Progressive Education Society's
Modern College of Engineering, Pune
MCA Department
A.Y.2023-24
(410908) Data Science Laboratory

****

Class: SY-MCA | Shift / Div.: A | Roll Number: 51061
Name: Shruti Singh | Assignment No: 02 | Date of Implementation: 27/08/2023

****

Q1.Write a R program to create three vectors a,b,c with 3 integers. Combine the three vectors to become a 3×3 matrix where each column represents a vector. Print the content of the matrix.

Solution:

```
> a <- c(1,2,3)
> b <- c(4,5,6)
> c <- c(7,8,9)
> matrix <- cbind(a,b,c)
> matrix
     a b c
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
```

Screenshot :

```
> a <- c(1,2,3)
> b <- c(4,5,6)
> c <- c(7,8,9)
> matrix <- cbind(a,b,c)
> matrix
     a b c
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
>
```

****

Q2 : Write a R program to create a list containing a vector, a matrix and a list and give names to the elements in the list. Access the first and second element of the list.

Solution:

```
>
> list1 <-
list(c(1,2,3,4,5),matrix(c(1,2,3,4,5,6,7,8,9),nrow=3),list(c("red","yello
w","green","blue")))
```

```
> list1
[[1]]
[1] 1 2 3 4 5

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

[[3]]
[[3]][[1]]
[1] "red"    "yellow" "green"  "blue"


> names(list1) <- c("numbers","matrix","colors")
> print("list with named elements : ");list1
[1] "list with named elements : "
$numbers
[1] 1 2 3 4 5

$matrix
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

$colors
$colors[[1]]
[1] "red"    "yellow" "green"  "blue"


> print("first element : ");list1[1]
[1] "first element : "
$numbers
[1] 1 2 3 4 5

> print("second element : ");list1[2]
[1] "second element : "
$matrix
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Screenshot :

```
>
> list1 <- list(c(1,2,3,4,5),matrix(c(1,2,3,4,5,6,7,8,9),nrow=3),list(c("red","yellow","green","blue")))
> list1
[[1]]
[1] 1 2 3 4 5

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

[[3]]
[[3]][[1]]
[1] "red"    "yellow" "green"  "blue"


> names(list1) <- c("numbers","matrix","colors")
> print("list with named elements : ");list1
[1] "list with named elements : "
$numbers
[1] 1 2 3 4 5

$matrix
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

$colors
$colors[[1]]
[1] "red"    "yellow" "green"  "blue"


> print("first element : ");list1[1]
[1] "first element : "
$numbers
[1] 1 2 3 4 5

> print("second element : ");list1[2]
[1] "second element : "
$matrix
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

****

Q3: Write a R program to create an array with three columns, three rows, and two "tables". taking two vectors as input to the array. Print the array.

Solution :

```
> vec1 <- c(1,2,3,4,5)
> vec2 <- c(1,4,9,16,25)
> arr = array(c(vec1,vec2),dim = c(3,3,2))
> arr
, , 1

     [,1] [,2] [,3]
[1,]    1    4    4
[2,]    2    5    9
[3,]    3    1   16

, , 2

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

```
      [1,]   25    3    1
      [2,]    1    4    4
      [3,]    2    5    9
```

Screenshot :

```
> vec1 <- c(1,2,3,4,5)
> vec2 <- c(1,4,9,16,25)
> arr = array(c(vec1,vec2),dim = c(3,3,2))
> arr
, , 1

     [,1] [,2] [,3]
[1,]    1    4    4
[2,]    2    5    9
[3,]    3    1   16

, , 2

     [,1] [,2] [,3]
[1,]   25    3    1
[2,]    1    4    4
[3,]    2    5    9
```

****

Q4 : Write a R program to create a data frame from four given vectors name=c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas') score=c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19)
attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1)
qualify=c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')

```
> names <-
c("Anastasia","Dima","Katherine","James","Emily","Michael","Mathew","Laura","Ke
vin","Jonas")
> score <- c(12.5,9,16.5,12,9,20,14.5,13.5,8,19)
> attempts = c(1,3,2,3,2,3,1,1,2,1)
> qualify <- c("yes","no","yes","no","no","yes","yes","no","no","yes")
> data_frame <- data.frame(names,score,attempts,qualify)
> data_frame
       names score attempts qualify
1  Anastasia  12.5        1     yes
2       Dima   9.0        3      no
3  Katherine  16.5        2     yes
4      James  12.0        3      no
5      Emily   9.0        2      no
6    Michael  20.0        3     yes
7     Mathew  14.5        1     yes
```

```
8       Laura  13.5        1       no
9       Kevin   8.0        2       no
10      Jonas  19.0        1      yes
```

Screenshot :

```
> names <- c("Anastasia","Dima","Katherine","James","Emily","Michael","Mathew","Laura","Kevin","Jonas")
> score <- c(12.5,9,16.5,12,9,20,14.5,13.5,8,19)
> attempts = c(1,3,2,3,2,3,1,1,2,1)
> qualify <- c("yes","no","yes","no","no","yes","yes","no","no","yes")
> data_frame <- data.frame(names,score,attempts,qualify)
> data_frame
       names score attempts qualify
1  Anastasia  12.5        1     yes
2       Dima   9.0        3      no
3  Katherine  16.5        2     yes
4      James  12.0        3      no
5      Emily   9.0        2      no
6    Michael  20.0        3     yes
7     Mathew  14.5        1     yes
8      Laura  13.5        1      no
9      Kevin   8.0        2      no
10     Jonas  19.0        1     yes
>
```

****

Q5 : Write a R program to create a factor corresponding to height of women data set, which contains height and weights for a sample of women

Solution :

```
women_data <- data.frame(
  Height = c(160, 165, 170, 175, 162, 168, 155, 160, 172, 178),
  Weight = c(55, 60, 65, 70, 58, 63, 52, 56, 68, 75)
)

# Define height categories (bins)
height_breaks <- c(150, 160, 170, 180)

# Create a factor variable for height based on the defined breaks
women_data$Height_Category <- cut(women_data$Height, breaks =
height_breaks, labels = c("Short", "Average", "Tall"))

# Display the updated data frame with the height category
print(women_data)
```

Screenshot :

```
> # Create a factor variable for height based on the defined breaks
> women_data$Height_Category <- cut(women_data$Height, breaks = height_breaks, labels = c("Short", "Average", "Tall"))
>
> # Display the updated data frame with the height category
> print(women_data)
   Height Weight Height_Category
1     160     55           Short
2     165     60         Average
3     170     65         Average
4     175     70            Tall
5     162     58         Average
6     168     63         Average
7     155     52           Short
8     160     56           Short
9     172     68            Tall
10    178     75            Tall
```

Second way:

```
data = women
print("Women data set of height and weights:")
print(data)
height_f = cut(women$height,3)
print("Factor corresponding to height:")
print(table(height_f))
```

```
> height_f = cut(women$height,3)
> print("Factor corresponding to height:")
[1] "Factor corresponding to height:"
> print(table(height_f))
height_f
  (58,62.7] (62.7,67.3]   (67.3,72]
          5           5           5
>
```

****

Q6: Use R to create the following two matrices and do the indicated matrix multiplication.

7	9	12	x	1	7	12	19
2	4	13		2	8	13	20
				3	9	14	21

What is the resulting matrix?

Solution:

```
mat1 <- matrix( c(7, 9, 12, 2, 4, 13), nrow = 2, ncol = 3)
mat2 <- matrix( c(1, 7, 12, 19, 2, 8, 13,20, 3, 9, 14,21), nrow = 3, ncol = 4)
mat1
mat2

mat3 <- mat1%*%mat2
```

```
print("multiplication of two matrices:")
print(mat3)
```

Screenshot:

```
> mat1 <- matrix( c(7, 9, 12, 2, 4, 13), nrow = 2, ncol = 3)
> mat2 <- matrix( c(1, 7, 12, 19, 2, 8, 13,20, 3, 9, 14,21), nrow = 3, ncol = 4)
> mat1
     [,1] [,2] [,3]
[1,]    7   12    4
[2,]    9    2   13
> mat2
     [,1] [,2] [,3] [,4]
[1,]    1   19   13    9
[2,]    7    2   20   14
[3,]   12    8    3   21
>
> mat3 <- mat1%*%mat2
>
> print("multiplication of two matrices:")
[1] "multiplication of two matrices:"
> print(mat3)
     [,1] [,2] [,3] [,4]
[1,]  139  189  343  315
[2,]  179  279  196  382
```

****

Q7: WAP to Print the Fibonacci Sequence.

Solution:

```
print_fibonacci <- function(n) {
  a <- 0
  b <- 1

  cat("Fibonacci Sequence:")
  for (i in 1:n) {
    cat(a, " ")
    next_num <- a + b
    a <- b
    b <- next_num
  }
}

# Example usage
number_of_terms <- 10
print_fibonacci(number_of_terms)
```

Solution:

```
> print_fibonacci <- function(n) {
+   a <- 0
+   b <- 1
+
+   cat("Fibonacci Sequence:")
+   for (i in 1:n) {
+     cat(a, " ")
+     next_num <- a + b
+     a <- b
+     b <- next_num
+   }
+ }
> number_of_terms <- 10
> print_fibonacci(number_of_terms)
Fibonacci Sequence:0  1  1  2  3  5  8  13  21  34
>
```

****

Q8: WAP to import data in R from csv, excel, txt file.

Solution:

```
# Read CSV into DataFrame
read_csv = read.csv("D:\\PESMCOE\\Data Science\\Assign
2\\username_onboarding.csv")
print(read_csv)

# Read excel into R
#install.packages("Rcpp")
#install.packages("readxl")
library(readxl)
read_exc = read_excel("D:\\PESMCOE\\Data Science\\Assign
2\\Financial_Sample.xlsx")
print(read_exc)

# Read text file into R
x<-read.csv("D:\\PESMCOE\\Data Science\\Assign 2\\products.txt")
print(x)
```

****

Q9: WAP to export data from R to CSV, Excel, Text File and Google drive.

Solution:

```
#Exporting to csv file
data_info <- data.frame( product = c("computer", "monitor", "keyboard", "printer", "tablet"),
                         price = c(800, 450, 100, 150, 300))
write.csv(data_info,"C:\\Users\\Shruti Singh\\Downloads\\exp_products.csv", row.names =
FALSE )
```

Screenshot:


```
#Exporting to excel file
#install.packages("writexl")
library("writexl")
excel_info <- data.frame( Name = c("Jon", "Bill", "Maria", "Ben", "Tina"),
                          Age = c(23, 41, 32, 58, 26))
write_xlsx(excel_info,"C:\\Users\\Shruti Singh\\Downloads\\exp_info.xlsx" )

#Exporting to text file
text_info <- data.frame( Name = c("Jon", "Bill", "Maria", "Ben", "Tina"),
                         Age = c(23, 41, 32, 58, 26))
write.table(text_info,"C:\\Users\\Shruti Singh\\Downloads\\exp_text.txt", quote=FALSE)
```

Screenshot:

```
Name Age
1 Jon 23
2 Bill 41
3 Maria 32
4 Ben 58
5 Tina 26
```

****

Q10: Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two vectors. Print the second row of the second matrix of the array and the element in the 3rd row and 3rd column of the 1st matrix.

Solution:

```
print("Two vectors of different lengths:")
v1 =  c(1,3,4,5)
v2 =  c(10,11,12,13,14,15)
print(v1)
print(v2)
result = array(c(v1,v2),dim = c(3,3,2))
print("New array:")
print(result)
print("The second row of the second matrix of the array:")
print(result[2,,2])
print("The element in the 3rd row and 3rd column of the 1st matrix:")
print(result[3,3,1])
```

Screenshot:

```
> print("Two vectors of different lengths:")
[1] "Two vectors of different lengths:"
> v1 =  c(1,3,4,5)
> v2 =  c(10,11,12,13,14,15)
> print(v1)
[1] 1 3 4 5
> print(v2)
[1] 10 11 12 13 14 15
> result = array(c(v1,v2),dim = c(3,3,2))
> print("New array:")
[1] "New array:"
> print(result)
, , 1

     [,1] [,2] [,3]
[1,]    1    5   12
[2,]    3   10   13
[3,]    4   11   14

, , 2

     [,1] [,2] [,3]
[1,]   15    4   11
[2,]    1    5   12
[3,]    3   10   13

> print("The second row of the second matrix of the array:")
[1] "The second row of the second matrix of the array:"
> print(result[2,,2])
[1]  1  5 12
> print("The element in the 3rd row and 3rd column of the 1st matrix:")
[1] "The element in the 3rd row and 3rd column of the 1st matrix:"
> print(result[3,3,1])
[1] 14
```

****

Q11: VAT has different rate according to the product purchased. Imagine we have three different kind of products with different VAT applied:

Categories	Product	VAT
A	Book, magazine, newspaper, etc...	8%
B	Vegetable, meat, beverage, etc...	10%
C	Tee-shirt, jean, pant, etc...	20%

Write a chain to apply the correct VAT rate to the product customer bought and calculate a price.

Solution:

```
vat_calculation <- function(x,p){

  if (category =='A'){
    cat('A vat rate of 8% is applied.','The total price is',p *1.08)
  } else if (category =='B'){
    cat('A vat rate of 10% is applied.','The total price is',p *1.10)
  } else if(category== 'C'){
    cat('A vat rate of 20% is applied.','The total price is',p *1.20)
  }else{
    cat('Invalid choice')
  }
}
```

```
category <- 'A'
price <- 100
vat_calculation(category,price)
```

Screenshot:

```
vat_calculation <- function(x,p){

  if (category =='A'){
    cat('A vat rate of 8% is applied.','The total price is',p *1.08)
  } else if (category =='B'){
    cat('A vat rate of 10% is applied.','The total price is',p *1.10)
  } else if(category== 'C'){
    cat('A vat rate of 20% is applied.','The total price is',p *1.20)
  }else{
    cat('Invalid choice')
  }
}

category <- 'A'
price <- 100
vat_calculation(category,price)
vat rate of 8% is applied. The total price is 108
```

****

Q12: A cloth showroom has announced the following seasonal discounts on purchase of items. Write a R program using switch and if statement to compute the net amount paid by a customer.

Purchase Amount	Discount	
	Mill Cloth	Handloom Items
0-100	-	5%
101-200	5%	7.5%
201-300	7.5%	10%
301 and Above	10%	15.0%

Solution:

```
amt <- 500

if (amt>=0 && amt<=100){
  print("0-100")
  print(amt * 1.05 )
}else if( amt<=200 && amt >=101){
  print("101-200")
  print((amt *1.05) + (amt * 1.75))
}else if(amt>=201 && amt<=300){
  print("201-300")
```

```
  print((amt *1.75) + (amt * 1.10))
}else if(amt>=301){
  print("301 above")
  print((amt *1.10) + (amt * 1.15))
}else{
  print("Invalid amount")
}
```

Screenshot:

```
> amt <- 500
>
> if (amt>=0 && amt<=100){
+   print("0-100")
+   print(amt * 1.05 )
+ }else if( amt<=200 && amt >=101){
+   print("101-200")
+   print((amt *1.05) + (amt * 1.75))
+ }else if(amt>=201 && amt<=300){
+   print("201-300")
+   print((amt *1.75) + (amt * 1.10))
+ }else if(amt>=301){
+   print("301 above")
+   print((amt *1.10) + (amt * 1.15))
+ }else{
+   print("Invalid amount")
+ }
[1] "301 above"
[1] 1125
>
```

****

Q13: Find Sum of Series $1^2+2^2+3^2+.....+n^2$.

Solution:

```
sum_of_series <- function(n){

  sum_tot <- 0
  for (i in 1:n){
    sum_tot<- sum_tot+ i^2
  }
  return(sum_tot)

}

n <- 10
result <- sum_of_series(n)
result
```

Screenshot:

```
> sum_of_series <- function(n){
+
+   sum_tot <- 0
+   for (i in 1:n){
+     sum_tot<- sum_tot+ i^2
+   }
+   return(sum_tot)
+
+ }
>
> n <- 10
> result <- sum_of_series(n)
> result
[1] 385
>
```

****

Q14: Write a R program to print the numbers from 1 to 100 and print "Fizz" for multiples of 3, print "Buzz" for multiples of 5, and print "FizzBuzz" for multiples of both.

Solution:

```
for (n in 1:100) {
 if (n %% 3 == 0 & n %% 5 == 0) {print("FizzBuzz")}
 else if (n %% 3 == 0) {print("Fizz")}
 else if (n %% 5 == 0) {print("Buzz")}
 else print(n)
}
```

Screenshot:

```
> for (n in 1:100) {
+   if (n %% 3 == 0 & n %% 5 == 0) {print("FizzBuzz")}
+   else if (n %% 3 == 0) {print("Fizz")}
+   else if (n %% 5 == 0) {print("Buzz")}
+   else print(n)
+ }
[1] 1
[1] 2
[1] "Fizz"
[1] 4
[1] "Buzz"
[1] "Fizz"
[1] 7
[1] 8
[1] "Fizz"
[1] "Buzz"
[1] 11
[1] "Fizz"
[1] 13
[1] 14
[1] "FizzBuzz"
[1] 16
[1] 17
[1] "Fizz"
[1] 19
[1] "Buzz"
[1] "Fizz"
[1] 22
[1] 23
[1] "Fizz"
[1] "Buzz"
[1] 26
```

****

Q15: Write a R Program to find the sum of digits of a number reducing it to one digit using repeat loop.

Solution:

Using repeat loop :

```
sum_of_digits_repeat <- function(n){

 x<- n
 repeat {

  if(x<9)
  {
   return(x)
  }
  res <- 0
  t <- x
```

```
    repeat{

      if(t==0){
        break
      }

      res <- res+t%%10
      t <- t%/%10

    }
    x <- res
  }
}

cat("sum of digits to one digit using repeat loop :
",sum_of_digits_repeat(98765678912398))
```

Screenshot:

```
> sum_of_digits_repeat <- function(n){
+
+   x<- n
+   repeat {
+
+     if(x<9)
+     {
+       return(x)
+     }
+     res <- 0
+     t <- x
+     repeat{
+
+       if(t==0){
+         break
+       }
+
+       res <- res+t%%10
+       t <- t%/%10
+
+     }
+     x <- res
+   }
+ }
>
> cat("sum of digits to one digit using repeat loop : ",sum_of_digits_repeat(98765678912398))
sum of digits to one digit using repeat loop :  7
```

Using while loop:

```
sum_of_digits <- function(n){
  x <- n
  while(x>9)
  {
```

```
    res <- 0
    t <- x
    while (t!=0) {
      res <- res+t%%10
      t<- t%/%10
    }
    x<-res
  }
  return(x)
}

cat("Sum of digits of a number reducing it to one digit :
“,sum_of_digits(98765678912398))
```

Screenshot:

```
> sum_of_digits <- function(n){
+   x <- n
+   while(x>9)
+   {
+     res <- 0
+     t <- x
+     while (t!=0) {
+       res <- res+t%%10
+       t<- t%/%10
+     }
+     x<-res
+   }
+   return(x)
+ }
>
>
> cat("Sum of digits of a number reducing it to one digit : ",sum_of_digits(98765678912398))
Sum of digits of a number reducing it to one digit :  7
>
```

****

Progressive Education Society's
Modern College of Engineering, Pune
MCA Department
A.Y.2023-24
(410908) Data Science Laboratory

****

Class: SY-MCA | Shift / Div.: A | Roll Number: 51061
Name: Shruti Singh | Assignment No: 03 | Date of Implementation: 27/08/2023

****

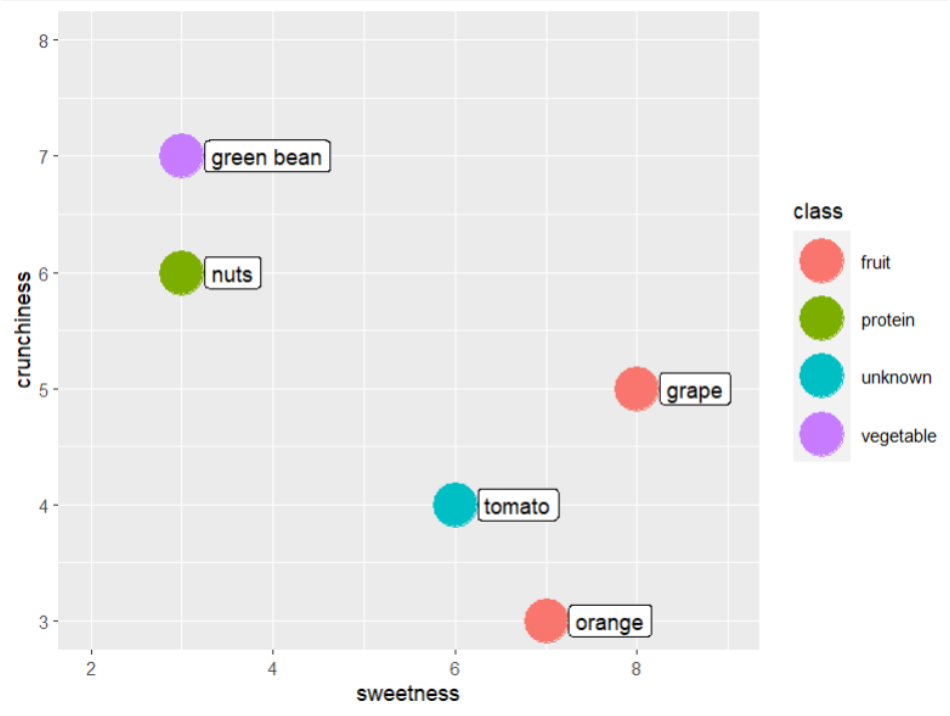
Q1.We have four things grape, green bean, nuts and orange with two characteristics sweetness (8, 3, 3, 7) and Crunchiness (5, 7, 6, 3). Among them two are fruits, one is protein and one is vegetable. Suppose we wanted to classify tomato into one of the classes. Is tomato a fruit, vegetable or protein? Tomato has the following characteristics: sweetness = 6, crunchiness = 4. Let's add Carrot with characteristics sweetness = 4 and crunchiness = 9 keep k=1. Try for k=4 also.

Solution:

```
things <- data.frame(ingredient =  c("grape", "green bean" , "nuts" , "orange"),
                     sweetness = c(8,3,3,7),
                     crunchiness = c(5,7,6,3),
                     class = c("fruit", "vegetable", "protein", "fruit"))

unknown <- data.frame(ingredient =  "tomato",
                      sweetness = 6,
                      crunchiness = 4,
                      class="unknown")

library(dplyr)
library(descr)
library(ggplot2)
ggplot(bind_rows(things, unknown)) +
  geom_point(aes(x=sweetness, y=crunchiness, color=class),size=10) +
  geom_label(aes(x=sweetness, y=crunchiness, label=ingredient), hjust = 0, nudge_x = 0.25)+
  xlim(2,9) + ylim(3,8)

library(class) #contains knn function
pred <- knn(select(things, sweetness, crunchiness),
            select(unknown,sweetness, crunchiness), things$class, k=1)
pred

unknown <- data.frame(ingredient =  c("tomato", "carrot"),
```

```
                sweetness = c(6,4),
                crunchiness = c(4,9),
                class=c("unknown", "unknown"))
unknown

pred <- knn(select(things, sweetness, crunchiness),
        select(unknown,sweetness, crunchiness), things$class, k=1)
pred

pred <- knn(select(things, sweetness, crunchiness),
        select(unknown,sweetness, crunchiness), things$class, k=4)
pred
```

Screenshot:



For k=1 :

```
> pred <- knn(select(things, sweetness, crunchiness),
+             select(unknown,sweetness, crunchiness), things$class, k=1)
> pred
[1] fruit     vegetable
Levels: fruit protein vegetable
```

For k=4 :

```
> pred <- knn(select(things, sweetness, crunchiness),
+             select(unknown,sweetness, crunchiness), things$class, k=4)
> pred
[1] fruit fruit
Levels: fruit protein vegetable
```

****

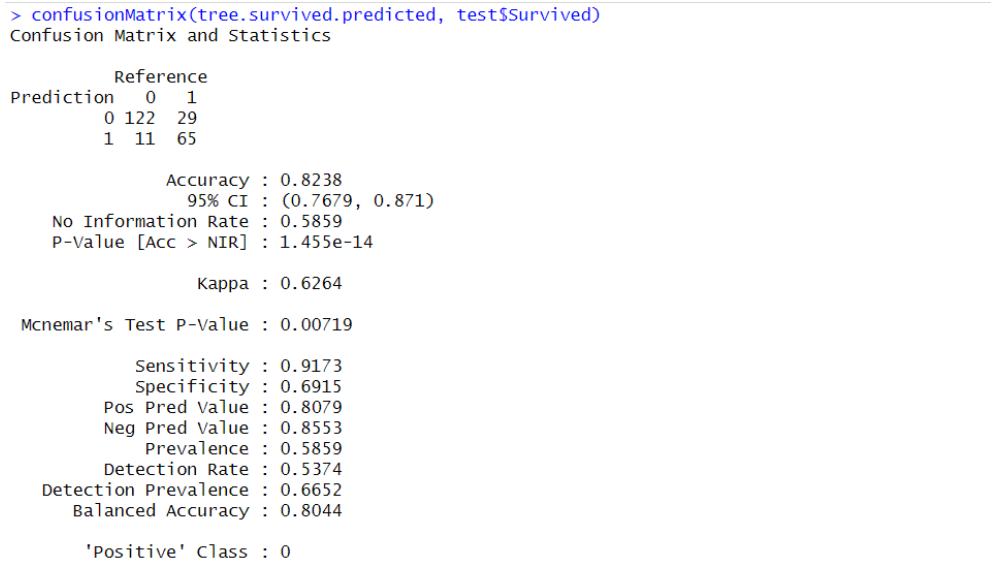
Q2: Using Titanic.CSV file, predict which people are more likely to survive after the collision with the iceberg using Decision Trees.

Solution:

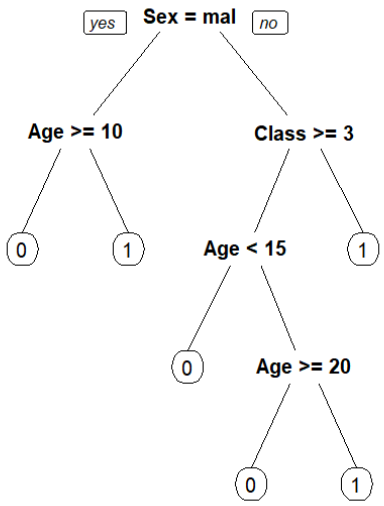
```
library(caret)
library(FSelector)
library(rpart)
library(rpart.plot)
library(dplyr)
library(xlsx)
library(data.tree)
library(caTools)


################### FIRST METHOD ##############################


df <- read.xlsx("C:\\Users\\Shruti Singh\\Desktop\\SY\\Data
science\\Rstudio\\Assignment 3\\Titanic.xlsx",sheetIndex = 1 )

df

summary(df)

#Titanic <- Titanic[,c('Class','Age','Sex','Survived')]

df <- select(df, Survived, Class, Sex, Age)
df <- na.omit(df)
df
df <- mutate(df, Survived = factor(Survived), Class = as.numeric(Class), Age =
as.numeric(Age))

set.seed(123)
sample = sample.split(df$Survived, SplitRatio = .70)
train = subset(df, sample==TRUE)
test = subset(df, sample == FALSE)
```

```
#Training the decision tree classifier

tree <- rpart(Survived ~., data = train)

#Predictions

tree.survived.predicted <- predict(tree, test, type = 'class')

#Confusion Matrix for evaluating the model

confusionMatrix(tree.survived.predicted, test$Survived)

#Visualizing Decision Tree

prp(tree)
```

SCREENSHOT:



```
> confusionMatrix(tree.survived.predicted, test$Survived)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 122  29
         1  11  65

               Accuracy : 0.8238
                 95% CI : (0.7679, 0.871)
    No Information Rate : 0.5859
    P-Value [Acc > NIR] : 1.455e-14

                  Kappa : 0.6264

 Mcnemar's Test P-Value : 0.00719

            Sensitivity : 0.9173
            Specificity : 0.6915
         Pos Pred Value : 0.8079
         Neg Pred Value : 0.8553
             Prevalence : 0.5859
         Detection Rate : 0.5374
   Detection Prevalence : 0.6652
      Balanced Accuracy : 0.8044

       'Positive' Class : 0
```



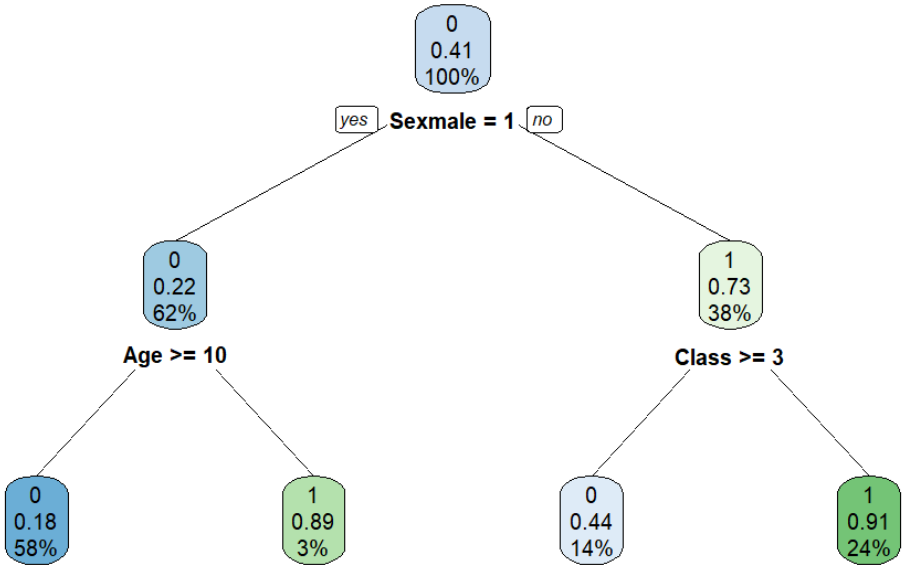
################### SECOND METHOD ##############################

```
df <- read.xlsx("C:\\Users\\Shruti Singh\\Desktop\\SY\\Data
science\\Rstudio\\Assignment 3\\Titanic.xlsx",sheetIndex = 1 )

df

summary(df)

#Titanic <- Titanic[,c('Class','Age','Sex','Survived')]

df <- select(df, Survived, Class, Sex, Age)
df

df <- mutate(df, Survived = factor(Survived), Class = as.numeric(Class), Age =
as.numeric(Age))
df <- na.omit(df)
set.seed(123)
sample = sample.split(df$Survived, SplitRatio = .70)
train = subset(df, sample==TRUE)
test = subset(df, sample == FALSE)
ctrl <- trainControl(method = 'cv', number=10)

fit.cv <- train(Survived ~ ., data = train, method = "rpart",
                trControl = ctrl,
                tuneLength = 30)
```

```
pred <- predict(fit.cv,test)
confusionMatrix(table(test[,"Survived"], pred))
print(fit.cv)
plot(fit.cv)
plot(fit.cv$finalModel)
text(fit.cv$finalModel)


library(rpart.plot)
rpart.plot(fit.cv$finalModel)
rpart.plot(fit.cv$finalModel, fallen.leaves = FALSE)
```

SCREENSHOT:

```
> confusionMatrix(table(test[,"Survived"], pred))
Confusion Matrix and Statistics

   pred
      0   1
  0 125   8
  1  33  61

               Accuracy : 0.8194
                 95% CI : (0.7631, 0.8672)
    No Information Rate : 0.696
    P-Value [Acc > NIR] : 1.651e-05

                  Kappa : 0.6127

 Mcnemar's Test P-Value : 0.0001781

            Sensitivity : 0.7911
            Specificity : 0.8841
         Pos Pred Value : 0.9398
         Neg Pred Value : 0.6489
             Prevalence : 0.6960
         Detection Rate : 0.5507
   Detection Prevalence : 0.5859
      Balanced Accuracy : 0.8376

       'Positive' Class : 0
```



****

Q3: Load the tissue_gene_expression dataset. Run a k-means clustering on the data with K=7. Make a table comparing the identified clusters to the actual tissue types. Run the algorithm several times to see how the answer changes.

SOLUTION:

```
install.packages("dslabs")
library(dslabs)
data("tissue_gene_expression")
df <- data.frame(tissue_gene_expression)
df
cl <- kmeans(tissue_gene_expression$x, centers = 7)
table(cl$cluster, tissue_gene_expression$y)
```

SCREENSHOT:

ITERATION 1:

```
> cl <- kmeans(tissue_gene_expression$x, centers = 7)
> table(cl$cluster, tissue_gene_expression$y)

  cerebellum colon endometrium hippocampus kidney liver placenta
1          0     0           0           0      0    24        0
2          5     0           0          31      0     0        0
3          2     0           0           0      2     2        0
4          0    34           1           0      0     0        6
5         31     0           0           0      0     0        0
6          0     0          14           0     15     0        0
7          0     0           0           0     22     0        0
```

ITERATION 2:

	cerebellum	colon	endometrium	hippocampus	kidney	liver	placenta
1	0	0	15	0	7	0	0
2	0	0	0	0	30	0	0
3	0	0	0	0	0	24	0
4	36	0	0	31	0	0	0
5	0	34	0	0	0	0	0
6	2	0	0	0	2	2	0
7	0	0	0	0	0	0	6

ITERATION 3:

	cerebellum	colon	endometrium	hippocampus	kidney	liver	placenta
1	0	0	15	0	7	0	6
2	0	34	0	0	0	0	0
3	0	0	0	0	13	0	0
4	0	0	0	0	10	0	0
5	0	0	0	0	0	26	0
6	38	0	0	31	0	0	0
7	0	0	0	0	9	0	0

ITERATION 4:

	cerebellum	colon	endometrium	hippocampus	kidney	liver	placenta
1	0	0	0	31	0	0	0
2	5	0	0	0	0	0	0
3	24	0	0	0	0	0	0
4	0	34	15	0	1	24	6
5	7	0	0	0	0	0	0
6	2	0	0	0	2	2	0
7	0	0	0	0	36	0	0

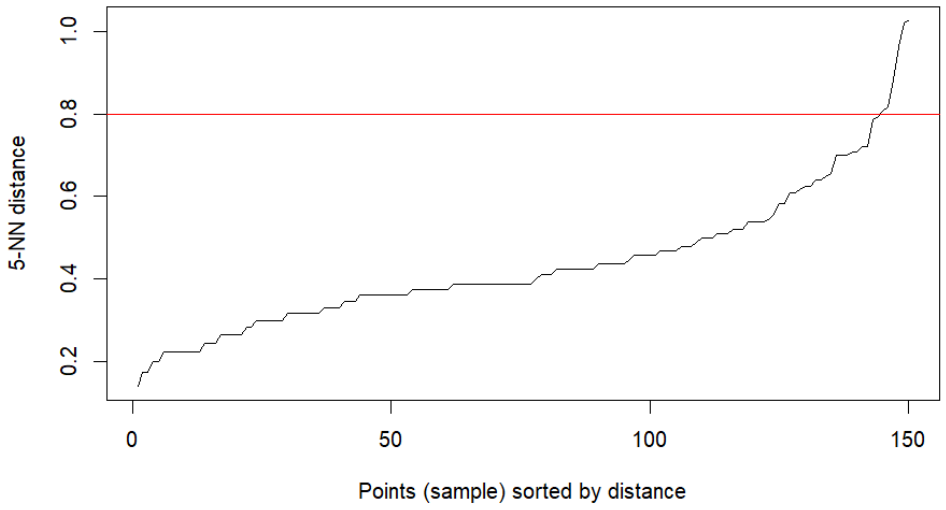
****

Q4: Plot the distribution of distances between data points and their fifth nearest neighbors using the kNNdistplot function from the dbscan package. Examine the plot and find a tentative threshold at which distances start increasing quickly. On the same plot, draw a horizontal line at the level of the threshold (use Iris dataset).

SOLUTION:

```
install.packages("dbscan")

library(dbscan)

df <- iris[, -ncol(iris)]
df
kNNdistplot(df, k = 5)
abline(h = 0.8, col = "red")
```

SCREENSHOT:



****

Progressive Education Society's
Modern College of Engineering, Pune
MCA Department
A.Y.2023-24
(410908) Data Science Laboratory

****

Class: SY-MCA | Shift / Div.: A | Roll Number: 52061
Name: Shruti Singh | Assignment No: 04 | Date of Implementation: 06/11/2023

****

Q1.Use the Apriori algorithm on the grocery dataset with minimum support to 0.001 and minimum confidence of 0.8 indicate the top 5 association rules that are generated and highlight the strong ones, sort them by confidence.

Program:

```
library(arules)
library(arulesViz)
library(datasets)

# Load the data set
data(Groceries)

# Create an item frequency plot for the top 20 items
itemFrequencyPlot(Groceries,topN=20,type="absolute")

rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))

options(digits=2)
inspect(rules[1:5])

rules<-sort(rules, by="confidence", decreasing=TRUE)

rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8,maxlen=3))

subset.matrix <- is.subset(rules, rules)
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
redundant <- colSums(subset.matrix, na.rm=T) >= 1
rules.pruned <- rules[!redundant]
rules<-rules.pruned

# Targeting Items
```

```
rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.08),
               appearance = list(default="lhs",rhs="whole milk"),
               control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="confidence")
inspect(rules[1:5])

rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.15,minlen=2),
               appearance = list(default="rhs",lhs="whole milk"),
               control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="confidence")
inspect(rules[1:5])

plot(rules,method="graph")
```

Output:

```
   lhs             rhs                support confidence coverage lift count
[1] {whole milk} => {other vegetables} 0.075   0.29       0.26     1.5  736
[2] {whole milk} => {rolls/buns}       0.057   0.22       0.26     1.2  557
[3] {whole milk} => {yogurt}           0.056   0.22       0.26     1.6  551
[4] {whole milk} => {root vegetables}  0.049   0.19       0.26     1.8  481
[5] {whole milk} => {tropical fruit}   0.042   0.17       0.26     1.6  416
```

```
> rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.15,minlen=2),
+                appearance = list(default="rhs",lhs="whole milk"),
+                control = list(verbose=F))
> rules<-sort(rules, decreasing=TRUE,by="confidence")
> inspect(rules[1:5])
  lhs             rhs                support confidence coverage lift count
1] {whole milk} => {other vegetables} 0.075   0.29       0.26     1.5  736
2] {whole milk} => {rolls/buns}       0.057   0.22       0.26     1.2  557
3] {whole milk} => {yogurt}           0.056   0.22       0.26     1.6  551
4] {whole milk} => {root vegetables}  0.049   0.19       0.26     1.8  481
5] {whole milk} => {tropical fruit}   0.042   0.17       0.26     1.6  416
```

****

Q2 : Use the Eclat algorithm on given Market Basket Dataset and predict the items which are bought frequently.

Program:

```
library(arules)
rules = eclat(data = Groceries, parameter = list(support = 0.004, minlen = 3))

print('the items which are bought frequently:')

inspect(sort(rules, by = 'support')[1:10])
```

Output:

```
 [1] "the items which are bought frequently:"
>
> inspect(sort(rules, by = 'support')[1:10])
     items                                              support count
[1]  {root vegetables, other vegetables, whole milk}    0.023   228
[2]  {other vegetables, whole milk, yogurt}             0.022   219
[3]  {other vegetables, whole milk, rolls/buns}         0.018   176
[4]  {tropical fruit, other vegetables, whole milk}     0.017   168
[5]  {whole milk, yogurt, rolls/buns}                   0.016   153
[6]  {tropical fruit, whole milk, yogurt}               0.015   149
[7]  {other vegetables, whole milk, whipped/sour cream} 0.015   144
[8]  {root vegetables, whole milk, yogurt}              0.015   143
[9]  {other vegetables, whole milk, soda}               0.014   137
[10] {pip fruit, other vegetables, whole milk}          0.014   133
```

```
 [1] "the items which are bought frequently:"
>
> inspect(sort(rules, by = 'support')[1:10])
     items                                              support count
[1]  {root vegetables, other vegetables, whole milk}    0.023   228
[2]  {other vegetables, whole milk, yogurt}             0.022   219
[3]  {other vegetables, whole milk, rolls/buns}         0.018   176
[4]  {tropical fruit, other vegetables, whole milk}     0.017   168
[5]  {whole milk, yogurt, rolls/buns}                   0.016   153
[6]  {tropical fruit, whole milk, yogurt}               0.015   149
[7]  {other vegetables, whole milk, whipped/sour cream} 0.015   144
[8]  {root vegetables, whole milk, yogurt}              0.015   143
[9]  {other vegetables, whole milk, soda}               0.014   137
[10] {pip fruit, other vegetables, whole milk}          0.014   133
```

****

Progressive Education Society's

Modern College of Engineering, Pune
MCA Department
A.Y.2023-24
(410908) Data Science Laboratory

****

Class: SY-MCA | Shift / Div.: A | Roll Number: 52061
Name: Shruti Singh | Assignment No: 05 | Date of Implementation: 06/11/2023

****

Q1.Find the mean, median, Mode, Range, Interquartile Range IQR and normal distribution of the physical-fitness scores. Third graders at Roth Elementary School were given a physical-fitness test. Their scores were:
a. 12 22 6 9 2 9 5 9 3 5 16 1 22 18
b. 6 12 21 23 9 10 24 21 17 11 18 19 17 5
c. 14 16 19 19 18 3 4 21 16 20 15 14 17 4
d. 5 22 12 15 18 20 8 10 13 20 6 9 2 17
e. 15 9 4 15 14 19 3 24

Program:

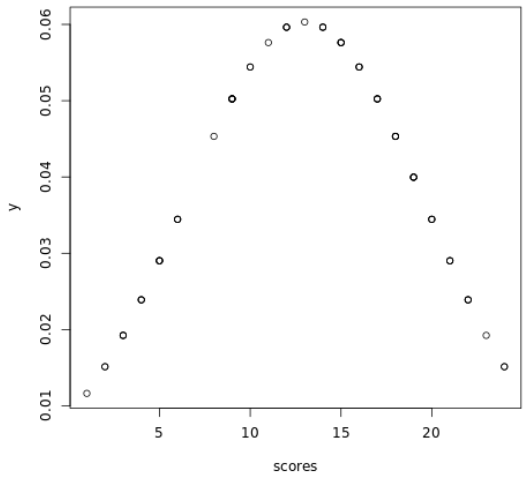
```
scores <- c(
                c(12,22,6,9,2,9,5,9,3,5,16,1,22,18),
                c(6,12,21,23,9,10,24,21,17,11,18,19,17,5),
                c(14,16,19,19,18,3,4,21,16,20,15,14,17,4),
                c(5,22,12,15,18,20,8,10,13,20,6,9,2,17),
                c(15,9,4,15,14,19,3,24)
            )

mean <- mean(scores)
paste("Mean : ",mean)

mode <- names(sort(-table(scores)))[1]
paste("Mode : ",mode)

median <- median(scores)
paste("Median : ",median)

range <- max(scores) - min(scores)
paste("Range : ",range)

iqr <- IQR(scores)
paste("InterQuantile Range : ",iqr)
```

```
#normal distribution

sdv <- sd(scores)

y <- dnorm(scores, mean, sdv)
plot(scores,y)

y <- pnorm(scores, mean, sdv)
plot(scores,y)
```

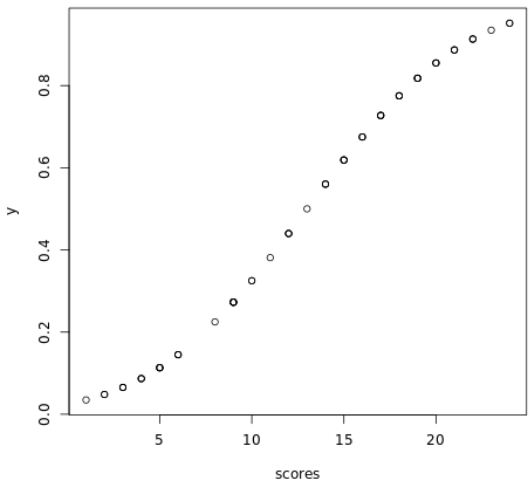
Output:

```
[1] "Mean :  13"
[1] "Mode :  9"
[1] "Median :  14"
[1] "Range :  23"
[1] "InterQuantile Range :  10.75"
```

Qnorm :



pnorm:



****

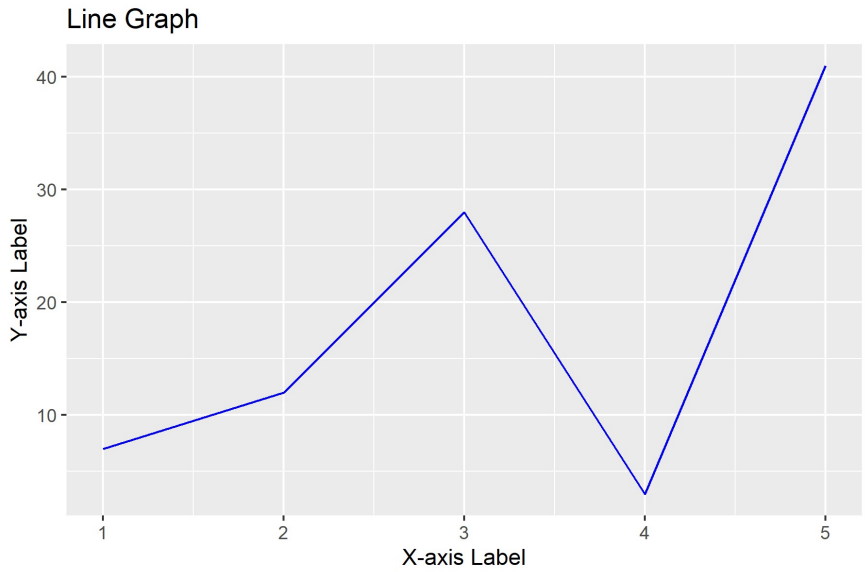
Q2: Plot the line graph using v<- c(7,12,28,3,41) and save the plot.

Program:

```
# Load the ggplot2 package
library(ggplot2)

# Define your data
v <- c(7, 12, 28, 3, 41)
x <- 1:length(v)
```

```
# Create the line graph
line_plot <- ggplot() +
  geom_line(aes(x = x, y = v), color = "blue") +
  xlab("X-axis Label") +
  ylab("Y-axis Label") +
  ggtitle("Line Graph")

# Save the plot to a file (e.g., in PNG format)
ggsave("C:\\Users\\sshru\\OneDrive\\Desktop\\SY\\Data Science\\Assignment 5\\line_plot.png",
plot = line_plot, width = 6, height = 4, dpi = 300)
```

Output:



****

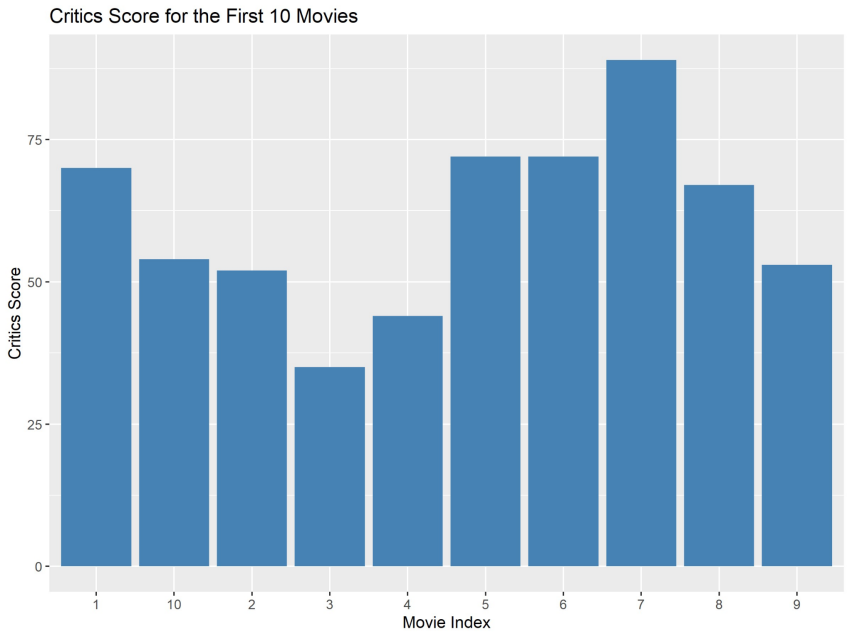
Q3: Read the file moviesData.csv create a bar chart of critics_score for the first 10 movies. Save the plot.

Program:

```
# Install and load the ggplot2 package if not already installed
# install.packages("ggplot2")
library(ggplot2)
# Read the CSV file
movies_data <- read.csv("C:\\Users\\sshru\\OneDrive\\Desktop\\SY\\Data
Science\\Assignment 5\\movies.csv")
movies_data
# Select the first 10 rows
first_10_movies <- movies_data[1:10, ]
# Create a bar chart
```

```
bar_plot <- ggplot(data = first_10_movies, aes(x = row.names(first_10_movies), y =
Audience.score..)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Critics Score for the First 10 Movies", x = "Movie Index", y = "Critics Score")

# Save the plot as an image file (e.g., in PNG format)
ggsave("C:\\Users\\sshru\\OneDrive\\Desktop\\SY\\Data Science\\Assignment
5\\critics_score_bar_chart.png", plot = bar_plot, width = 8, height = 6, dpi = 300)
```

Output:



****

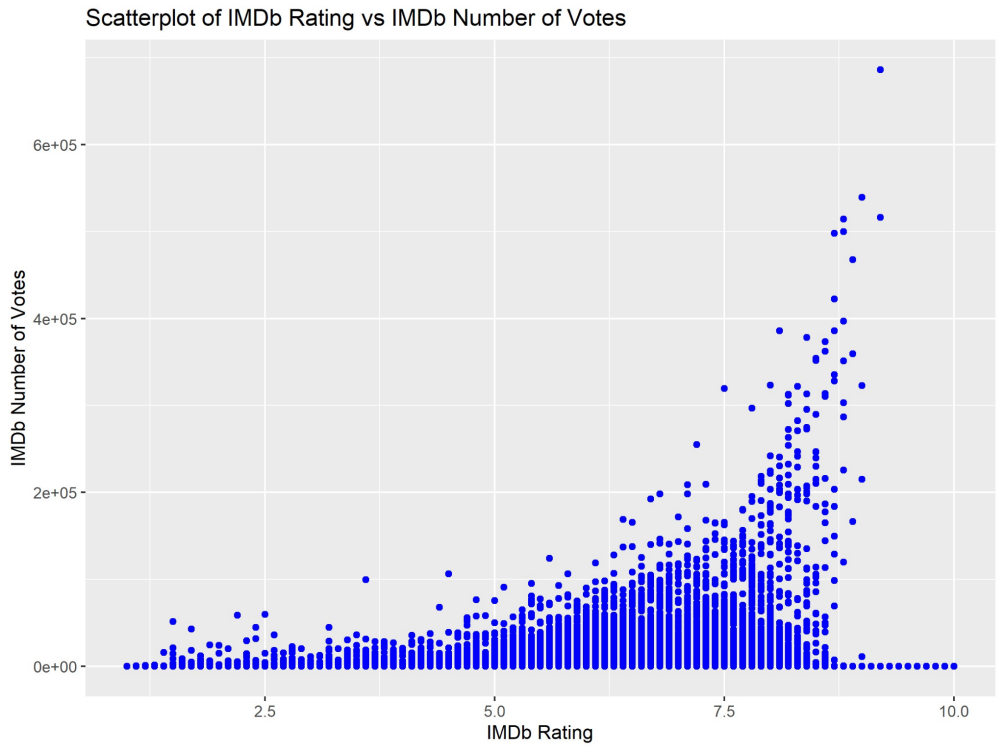
Q4: Create a scatterplot of imdb_rating and imdb_num_votes to see their relation and save the plot.

Program:

```
library(ggplot2)
load("C:\\Users\\sshru\\OneDrive\\Desktop\\SY\\Data Science\\Assignment
5\\movies.rdata")
ls()
movie_sub <- tibble::as_tibble(movies)
movie_sub <- dplyr::select(movie_sub, title, rating,votes)
movie_sub


scatter_plot <- ggplot(data = movie_sub, aes(x = rating, y = votes)) +
  geom_point(color = "blue") +
```

```
  labs(title = "Scatterplot of IMDb Rating vs IMDb Number of Votes", x = "IMDb Rating", y
= "IMDb Number of Votes")

# Save the plot as an image file (e.g., in PNG format)
ggsave("C:\\Users\\sshru\\OneDrive\\Desktop\\SY\\Data Science\\Assignment
5\\imdb_rating_vs_num_votes_scatterplot.png", plot = scatter_plot, width = 8, height = 6,
dpi = 300)
```

Output:

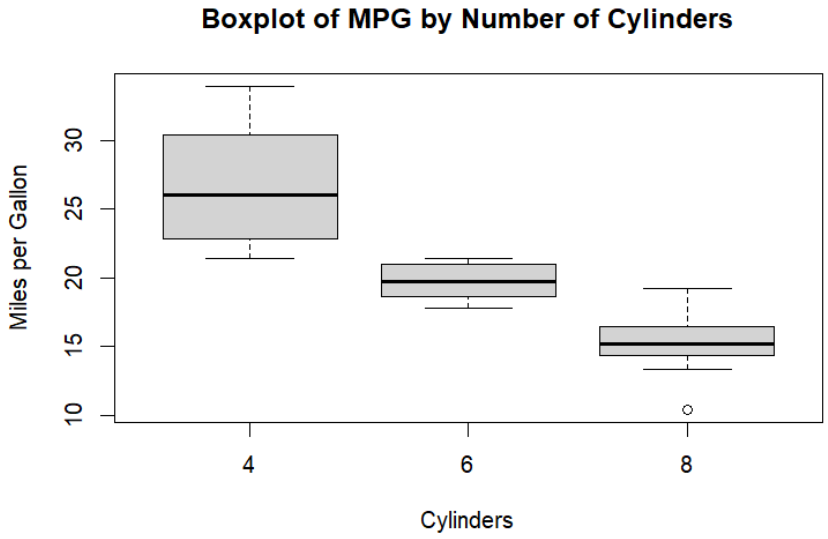


****

Q5: Use the data set “mtcars”and create a boxplot for “mpg” and “cyl” columns.
Program:

```
data(mtcars)
mtcars
boxplot(mpg ~ cyl, data = mtcars,
        main = "Boxplot of MPG by Number of Cylinders",
        xlab = "Cylinders",
        ylab = "Miles per Gallon")
```

Output:



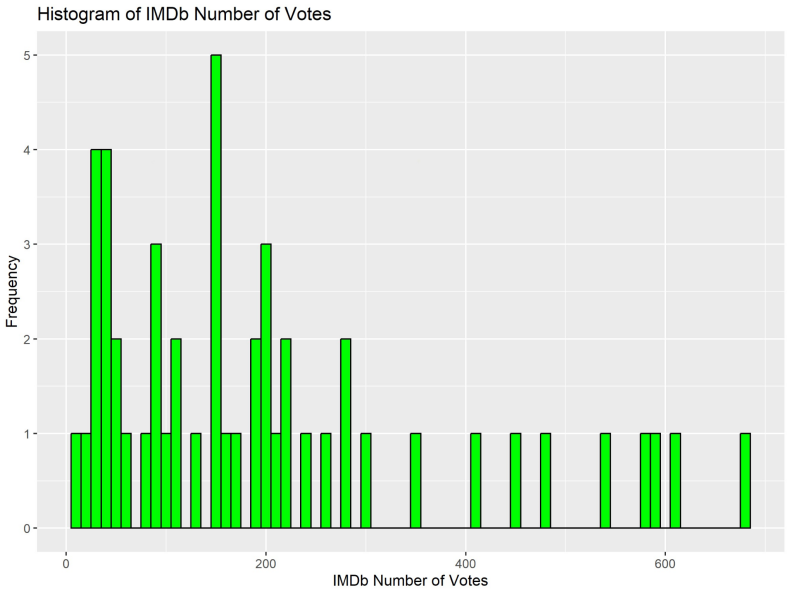
****

Q6: Read the file movies Data.csv, create a histogram of the object named imdb_num_votes in this file. Save the plot.

Program:

```
library(ggplot2)
load("C:\\Users\\sshru\\OneDrive\\Desktop\\SY\\Data Science\\Assignment 5\\movies.rdata")
ls()
movie_sub <- tibble::as_tibble(movies)
movie_sub <- dplyr::select(movie_sub,votes)
movie_sub <- head(movie_sub, 50)

histogram_plot <- ggplot(data = movie_sub, aes(x = votes)) +
  geom_histogram(binwidth = 10, fill = "green", color = "black") +
  labs(title = "Histogram of IMDb Number of Votes", x = "IMDb Number of Votes", y = "Frequency")

# Save the plot as an image file (e.g., in PNG format)
ggsave("C:\\Users\\sshru\\OneDrive\\Desktop\\SY\\Data Science\\Assignment 5\\imdb_num_votes_histogram1.png", plot = histogram_plot, width = 8, height = 6, dpi = 300)
```

Output:


Histogram of IMDb Number of Votes
Frequency
0
1
2
3
4
5
0
200
400
600
IMDb Number of Votes

****