MATURI VENKATA SUBBA RAO ENGINEERING COLLEGE

(An Autonomous Institution)

(Affiliated to Osmania University & Recognized by AICTE) Nadergul, RangaReddy Dist.

CERTIFICATE

Department of COMPUTER SCIENCE & ENGINEERING

Certified that this is a bonafide work of lab experiments carried out by Mr/Ms. ______________________________bearing Roll.No.___________________under the course of Data Science Laboratory prescribed by Osmania University for B.E. Sem-VII of Computer Science & Engineering during the academic year 2021–2022.

Internal Examiner ***External Examiner***

VISION AND MISSION

VISION

- To impart technical education of the highest standards, producing competent and confident engineers with an ability to use computer science knowledge to solve societal problems.

MISSION

- To make learning process exciting, stimulating and interesting.
- To impart adequate fundamental knowledge and soft skills to students.
- To expose students to advanced computer technologies in order to excel in engineering practices by bringing out the creativity in students.
- To develop economically feasible and socially acceptable software.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

The Bachelor's program in Computer Science and Engineering is aimed at preparing graduates who will:-
PEO-1: Achieve recognition through demonstration of technical competence for successful execution of software projects to meet customer business objectives.
PEO-2: Practice life-long learning by pursuing professional certifications, higher education or research in the emerging areas of information processing and intelligent systems at a global level.
PEO-3: Contribute to society by understanding the impact of computing using a multidisciplinary and ethical approach.

(A) PROGRAM OUTCOMES (POs)

At the end of the program the students (Engineering Graduates) will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principle and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Lifelong learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

(B) PROGRAM SPECIFIC OUTCOMES (PSOs)

13. (PSO-1) Demonstrate competence to build effective solutions for computational real-world problems using software and hardware across multi-disciplinary domains.
14. (PSO-2) Adapt to current computing trends for meeting the industrial and societal needs through a holistic professional development leading to pioneering careers or entrepreneurship

Course Name: Data Science Lab

Course Code: PC 751 CS

Academic Year: 2021-2022

Semester: VII

Course objectives:

- To understand the R Programming Language
- Exposure on solving of data science problem
- Understand Classification and Regression Modelling

Course outcomes:

- Work with data science using R Programming environment
- Implement various statistical concepts like linear and logistic regression
- Perform Classification and Clustering over a given data set

INDEX

S.No.	Name of the Experiment	Experiment Date	Date of Submission	Page No.
1	**R AS CALCULATOR APPLICATION**			
1.1	Using with and without R objects on console			
1.2	Using mathematical functions on console			
1.3	Write an R script, to create R objects for calculator application and save in a specified location in disk.			
2	**DESCRIPTIVE STATISTICS IN R**			
2.1	Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars & cars datasets.			
2.2	Write an R script to find subset of dataset by using subset(), aggregate() functions on iris dataset.			
3	**READING AND WRITING DIFFERENT TYPES OF DATASETS**			
3.1	Reading different types of data sets (.txt, .csv) from web and disk and writing in file in specific disk location.			
3.2	Reading Excel data sheet in R.			
4	**VISUALIZATIONS**			
4.1	Find the data distributions using box and scatterplot.			
4.2	Find the outliers using plot.			
4.3	Plot the histogram, bar chart and pie chart on sample data.			
5	**CORRELATION AND COVARIANCE**			
5.1	Find the correlation matrix.			
5.2	Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data.			
5.3	Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data.			
6	**REGRESSION MODEL**			
6.1	Import a data from web storage. Name the dataset and perform Logistic Regression to find out relation between variables the model. Also check the model is fit or not [require (foreign), require(MASS)]			
7	**CLASSIFICATION MODEL**			
7.1	Install relevant package for classification			
7.2	Choose classifier for classification problem			
7.3	Evaluate the performance of classifier			
8	**CLUSTERING MODEL**			
8.1	Clustering algorithms for unsupervised classification			
8.2	Plot the cluster data using R visualizations			

1. R AS CALCULATOR APPLICATION
(i) Without using R objects on console

```
> 25+32
```

Output:
```
[1] 57
```

```
> 36-15
```

Output:
```
[1] 21
```

```
> 145*8
[1] 1160
```

```
> 365/7
```
Output:
```
[1] 52.14
```

(ii) Using R objects on console:
```
>A=45
>B=2
>c=A+B
>c
```

Output:
```
[1]47
```

```
>A=5
>B=2
>c=A-B
>c
```

Output:
```
[1]3
>A=10
>B=2
>c=A*B
>c
```

Output:
```
[1]20
```

```
>A=4
>B=2
>c=A/B
>c
```

Output:
[1]2

(c) Program make a simple calculator that can add, subtract, multiply and divide using functions

```
add <- function(x, y) {
  return(x + y)
}
subtract <- function(x, y) {
  return(x - y)
}
multiply <- function(x, y) {
  return(x * y)
}
divide <- function(x, y) {
  return(x / y)
}

# take input from the user
print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")
choice = as.integer(readline(prompt="Enter choice[1/2/3/4]: "))
num1 = as.integer(readline(prompt="Enter first number: "))
num2 = as.integer(readline(prompt="Enter second number: "))
operator <- switch(choice,"+","-","*","/")
result <- switch(choice, add(num1, num2), subtract(num1, num2), multiply(num1, num2), divide(num1,
num2))
print(paste(num1, operator, num2, "=", result))
```

OUTPUT

```
> source("~/Documents/305.R")
[1] "Select operation."
[1] "1.Add"
[1] "2.Subtract"
[1] "3.Multiply"
[1] "4.Divide"
Enter choice[1/2/3/4]: 3
Enter first number: 3
Enter second number: 4
[1] "3 * 4 = 12"
```

2.DESCRIPTIVE STATISTICS IN R

a. Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars& cars datasets.

load data mtcars:

```
data(mtcars)
```

structure of mtcars:

```
str(mtcars)
```

Output:

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

str( ) → This function compactly displays the internal structure of an R object, a diagnostic function and an alternative to summary. Ideally, only one line for each ‘basic’ structure is displayed. It is especially well suited to compactly display the (abbreviated) contents of (possibly nested) lists.

dimension of dataset:

```
dim(mtcars)
```

Output:

```
[1] 32 11
```

get names of each variables or columns:

```
names(mtcars)
```

Output:

```
[1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"  "carb"
```

Summaries of the datasets:

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

Usage:

```
>summary(mtcars)
```

Output:

```
     mpg             cyl             disp             hp             drat
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0   Min.   :2.760
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080
 Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930
       wt             qsec             vs               am              gear
 Min.   :1.513   Min.   :14.50   Min.   :0.0000   Min.   :0.0000   Min.   :3.000
 1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:3.000
 Median :3.325   Median :17.71   Median :0.0000   Median :0.0000   Median :4.000
 Mean   :3.217   Mean   :17.85   Mean   :0.4375   Mean   :0.4062   Mean   :3.688
 3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:4.000
 Max.   :5.424   Max.   :22.90   Max.   :1.0000   Max.   :1.0000   Max.   :5.000
      carb
 Min.   :1.000
 1st Qu.:2.000
 Median :2.000
 Mean   :2.812
 3rd Qu.:4.000
 Max.   :8.000
```

quantiles of dataset:

quantile()

The generic function quantile produces sample quantiles corresponding to the given probabilities. The smallest observation corresponds to a probability of 0 and the largest to a probability of 1.

Common quantiles have special names, such as quartiles (four groups), deciles (ten groups), and percentiles (100 groups). The groups created are termed halves, thirds, quarters, etc., though sometimes the terms for the quantile are used for the groups created, rather than for the cut points.

Ex:

```
x=1:10
> x
```

Output:

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

> quantile(x)

Output:

```
   0%    25%    50%    75%   100%
 1.00  3.25  5.50  7.75 10.00
```

> quantile(mtcars$mpg)

Output:

```
    0%     25%     50%     75%    100%
10.400 15.425 19.200 22.800 33.900
```

select quantiles by percent

quantile(mtcars$wt, c(.2, .4, .8))

Output:

```
 20%    40%    80%
 2.349 3.158 3.770
```

variance

variance of weight:

var(mtcars$wt)

Output:

[1] 0.957379

Covariance

get covariance between mpg and gear:

cov(mtcars$mpg, mtcars$gear)

Output:

[1] 2.135685

get covariance all variables:

cov(mtcars[,1:11])

Output:

```
  mpg        cyl       disp        hp        drat
## mpg    36.324103  -9.1723790  -633.09721 -320.732056   2.19506351
## cyl    -9.172379   3.1895161   199.66028  101.931452  -0.66836694
## disp -633.097208 199.6602823 15360.79983 6721.158669 -47.06401915
## hp   -320.732056 101.9314516  6721.15867 4700.866935 -16.45110887
## drat    2.195064  -0.6683669   -47.06402  -16.451109   0.28588135
## wt     -5.116685   1.3673710   107.68420   44.192661  -0.37272073
## qsec    4.509149  -1.8868548   -96.05168  -86.770081   0.08714073
## vs      2.017137  -0.7298387   -44.37762  -24.987903   0.11864919
## am      1.803931  -0.4657258   -36.56401   -8.320565   0.19015121
## gear    2.135685  -0.6491935   -50.80262   -6.358871   0.27598790
## carb   -5.363105   1.5201613    79.06875   83.036290  -0.07840726
##             wt        qsec          vs          am       gear
## mpg   -5.1166847   4.50914919   2.01713710   1.80393145   2.1356855
## cyl    1.3673710  -1.88685484  -0.72983871  -0.46572581  -0.6491935
## disp 107.6842040 -96.05168145 -44.37762097 -36.56401210 -50.8026210
## hp    44.1926613 -86.77008065 -24.98790323  -8.32056452  -6.3588710
## drat  -0.3727207   0.08714073   0.11864919   0.19015121   0.2759879
## wt     0.9573790  -0.30548161  -0.27366129  -0.33810484  -0.4210806
## qsec  -0.3054816   3.19316613   0.67056452  -0.20495968  -0.2804032
## vs    -0.2736613   0.67056452   0.25403226   0.04233871   0.0766129
## am    -0.3381048  -0.20495968   0.04233871   0.24899194   0.2923387
## gear  -0.4210806  -0.28040323   0.07661290   0.29233871   0.5443548
## carb   0.6757903  -1.89411290  -0.46370968   0.04637097   0.3266129
##             carb
## mpg  -5.36310484
## cyl   1.52016129
## disp 79.06875000
## hp   83.03629032
## drat -0.07840726
## wt    0.67579032
## qsec -1.89411290
## vs   -0.46370968
## am    0.04637097
## gear  0.32661290
## carb  2.60887097
```

Correlation

get correlation between mpg and gear:

```
cor(mtcars$mpg, mtcars$gear)
```

Output:

```
[1] 0.4802848
```

get correlation all variables:

```
cor(mtcars[,1:11])
```

Output:

```
          mpg        cyl       disp         hp       drat         wt
## mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684  0.68117191 -0.8676594
## cyl  -0.8521620  1.0000000  0.9020329  0.8324475 -0.69993811  0.7824958
## disp -0.8475514  0.9020329  1.0000000  0.7909486 -0.71021393  0.8879799
## hp   -0.7761684  0.8324475  0.7909486  1.0000000 -0.44875912  0.6587479
## drat  0.6811719 -0.6999381 -0.7102139 -0.4487591  1.00000000 -0.7124406
## wt   -0.8676594  0.7824958  0.8879799  0.6587479 -0.71244065  1.0000000
## qsec  0.4186840 -0.5912421 -0.4336979 -0.7082234  0.09120476 -0.1747159
## vs    0.6640389 -0.8108118 -0.7104159 -0.7230967  0.44027846 -0.5549157
## am    0.5998324 -0.5226070 -0.5912270 -0.2432043  0.71271113 -0.6924953
## gear  0.4802848 -0.4926866 -0.5555692 -0.1257043  0.69961013 -0.5832870
## carb -0.5509251  0.5269883  0.3949769  0.7498125 -0.09078980  0.4276059
##            qsec         vs          am       gear        carb
## mpg   0.41868403  0.6640389  0.59983243  0.4802848 -0.55092507
## cyl  -0.59124207 -0.8108118 -0.52260705 -0.4926866  0.52698829
## disp -0.43369788 -0.7104159 -0.59122704 -0.5555692  0.39497686
## hp   -0.70822339 -0.7230967 -0.24320426 -0.1257043  0.74981247
## drat  0.09120476  0.4402785  0.71271113  0.6996101 -0.09078980
## wt   -0.17471588 -0.5549157 -0.69249526 -0.5832870  0.42760594
## qsec  1.00000000  0.7445354 -0.22986086 -0.2126822 -0.65624923
## vs    0.74453544  1.0000000  0.16834512  0.2060233 -0.56960714
## am   -0.22986086  0.1683451  1.00000000  0.7940588  0.05753435
## gear -0.21268223  0.2060233  0.79405876  1.0000000  0.27407284
## carb -0.65624923 -0.5696071  0.05753435  0.2740728  1.00000000
```

b. Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset.

Subset

It return subsets of vectors, matrices or data frames meeting given conditions.

```
>subset(iris,iris$Sepal.Length>7)
```

Output:

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
103          7.1         3.0          5.9         2.1 virginica
106          7.6         3.0          6.6         2.1 virginica
108          7.3         2.9          6.3         1.8 virginica
110          7.2         3.6          6.1         2.5 virginica
118          7.7         3.8          6.7         2.2 virginica
119          7.7         2.6          6.9         2.3 virginica
123          7.7         2.8          6.7         2.0 virginica
126          7.2         3.2          6.0         1.8 virginica
130          7.2         3.0          5.8         1.6 virginica
131          7.4         2.8          6.1         1.9 virginica
132          7.9         3.8          6.4         2.0 virginica
136          7.7         3.0          6.1         2.3 virginica
```

aggregate ()

aggregate() Function in R Splits the data into subsets, computes summary statistics for each subsets and returns the result in a group by form. Aggregate function in R is similar to group by in SQL. Aggregate() function is useful in performing all the aggregate operations like sum,count,mean, minimum and Maximum. It splits the data into subsets, computes summary statistics for each, and returns the result in a convenient form.

The most basic uses of aggregate involve base functions such as mean and sd. It is indeed one of the most common uses of aggregate to compare the mean or other properties of sample groups.

Aggregate() function in R applied on iris dataset:

Step1:

Load the dataset

```
data(iris)
```

Step 2:

Apply aggregate() to calculate mean, sum

```
agg_mean = aggregate(iris[,1:4], by=list(iris$Species),FUN=mean)
```

Step 3:

Display the result

```
agg_mean
```

Similarly use aggegate() function to calculate:

- Sum:
  agg_sum = aggregate(iris[,1:4], by=list(iris$Species),FUN=sum)
- standard deviation:
  agg_sd = aggregate(iris[,1:4], by=list(iris$Species),FUN=sd)
- Min value
  agg_min = aggregate(iris[,1:4], by=list(iris$Species),FUN=min)
- Max value
  agg_max = aggregate(iris[,1:4], by=list(iris$Species),FUN=max)

3. READING AND WRITING DIFFERENT TYPES OF DATASETS

a. Reading different types of data sets (.txt, .csv) from web and disk and writing in file in specific disk location.

```
# Set current working directory.
setwd("/DH/rprog”)
#Read the file
data <- read.csv("input.csv")
print(data)
```

b. Reading Excel data sheet in R.

```
#Install xlsx Package
install.packages("xlsx")

# Read the file input.xlsx.
data <- read.xlsx("input.xlsx")
print(data)
```

4.VISUALIZATIONS

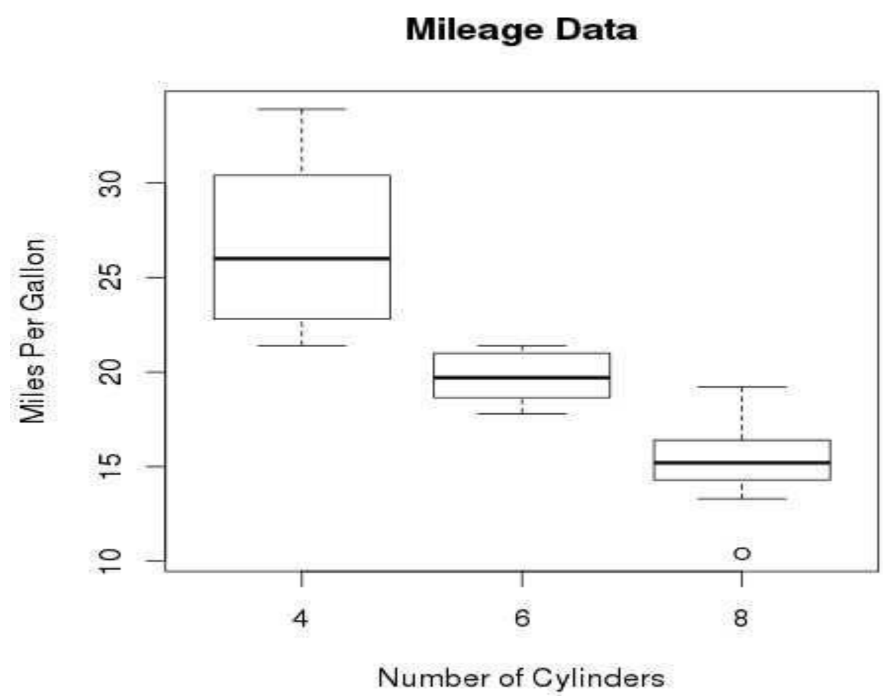
Find the data distributions using box and scatter plot.

```
Install.packages("ggplot2")
Library(ggplot2)
Input <- mtcars[,c('mpg','cyl')]input

Boxplot(mpg ~ cyl, data = mtcars, xlab = "number of cylinders",ylab =
"miles per gallon", main = "mileage data")
Dev.off()
```

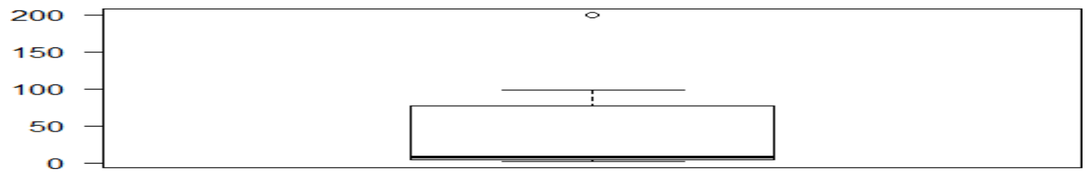
Output :-

	mpg	cyl
Mazda rx4	21.0	6
Mazda rx4 wag	21.0	6
Datsun 710	22.8	4
Hornet 4 drive	21.4	6
Hornet sportabout	18.7	8
Valiant	18.1	6



a. **Find the outliers using plot.**

```
v=c(50,75,100,125,150,175,200)
boxplot(v)
```



b. Plot the histogram, bar chart and pie chart on sample data.

<u>Histogram</u>

library(graphics)

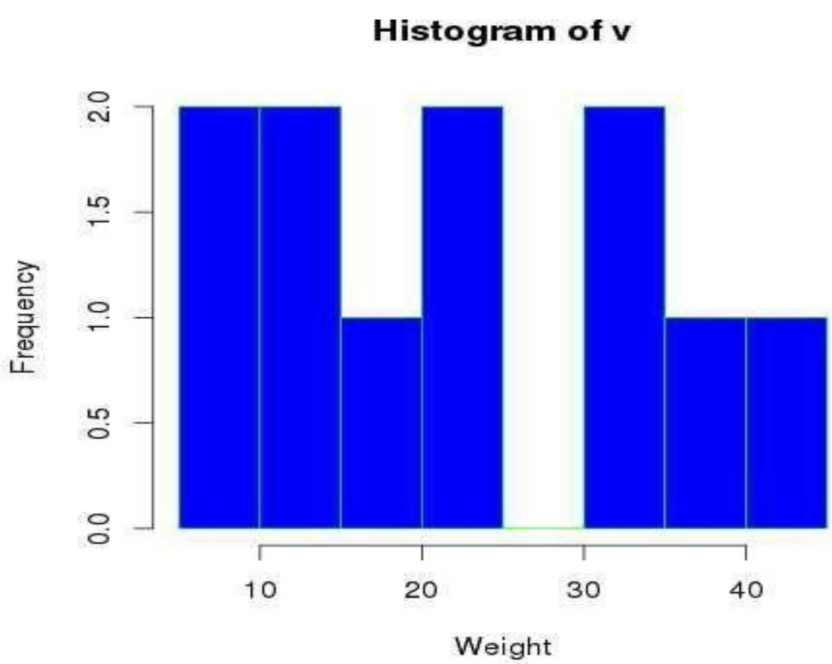
v <- c(9,13,21,8,36,22,12,41,31,33,19)

Create the histogram.

hist(v,xlab = "Weight",col = "blue",border = "green")

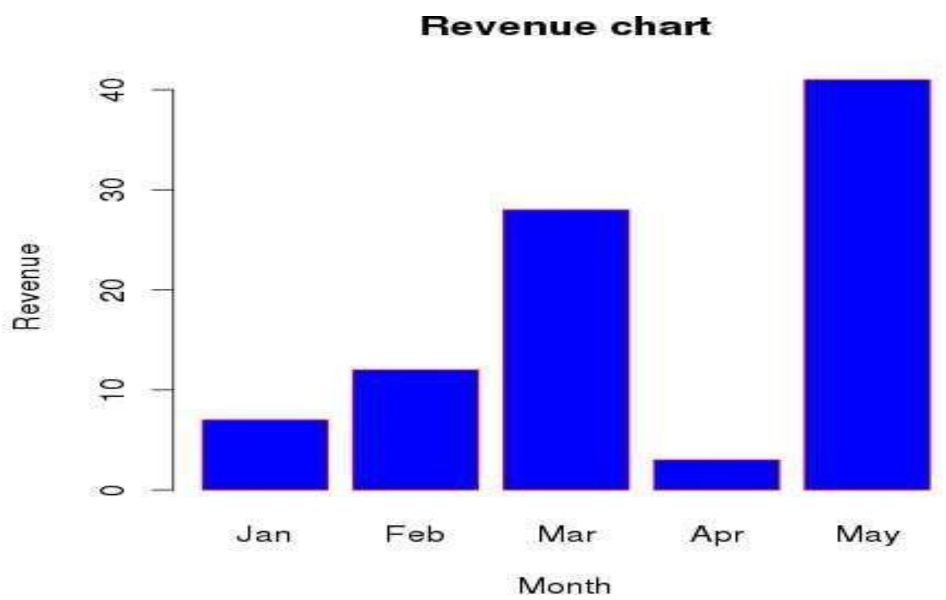
dev.off()

Output:-



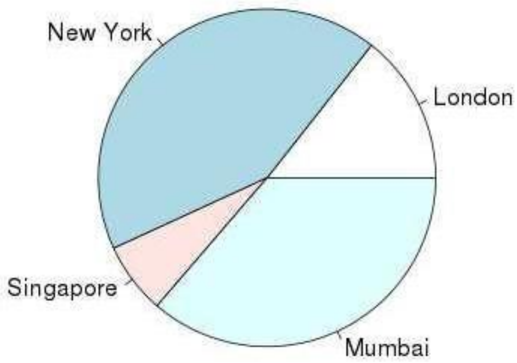
Bar chart

```
library(graphics)
H <- c(7,12,28,3,41)
M <- c("Jan","Feb","Mar","Apr","May")
# Plot the bar chart.
barplot(H, names. arg = M, xlab = "Month", ylab = "Revenue", col = "blue", main = "Revenue chart")
dev.off()
```



Pie Chart

```
library(graphics)
x <- c(21, 62, 10, 53)
labels<- c("London", "NewYork", "Singapore", "Mumbai")
# Plot the Pie chart.
pie(x,labels)
dev.off()
```



5.CORRELATION AND COVARIANCE

a) Find the corelation matrix

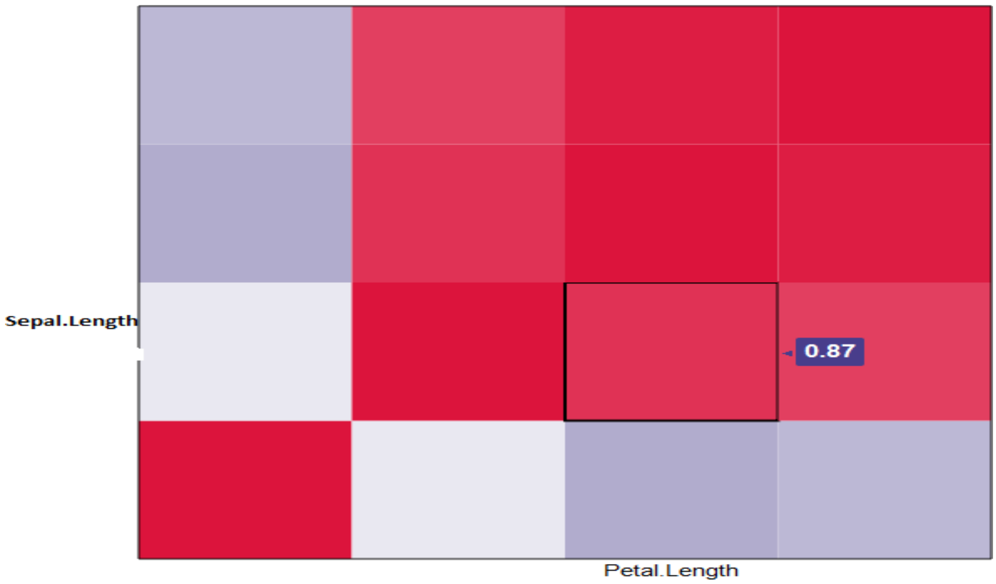
```
d<-data.frame(x1=rnorm(!0),x2=rnorm(10),x3=rnorm(10))
cor(d)
m<-cor(d) #get correlations
library(„corrplot‟)
corrplot(m,method=”square”)
x<-matrix(rnorm(2),,nrow=5,ncol=4)
y<-matrix(rnorm(15),nrow=5,ncol=3)
COR<-cor(x,y)
COR
```

b) Plot the correlation plot on dataset and visualize giving an overview of relationships amongdata on iris data.

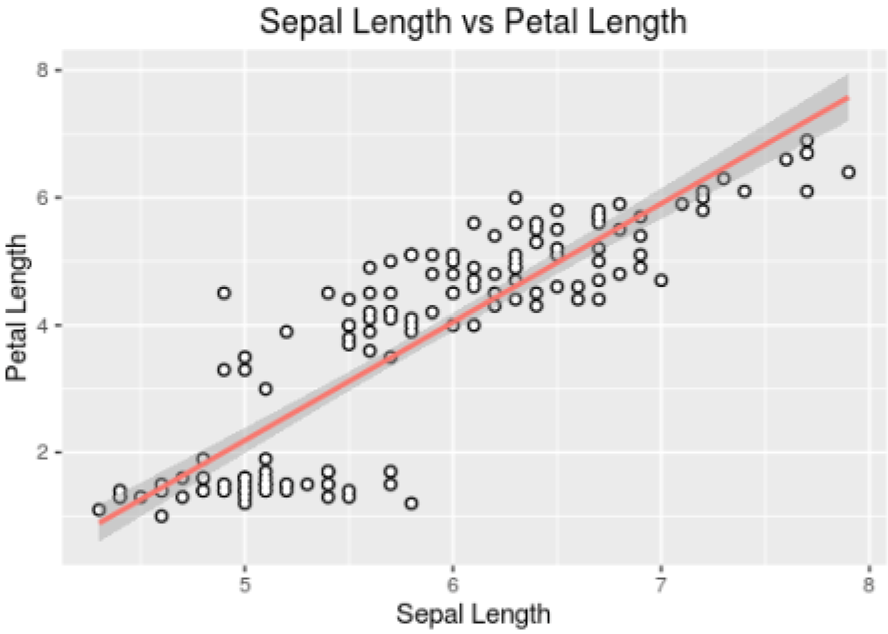
```
Image(x=seq(dim(x)[2])
Y<-seq(dim(y)[2])
Z=COR,xlab=”xcolumn”,ylab=”y
column”)Library(gtlcharts)
Data(iris) Iris$species<-NULL
Iplotcorr(iris,reoder=TRUE)
```

c) Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data.library(ggplot2)data(iris) str(iris)

```
ggplot(data=iris,aes(x=sepal.length,y=petal.length))+geom_point(size=2,colour=”black”)+ge
om_
point(size=1,colour=”white”)+geom_smooth(aes(colour=”black”),method=”lm‟)+ggtitle(“se
pal.l
engthvspetal.length”)+xlab(“sepal.length”)+ylab(“petal.length”)+these(legend.position=”no
ne”)
```



OUTPUT:



6. REGRESSION MODEL

Import a data from web storage. Name the dataset and perform Logistic Regression to find out relation between variables the model. Also check the model is fit or not [require (foreign), require(MASS)]

>mydata$rank<-factor(mydata$rank)
>mylogit<-glm(admit~gre+gpa+rank,data=mydata,family=”binomial”)
>summary(mylogit)

OUTPUT:

```
> mydata$rank <- factor(mydata$rank)
> mylogit <- glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")
> summary(mylogit)

Call:
glm(formula = admit ~ gre + gpa + rank, family = "binomial",
    data = mydata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6268  -0.8662  -0.6388   1.1490   2.0790

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.989979   1.139951  -3.500 0.000465 ***
gre          0.002264   0.001094   2.070 0.038465 *
gpa          0.804038   0.331819   2.423 0.015388 *
rank2       -0.675443   0.316490  -2.134 0.032829 *
rank3       -1.340204   0.345306  -3.881 0.000104 ***
rank4       -1.551464   0.417832  -3.713 0.000205 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 458.52  on 394  degrees of freedom
AIC: 470.52

Number of Fisher Scoring iterations: 4
```

The R package "party" is used to create decision trees.

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

The package "party" has the function ctree() which is used to create and analyze decison tree.

Syntax
The basic syntax for creating a decision tree in R is −
 ctree(formula, data)
Here,
formula is a formula describing the predictor and response variables.
data is the name of the data set used.

Input Data

We will use the R in-built data set named readingSkills to create a decision tree. It describes the score of someone's readingSkills if we know the variables "age", "shoesize", "score" and whether the person is a native speaker or not.
Sample data from dataset readingSkills

Sno	nativeSpeaker	age	shoeSize	score
1	yes	5	24.83189	32.29385
2	yes	6	25.95238	36.63105
3	no	11	30.42170	49.60593
4	yes	7	28.66450	40.28456
5	yes	11	31.88207	55.46085
6	yes	10	30.07843	52.83124

Program:

```
# Load the party package. It will automatically load other dependent packages.
library(party)
# Create the input data frame.
input.dat <- readingSkills[c(1:105),]
# Give the chart file a name.
png(file = "decision_tree.png")
# Create the tree.
  output.tree <- ctree(  nativeSpeaker ~ age + shoeSize + score,   data = input.dat)
# Plot the tree.
plot(output.tree)
# Save the file.
dev.off()
```

Output:

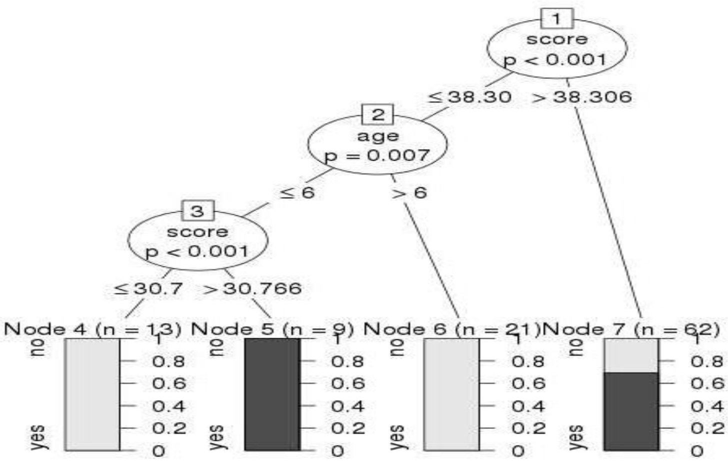
null device
1
Loading required package: methods
Loading required package: grid
Loading required package: mvtnorm
Loading required package: modeltools
Loading required package: stats4
Loading required package: strucchange
Loading required package: zoo

Attaching package: 'zoo'

The following objects are masked from 'package:base':

as.Date, as.Date.numeric

Loading required package: sandwich



8.CLUSTERING MODEL

a. Clustering algorithms for unsupervised classification.

Using iris dataset and K-means Clustering algorithm

```
library(cluster)
> set.seed(20)
> irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)
# nstart = 20. This means that R will try 20 different random starting assignments and then
select the onewith the lowest within cluster variation.
> irisCluster
```

Output:

	Petal.Length	Petal.Width
1	1.462000	0.246000
2	4.269231	1.342308
3	5.595833	2.037500

Clustering vector:

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [42] 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2
 [83] 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3
[124] 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3
```

Within cluster sum of squares by cluster:

[1] 2.02200 13.05769 16.29167
 (between_SS / total_SS = 94.3 %)

Available components:

```
[1] "cluster"    "centers"    "totss"      "withinss"    "tot.withinss"
[6] "betweenss"  "size"       "iter"       "ifault"
```

b. Plot the cluster data using R visualizations

```
> irisCluster$cluster <- as.factor(irisCluster$cluster)
> ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom_point()
```

Output: