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 **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 31. 8. 23

****

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.

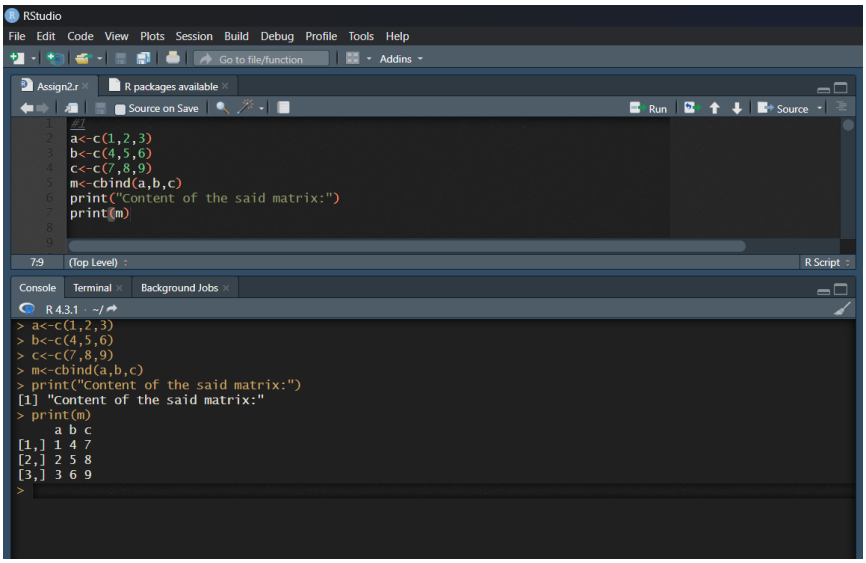
Code:

```
a<-c(1,2,3)
b<-c(4,5,6)
c<-c(7,8,9)
m<-cbind(a,b,c)
print("Content of the said matrix:")
print(m)
```

Output:a

```
> a<-c(1,2,3)
> b<-c(4,5,6)
> c<-c(7,8,9)
> m<-cbind(a,b,c)
> print("Content of the said matrix:")
[1] "Content of the said matrix:"
> print(m)
     a b c
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
```

Screenshot:



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 31. 8. 23

****

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.

Code:

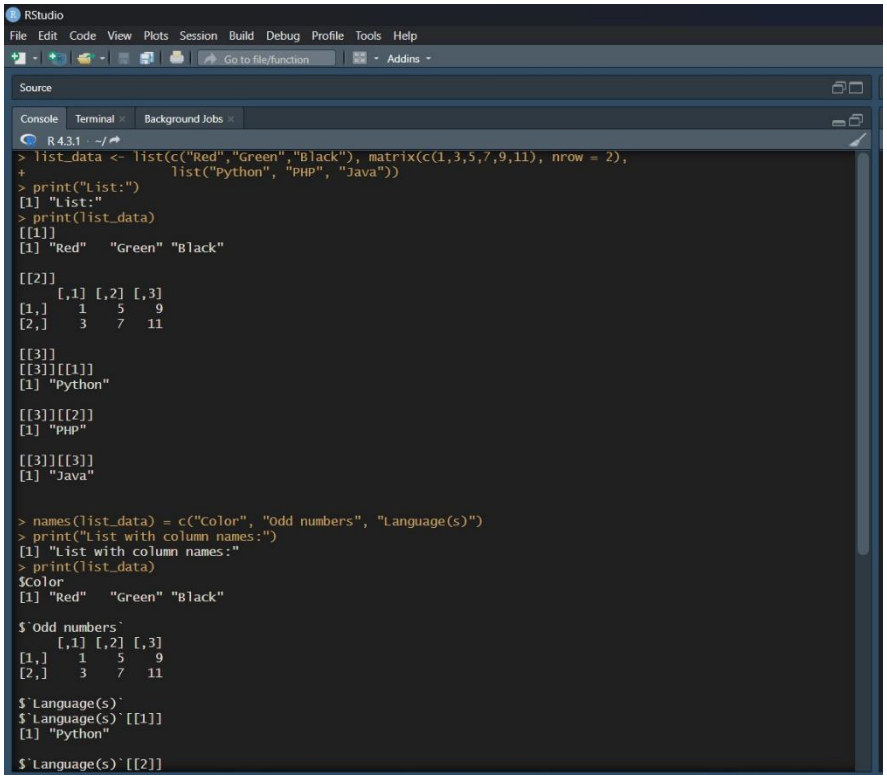
```
list_data <- list(c("Red","Green","Black"), matrix(c(1,3,5,7,9,11), nrow = 2),
                  list("Python", "PHP", "Java"))
print("List:")
print(list_data)
names(list_data) = c("Color", "Odd numbers", "Language(s)")
print("List with column names:")
print(list_data)
print('1st element:')
print(list_data[1])
print('2nd element:')
print(list_data[2])
```

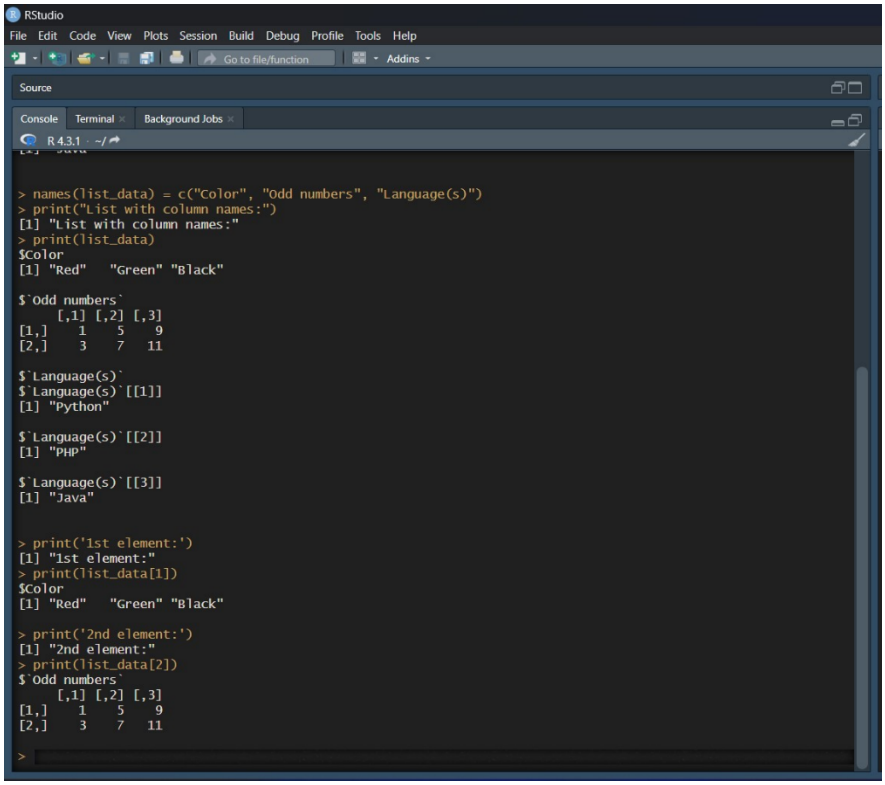
Output:

```
> list_data <- list(c("Red","Green","Black"), matrix(c(1,3,5,7,9,11), nrow = 2),
+                   list("Python", "PHP", "Java"))
> print("List:")
[1] "List:"
> print(list_data)
[[1]]
[1] "Red"   "Green" "Black"

[[2]]
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11

[[3]]
[[3]][[1]]
[1] "Python"

[[3]][[2]]
[1] "PHP"


[[3]][[3]]
[1] "Java"
```

```
> names(list_data) = c("Color", "Odd numbers", "Language(s)")
> print("List with column names:")
[1] "List with column names:"
> print(list_data)
$Color
[1] "Red"   "Green" "Black"

$`Odd numbers`
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11

$`Language(s)`
$`Language(s)`[[1]]
[1] "Python"

$`Language(s)`[[2]]
[1] "PHP"

$`Language(s)`[[3]]
[1] "Java"


> print('1st element:')
[1] "1st element:"
> print(list_data[1])
$Color
[1] "Red"   "Green" "Black"

> print('2nd element:')
[1] "2nd element:"
> print(list_data[2])
$`Odd numbers`
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11
```

Screenshot:


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R 4.3.1 · ~/
> list_data <- list(c("Red","Green","Black"), matrix(c(1,3,5,7,9,11), nrow = 2),
+ list("Python", "PHP", "Java"))
> print("List:")
[1] "List:"
> print(list_data)
[[1]]
[1] "Red" "Green" "Black"
[[2]]
[,1] [,2] [,3]
[1,] 1 5 9
[2,] 3 7 11
[[3]]
[[3]][[1]]
[1] "Python"
[[3]][[2]]
[1] "PHP"
[[3]][[3]]
[1] "Java"
> names(list_data) = c("Color", "Odd numbers", "Language(s)")
> print("List with column names:")
[1] "List with column names:"
> print(list_data)
$Color
[1] "Red" "Green" "Black"
$`Odd numbers`
[,1] [,2] [,3]
[1,] 1 5 9
[2,] 3 7 11
$`Language(s)`
$`Language(s)`[[1]]
[1] "Python"
$`Language(s)`[[2]]


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R 4.3.1 · ~/
> names(list_data) = c("Color", "Odd numbers", "Language(s)")
> print("List with column names:")
[1] "List with column names:"
> print(list_data)
$Color
[1] "Red" "Green" "Black"
$`Odd numbers`
[,1] [,2] [,3]
[1,] 1 5 9
[2,] 3 7 11
$`Language(s)`
$`Language(s)`[[1]]
[1] "Python"
$`Language(s)`[[2]]
[1] "PHP"
$`Language(s)`[[3]]
[1] "Java"
> print('1st element:')
[1] "1st element:"
> print(list_data[1])
$Color
[1] "Red" "Green" "Black"
> print('2nd element:')
[1] "2nd element:"
> print(list_data[2])
$`Odd numbers`
[,1] [,2] [,3]
[1,] 1 5 9
[2,] 3 7 11
>

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 **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 31. 8. 23

****

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.

Code:

```
a1 = c(20, 22, 24, 26)
a2 = c(21, 23, 25, 27, 29)
array1 = array(c(a1, a2),dim = c(3,3,2))
print(array1)
```

Output:

```
> a1 = c(20, 22, 24, 26)
> a2 = c(21, 23, 25, 27, 29)
> array1 = array(c(a1, a2),dim = c(3,3,2))
> print(array1)
, , 1

     [,1] [,2] [,3]
[1,]   20   26   25
[2,]   22   21   27
[3,]   24   23   29

, , 2

     [,1] [,2] [,3]
[1,]   20   26   25
[2,]   22   21   27
[3,]   24   23   29
```

Screenshot:

****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 31. 8. 23

****

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')

Code:

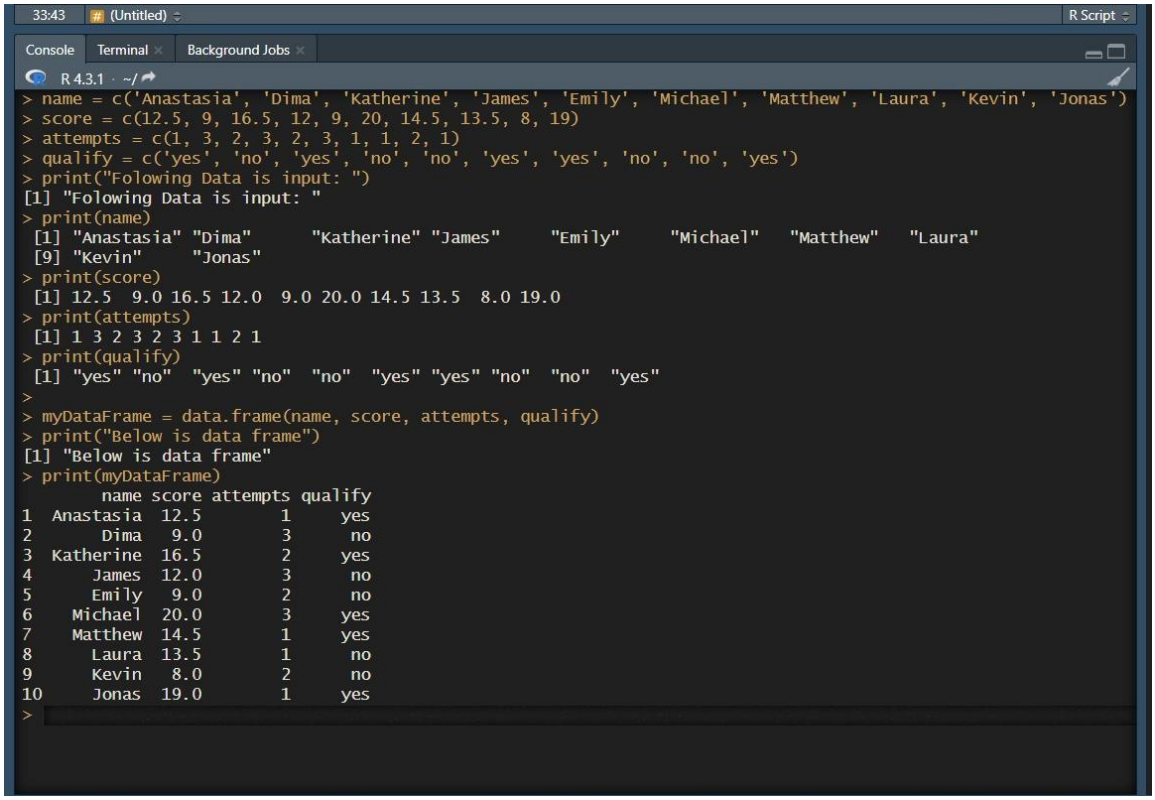
```
name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas')
score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19)
attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1)
qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
print("Folowing Data is input: ")
print(name)
print(score)
print(attempts)
print(qualify)

myDataFrame = data.frame(name, score, attempts, qualify)
print("Below is data frame")
print(myDataFrame)
```

Output:

```
> 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')
> print("Folowing Data is input: ")
[1] "Folowing Data is input: "
> print(name)
 [1] "Anastasia" "Dima"      "Katherine" "James"     "Emily"     "Michael"   "Matthew"   "Laura"
 [9] "Kevin"     "Jonas"
> print(score)
 [1] 12.5  9.0 16.5 12.0  9.0 20.0 14.5 13.5  8.0 19.0
> print(attempts)
 [1] 1 3 2 3 2 3 1 1 2 1
> print(qualify)
 [1] "yes" "no"  "yes" "no"  "no"  "yes" "yes" "no"  "no"  "yes"
>
> myDataFrame = data.frame(name, score, attempts, qualify)
> print("Below is data frame")
```

```
[1] "Below is data frame"
> print(myDataFrame)
       name 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    Matthew  14.5        1     yes
8      Laura  13.5        1      no
9      Kevin   8.0        2      no
10     Jonas  19.0        1     yes
```

Screenshot:



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 31. 8. 23

****

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

Code:

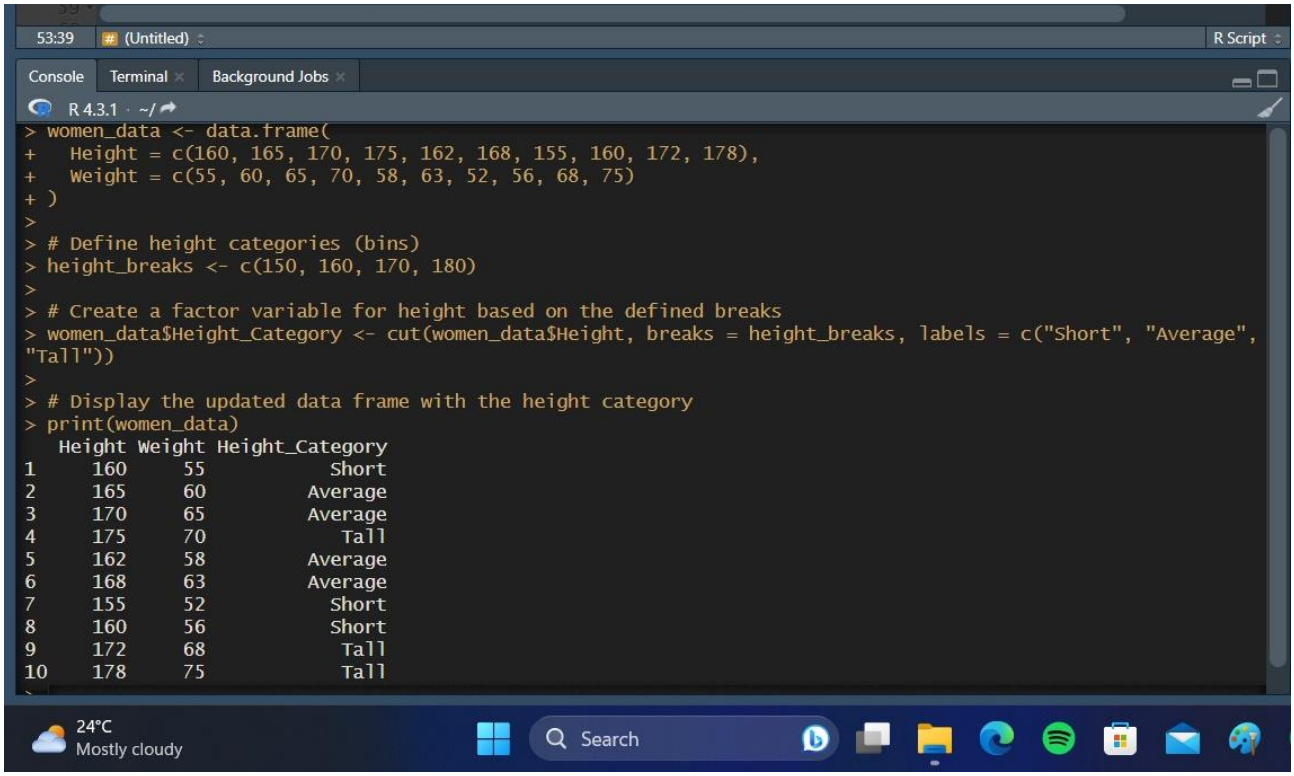
```
women_data <- data.frame(
  Height = c(160, 165, 170, 175, 162, 168, 155, 160, 172, 178),
  Weight = c(55, 60, 65, 70, 58, 63, 52, 56, 68, 75)
)

# 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)
```

Output:

```
> 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)
   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

Screenshot:



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 31. 8. 23

****

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

$$\begin{bmatrix} 7 & 9 & 12 \\ 2 & 4 & 13 \end{bmatrix} \times \begin{bmatrix} 1 & 7 & 12 & 19 \\ 2 & 8 & 13 & 20 \\ 3 & 9 & 14 & 21 \end{bmatrix}$$

What is the resulting matrix?

Code:

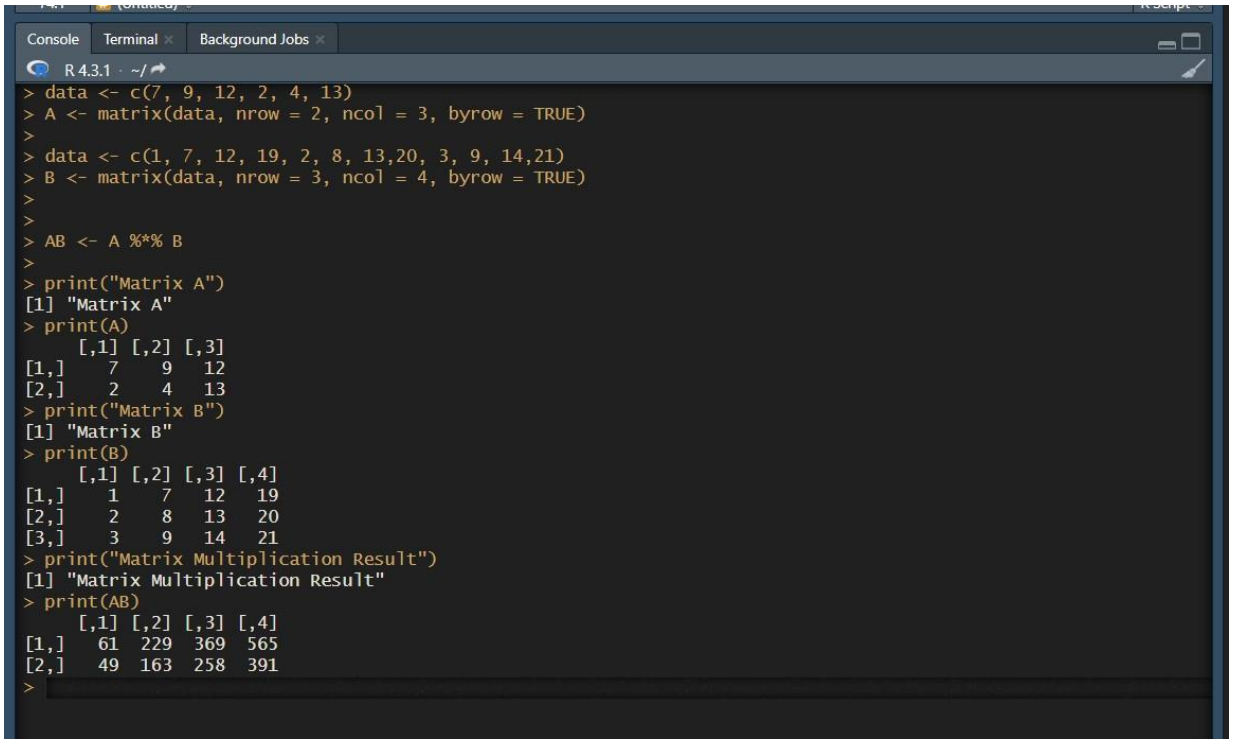
```
data <- c(7, 9, 12, 2, 4, 13)
A <- matrix(data, nrow = 2, ncol = 3, byrow = TRUE)

data <- c(1, 7, 12, 19, 2, 8, 13,20, 3, 9, 14,21)
B <- matrix(data, nrow = 3, ncol = 4, byrow = TRUE)


AB <- A %*% B

print("Matrix A")
print(A)
print("Matrix B")
print(B)
print("Matrix Multiplication Result")
print(AB)
```

Output:

```
> data <- c(7, 9, 12, 2, 4, 13)
> A <- matrix(data, nrow = 2, ncol = 3, byrow = TRUE)
>
> data <- c(1, 7, 12, 19, 2, 8, 13,20, 3, 9, 14,21)
> B <- matrix(data, nrow = 3, ncol = 4, byrow = TRUE)
>
>
> AB <- A %*% B
>
> print("Matrix A")
[1] "Matrix A"
> print(A)
     [,1] [,2] [,3]
[1,]    7    9   12
[2,]    2    4   13
```

```
> print("Matrix B")
[1] "Matrix B"
> print(B)
     [,1] [,2] [,3] [,4]
[1,]    1    7   12   19
[2,]    2    8   13   20
[3,]    3    9   14   21
> print("Matrix Multiplication Result")
[1] "Matrix Multiplication Result"
> print(AB)
     [,1] [,2] [,3] [,4]
[1,]   61  229  369  565
[2,]   49  163  258  391
```

Screenshot:



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 31. 8. 23

****

Q7. WAP to Print the Fibonacci Sequence.

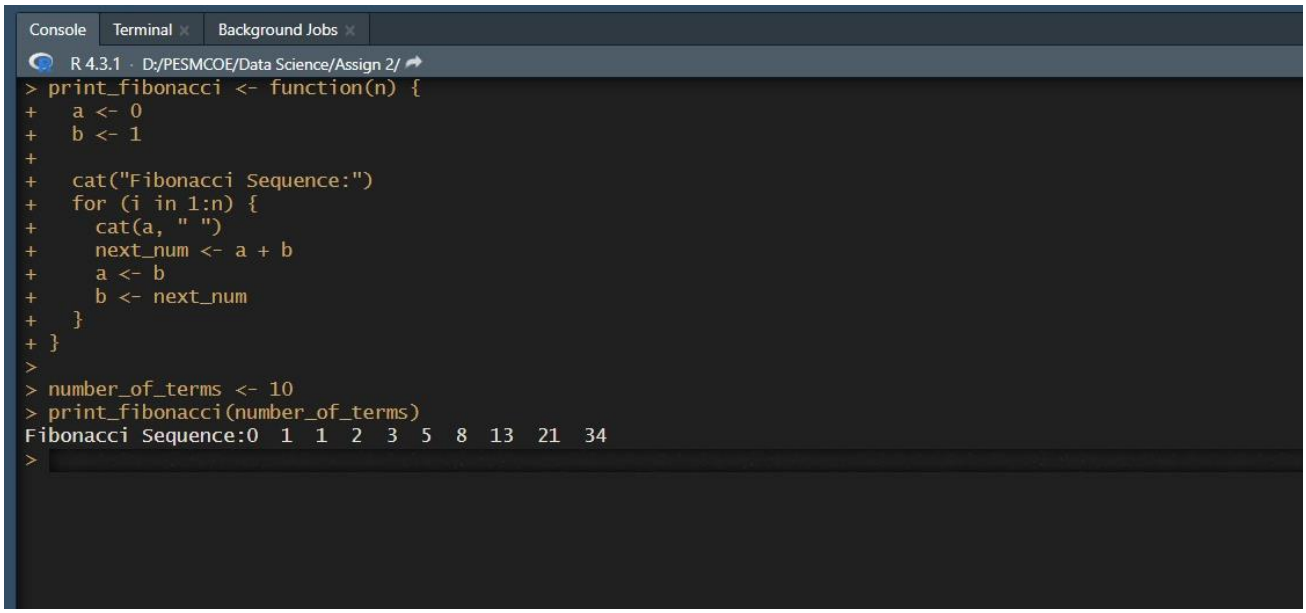
Code:

```
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)
```

Output:

```
> # Function to print the Fibonacci sequence using a loop
> 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
```

Screenshot:


Console
Terminal
Background Jobs
R 4.3.1 · D:/PESMCOE/Data Science/Assign 2/
> 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
>

****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 31. 8. 23

****

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

Code:

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

Output:

```
> read_csv = read.csv("D:\\PESMCOE\\Data Science\\Assign 2\\username_onboarding.csv")
> print(read_csv)
  Username..Identifier.First.name.Last.name
1              booker12;9012;Rachel;Booker
2                 grey07;2070;Laura;Grey
3              johnson81;4081;Craig;Johnson
4              jenkins46;9346;Mary;Jenkins
5                smith79;5079;Jamie;Smith
```

Code:

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

Output:

```
> library(readxl)
> read_exc = read_excel("D:\\PESMCOE\\Data Science\\Assign 2\\Financial_Sample.xlsx")
> print(head(read_exc))
# A tibble: 6 × 16
  Segment    Country Product `Discount Band` `Units Sold` `Manufacturing Price` `Sale Price` `Gross
Sales`
  <chr>      <chr>   <chr>   <chr>                <dbl>              <dbl>       <dbl>       <dbl>
1 Government Canada  Carret… None                1618.                  3          20       32370
2 Government Germany Carret… None                1321                   3          20       26420
3 Midmarket  France  Carret… None                2178                   3          15       32670
4 Midmarket  Germany Carret… None                 888                   3          15       13320
5 Midmarket  Mexico  Carret… None                2470                   3          15       37050
6 Government Germany Carret… None                1513                   3         350      529550
# ℹ 8 more variables: Discounts <dbl>, Sales <dbl>, COGS <dbl>, Profit <dbl>, Date <dttm>,
#   `Month Number` <dbl>, `Month Name` <chr>, Year <chr>
```

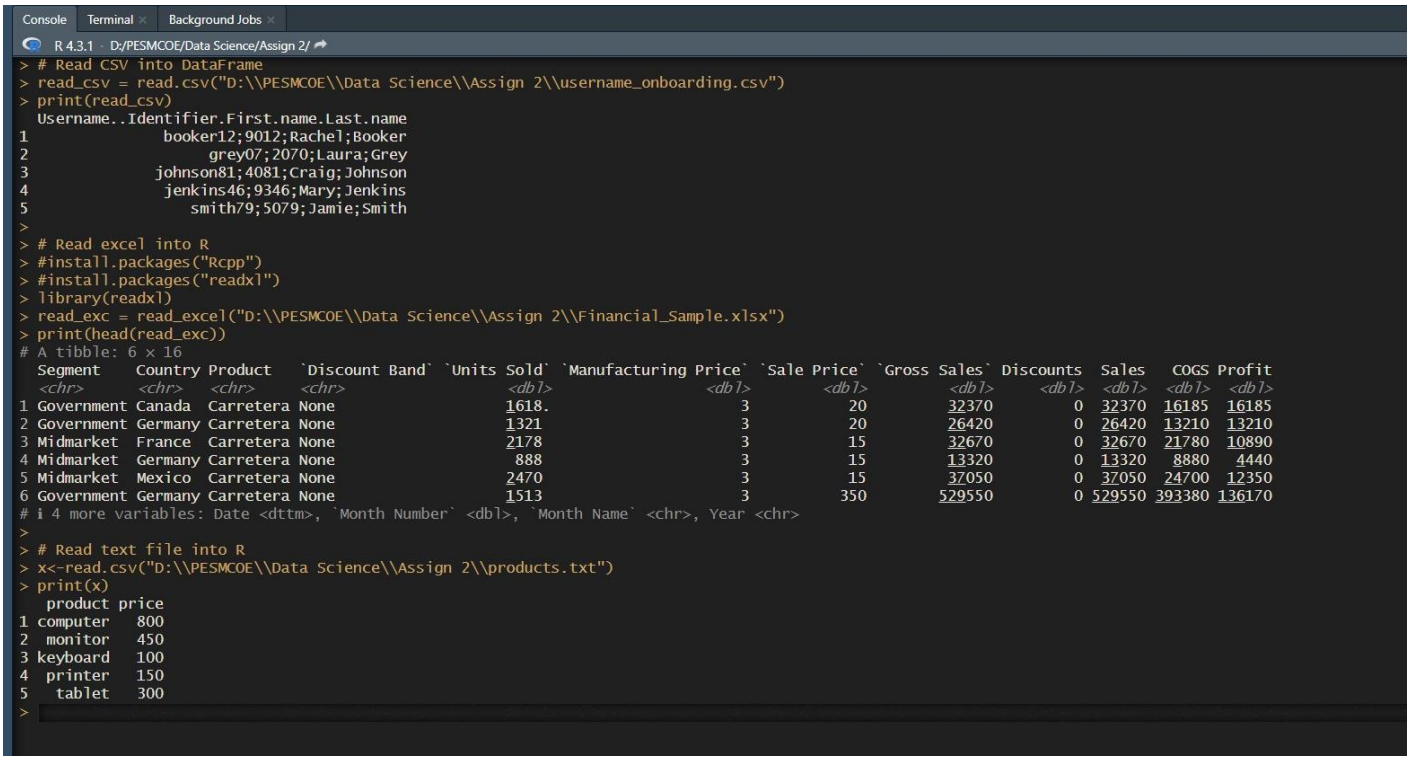
Code:

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

Output:

```
> x<-read.csv("D:\\PESMCOE\\Data Science\\Assign 2\\products.txt")
> print(x)
   product price
1 computer   800
2  monitor   450
3 keyboard   100
4  printer   150
5   tablet   300
```

Screenshot:



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 6. 9. 23

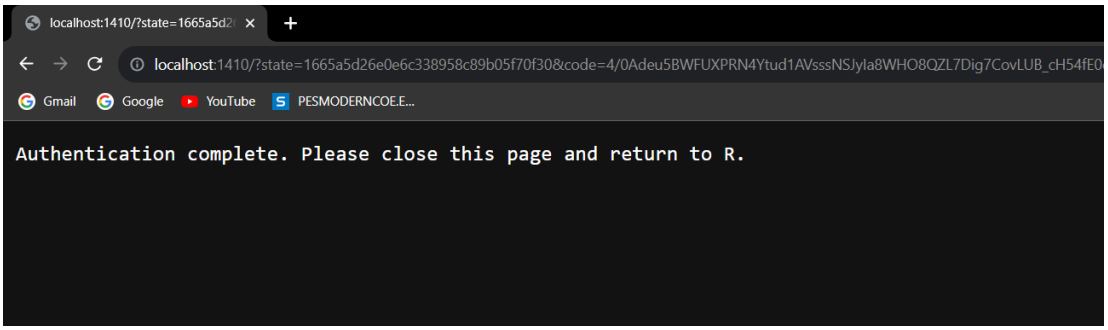
****

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

Code:

```
#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,"D:\\PESMCOE\\Data Science\\Assign 2\\exp_products.csv", row.names = FALSE )

#Exporting to excel file
library("writexl")
excel_info <- data.frame( Name = c("Jon", "Bill", "Maria", "Ben", "Tina"),
                          Age = c(23, 41, 32, 58, 26))
write_xlsx(excel_info,"D:\\PESMCOE\\Data Science\\Assign 2\\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,"D:\\PESMCOE\\Data Science\\Assign 2\\exp_text.txt", quote=FALSE)

#Exporting to google drive
#install.packages("googledrive")
library(googledrive)
write.csv(x=iris,file="iris.csv")
drive_upload(media="iris.csv",type='spreadsheet')
ss<- gs4_create(name="ds_assign_1",sheets="Sheet 1")
sheet_write(data=iris,ss=ss,sheet="Sheet 1")
drive_mv(file=ss,path="My Drive/Assignments/")
```

Output:



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 6. 9. 23

****

Q10. Write an 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

Code:

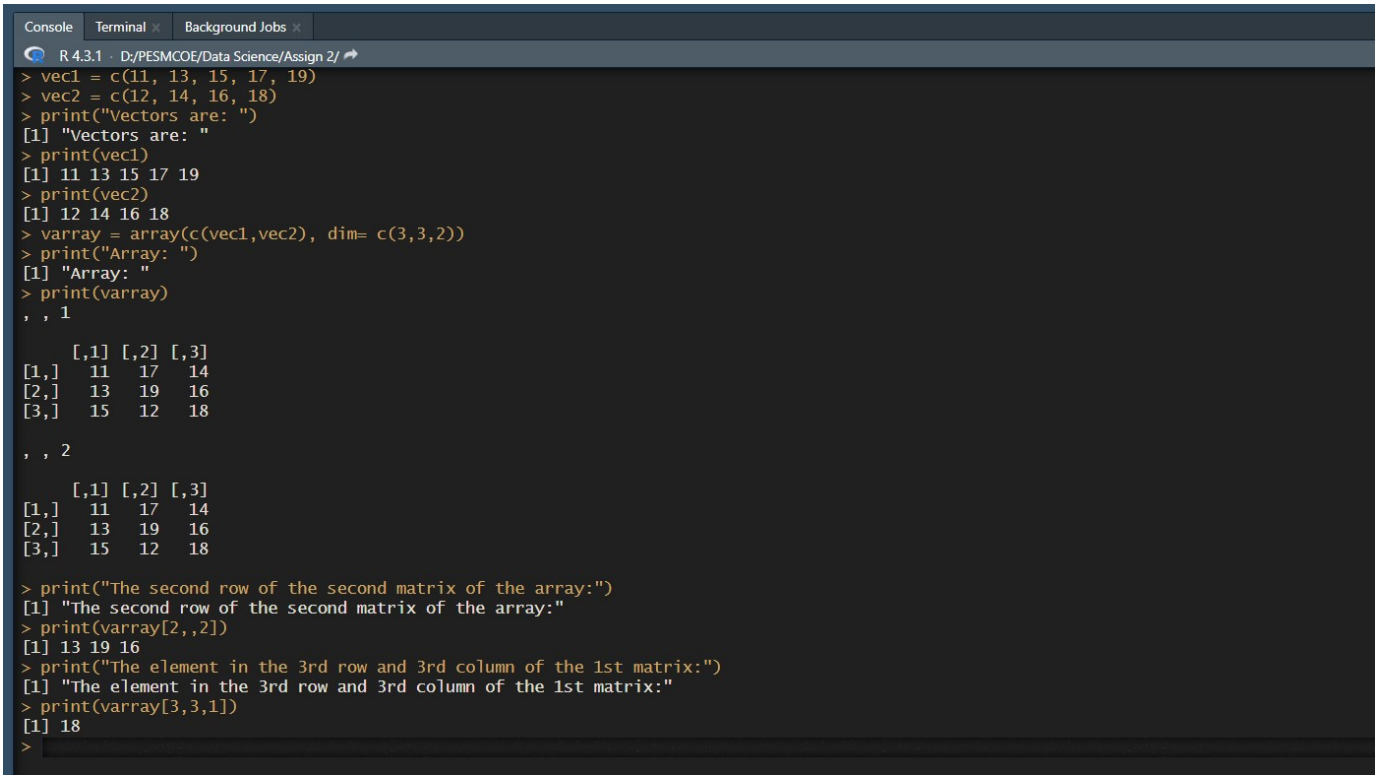
```
vec1 = c(11, 13, 15, 17, 19)
vec2 = c(12, 14, 16, 18)
print("Vectors are: ")
print(vec1)
print(vec2)
varray = array(c(vec1,vec2), dim= c(3,3,2))
print("Array: ")
print(varray)
print("The second row of the second matrix of the array:")
print(varray[2,,2])
print("The element in the 3rd row and 3rd column of the 1st matrix:")
print(varray[3,3,1])
```

Output:

```
> vec1 = c(11, 13, 15, 17, 19)
> vec2 = c(12, 14, 16, 18)
> print("Vectors are: ")
[1] "Vectors are: "
> print(vec1)
[1] 11 13 15 17 19
> print(vec2)
[1] 12 14 16 18
> varray = array(c(vec1,vec2), dim= c(3,3,2))
> print("Array: ")
[1] "Array: "
> print(varray)
, , 1

     [,1] [,2] [,3]
[1,]   11   17   14
[2,]   13   19   16
[3,]   15   12   18

, , 2
```

```
     [,1] [,2] [,3]
[1,]   11   17   14
[2,]   13   19   16
[3,]   15   12   18

> print("The second row of the second matrix of the array:")
[1] "The second row of the second matrix of the array:"
> print(varray[2,,2])
[1] 13 19 16
> 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(varray[3,3,1])
[1] 18
```

Screenshot:



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 6. 9. 23

****

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.

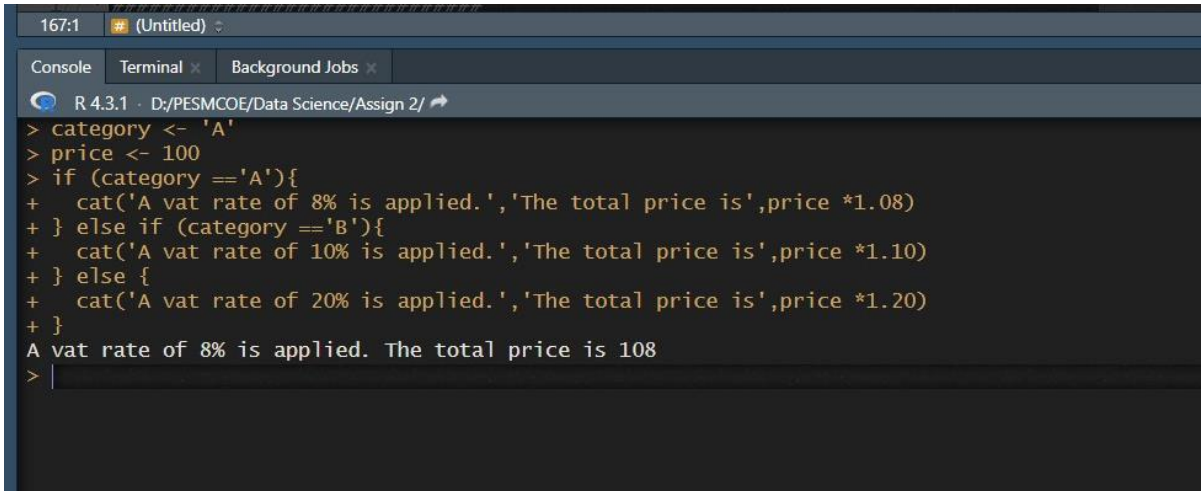
Code:

```
category <- 'A'
price <- 100
if (category =='A'){
  cat('A vat rate of 8% is applied.','The total price is',price *1.08)
} else if (category =='B'){
  cat('A vat rate of 10% is applied.','The total price is',price *1.10)
} else {
  cat('A vat rate of 20% is applied.','The total price is',price *1.20)
}
```

Output:

```
> category <- 'A'
> price <- 100
> if (category =='A'){
+   cat('A vat rate of 8% is applied.','The total price is',price *1.08)
+ } else if (category =='B'){
+   cat('A vat rate of 10% is applied.','The total price is',price *1.10)
+ } else {
+   cat('A vat rate of 20% is applied.','The total price is',price *1.20)
+ }
A vat rate of 8% is applied. The total price is 108
```

Screenshot:


167:1
(Untitled)
Console
Terminal
Background Jobs
R 4.3.1 · D:/PESMCOE/Data Science/Assign 2/
> category <- 'A'
> price <- 100
> if (category =='A'){
+ cat('A vat rate of 8% is applied.','The total price is',price *1.08)
+ } else if (category =='B'){
+ cat('A vat rate of 10% is applied.','The total price is',price *1.10)
+ } else {
+ cat('A vat rate of 20% is applied.','The total price is',price *1.20)
+ }
A vat rate of 8% is applied. The total price is 108
>

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 **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 6. 9. 23

****

Q12. A cloth showroom has announced the following seasonal discounts on purchase of items. Write an 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%

Code:

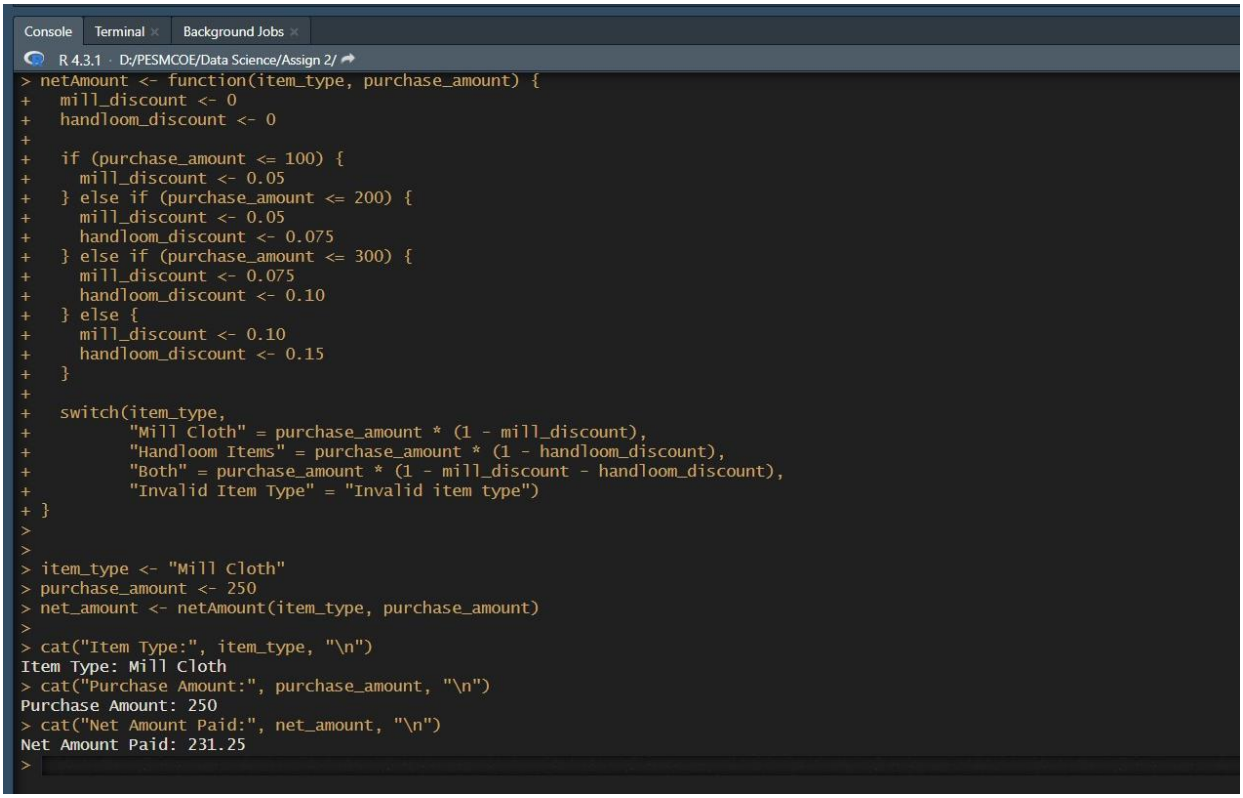
```
netAmount <- function(item_type, purchase_amount) {
 mill_discount <- 0
 handloom_discount <- 0

 if (purchase_amount <= 100) {
  mill_discount <- 0.05
 } else if (purchase_amount <= 200) {
  mill_discount <- 0.05
  handloom_discount <- 0.075
 } else if (purchase_amount <= 300) {
  mill_discount <- 0.075
  handloom_discount <- 0.10
 } else {
  mill_discount <- 0.10
  handloom_discount <- 0.15
 }

 switch(item_type,
       "Mill Cloth" = purchase_amount * (1 - mill_discount),
       "Handloom Items" = purchase_amount * (1 - handloom_discount),
       "Both" = purchase_amount * (1 - mill_discount - handloom_discount),
       "Invalid Item Type" = "Invalid item type")
}




item_type <- "Mill Cloth"
purchase_amount <- 250
```

```
net_amount <- netAmount(item_type, purchase_amount)

cat("Item Type:", item_type, "\n")
cat("Purchase Amount:", purchase_amount, "\n")
cat("Net Amount Paid:", net_amount, "\n")
```

Output:

```
> netAmount <- function(item_type, purchase_amount) {
+   mill_discount <- 0
+   handloom_discount <- 0
+
+   if (purchase_amount <= 100) {
+     mill_discount <- 0.05
+   } else if (purchase_amount <= 200) {
+     mill_discount <- 0.05
+     handloom_discount <- 0.075
+   } else if (purchase_amount <= 300) {
+     mill_discount <- 0.075
+     handloom_discount <- 0.10
+   } else {
+     mill_discount <- 0.10
+     handloom_discount <- 0.15
+   }
+
+   switch(item_type,
+          "Mill Cloth" = purchase_amount * (1 - mill_discount),
+          "Handloom Items" = purchase_amount * (1 - handloom_discount),
+          "Both" = purchase_amount * (1 - mill_discount - handloom_discount),
+          "Invalid Item Type" = "Invalid item type")
+ }
>
>
> item_type <- "Mill Cloth"
> purchase_amount <- 250
> net_amount <- netAmount(item_type, purchase_amount)
>
> cat("Item Type:", item_type, "\n")
Item Type: Mill Cloth
> cat("Purchase Amount:", purchase_amount, "\n")
Purchase Amount: 250
> cat("Net Amount Paid:", net_amount, "\n")
Net Amount Paid: 231.25
```

Screenshot:


Console
Terminal
Background Jobs
R 4.3.1 · D:/PESMCOE/Data Science/Assign 2/
> netAmount <- function(item_type, purchase_amount) {
+ mill_discount <- 0
+ handloom_discount <- 0
+
+ if (purchase_amount <= 100) {
+ mill_discount <- 0.05
+ } else if (purchase_amount <= 200) {
+ mill_discount <- 0.05
+ handloom_discount <- 0.075
+ } else if (purchase_amount <= 300) {
+ mill_discount <- 0.075
+ handloom_discount <- 0.10
+ } else {
+ mill_discount <- 0.10
+ handloom_discount <- 0.15
+ }
+
+ switch(item_type,
+ "Mill Cloth" = purchase_amount * (1 - mill_discount),
+ "Handloom Items" = purchase_amount * (1 - handloom_discount),
+ "Both" = purchase_amount * (1 - mill_discount - handloom_discount),
+ "Invalid Item Type" = "Invalid item type")
+ }
>
>
> item_type <- "Mill Cloth"
> purchase_amount <- 250
> net_amount <- netAmount(item_type, purchase_amount)
>
> cat("Item Type:", item_type, "\n")
Item Type: Mill Cloth
> cat("Purchase Amount:", purchase_amount, "\n")
Purchase Amount: 250
> cat("Net Amount Paid:", net_amount, "\n")
Net Amount Paid: 231.25
>

****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 7. 9. 23

****

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

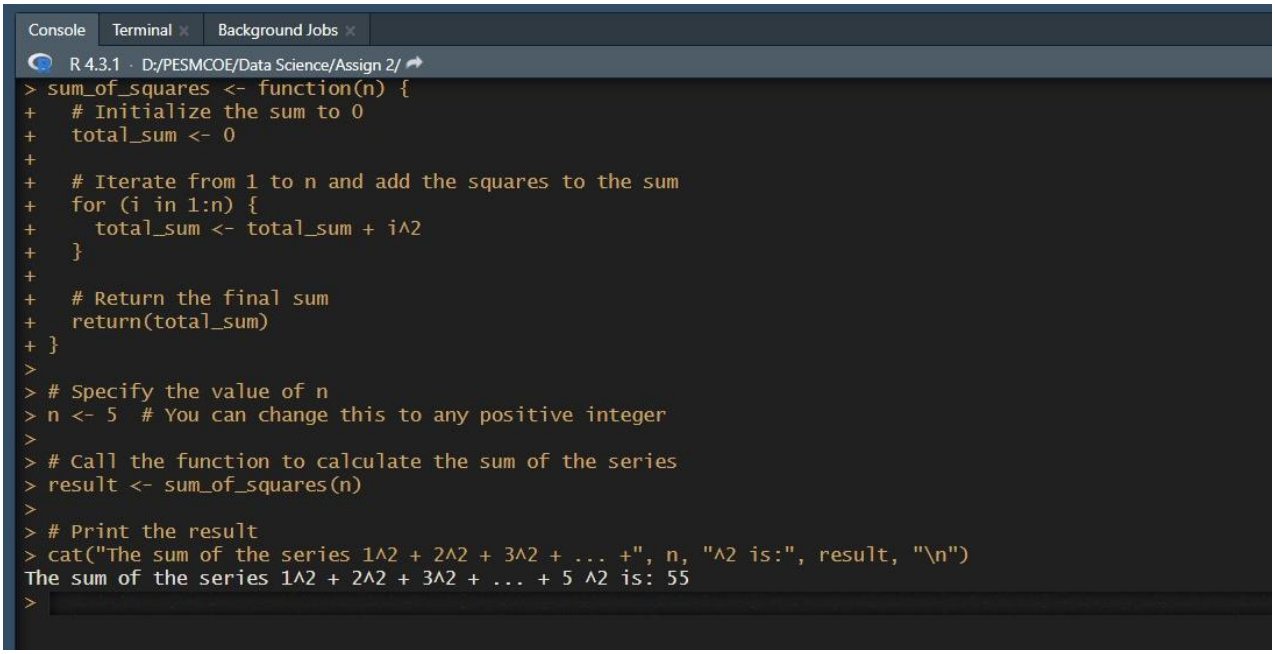
Code:

```
sum_of_squares <- function(n) {
  total_sum <- 0

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

n <- 5
result <- sum_of_squares(n)
cat("The sum of the series 1^2 + 2^2 + 3^2 + ... +", n, "^2 is:", result, "\n")
```

Output:

```
> sum_of_squares <- function(n) {
+
+   total_sum <- 0
+
+   for (i in 1:n) {
+     total_sum <- total_sum + i^2
+   }
+   return(total_sum)
+ }
>
>
> n <- 5
>
> result <- sum_of_squares(n)
>
> cat("The sum of the series 1^2 + 2^2 + 3^2 + ... +", n, "^2 is:", result, "\n")
The sum of the series 1^2 + 2^2 + 3^2 + ... + 5 ^2 is: 55
```

Screenshot:


Console
Terminal
Background Jobs
R 4.3.1 · D:/PESMCOE/Data Science/Assign 2/
> sum_of_squares <- function(n) {
+   # Initialize the sum to 0
+   total_sum <- 0
+
+   # Iterate from 1 to n and add the squares to the sum
+   for (i in 1:n) {
+   total_sum <- total_sum + i^2
+   }
+
+   # Return the final sum
+   return(total_sum)
+ }
>
> # Specify the value of n
> n <- 5 # You can change this to any positive integer
>
> # Call the function to calculate the sum of the series
> result <- sum_of_squares(n)
>
> # Print the result
> cat("The sum of the series 1^2 + 2^2 + 3^2 + ... +", n, "^2 is:", result, "\n")
The sum of the series 1^2 + 2^2 + 3^2 + ... + 5 ^2 is: 55
>

****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 7. 9. 23

****

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.

Code:

```
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)
}
```

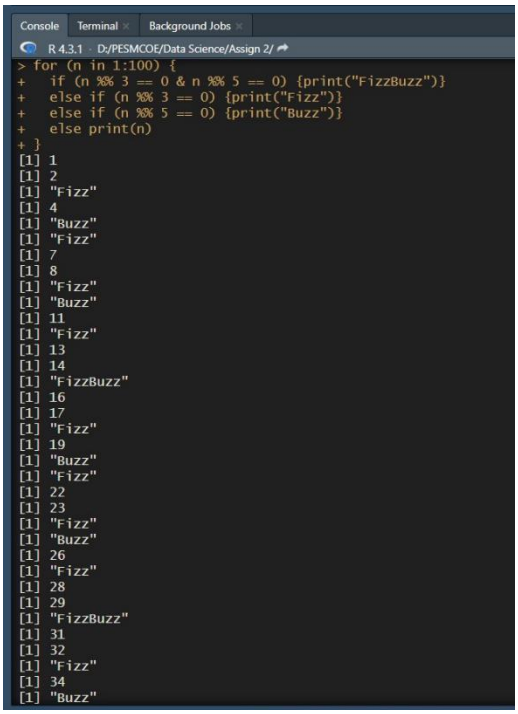
Output:

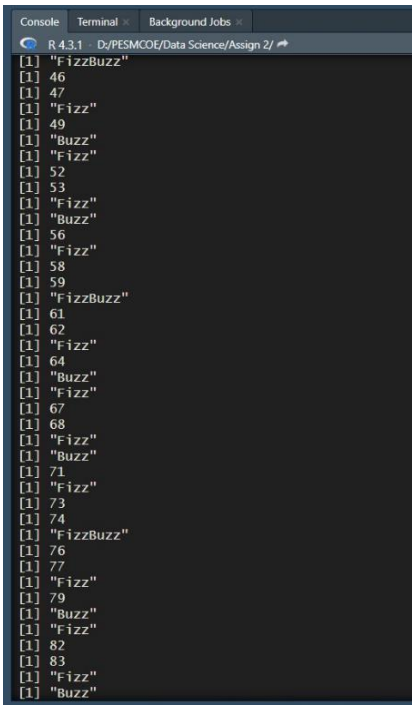
```
> for (n in 1:100) {
+   if (n %% 3 == 0 & n %% 5 == 0) {print("FizzBuzz")}
+   else if (n %% 3 == 0) {print("Fizz")}
+   else if (n %% 5 == 0) {print("Buzz")}
+   else print(n)
+ }
[1] 1
[1] 2
[1] "Fizz"
[1] 4
[1] "Buzz"
[1] "Fizz"
[1] 7
[1] 8
[1] "Fizz"
[1] "Buzz"
[1] 11
[1] "Fizz"
[1] 13
[1] 14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

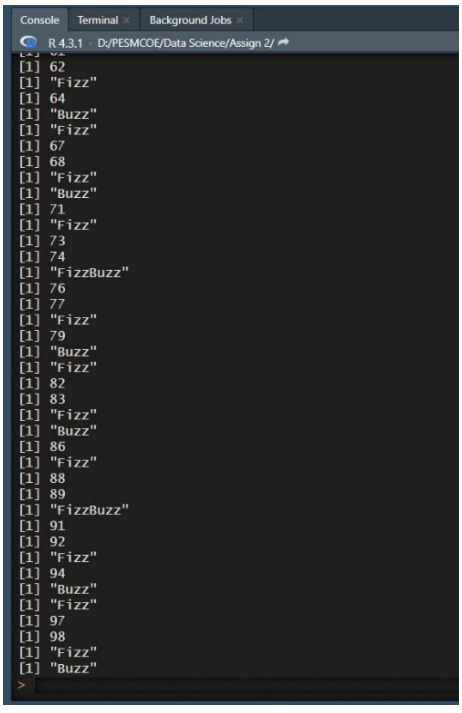
[1] 23
[1] "Fizz"
[1] "Buzz"
[1] 26
[1] "Fizz"
[1] 28
[1] 29
[1] "FizzBuzz"
[1] 31
[1] 32
[1] "Fizz"
[1] 34
[1] "Buzz"
[1] "Fizz"
[1] 37
[1] 38
[1] "Fizz"
[1] "Buzz"
[1] 41
[1] "Fizz"
[1] 43
[1] 44
[1] "FizzBuzz"
[1] 46
[1] 47
[1] "Fizz"
[1] 49
[1] "Buzz"
[1] "Fizz"
[1] 52
[1] 53
[1] "Fizz"
[1] "Buzz"
[1] 56
[1] "Fizz"
[1] 58
[1] 59
[1] "FizzBuzz"
[1] 61
[1] 62
[1] "Fizz"
[1] 64
[1] "Buzz"
[1] "Fizz"
[1] 67
[1] 68
[1] "Fizz"
[1] "Buzz"
[1] 71
[1] "Fizz"
[1] 73

```
[1] 74
[1] "FizzBuzz"
[1] 76
[1] 77
[1] "Fizz"
[1] 79
[1] "Buzz"
[1] "Fizz"
[1] 82
[1] 83
[1] "Fizz"
[1] "Buzz"
[1] 86
[1] "Fizz"
[1] 88
[1] 89
[1] "FizzBuzz"
[1] 91
[1] 92
[1] "Fizz"
[1] 94
[1] "Buzz"
[1] "Fizz"
[1] 97
[1] 98
[1] "Fizz"
[1] "Buzz"
```

Screenshot:







****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 51043

Name: Vanessa Reetu Prashant More **Assignment No:** 2 **Date of Implementation:** 7. 9. 23

****

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

Code:

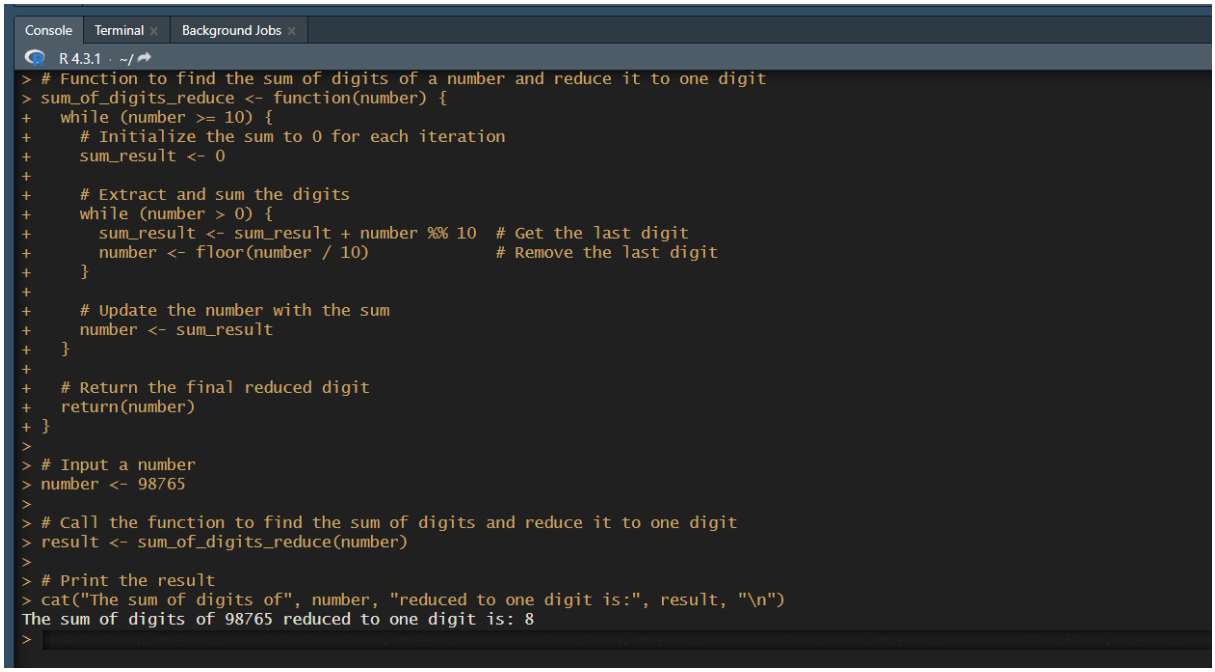
```
sum_of_digits_reduce <- function(number) {
  while (number >= 10) {
    sum_result <- 0

    while (number > 0) {
      sum_result <- sum_result + number %% 10
      number <- floor(number / 10)
    }

    number <- sum_result
  }

  return(number)
}

number <- 98765

result <- sum_of_digits_reduce(number)

cat("The sum of digits of", number, "reduced to one digit is:", result, "\n")
```

Output:

```
> sum_of_digits_reduce <- function(number) {
+   while (number >= 10) {
+
+     sum_result <- 0
+
+     while (number > 0) {
+       sum_result <- sum_result + number %% 10
+       number <- floor(number / 10)
+     }
+
+     number <- sum_result
+   }
+
+   return(number)
+ }
```

```
>
> number <- 98765
>
> result <- sum_of_digits_reduce(number)
>
> cat("The sum of digits of", number, "reduced to one digit is:", result, "\n")
The sum of digits of 98765 reduced to one digit is: 8
```

Screenshot:



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 3 **Date of Implementation:** 11. 10. 23

****

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.

Code:

```
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"))

things

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

                      class="unknown")
unknown

#install.packages("dplyr")
#install.packages("descr")
#install.packages("ggplot2")

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
```

Output:

```
> 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"))
>
> things
  ingredient sweetness crunchiness     class
1      grape         8           5     fruit
2 green bean         3           7 vegetable
3       nuts         3           6   protein
4     orange         7           3     fruit
>
> unknown <- data.frame(ingredient =  "tomato",
+                  sweetness = 6,
+                  crunchiness = 4,
+
+                  class="unknown")
> unknown
  ingredient sweetness crunchiness   class
1     tomato         6           4 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
[1] fruit
Levels: fruit protein vegetable
>
> unknown <- data.frame(ingredient =  c("tomato", "carrot"),
+                   sweetness = c(6,4),
+                   crunchiness = c(4,9),
+                   class=c("unknown", "unknown"))
> unknown
```

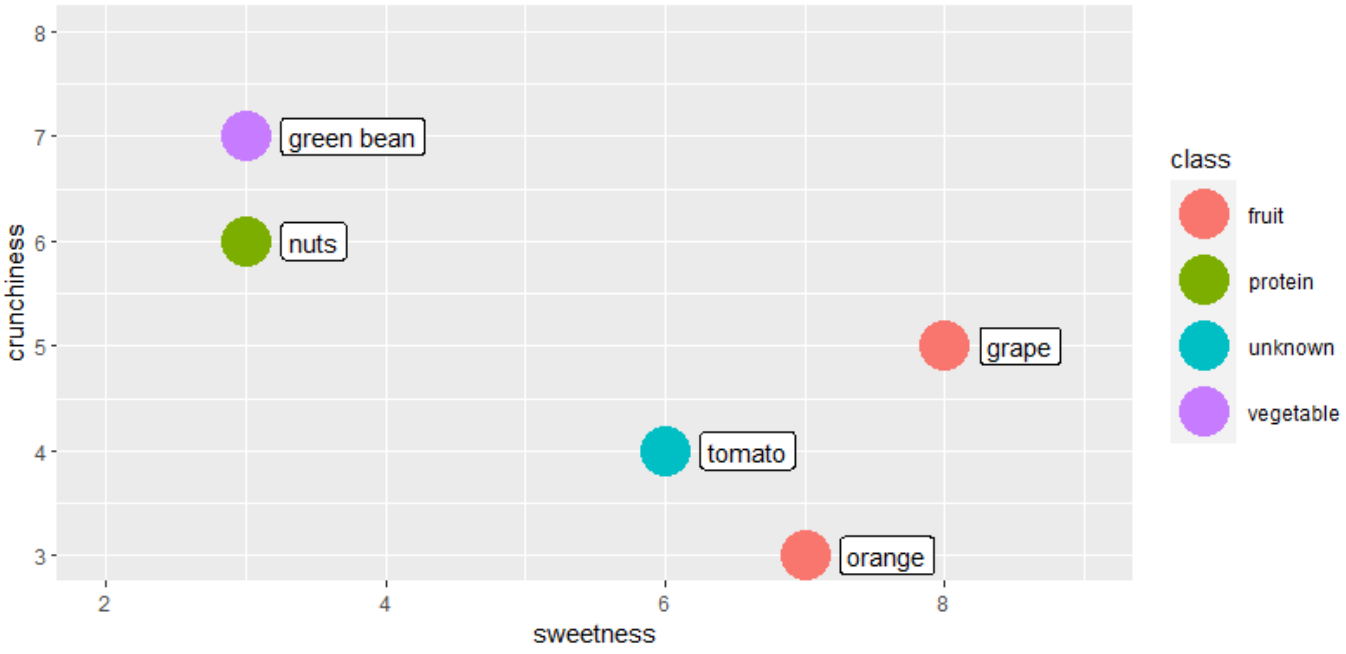
```
  ingredient sweetness crunchiness   class
1     tomato         6           4 unknown
2     carrot         4           9 unknown
```

For K value=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 value=4

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



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 3 **Date of Implementation:** 11. 10. 23

****

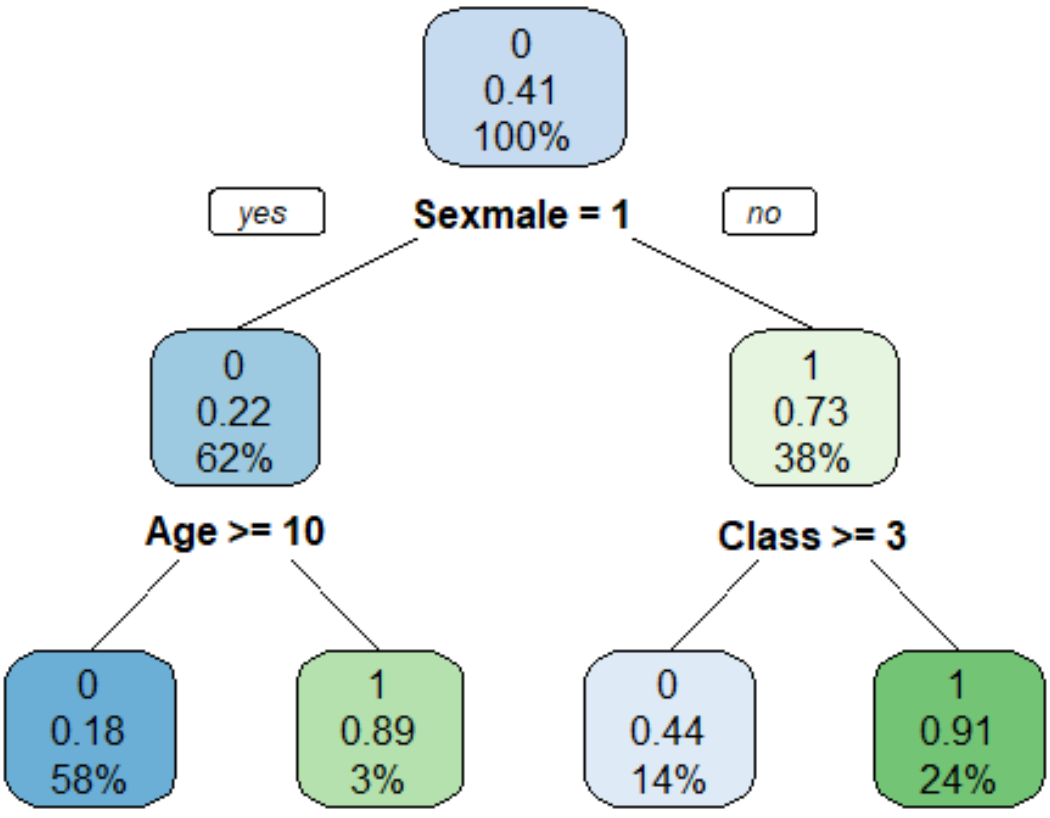
Q2. Using Titanic.CSV file predict which people are more likely to survive after the collision with the iceberg using Decision Trees.

Code:

```
library(caret)
library(FSelector)
library(rpart)
library(rpart.plot)
library(dplyr)
library(xlsx)
library(data.tree)
library(caTools)
df <- read.xlsx("D:\\PESMCOE\\Data Science\\Assign 3\\Titanic.xlsx", sheetIndex=1)
df
summary(df)

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)
```

Output:

```
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
```



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 **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 3 **Date of Implementation:** 11. 10. 23

****

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.

Code:

```
#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)
```

Output:

1st Run:

```
  cerebellum colon endometrium hippocampus kidney liver placenta
1         31     0           0           0      0     0         0
2          5     0           0          31      0     0         0
3          0    34           0           0      0     0         0
4          0     0           0           0     30     0         0
5          2     0           0           0      2     0         6
6          0     0          15           0      7     0         0
7          0     0           0           0      0    26         0
>
```

2nd Run:

```
> table(cl$cluster, tissue_gene_expression$y)

  cerebellum colon endometrium hippocampus kidney liver placenta
1          0    34           0           0      0     0         0
2          2     0           0           2      3     2         0
3          0     0          15           0      0     0         0
4         36     0           0          29      0     0         0
5          0     0           0           0     36     0         0
6          0     0           0           0      0     0         6
7          0     0           0           0      0    24         0
>
```

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 **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 3 **Date of Implementation:** 11. 10. 23

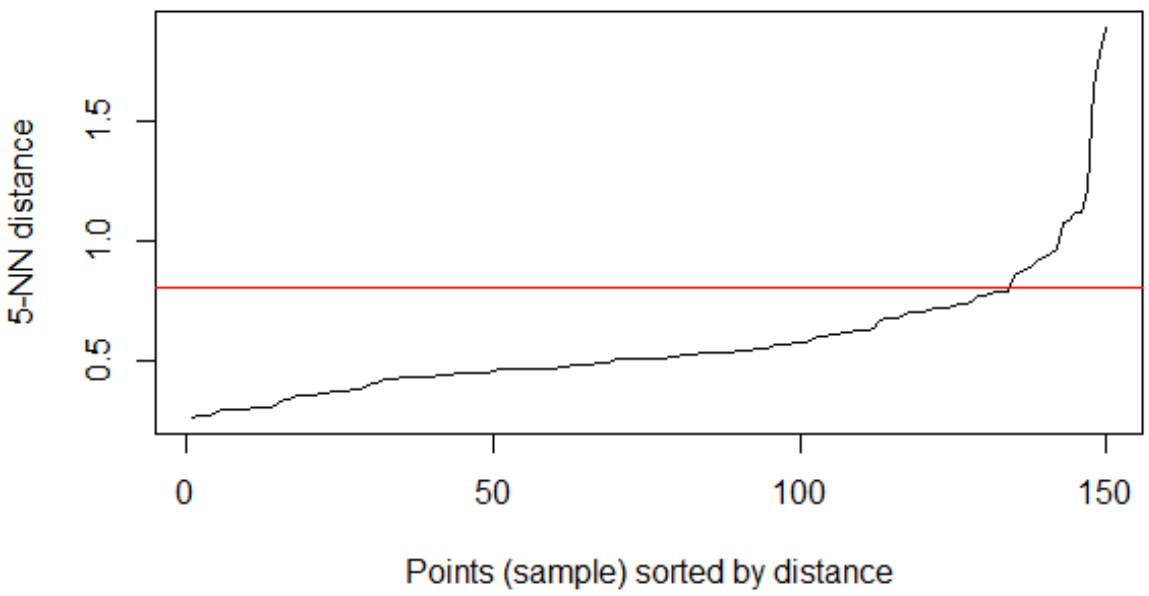
****

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)

Code:

```
df <- iris[, -ncol(iris)]

df <- scale(df)
df <- as.data.frame(df)

library(dbscan)
kNNdistplot(df, k = 5)
abline(h = 0.8, col = "red")
```

Output:



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 4 **Date of Implementation:** 30. 10. 23

****

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.

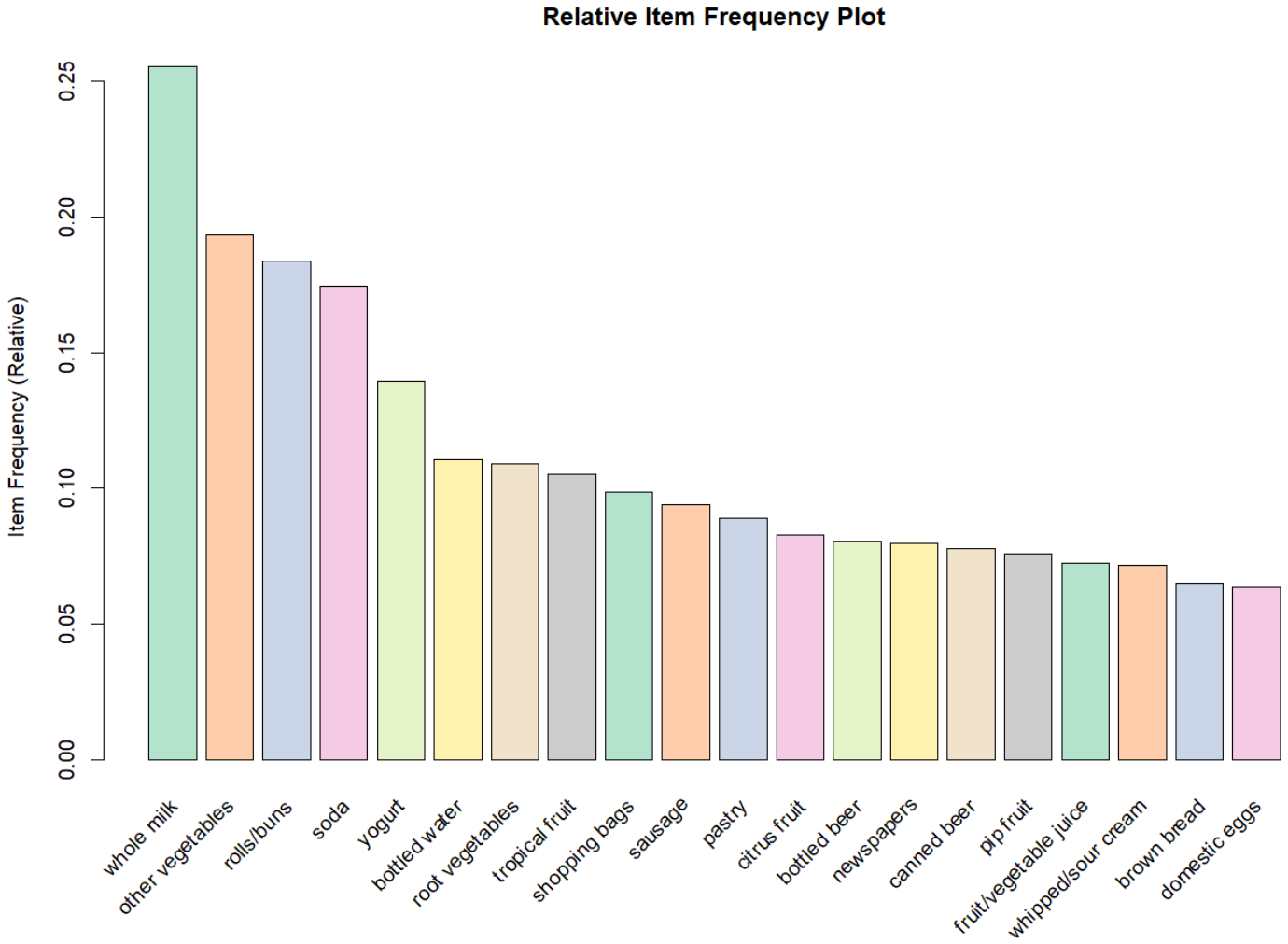
Code:

```
library(arules)
library(arulesViz)
library(RColorBrewer)
library(datasets)
# import dataset
data("Groceries")

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


# using inspect() function
inspect(rules[1:5])

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

# Display the top 5 rules
#top_5_rules <- head(rules, 5)
# Display the top 5 rules
top_5_rules <- head(rules, 5)

# Highlight the strong rules (you can customize the threshold)
strong_rules <- subset(top_5_rules, confidence > 0.9)

# Display the top 5 rules and highlight the strong ones
inspect(strong_rules)
 #using itemFrequencyPlot() function
arules::itemFrequencyPlot(Groceries, topN = 20,
                    col = brewer.pal(8, 'Pastel2'),
                     main = 'Relative Item Frequency Plot',
                     type = "relative",
                     ylab = "Item Frequency (Relative)")
```

Output:

```
> # using inspect() function
> inspect(rules[1:5])
    lhs                          rhs            support     confidence coverage
[1] {liquor, red/blush wine} => {bottled beer} 0.001931876 0.9047619  0.002135231
[2] {curd, cereals}          => {whole milk}   0.001016777 0.9090909  0.001118454
[3] {yogurt, cereals}        => {whole milk}   0.001728521 0.8095238  0.002135231
[4] {butter, jam}            => {whole milk}   0.001016777 0.8333333  0.001220132
[5] {soups, bottled beer}    => {whole milk}   0.001118454 0.9166667  0.001220132
    lift      count
[1] 11.235269 19
[2]  3.557863 10
[3]  3.168192 17
[4]  3.261374 10
[5]  3.587512 11
```

```
>
> # Display the top 5 rules and highlight the strong ones
> inspect(strong_rules)
    lhs                                                rhs          support
[1] {rice, sugar}                                   => {whole milk} 0.001220132
[2] {canned fish, hygiene articles}                 => {whole milk} 0.001118454
[3] {root vegetables, butter, rice}                 => {whole milk} 0.001016777
[4] {root vegetables, whipped/sour cream, flour}    => {whole milk} 0.001728521
[5] {butter, soft cheese, domestic eggs}            => {whole milk} 0.001016777
    confidence coverage    lift     count
[1] 1          0.001220132 3.913649 12
[2] 1          0.001118454 3.913649 11
[3] 1          0.001016777 3.913649 10
[4] 1          0.001728521 3.913649 17
[5] 1          0.001016777 3.913649 10
```



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 **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 4 **Date of Implementation:** 30. 10. 23

****

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

Code:

```
library(arules)

data("Groceries")
#data <- read.csv('your_dataset.csv', header = FALSE)

# Convert the dataset into transactions format
#transactions <- as(Groceries, 'transactions')

# Apply the Eclat algorithm with minimum support
frequent_itemsets <- eclat(Groceries, parameter = list(support = 0.003, minlen=4))

# Display the frequent itemsets
inspect(frequent_itemsets[1:10])

print("Top 10 items frequently bought together")
inspect(sort(frequent_itemsets, by = 'support', decreasing = TRUE)[1:10])
```

Output:

```
>
> # Display the frequent itemsets
> inspect(frequent_itemsets[1:10])
     items                                                                  support     count
[1]  {root vegetables, onions, other vegetables, whole milk}                0.003253686 32
[2]  {other vegetables, whole milk, yogurt, cream cheese }                  0.003457041 34
[3]  {root vegetables, other vegetables, whole milk, frozen vegetables}     0.003863752 38
[4]  {other vegetables, whole milk, yogurt, frozen vegetables}              0.003558719 35
[5]  {beef, root vegetables, other vegetables, whole milk}                  0.003965430 39
[6]  {beef, other vegetables, whole milk, yogurt}                           0.003050330 30
[7]  {beef, other vegetables, whole milk, rolls/buns}                       0.003355363 33
[8]  {tropical fruit, whole milk, curd, yogurt}                             0.003965430 39
[9]  {tropical fruit, other vegetables, whole milk, curd}                   0.003152008 31
[10] {root vegetables, other vegetables, whole milk, curd}                  0.003152008 31
>
> print("Top 10 items frequently bought together")
[1] "Top 10 items frequently bought together"
> inspect(sort(frequent_itemsets, by = 'support', decreasing = TRUE)[1:10])
     items                                                                  support     count
[1]  {root vegetables, other vegetables, whole milk, yogurt}                0.007829181 77
[2]  {tropical fruit, other vegetables, whole milk, yogurt}                 0.007625826 75
[3]  {tropical fruit, root vegetables, other vegetables, whole milk}        0.007015760 69
[4]  {root vegetables, other vegetables, whole milk, rolls/buns}            0.006202339 61
[5]  {other vegetables, whole milk, yogurt, rolls/buns}                     0.005998983 59
[6]  {citrus fruit, root vegetables, other vegetables, whole milk}          0.005795628 57
[7]  {tropical fruit, root vegetables, whole milk, yogurt}                  0.005693950 56
[8]  {other vegetables, whole milk, yogurt, whipped/sour cream}             0.005592272 55
[9]  {pip fruit, root vegetables, other vegetables, whole milk}             0.005490595 54
[10] {root vegetables, other vegetables, whole milk, whipped/sour cream}    0.005185562 51
>
```

****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 5 **Date of Implementation:** 30. 10. 23

****

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

Code:

```
# Physical fitness scores for third graders
scores_a <- c(12, 22, 6, 9, 2, 9, 5, 9, 3, 5, 16, 1, 22, 18)
scores_b <- c(6, 12, 21, 23, 9, 10, 24, 21, 17, 11, 18, 19, 17, 5)
scores_c <- c(14, 16, 19, 19, 18, 3, 4, 21, 16, 20, 15, 14, 17, 4)
scores_d <- c(5, 22, 12, 15, 18, 20, 8, 10, 13, 20, 6, 9, 2, 17)
scores_e <- c(15, 9, 4, 15, 14, 19, 3, 24)

# Combine all scores into one vector
all_scores <- c(scores_a, scores_b, scores_c, scores_d, scores_e)

# Mean
mean_score <- mean(all_scores)
print(paste("Mean:", mean_score))

# Median
median_score <- median(all_scores)
print(paste("Median:", median_score))

# Mode
mode_score <- as.numeric(names(sort(table(all_scores), decreasing = TRUE)[1]))
print(paste("Mode:", mode_score))

# Range
range_score <- max(all_scores) - min(all_scores)
print(paste("Range:", range_score))

# Interquartile Range (IQR)
Q1 <- quantile(all_scores, 0.25)
Q3 <- quantile(all_scores, 0.75)
iqr_score <- Q3 - Q1
print(paste("Interquartile Range (IQR):", iqr_score))
```

Output:

```
> # Mean
> mean_score <- mean(all_scores)
> print(paste("Mean:", mean_score))
[1] "Mean: 13"
>
> # Median
> median_score <- median(all_scores)
> print(paste("Median:", median_score))
[1] "Median: 14"
>
> # Mode
> mode_score <- as.numeric(names(sort(table(all_scores), decreasing = TRUE)[1]))
> print(paste("Mode:", mode_score))
[1] "Mode: 9"
>
> # Range
> range_score <- max(all_scores) - min(all_scores)
> print(paste("Range:", range_score))
[1] "Range: 23"
>
> # Interquartile Range (IQR)
> Q1 <- quantile(all_scores, 0.25)
> Q3 <- quantile(all_scores, 0.75)
> iqr_score <- Q3 - Q1
> print(paste("Interquartile Range (IQR):", iqr_score))
[1] "Interquartile Range (IQR): 10.75"
```

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 **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 5 **Date of Implementation:** 30. 10. 23

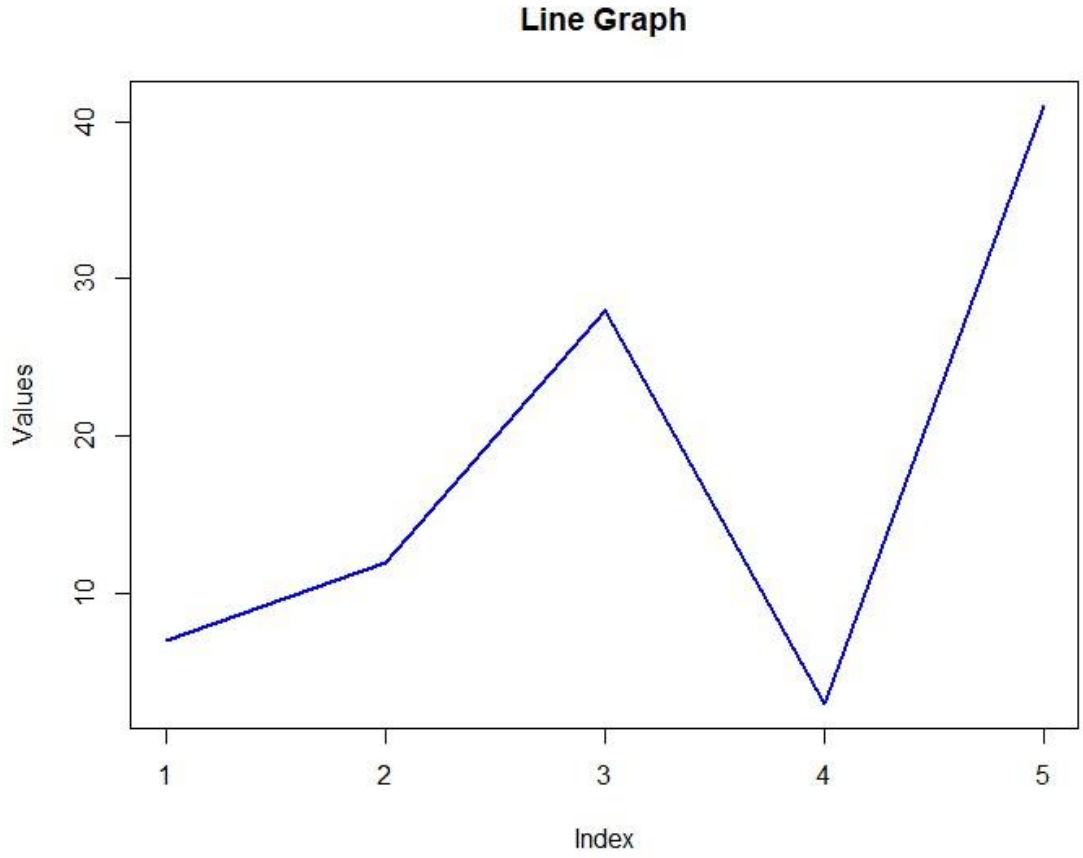
****

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

Code:

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

# Create a line plot
plot(v, type = "l", col = "blue", lwd = 2, xlab = "Index", ylab = "Values", main = "Line Graph")

# Save the plot
png("line_plot.png", width = 800, height = 400)  # Adjust width and height as needed
plot(v, type = "l", col = "blue", lwd = 2, xlab = "Index", ylab = "Values", main = "Line Graph")
dev.off()

# Display the plot (optional)
# You can uncomment the following line if you want to display the plot in the R environment
plot(v, type = "l", col = "blue", lwd = 2, xlab = "Index", ylab = "Values", main = "Line Graph")
```

Output:



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 5 **Date of Implementation:** 30. 10. 23

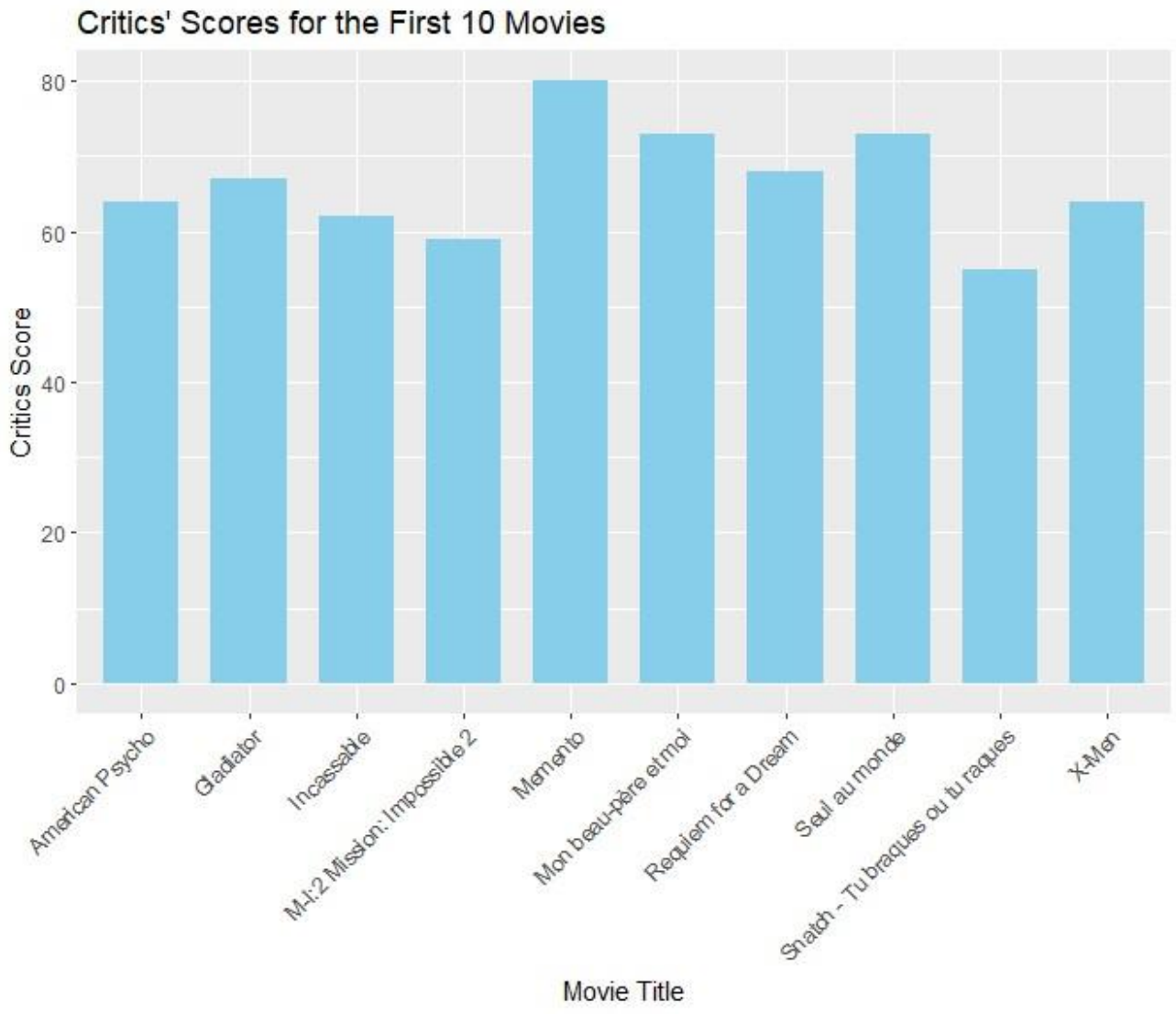
****

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

Code:

```
# Load libraries
library(ggplot2)
library(dplyr)
# Read the CSV file
movies_data <- read.csv("D:\\PESMCOE\\Data Science\\Assign 5\\movies_data.csv")
# Subset the data for the first 10 movies
first_10_movies <- head(movies_data, 10)
# Create a bar chart of critics_score for the first 10 movies
bar_plot <- ggplot(first_10_movies, aes(x = Movie, y = criticRating)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.7) +
  labs(title = "Critics' Scores for the First 10 Movies",
       x = "Movie Title",
       y = "Critics Score") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for better readability
# Save the plot as a PNG file
ggsave("critics_score_bar_chart.png", bar_plot, width = 10, height = 6, units = "in")

# Display the plot (optional)
print(bar_plot)
```

Output:


Critics' Scores for the First 10 Movies
Critics Score
80
60
40
20
0
American Psycho
Gladiator
Incassable
M:I:2 Mission: Impossible 2
Memento
Mon beau-père et moi
Requiem for a Dream
Seul au monde
Snatch - Tu braques ou tu raques
X-Men
Movie Title

****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 5 **Date of Implementation:** 30. 10. 23

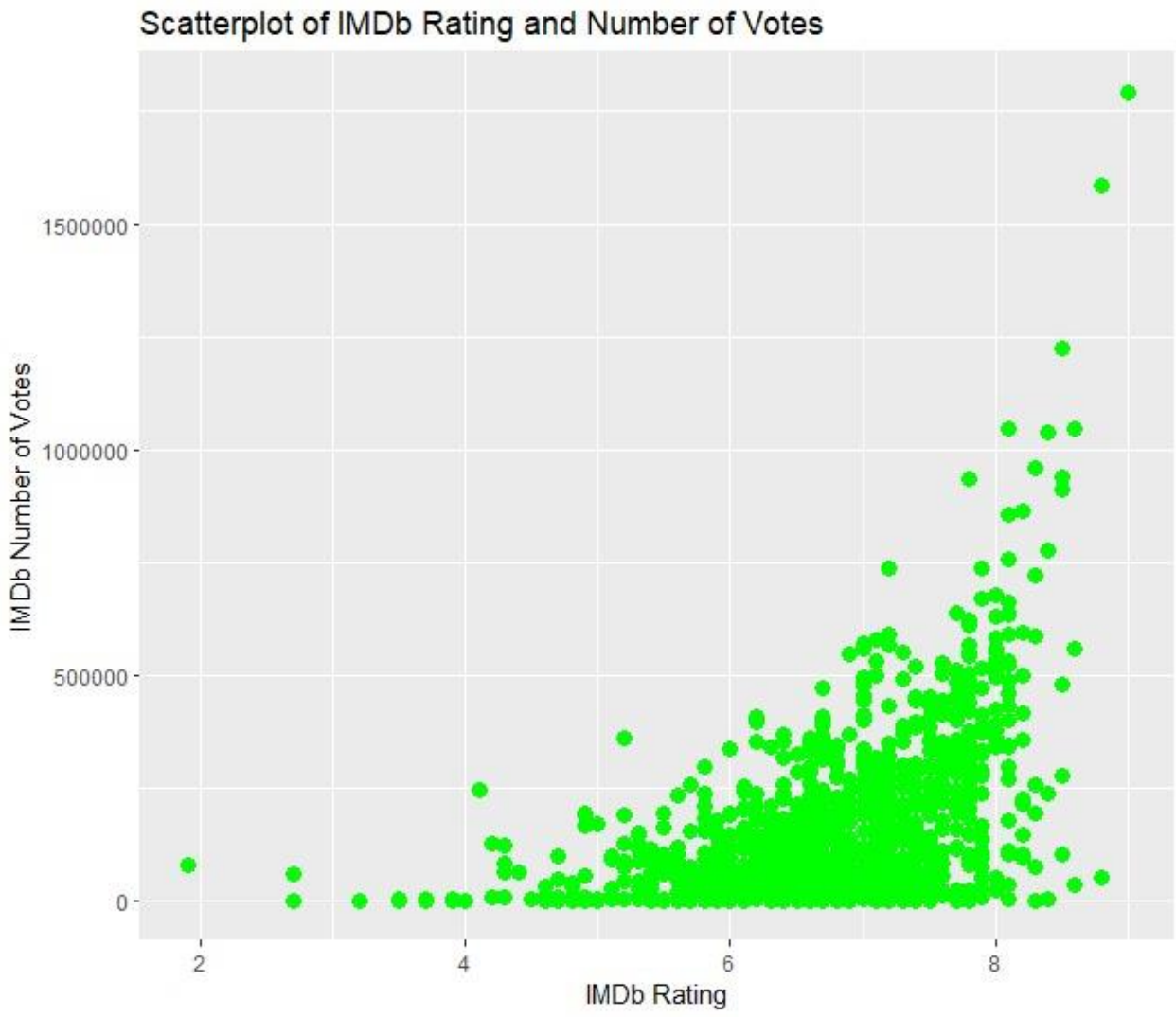
****

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

Code:

```
# Load libraries
library(ggplot2)
library(dplyr)

# Read the CSV file
movies_data <- read.csv("D:\\PESMCOE\\Data Science\\Assign 5\\IMDB-Movie-Data.csv")

# Create a scatterplot of imdb_rating and imdb_num_votes
scatter_plot <- ggplot(movies_data, aes(x = Rating, y = Votes)) +
  geom_point(color = "green", size = 3) +
  labs(title = "Scatterplot of IMDb Rating and Number of Votes",
       x = "IMDb Rating",
       y = "IMDb Number of Votes")

# Save the plot as a PNG file
ggsave("imdb_rating_vs_num_votes_scatterplot.png", scatter_plot, width = 10, height = 6, units = "in")

# Display the plot (optional)
print(scatter_plot)
```

Output:


Scatterplot of IMDb Rating and Number of Votes
IMDb Number of Votes
1500000
1000000
500000
0
2
4
6
8
IMDb Rating

****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 5 **Date of Implementation:** 30. 10. 23

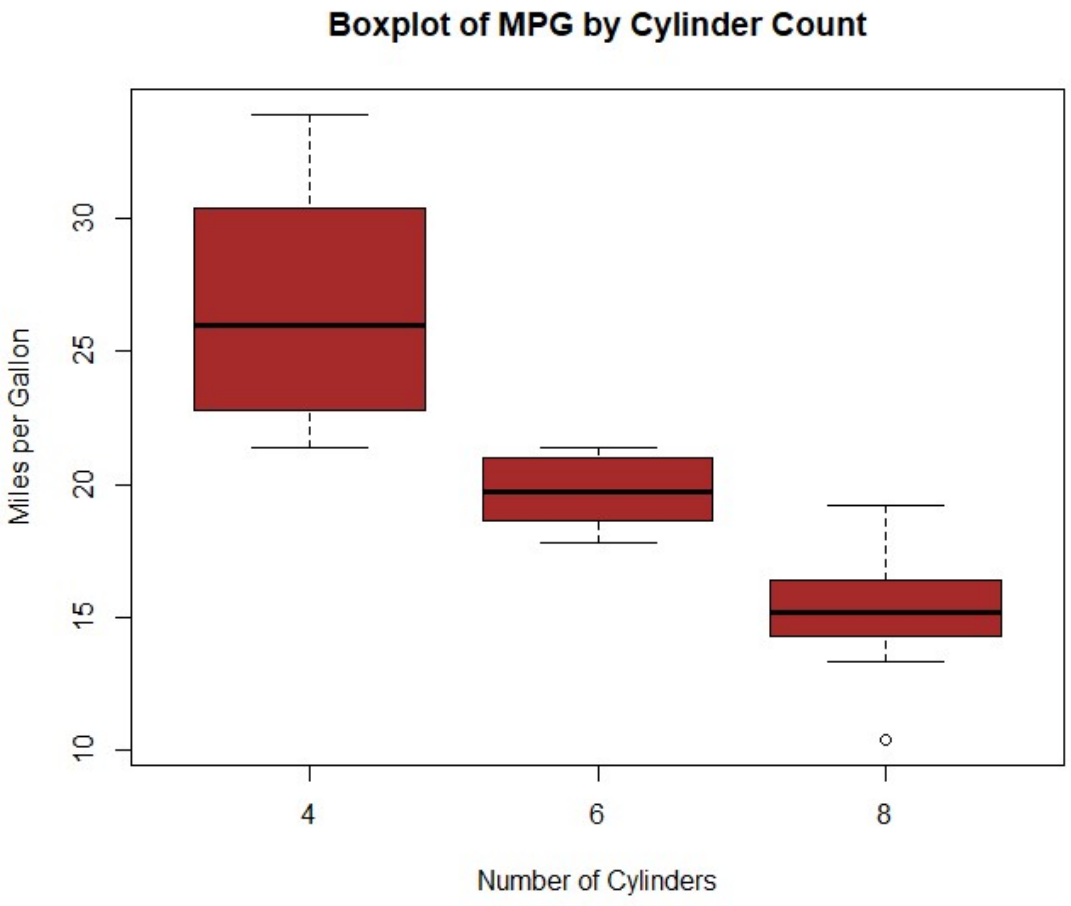
****

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

Code:

```
# Load the dataset
data(mtcars)

# Create a boxplot for mpg and cyl
boxplot(mpg ~ cyl, data = mtcars,
    col = "brown",
    main = "Boxplot of MPG by Cylinder Count",
    xlab = "Number of Cylinders",
    ylab = "Miles per Gallon")
```

Output:



****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 5 **Date of Implementation:** 30. 10. 23

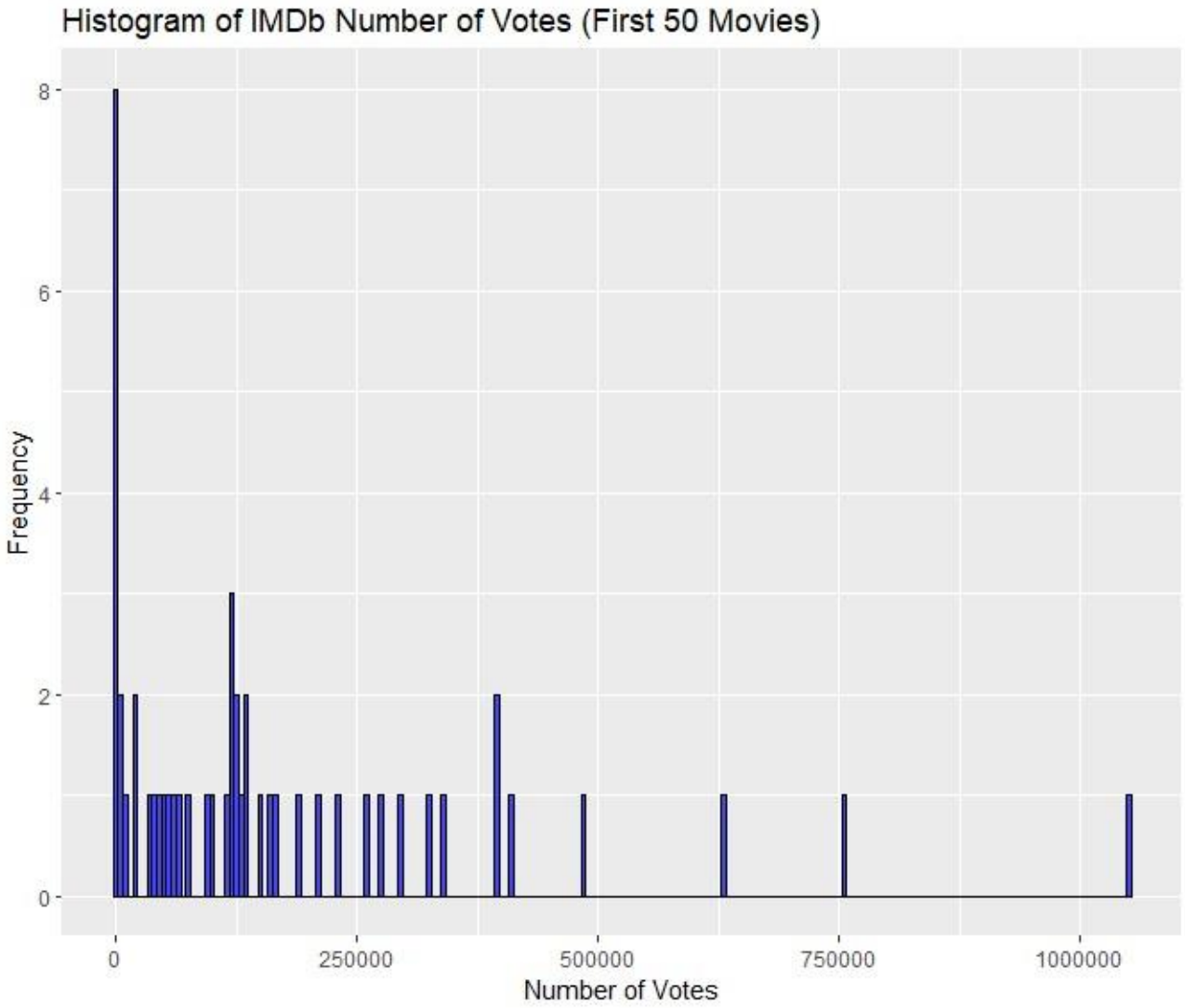
****

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

Code:

```
# Load libraries
library(ggplot2)
library(dplyr)

# Read the CSV file
movies_data <- read.csv("D:\\PESMCOE\\Data Science\\Assign 5\\IMDB-Movie-Data.csv")

# Subset the data for the first 50 movies
first_50_movies <- head(movies_data, 50)

# Create a histogram of imdb_num_votes
histogram_plot <- ggplot(first_50_movies, aes(x = Votes)) +
  geom_histogram(binwidth = 5000, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Histogram of IMDb Number of Votes (First 50 Movies)",
       x = "Number of Votes",
       y = "Frequency")

# Save the plot as a PNG file
ggsave("imdb_num_votes_histogram_50_movies.png", histogram_plot, width = 20, height = 6, units = "in")

# Display the plot (optional)
print(histogram_plot)
```

Output:


Histogram of IMDb Number of Votes (First 50 Movies)
Frequency
0
2
4
6
8
0
250000
500000
750000
1000000
Number of Votes

****

Class: SY-MCA **Shift / Div:** A **Batch:** S2 **Roll Number:** 52043

Name: Vanessa Reetu Prashant More **Assignment No:** 6 **Date of Implementation:** 23. 11. 23

****

Q. Design and Develop real-time Data Science Application (e.g. Image Recognition/ Intelligent Assistant/ Recommendation System/ Fake News Detection/Emotion Recognition/Chatbot /Other)

Credit Card Fraud Detection

Code:

```
library(dplyr) # for data manipulation
library(stringr) # for data manipulation
library(caret) # for sampling
library(caTools) # for train/test split
library(ggplot2) # for data visualization
library(corrplot) # for correlations
library(Rtsne) # for tsne plotting
library(DMwR) # for smote implementation
library(ROSE)# for ROSE sampling
library(rpart)# for decision tree model
library(Rborist)# for random forest model
library(xgboost) # for xgboost model

# function to set plot height and width
fig <- function(width, heigth){
    options(repr.plot.width = width, repr.plot.height = heigth)
}

# loading the data
df = read.csv('../input/creditcardfraud/creditcard.csv')

head(df)
```

```
  Time        V1         V2        V3         V4
1    0 -1.3598071 -0.07278117 2.5363467  1.3781552
2    0  1.1918571  0.26615071 0.1664801  0.4481541
3    1 -1.3583541 -1.34016307 1.7732093  0.3797796
4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913
5    2 -1.1582331  0.87773675 1.5487178  0.4030339
6    2 -0.4259659  0.96052304 1.1411093 -0.1682521
         V5         V6         V7         V8
1 -0.33832077  0.46238778  0.23959855  0.09869790
2  0.06001765 -0.08236081 -0.07880298  0.08510165
3 -0.50319813  1.80049938  0.79146096  0.24767579
4 -0.01030888  1.24720317  0.23760894  0.37743587
5 -0.40719338  0.09592146  0.59294075 -0.27053268
6  0.42098688 -0.02972755  0.47620095  0.26031433
        V9        V10        V11        V12
```

```
1  0.3637870  0.09079417 -0.5515995 -0.61780086
2 -0.2554251 -0.16697441  1.6127267  1.06523531
3 -1.5146543  0.20764287  0.6245015  0.06608369
4 -1.3870241 -0.05495192 -0.2264873  0.17822823
5  0.8177393  0.75307443 -0.8228429  0.53819555
6 -0.5686714 -0.37140720  1.3412620  0.35989384
         V13        V14        V15        V16
1 -0.9913898 -0.3111694  1.4681770 -0.4704005
2  0.4890950 -0.1437723  0.6355581  0.4639170
3  0.7172927 -0.1659459  2.3458649 -2.8900832
4  0.5077569 -0.2879237 -0.6314181 -1.0596472
5  1.3458516 -1.1196698  0.1751211 -0.4514492
6 -0.3580907 -0.1371337  0.5176168  0.4017259
          V17         V18         V19         V20
1  0.20797124  0.02579058  0.40399296  0.25141210
2 -0.11480466 -0.18336127 -0.14578304 -0.06908314
3  1.10996938 -0.12135931 -2.26185710  0.52497973
4 -0.68409279  1.96577500 -1.23262197 -0.20803778
5 -0.23703324 -0.03819479  0.80348692  0.40854236
6 -0.05813282  0.06865315 -0.03319379  0.08496767
           V21          V22         V23
1 -0.018306778  0.277837576 -0.11047391
2 -0.225775248 -0.638671953  0.10128802
3  0.247998153  0.771679402  0.90941226
4 -0.108300452  0.005273597 -0.19032052
5 -0.009430697  0.798278495 -0.13745808
6 -0.208253515 -0.559824796 -0.02639767
          V24        V25        V26          V27
1  0.06692807  0.1285394 -0.1891148  0.133558377
2 -0.33984648  0.1671704  0.1258945 -0.008983099
3 -0.68928096 -0.3276418 -0.1390966 -0.055352794
4 -1.17557533  0.6473760 -0.2219288  0.062722849
5  0.14126698 -0.2060096  0.5022922  0.219422230
6 -0.37142658 -0.2327938  0.1059148  0.253844225
          V28 Amount Class
1 -0.02105305 149.62      0
2  0.01472417   2.69      0
3 -0.05975184 378.66      0
4  0.06145763 123.50      0
5  0.21515315  69.99      0
6  0.08108026   3.67      0
```

str(df)

```
'data.frame':    284807 obs. of  31 variables:
 $ Time  : num  0 0 1 1 2 2 4 7 7 9 ...
 $ V1    : num  -1.36 1.192 -1.358 -0.966 -1.158 ...
 $ V2    : num  -0.0728 0.2662 -1.3402 -0.1852 0.8777 ...
 $ V3    : num  2.536 0.166 1.773 1.793 1.549 ...
 $ V4    : num  1.378 0.448 0.38 -0.863 0.403 ...
 $ V5    : num  -0.3383 0.06 -0.5032 -0.0103 -0.4072 ...
 $ V6    : num  0.4624 -0.0824 1.8005 1.2472 0.0959 ...
```

```
 $ V7    : num  0.2396 -0.0788 0.7915 0.2376 0.5929 ...
 $ V8    : num  0.0987 0.0851 0.2477 0.3774 -0.2705 ...
 $ V9    : num  0.364 -0.255 -1.515 -1.387 0.818 ...
 $ V10   : num  0.0908 -0.167 0.2076 -0.055 0.7531 ...
 $ V11   : num  -0.552 1.613 0.625 -0.226 -0.823 ...
 $ V12   : num  -0.6178 1.0652 0.0661 0.1782 0.5382 ...
 $ V13   : num  -0.991 0.489 0.717 0.508 1.346 ...
 $ V14   : num  -0.311 -0.144 -0.166 -0.288 -1.12 ...
 $ V15   : num  1.468 0.636 2.346 -0.631 0.175 ...
 $ V16   : num  -0.47 0.464 -2.89 -1.06 -0.451 ...
 $ V17   : num  0.208 -0.115 1.11 -0.684 -0.237 ...
 $ V18   : num  0.0258 -0.1834 -0.1214 1.9658 -0.0382 ...
 $ V19   : num  0.404 -0.146 -2.262 -1.233 0.803 ...
 $ V20   : num  0.2514 -0.0691 0.525 -0.208 0.4085 ...
 $ V21   : num  -0.01831 -0.22578 0.248 -0.1083 -0.00943 ...
 $ V22   : num  0.27784 -0.63867 0.77168 0.00527 0.79828 ...
 $ V23   : num  -0.11 0.101 0.909 -0.19 -0.137 ...
 $ V24   : num  0.0669 -0.3398 -0.6893 -1.1756 0.1413 ...
 $ V25   : num  0.129 0.167 -0.328 0.647 -0.206 ...
 $ V26   : num  -0.189 0.126 -0.139 -0.222 0.502 ...
 $ V27   : num  0.13356 -0.00898 -0.05535 0.06272 0.21942 ...
 $ V28   : num  -0.0211 0.0147 -0.0598 0.0615 0.2152 ...
 $ Amount: num  149.62 2.69 378.66 123.5 69.99 ...
 $ Class : int  0 0 0 0 0 0 0 0 0 0 ...

summary(df)

      Time             V1                  V2                 V3
 Min.   :     0   Min.   :-56.40751   Min.   :-72.71573   Min.   :-48.3256
 1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -0.59855   1st Qu.: -0.8904
 Median : 84692   Median :  0.01811   Median :  0.06549   Median :  0.1799
 Mean   : 94814   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.0000
 3rd Qu.:139320   3rd Qu.:  1.31564   3rd Qu.:  0.80372   3rd Qu.:  1.0272
 Max.   :172792   Max.   :  2.45493   Max.   : 22.05773   Max.   :  9.3826
       V4                 V5                  V6                 V7
 Min.   :-5.68317   Min.   :-113.74331   Min.   :-26.1605   Min.   :-43.5572
 1st Qu.:-0.84864   1st Qu.:  -0.69160   1st Qu.: -0.7683   1st Qu.: -0.5541
 Median :-0.01985   Median :  -0.05434   Median : -0.2742   Median :  0.0401
 Mean   : 0.00000   Mean   :   0.00000   Mean   :  0.0000   Mean   :  0.0000
 3rd Qu.: 0.74334   3rd Qu.:   0.61193   3rd Qu.:  0.3986   3rd Qu.:  0.5704
 Max.   :16.87534   Max.   :  34.80167   Max.   : 73.3016   Max.   :120.5895
       V8                 V9                 V10                V11
 Min.   :-73.21672   Min.   :-13.43407   Min.   :-24.58826   Min.   :-4.79747
 1st Qu.: -0.20863   1st Qu.: -0.64310   1st Qu.: -0.53543   1st Qu.:-0.76249
 Median :  0.02236   Median : -0.05143   Median : -0.09292   Median :-0.03276
 Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.00000
 3rd Qu.:  0.32735   3rd Qu.:  0.59714   3rd Qu.:  0.45392   3rd Qu.: 0.73959
 Max.   : 20.00721   Max.   : 15.59500   Max.   : 23.74514   Max.   :12.01891
      V12                V13                V14                V15
 Min.   :-18.6837   Min.   :-5.79188   Min.   :-19.2143   Min.   :-4.49894
 1st Qu.: -0.4056   1st Qu.:-0.64854   1st Qu.: -0.4256   1st Qu.:-0.58288
 Median :  0.1400   Median :-0.01357   Median :  0.0506   Median : 0.04807
 Mean   :  0.0000   Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000
```

```
 3rd Qu.:  0.6182   3rd Qu.: 0.66251   3rd Qu.:  0.4931   3rd Qu.: 0.64882
 Max.   :  7.8484   Max.   : 7.12688   Max.   : 10.5268   Max.   : 8.87774
     V16                V17                V18
 Min.   :-14.12985   Min.   :-25.16280   Min.   :-9.498746
 1st Qu.: -0.46804   1st Qu.: -0.48375   1st Qu.:-0.498850
 Median :  0.06641   Median : -0.06568   Median :-0.003636
 Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.000000
 3rd Qu.:  0.52330   3rd Qu.:  0.39968   3rd Qu.: 0.500807
 Max.   : 17.31511   Max.   :  9.25353   Max.   : 5.041069
     V19                V20                V21
 Min.   :-7.213527   Min.   :-54.49772   Min.   :-34.83038
 1st Qu.:-0.456299   1st Qu.: -0.21172   1st Qu.: -0.22839
 Median : 0.003735   Median : -0.06248   Median : -0.02945
 Mean   : 0.000000   Mean   :  0.00000   Mean   :  0.00000
 3rd Qu.: 0.458949   3rd Qu.:  0.13304   3rd Qu.:  0.18638
 Max.   : 5.591971   Max.   : 39.42090   Max.   : 27.20284
     V22                V23                V24
 Min.   :-10.933144   Min.   :-44.80774   Min.   :-2.83663
 1st Qu.: -0.542350   1st Qu.: -0.16185   1st Qu.:-0.35459
 Median :  0.006782   Median : -0.01119   Median : 0.04098
 Mean   :  0.000000   Mean   :  0.00000   Mean   : 0.00000
 3rd Qu.:  0.528554   3rd Qu.:  0.14764   3rd Qu.: 0.43953
 Max.   : 10.503090   Max.   : 22.52841   Max.   : 4.58455
     V25                V26                V27
 Min.   :-10.29540   Min.   :-2.60455   Min.   :-22.565679
 1st Qu.: -0.31715   1st Qu.:-0.32698   1st Qu.: -0.070840
 Median :  0.01659   Median :-0.05214   Median :  0.001342
 Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.000000
 3rd Qu.:  0.35072   3rd Qu.: 0.24095   3rd Qu.:  0.091045
 Max.   :  7.51959   Max.   : 3.51735   Max.   : 31.612198
     V28               Amount             Class
 Min.   :-15.43008   Min.   :    0.00   Min.   :0.000000
 1st Qu.: -0.05296   1st Qu.:    5.60   1st Qu.:0.000000
 Median :  0.01124   Median :   22.00   Median :0.000000
 Mean   :  0.00000   Mean   :   88.35   Mean   :0.001728
 3rd Qu.:  0.07828   3rd Qu.:   77.17   3rd Qu.:0.000000
 Max.   : 33.84781   Max.   :25691.16   Max.   :1.000000
```

```
colSums(is.na(df))
```

```
Time0V10V20V30V40V50V60V70V80V90V100V110V120V130V140V150V160V170V180V190V20
0V210V220V230V240V250V260V270V280Amount0Class0
```

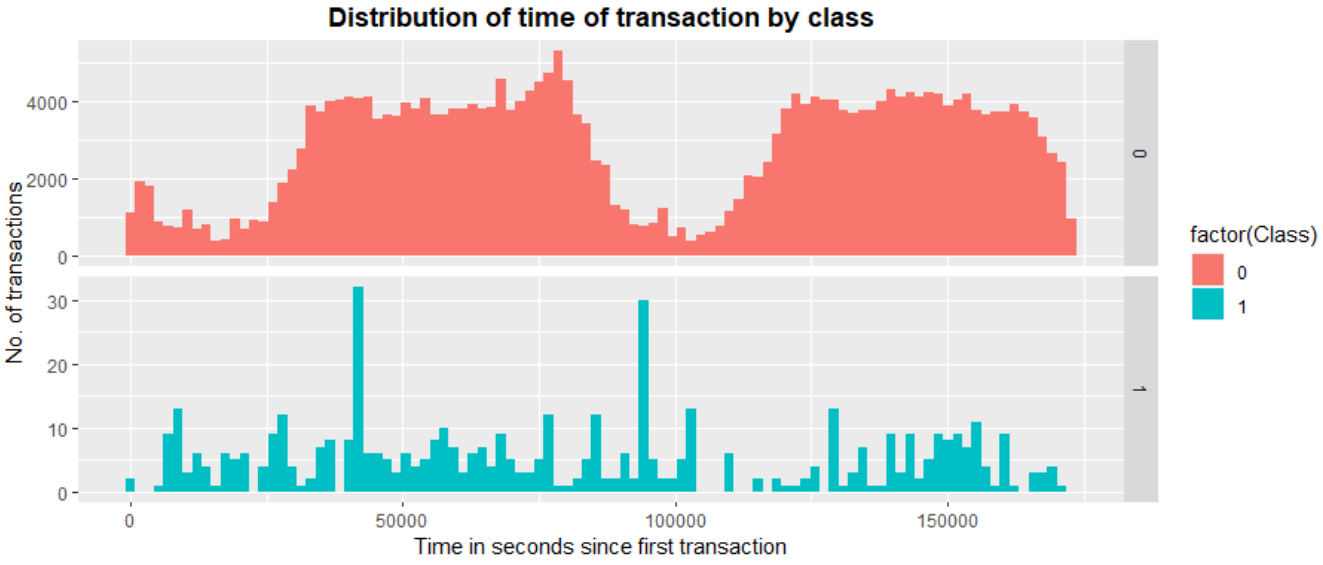
```
# checking class imbalance
table(df$Class)
```

```
0      1
284315    492
```

```
# class imbalance in percentage
prop.table(table(df$Class))
         0          1
0.998272514 0.001727486
```

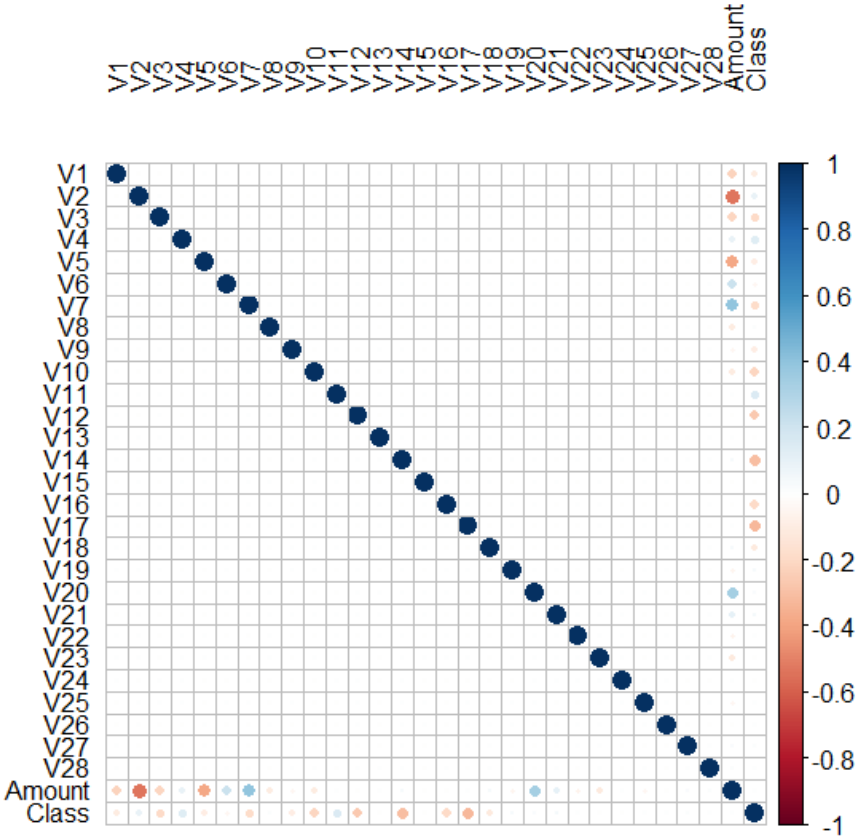
```
fig(12, 8)
common_theme <- theme(plot.title = element_text(hjust = 0.5, face = "bold"))

ggplot(data = df, aes(x = factor(Class),
                      y = prop.table(stat(count)), fill = factor(Class),
                      label = scales::percent(prop.table(stat(count))))) +
  geom_bar(position = "dodge") +
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_x_discrete(labels = c("no fraud", "fraud"))+
  scale_y_continuous(labels = scales::percent)+
  labs(x = 'Class', y = 'Percentage') +
  ggtitle("Distribution of class labels") +
  common_theme
```

```
fig(14, 8)
df %>%
  ggplot(aes(x = Time, fill = factor(Class))) + geom_histogram(bins = 100)+
  labs(x = 'Time in seconds since first transaction', y = 'No. of transactions') +
  ggtitle('Distribution of time of transaction by class') +
  facet_grid(Class ~ ., scales = 'free_y') + common_theme
```




```
fig(14, 8)
ggplot(df, aes(x = factor(Class), y = Amount)) + geom_boxplot() +
labs(x = 'Class', y = 'Amount') +
ggtitle("Distribution of transaction amount by class") + common_theme
```

```
fig(14, 8)
correlations <- cor(df[,-1],method="pearson")
corrplot(correlations, number.cex = .9, method = "circle", type = "full", tl.cex=0.8,tl.col = "black")
```




```
#Remove 'Time' variable
df <- df[,-1]

#Change 'Class' variable to factor
df$Class <- as.factor(df$Class)
levels(df$Class) <- c("Not_Fraud", "Fraud")

#Scale numeric variables

df[,-30] <- scale(df[,-30])

head(df)
```

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	…	V21	V22	V23
V24	V25	V26	V27	V28	Amount	Class							
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	…	<dbl>	<dbl>	<dbl>
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>						
1	-0.6942411	-0.04407485	1.6727706	0.9733638	-0.245116153	0.34706734							
0.1936786	0.08263713	0.3311272	0.08338540	…	-0.02492332	0.382853766	-0.17691102	0.1105067	0.2465850	-0.3921697	0.33089104	-0.06378104	0.24496383
Not_Fraud													
2	0.6084953	0.16117564	0.1097969	0.3165224	0.043483276	-0.06181986	-0.0637001	0.07125336	-0.2324938	-0.15334936	…	-0.30737626	-0.880075209
0.16220090	-0.5611296	0.3206933	0.2610690	-0.02225564	0.04460744	-0.34247394	Not_Fraud						

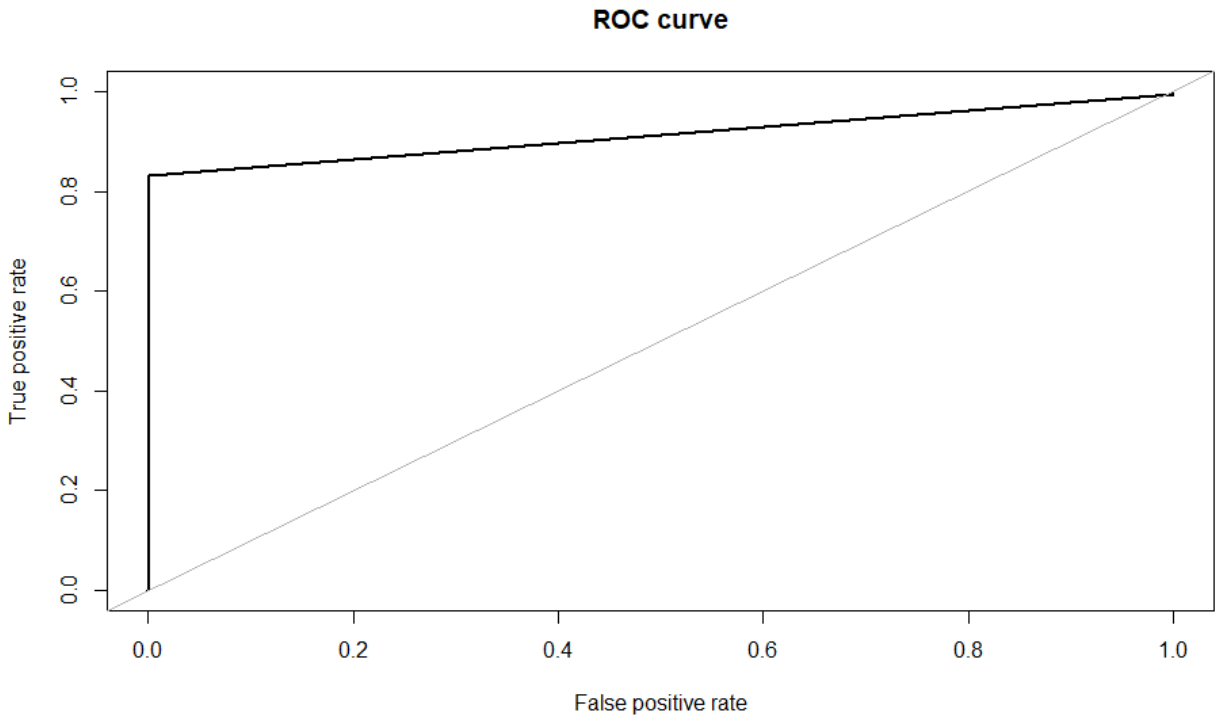
```
3       -0.6934992     -0.81157640   1.1694664      0.2682308      -0.364571146  1.35145121
        0.6397745      0.20737237    -1.3786729     0.19069928     ⋯        0.33763110     1.063356404
        1.45631719     -1.1380901    -0.6285356     -0.2884462     -0.13713661    -0.18102051
        1.16068389     Not_Fraud
4       -0.4933240     -0.11216923   1.1825144      -0.6097256     -0.007468867  0.93614819
        0.1920703      0.31601704    -1.2625010     -0.05046786    ⋯        -0.14744304    0.007266895  -
0.30477601   -1.9410237     1.2419015      -0.4602165     0.15539593     0.18618826     0.14053401
        Not_Fraud
5       -0.5913287     0.53154012    1.0214099      0.2846549      -0.295014918  0.07199846
        0.4793014      -0.22650983   0.7443250      0.69162382     ⋯        -0.01283920    1.100009340  -
0.22012301   0.2332497      -0.3952009     1.0416095      0.54361884     0.65181477     -0.07340321
        Not_Fraud
6       -0.2174742     0.58167387    0.7525841      -0.1188331     0.305008424   -0.02231344
        0.3849353      0.21795429    -0.5176177     -0.34110050    ⋯        -0.28352172    -0.771425648 -
0.04227277   -0.6132723     -0.4465828     0.2196368      0.62889938     0.24563577     -0.33855582
        Not_Fraud
```

```
set.seed(123)
split <- sample.split(df$Class, SplitRatio = 0.7)
train <-  subset(df, split == TRUE)
test <- subset(df, split == FALSE)

# class ratio initially
table(train$Class)
Not_Fraud     Fraud
  199020       344

# downsampling
set.seed(9560)
down_train <- downSample(x = train[, -ncol(train)],
                         y = train$Class)
table(down_train$Class)
Not_Fraud     Fraud
     344       344

# upsampling
set.seed(9560)
up_train <- upSample(x = train[, -ncol(train)],
                         y = train$Class)
table(up_train$Class)
Not_Fraud     Fraud
  199020    199020



# smote
set.seed(9560)
smote_train <- SMOTE(Class ~ ., data  = train)

table(smote_train$Class)
Not_Fraud     Fraud
    1376      1032
```

```
# rose
set.seed(9560)
rose_train <- ROSE(Class ~ ., data  = train)$data

table(rose_train$Class)
Not_Fraud     Fraud
   99844     99520
```

Decision Trees (CART)

```
#CART Model Performance on imbalanced data
set.seed(5627)

orig_fit <- rpart(Class ~ ., data = train)

#Evaluate model performance on test set
pred_orig <- predict(orig_fit, newdata = test, method = "class")

roc.curve(test$Class, pred_orig[,2], plotit = TRUE)
```



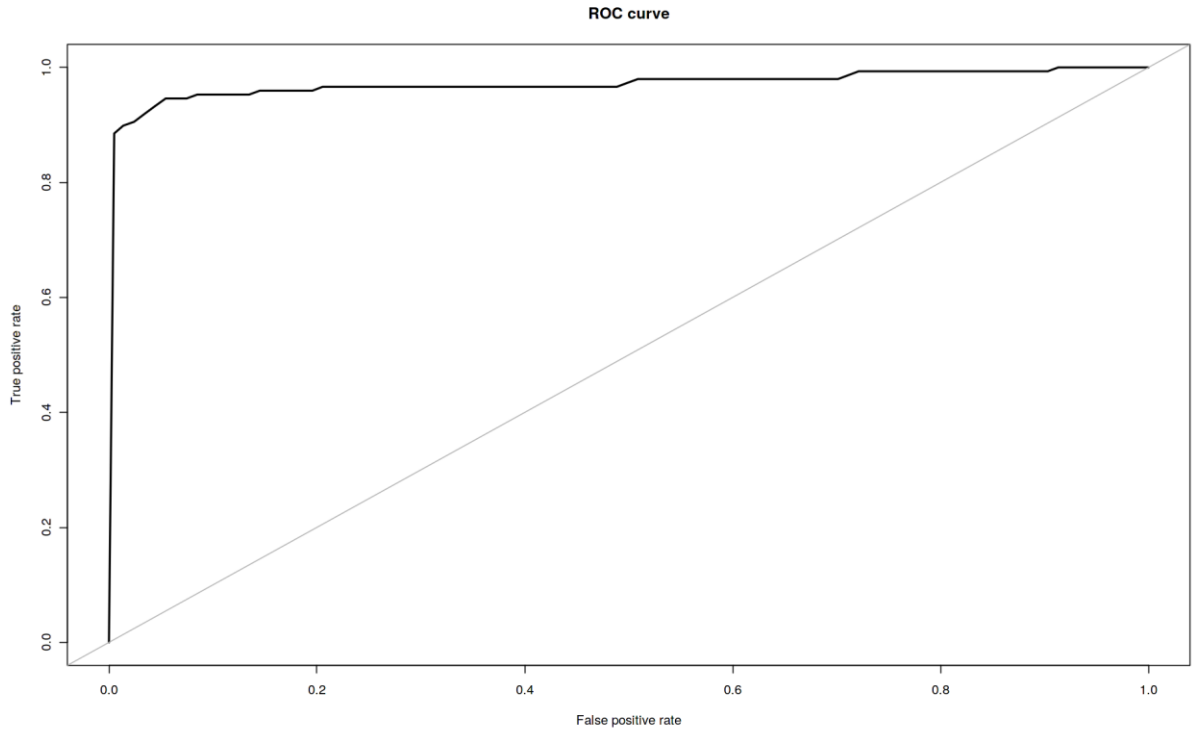

```
set.seed(5627)
# Build down-sampled model


down_fit <- rpart(Class ~ ., data = down_train)


set.seed(5627)
# Build up-sampled model
```

```
up_fit <- rpart(Class ~ ., data = up_train)


set.seed(5627)
# Build smote model


smote_fit <- rpart(Class ~ ., data = smote_train)

set.seed(5627)
# Build rose model


rose_fit <- rpart(Class ~ ., data = rose_train)
# AUC on down-sampled data
pred_down <- predict(down_fit, newdata = test)

print('Fitting model to downsampled data')
roc.curve(test$Class, pred_down[,2], plotit = FALSE)
[1] "Fitting model to downsampled data"
Area under the curve (AUC): 0.942

# AUC on up-sampled data
pred_up <- predict(up_fit, newdata = test)

print('Fitting model to upsampled data')
roc.curve(test$Class, pred_up[,2], plotit = FALSE)
[1] "Fitting model to upsampled data"
Area under the curve (AUC): 0.943
# AUC on up-sampled data
pred_smote <- predict(smote_fit, newdata = test)

print('Fitting model to smote data')
roc.curve(test$Class, pred_smote[,2], plotit = FALSE)
[1] "Fitting model to smote data"
Area under the curve (AUC): 0.934
# AUC on up-sampled data
pred_rose <- predict(rose_fit, newdata = test)

print('Fitting model to rose data')
roc.curve(test$Class, pred_rose[,2], plotit = FALSE)
[1] "Fitting model to rose data"
Area under the curve (AUC): 0.942
```

Logistic Regression

```
glm_fit <- glm(Class ~ ., data = up_train, family = 'binomial')

pred_glm <- predict(glm_fit, newdata = test, type = 'response')

roc.curve(test$Class, pred_glm, plotit = TRUE)
Warning message:
```

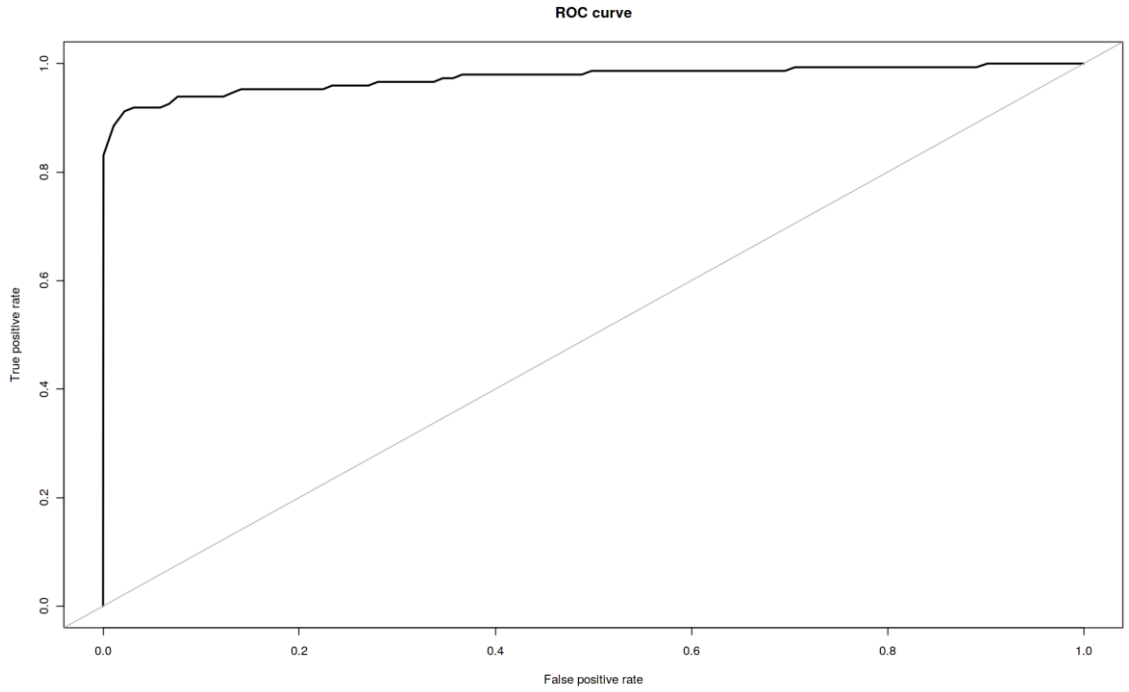
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
Area under the curve (AUC): 0.971



Random Forest

```
x = up_train[, -30]
y = up_train[,30]

rf_fit <- Rborist(x, y, ntree = 1000, minNode = 20, maxLeaf = 13)


rf_pred <- predict(rf_fit, test[,-30], ctgCensus = "prob")
prob <- rf_pred$prob

roc.curve(test$Class, prob[,2], plotit = TRUE)
```

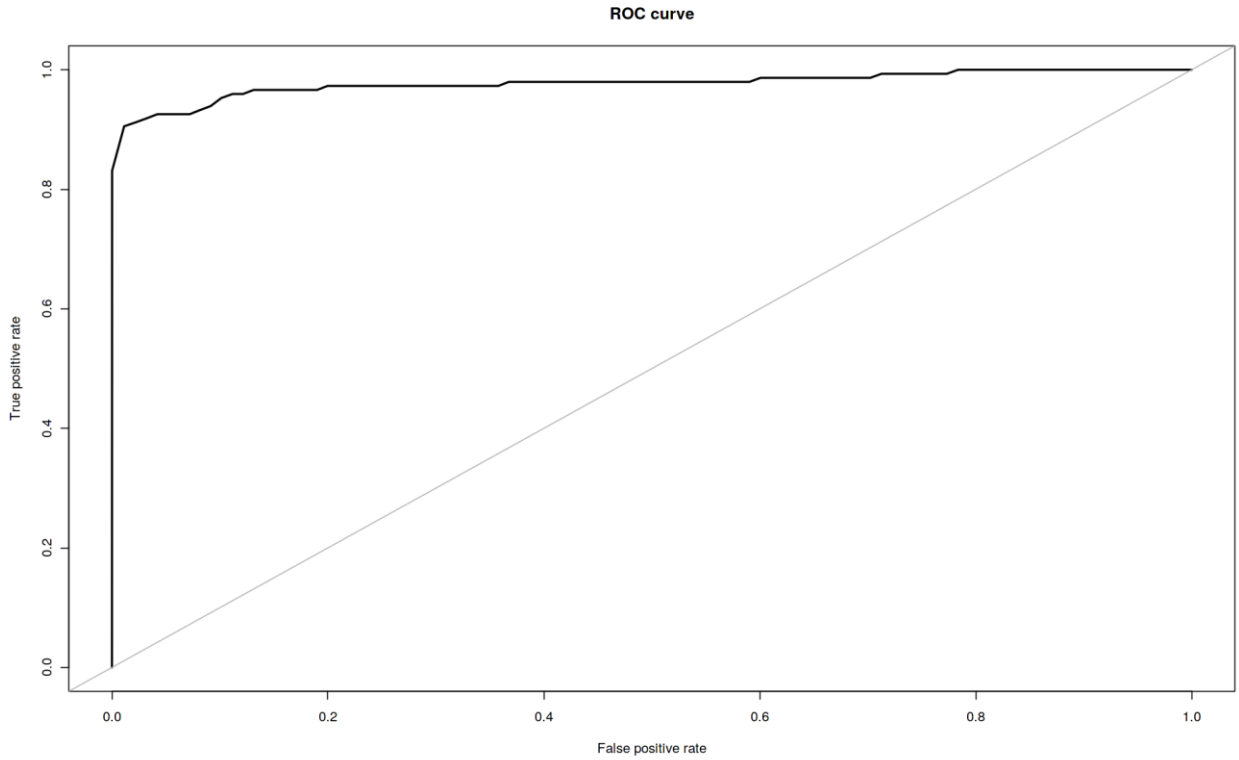
Area under the curve (AUC): 0.973



XGBoost

```
# Convert class labels from factor to numeric

labels <- up_train$Class

y <- recode(labels, 'Not_Fraud' = 0, "Fraud" = 1)
set.seed(42)
xgb <- xgboost(data = data.matrix(up_train[,-30]),
 label = y,
 eta = 0.1,
 gamma = 0.1,
 max_depth = 10,
 nrounds = 300,
 objective = "binary:logistic",
 colsample_bytree = 0.6,
 verbose = 0,
 nthread = 7,
)
xgb_pred <- predict(xgb, data.matrix(test[,-30]))

roc.curve(test$Class, xgb_pred, plotit = TRUE)
```

Area under the curve (AUC): 0.977




```
names <- dimnames(data.matrix(up_train[,-30]))[[2]]

# Compute feature importance matrix
importance_matrix <- xgb.importance(names, model = xgb)
# Nice graph
xgb.plot.importance(importance_matrix[1:10,])
```