [2, 27, 2, 21, 23]

arr2 = [2, 3, 4, 5, 6]

print ("arr1 : ", arr1)

print ("arr2 : ", arr2)

# output\_array

out = np.divide(arr1, arr2)

print ("\nOutputarray : \n", out)

**Output:**

arr1 : [2, 27, 2, 21, 23]

arr2 : [2, 3, 4, 5, 6]

Output array :

[1. 9. 0.5 4.2 3.83333333]

**numpy.isreal() method**

import numpy as geek

print("Is Real : ", geek.isreal([1+1j, 0j]), "\n")

print("Is Real : ", geek.isreal([1, 0]), "\n")

**Output:**

Is Real : [False True]

Is Real : [ True True]

**importing numpy**

import numpy as np

in\_complx1 = 2+4j

out\_complx1 = np.conj(in\_complx1)

print ("Output conjugated complex number of 2+4j : ", out\_complx1)

in\_complx2 =5-8j

out\_complx2 = np.conj(in\_complx2)

print ("Output conjugated complex number of 5-8j: ", out\_complx2)

**Output:**

Output conjugated complex number of 2+4j : (2-4j)

Output conjugated complex number of 5-8j: (5+8j)

**cbrt () function**

import numpy as np

arr1 = [1, 27000, 64, -1000]

print ("cbrt Value of arr1 : \n", np.cbrt(arr1))

arr2 = [1024 ,-128]

print ("\ncbrt Value of arr2 : ", np.cbrt(arr2))

**Output:**

cbrt Value of arr1 :

[ 1. 30. 4. -10.]

cbrt Value of arr2 : [10.0793684 -5.0396842]

**importing the numpy**

import numpy as np

in\_array = [1, 2, 3, 4, 5, 6, 7, 8 ]

print ("Input array : ", in\_array)

out\_array = np.clip(in\_array, a\_min = 2, a\_max = 6)

print ("Output array : ", out\_array)

**Output:**

Input array : [1, 2, 3, 4, 5, 6, 7, 8]

Output array : [2 2 3 4 5 6 6 6]

**23.Data analytics using Pandas**

**Aim:**

To write a program in Data analytics using Pandas.

**Algorithm:**

Step1: Start the program

Step2: Load the PANDAS package

Step3 Import and read the CSV File

Step4: Run the CSV file

Step5: To take a action data.head(), data.info(), data.shape(), data.describe(), data.dropna()

Step6: Verify the output

Step7: Exit the program

**PROGRAM:**

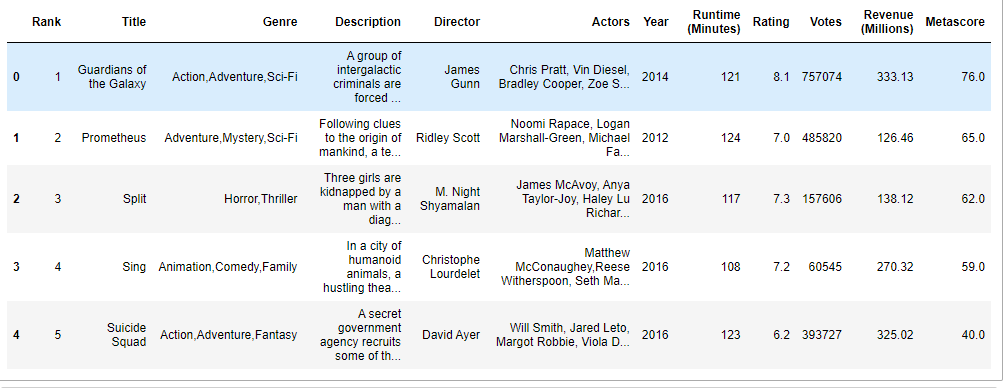
import pandas as pd

data = pd.read\_csv("/content/IMDB-Movie-Data.csv")

data\_indexed = pd.read\_csv('IMDB-Movie-Data.csv', index\_col="Title")

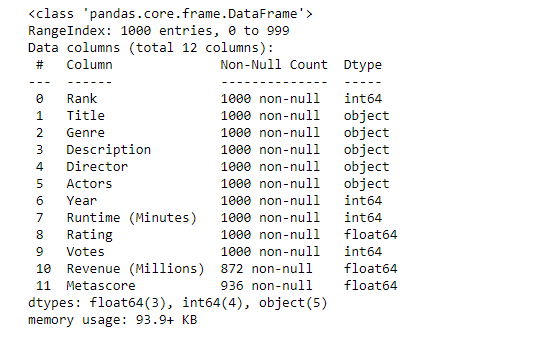
data.head()

**Output:**



data.info()

**Output:**



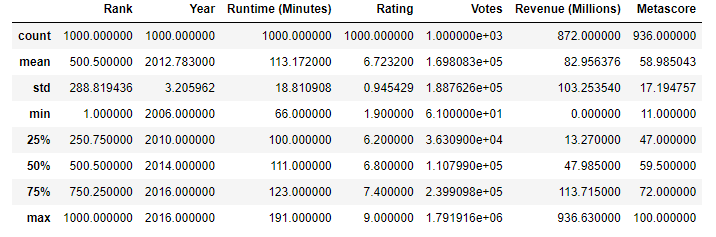
data.shape

**Output:**

(1000, 12)

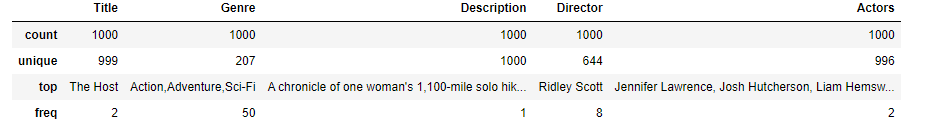
data.describe()

**Output:**

****

data.describe(include=['object','bool'])

**Output:**



genre = data['Genre']

data[['Genre']]

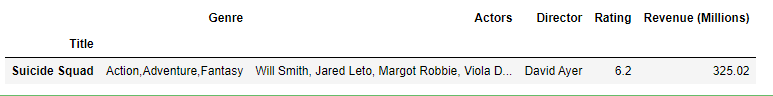
**Output:**



some\_cols = data[['Title','Genre','Actors','Director','Rating']]

data\_indexed.loc[['Suicide Squad']][['Genre','Actors','Director','Rating','Revenue (Millions)']]

**Output:**

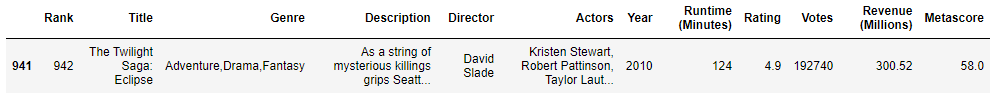


data[((data['Year'] >= 2010) & (data['Year'] <= 2016))

& (data['Rating'] < 6.0)

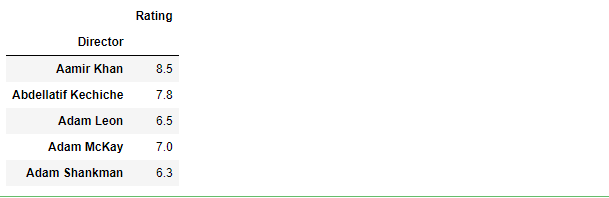
& (data['Revenue (Millions)'] > data['Revenue (Millions)'].quantile(0.95))]

**Output:**



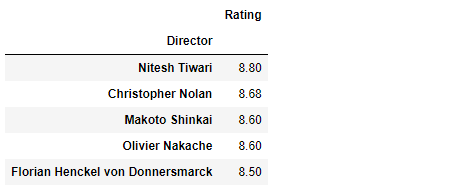
data.groupby('Director')[['Rating']].mean().head()

**Output:**



data.groupby('Director')[['Rating']].mean().sort\_values(['Rating'], ascending=False).head()

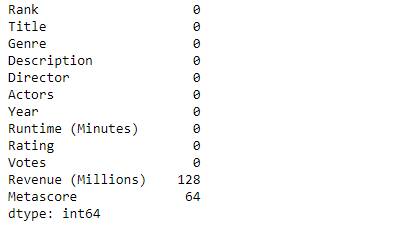
**Output:**



# To check null values row-wise

data.isnull().sum()

**Output:**



# Use drop function to drop columns

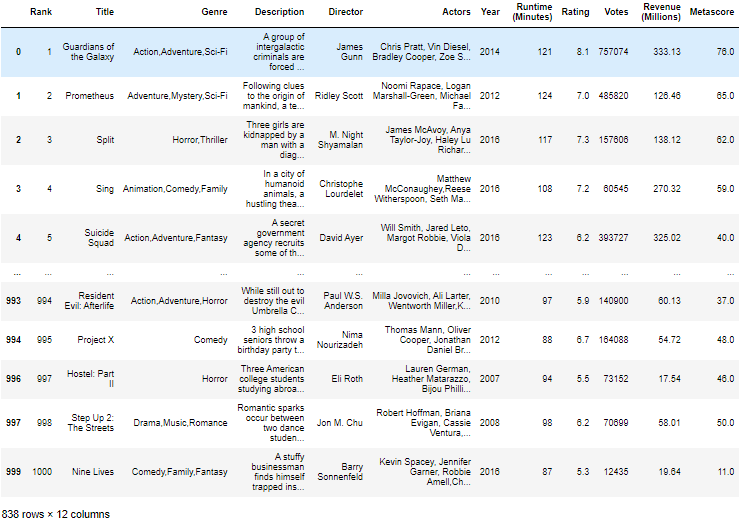
data.drop('Metascore', axis=1).head()

**Output:**



data.dropna()

**Output:**



revenue\_mean = data\_indexed['Revenue (Millions)'].mean()

print("The mean revenue is: ", revenue\_mean)

data\_indexed['Revenue (Millions)'].fillna(revenue\_mean, inplace=True)

def rating\_group(rating):

if rating >= 7.5:

return 'Good'

elif rating >= 6.0:

return 'Average'

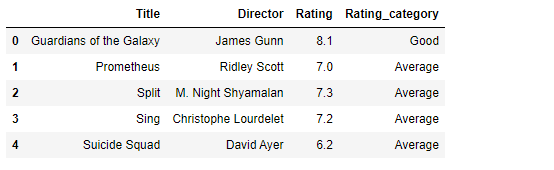
else:

return 'Bad'

data['Rating\_category'] = data['Rating'].

apply(rating\_group)

**Output:**



**24.Visual representations using Matplotlib**

**Aim:**

To write a program in Visual representations using Matplotlib.

**Algorithm:**

Step1: Start the program

Step2: Load the library RETICULATE in the program

Step3 Import and read the MATPLOTLIB package

Step4: Then use keyword “force = TRUE"

Step5: To use logical function to calculate receiving salary greater than 5000 and employees whose age is greater than 50

Step6: Run the Matplotlib program

Step7: Verify the output

Step8: Exit the program

**Program:**

 library(reticulate)

use\_python("/usr/bin/python3") # Change accordingly to your Python version

matplotlib <- import("matplotlib")

matplotlib$use("Agg", force = TRUE)

**Output:**

