DT:10-02-23 **LAB EXERCISES**

ITAO443-STATISTICS WITH R-PROGRAMMING

NAME:M.Vamsi

REG.NO:192011429

GITHUB LINK:- https://github.com/Vamsim29/ITA0443-STATISTICS-WITH-R-PROGRAMMING

FEB 10 2023 DAY 3 LAB ASSIGNMENT

1. (i) Write a function in R programming to print generate Fibonacci sequence using Recursion in R

.

(ii) Find sum of natural numbers up-to 10, without formula using loop statement.

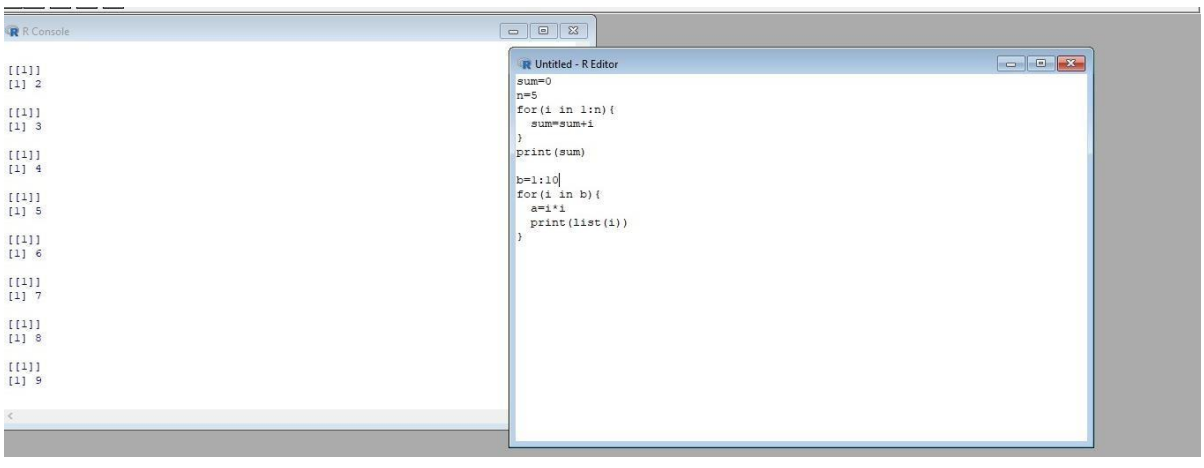
(iii) create a vector 1:10 and Find a square of each number and store that in a separate list.

P1(ii) sum=0 n=5

```
for(i in
 1:n){ sum=s
 um+i
}
print(sum)
```

```
b=1:10
for(i in
 b){ a=i*i
 print(list(i))
}
```

OUTPUT:-



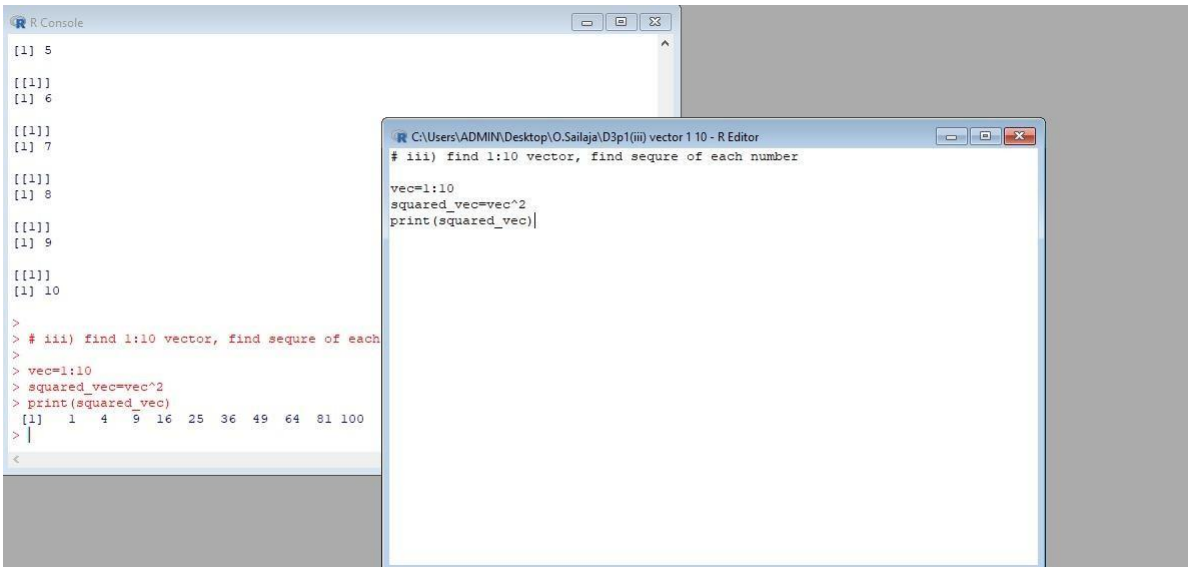
iii) find 1:10 vector, find sequre of each number

vec=1:10

squared_vec=vec^2

print(squared_vec)

OUTPUT:-



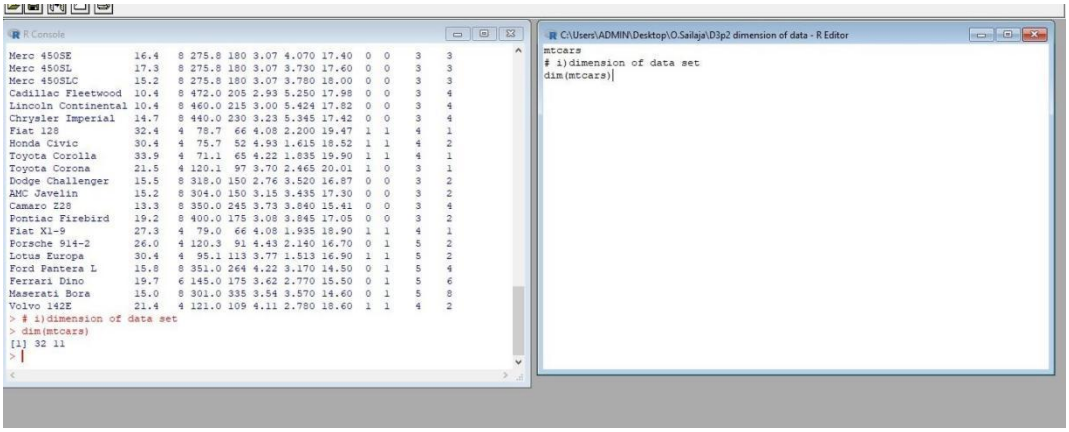
2 question 2. (motor trend car road test) comprises fuel consumption, performance and 10 aspects
of automobile
design for 32 automobiles. It comes pre-installed with package in R.
(i)Find the dimension of the dataset
(ii)Give the statistical summary of the features.
(iii)Print the categorical features in Dataset
(iv)Find the average weight(wt) grouped by Engine shape(vs)
(v)Find the largest and smallest value of the variable weight with respect to Engine shape

mtcars

i)dimension of data set

dim(mtcars)

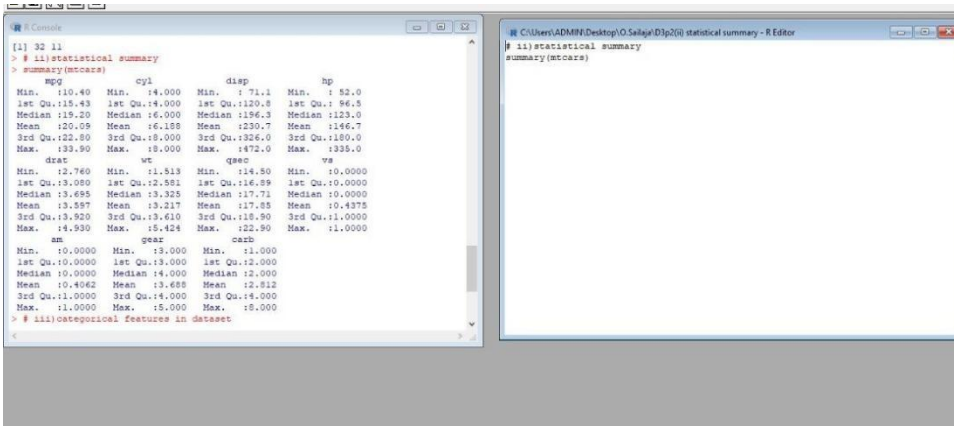
Output:-


R Console
Merc 450SE 16.4 8 275.8 180 3.07 4.070 17.40 0 0 3 3
Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3
Merc 450SLC 15.2 8 275.8 180 3.07 3.780 18.00 0 0 3 3
Cadillac Fleetwood 10.4 8 472.0 205 2.93 5.250 17.98 0 0 3 4
Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82 0 0 3 4
Chrysler Imperial 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4
Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1
Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2
Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1
Toyota Corona 21.5 4 120.1 97 3.70 2.465 20.01 1 0 3 1
Dodge Challenger 15.5 8 318.0 150 2.76 3.520 16.87 0 0 3 2
AMC Javelin 15.2 8 304.0 150 3.15 3.435 17.30 0 0 3 2
Camaro Z28 13.3 8 350.0 245 3.73 3.840 15.41 0 0 3 4
Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2
Fiat X1-9 27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1
Porsche 914-2 26.0 4 120.3 91 4.43 2.140 16.70 0 1 5 2
Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2
Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4
Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.50 0 1 5 6
Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.60 0 1 5 8
Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2
> # i)dimension of data set
> dim(mtcars)
[1] 32 11
>
C:\Users\ADMIN\Desktop\O.Sailaja\D3p2 dimension of data - R Editor
mtcars
# i)dimension of data set
dim(mtcars)

ii)statistical summary

summary(mtcars)

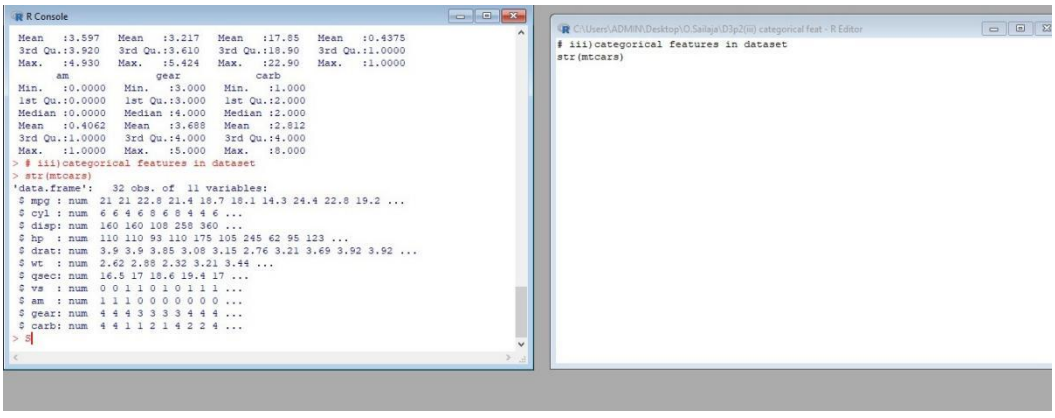
Output:-



iii)categorical features in dataset

str(mtcars)

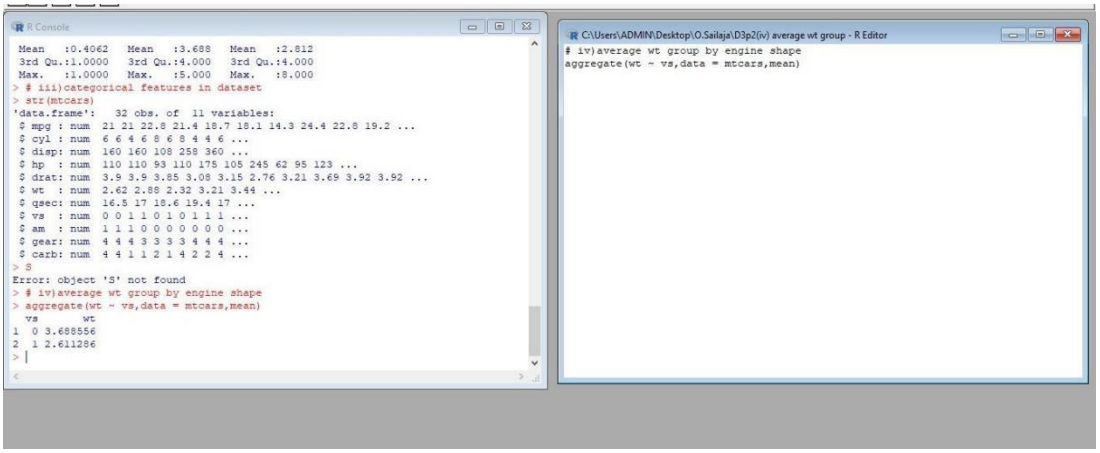
output:-



iv)averagewt group by engine shape

aggregate(wt ~ vs,data = mtcars,mean)

output:-



v)find largest & smallest weight with engine shape

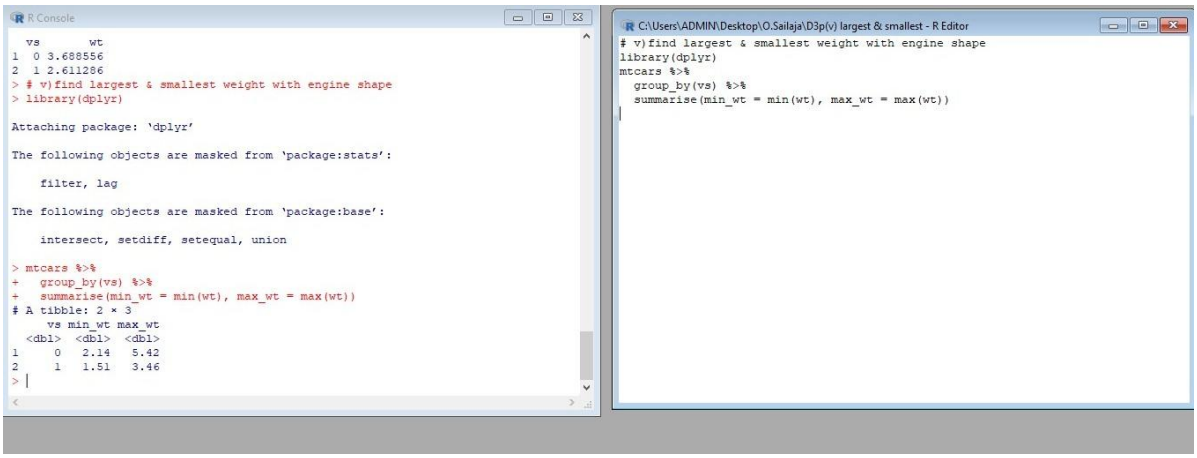
library(dplyr)

mtcars %>%

group_by(vs) %>%

summarise(min_wt = min(wt), max_wt = max(wt))

output:-



3 rd question 3.Use ggplot package to plot below EDA questions label the plot accordingly
(i)Create weight(wt) vs displacement(disp) scatter plot factor by Engine Shape(vs)
(ii) Create horsepower(hp) vs mileage (mgp) scatter plot factor by Engine Shape(vs)
(iv)In above plot , Separate columns according to cylinders(cyl) size
(v) Create histogram plot for horsepower (hp) with bin-width size of 5

library(ggplot2)

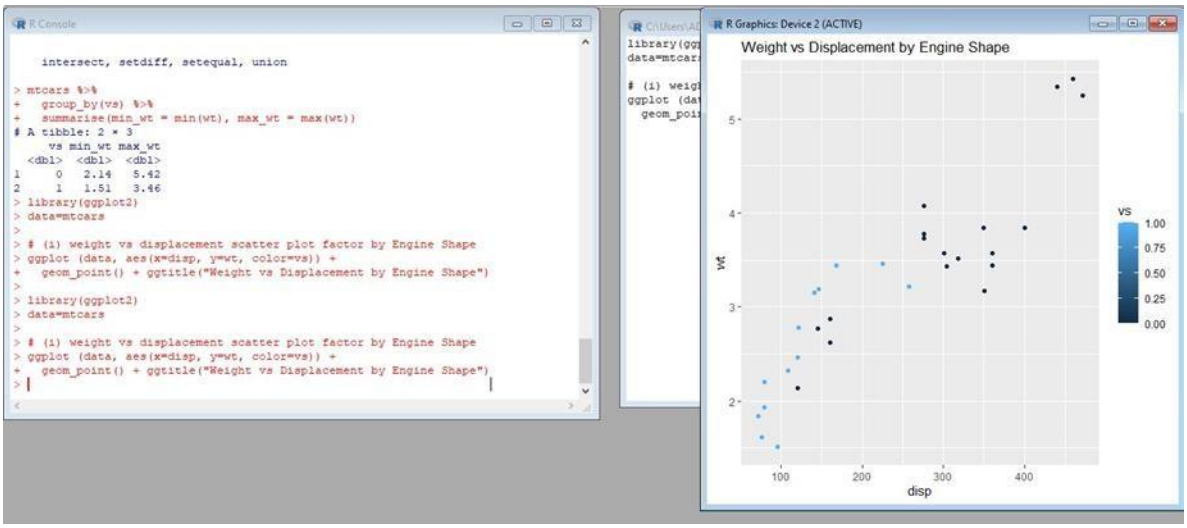
data=mtcars

(i) weight vs displacement scatter plot factor by Engine Shape

ggplot (data, aes(x=disp, y=wt, color=vs)) +

geom_point() + ggtitle("Weight vs Displacement by Engine Shape")

output:-

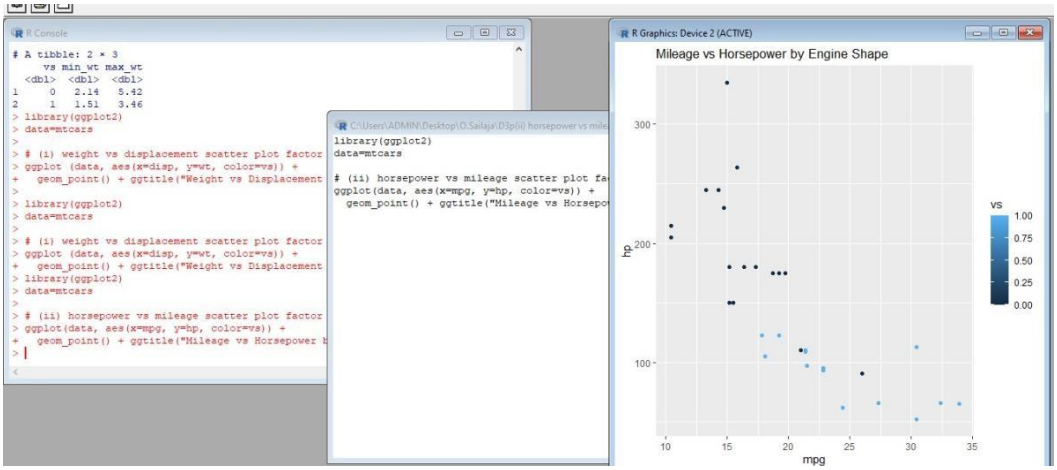


(ii) horsepower vs mileage scatter plot factor by Engine Shape

ggplot(data, aes(x=mpg, y=hp, color=vs)) +

geom_point() + ggtitle("Mileage vs Horsepower by Engine Shape")

output:-



(iii) horsepower vs mileage scatter plot factor by Cylinder Size

ggplot(data, aes(x=mpg, y=hp, color=cyl)) +

geom_point() + ggtitle("Mileage vs Horsepower by Cylinder Size")

4.Performing Logistic regression on dataset to predict the cars

Engine shape(vs) .

(i)Do the EDA analysis and find the features which is impact the

Engine shape and use

this for model.

(ii) Split the data set randomly with 80:20 ration to create train

and test dataset and create

logistic model

(iii)Create the Confusion matrix among prediction and test data.

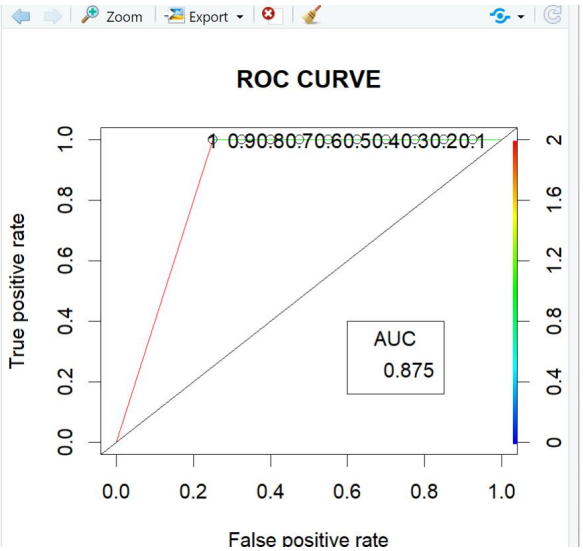
PROGRAM:-

```
# Installing the package
install.packages("caTools") # For Logistic regression
install.packages("ROCR")    # For ROC curve to evaluate model

# Loading package
library(caTools)
library(ROCR)

# Splitting dataset
split <- sample.split(mtcars, SplitRatio = 0.8)
split

train_reg <- subset(mtcars, split == "TRUE")
test_reg <- subset(mtcars, split == "FALSE")

# Training model
logistic_model <- glm(vs ~ wt + disp,
                      data = train_reg,
                      family = "binomial")
logistic_model

# Summary
summary(logistic_model)

# Predict test data based on model
predict_reg <- predict(logistic_model,
                       test_reg, type = "response")
predict_reg
```

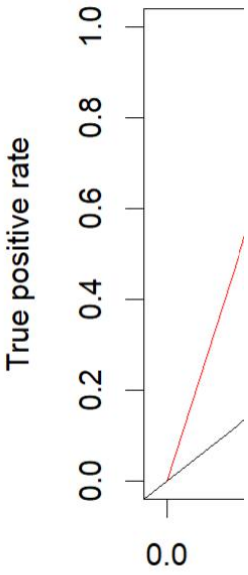
```
# Changing probabilities
predict_reg <- ifelse(predict_reg >0.5, 1, 0)

# Evaluating model accuracy
# using confusion matrix
table(test_reg$vs, predict_reg)

missing_classerr <- mean(predict_reg != test_reg$vs)
print(paste('Accuracy =', 1 - missing_classerr))

# ROC-AUC Curve
ROCPred <- prediction(predict_reg, test_reg$vs)
ROCPer <- performance(ROCPred, measure = "tpr",
                      x.measure = "fpr")

auc <- performance(ROCPred, measure = "auc")
auc <- auc@y.values[[1]]
auc

# Plotting curve
plot(ROCPer)
plot(ROCPer, colorize = TRUE,
    print.cutoffs.at = seq(0.1, by = 0.1),
    main = "ROC CURVE")
abline(a = 0, b = 1)

auc <- round(auc, 4)
legend(.6, .4, auc, title = "AUC", cex = 1)
```

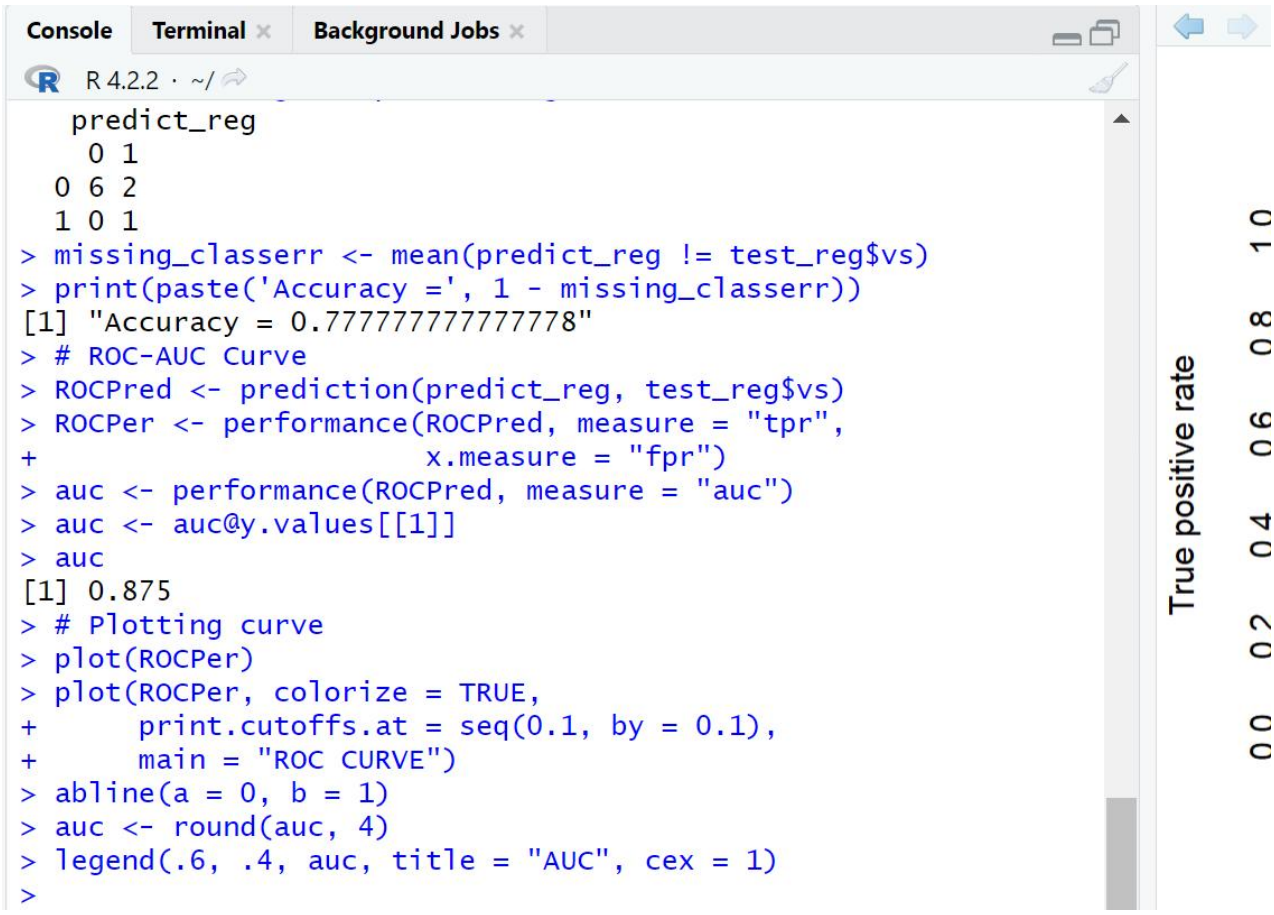
OUTPUT:-

```
Console  Terminal  Background Jobs
R 4.2.2 · ~/
packages
> # Loading package
> library(caTools)
> library(ROCR)
> # Splitting dataset
> split <- sample.split(mtcars, SplitRatio = 0.8)
> split
 [1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
[10]  TRUE  TRUE
> train_reg <- subset(mtcars, split == "TRUE")
> test_reg <- subset(mtcars, split == "FALSE")
> # Training model
> logistic_model <- glm(vs ~ wt + disp,
+                       data = train_reg,
+                       family = "binomial")
> logistic_model

Call:  glm(formula = vs ~ wt + disp, family = "binomial", data =
train_reg)

Coefficients:
(Intercept)           wt         disp
    1.69202      1.70855     -0.03214
```




```
R 4.2.2 · ~/
Call:
glm(formula = vs ~ wt + disp, family = "binomial", data = train_
reg)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9611  -0.2375   0.3879   0.4621   1.6686

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.69202    2.82563   0.599   0.5493
wt           1.70855    1.72447   0.991   0.3218
disp        -0.03214    0.01614  -1.991   0.0464 *
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 31.492  on 22  degrees of freedom
Residual deviance: 14.917  on 20  degrees of freedom
AIC: 20.917

Number of Fisher Scoring iterations: 5
```



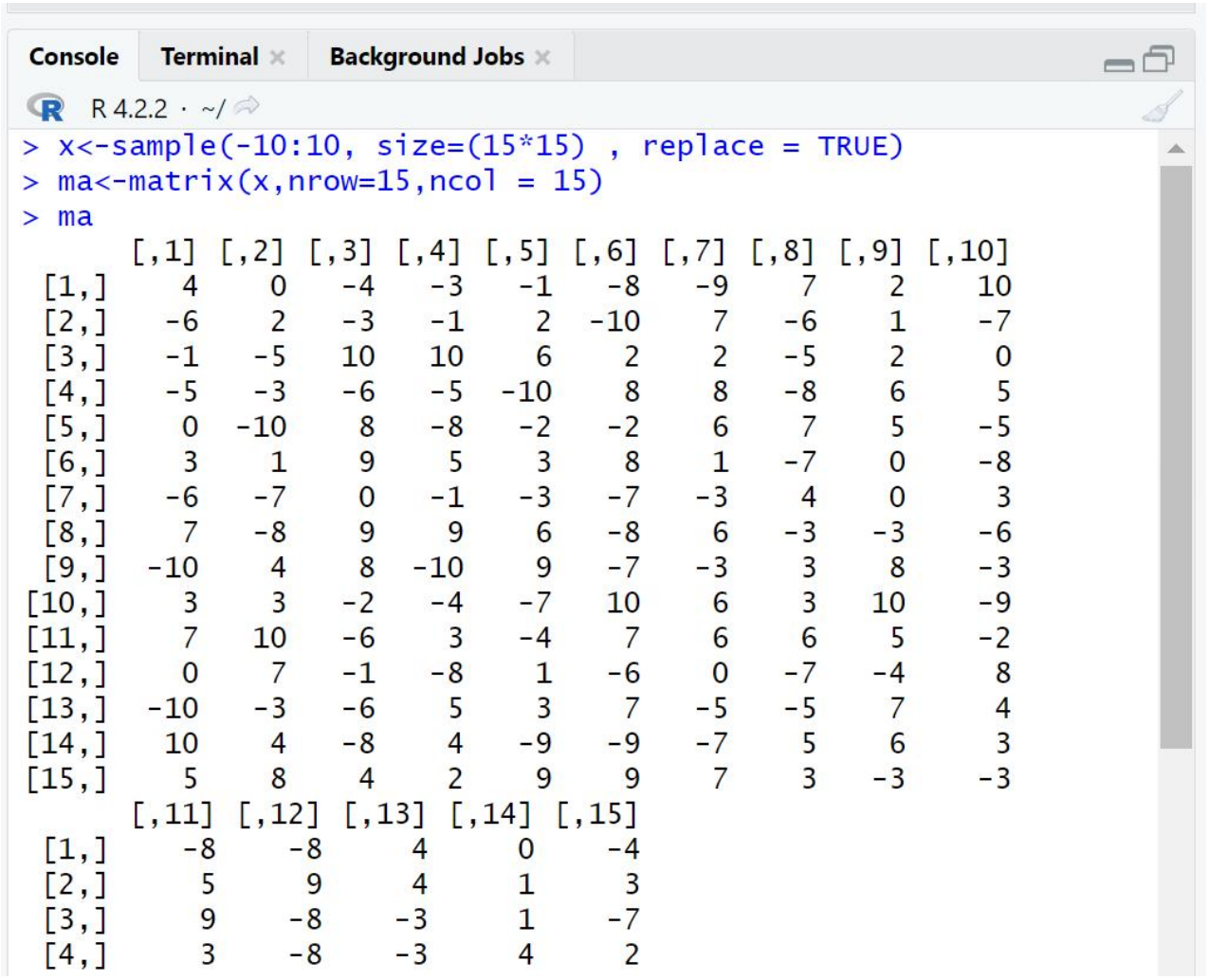


5. (I) Write R Program to create 15 x15 matrix filled with random numbers between -10 to 10, numbers can repeat.

PROGRAM:-

```
x<-sample(-10:10, size=(15*15) , replace = TRUE)
ma<-matrix(x,nrow=15,ncol = 15)
Ma
```

OUTPUT:-



(ii)Write R Program to display Lower Diagonal and upper

Diagonal matrix

PROGRAM:-

```
x<-sample(-10:10, size=(15*15) , replace = TRUE)
ma<-matrix(x,nrow=15,ncol = 15)
ma1<-ma
ma2<-ma
ma1[lower.tri(ma1)] <- 0
print("Lower diagonal matrix")
ma1
print("Upper diagonal matrix")
ma2[upper.tri(ma2)]<-0
ma2
```

OUTPUT:-

Console Terminal Background Jobs

R 4.2.2 · ~/

```
> x<-sample(-10:10, size=(15*15) , replace = TRUE)
> ma<-matrix(x,nrow=15,ncol = 15)
> ma1<-ma
> ma2<-ma
> ma1[lower.tri(ma1)] <- 0
> print("Lower diagonal matrix")
[1] "Lower diagonal matrix"
> ma1
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
 [1,]    1    6  -10    8    9   10   -9   -4   10     8    -6     9    10     9    -8
 [2,]    0    3   10   -9   -4    6  -10    5    9    -6     1    -7    -6   -10     0
 [3,]    0    0   -9    9    8   -5   -5    0   -8     3     7   -10    10    -4   -10
 [4,]    0    0    0    0    0   -7    0    7   -1     1    -9     3     3    -2     5
 [5,]    0    0    0    0   -4   -3   -5   -7    6     4    -1     3    -9    -4    -3
 [6,]    0    0    0    0    0    1   -9   -1    0    -6    -4     2     2     3    10
 [7,]    0    0    0    0    0    0  -10   -7   -5    -1     1    -4     3     1     5
 [8,]    0    0    0    0    0    0    0   -5   -4    -5    -9     4     4    -3    -5
 [9,]    0    0    0    0    0    0    0    0    5     9     5     6     4     1    -8
[10,]    0    0    0    0    0    0    0    0    0    -6     5    -2    -3     6    -7
[11,]    0    0    0    0    0    0    0    0    0     0    -5    -6     5    -3     1
[12,]    0    0    0    0    0    0    0    0    0     0     0    10     1    -1    -1
[13,]    0    0    0    0    0    0    0    0    0     0     0     0    -7    10    -4
[14,]    0    0    0    0    0    0    0    0    0     0     0     0     0    -6     5
[15,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     5
```

```
> print("Upper diagonal matrix")
[1] "Upper diagonal matrix"
> ma2[upper.tri(ma2)]<-0
> ma2
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
 [1,]    1    0    0    0    0    0    0    0    0     0     0     0     0     0     0
 [2,]   -7    3    0    0    0    0    0    0    0     0     0     0     0     0     0
 [3,]   -8   -9   -9    0    0    0    0    0    0     0     0     0     0     0     0
 [4,]    7   -9   -3    0    0    0    0    0    0     0     0     0     0     0     0
 [5,]  -10   -3   -5   -6   -4    0    0    0    0     0     0     0     0     0     0
 [6,]    4    5    0    1    0    1    0    0    0     0     0     0     0     0     0
 [7,]    1   -1   -6    3   -4   -6  -10    0    0     0     0     0     0     0     0
 [8,]    1   -7    5    8   -2   -6   -4   -5    0     0     0     0     0     0     0
 [9,]    2  -10   -5   -7   -1    7    4   -4    5     0     0     0     0     0     0
[10,]    2   -7    1   -2   -1  -10    3    9   -6    -6     0     0     0     0     0
[11,]   -5   -5    5   -5    1    9   -9   -8    9     3    -5     0     0     0     0
[12,]   -5    4   -2    1   10    9    8    3    4     8     6    10     0     0     0
[13,]   -2   -8   -6    5    5   -4   -5    8    0     4     1    -7    -7     0     0
[14,]    7   -9   -7   -6   10    9    4   -7   -5     0     3     1     3    -6     0
[15,]    6    1   -6    5   -9    4   10  -10   -3     8     3    10    -3    -4     5
```

(iii)Write R Program to count 0's in the matrix and check

the matrix is sparse matrix or not

PROGRAM:-

```
x<-sample(-10:10, size=(15*15) , replace = TRUE)

ma<-matrix(as.integer(x),nrow=15,ncol = 15)

y<-0

for(i in 1:15)

{

 for(j in 1:15)
```

```
 {
   if(ma[i][j]==0)
   {
     y=y+1
   }
 }
}
paste("no of zeros in matrix equal to ",y)
z=y/(15*15)
if(z>0.5)
{
  print("The matrix is sparse matrix")
}
else
{
  print("The matrix is not a sparse matarix")
}
```