**Institute of Technology & Management**

**GIDA Gorakhpur**



**DATA ANALYTICS LAB**

**Subject Code: KIT-651**

|  |  |
| --- | --- |
| **Prepared by:** | **Submitted to:** |
| **PANKAJ KUMAR GOND**  **1812013022**  **IT 3rd YEAR** | **Mr. RAHUL CHAKRAVORTY**  **Assistant Professor** |

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING AND INFORMATION TECHNOLOGY**

**Data Analytics Lab (KIT-651)**

## **INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **OBJECTS** | **DATE** | **GRADE** |
| 1 | **Program to print the Fibonacci Series.** |  |  |
| 2 | **Write a program to find Mean, Median and Mode.** |  |  |
| 3 | **Write a program in R for Linear Regression of Salary.** |  |  |
| 4 | **Write a program in R for logistic regression.** |  |  |
| 5 | **Program in R to perform matrix addition, subtraction, multiplication, division** |  |  |
| 6 | **Program in R for dimensionality reduction using PCA.** |  |  |
| 7 | **To perform K-Means clustering operation and visualization of iris dataset.** |  |  |
| 8 | **Program to perform Apriori Algorithm in R.** |  |  |
| 9 | **To perform KNN classifier in R.** |  |  |
| 10 | **Program to perform Time series analysis in R.** |  |  |

**Program 1**

**Object: Program to print the Fibonacci Series**

# take input from the user

nterms = as.integer(readline(prompt="How many terms? "))

# first two terms

n1 = 0

n2 = 1

count = 2

# check if the number of terms is valid

if(nterms<= 0) {

print("Plese enter a positive integer")

} else {

if(nterms == 1) {

print("Fibonacci sequence:")

print(n1)

} else {

print("Fibonacci sequence:")

print(n1)

print(n2)

while(count <nterms) {

nth = n1 + n2

print(nth)

# update values

n1 = n2

n2 = nth

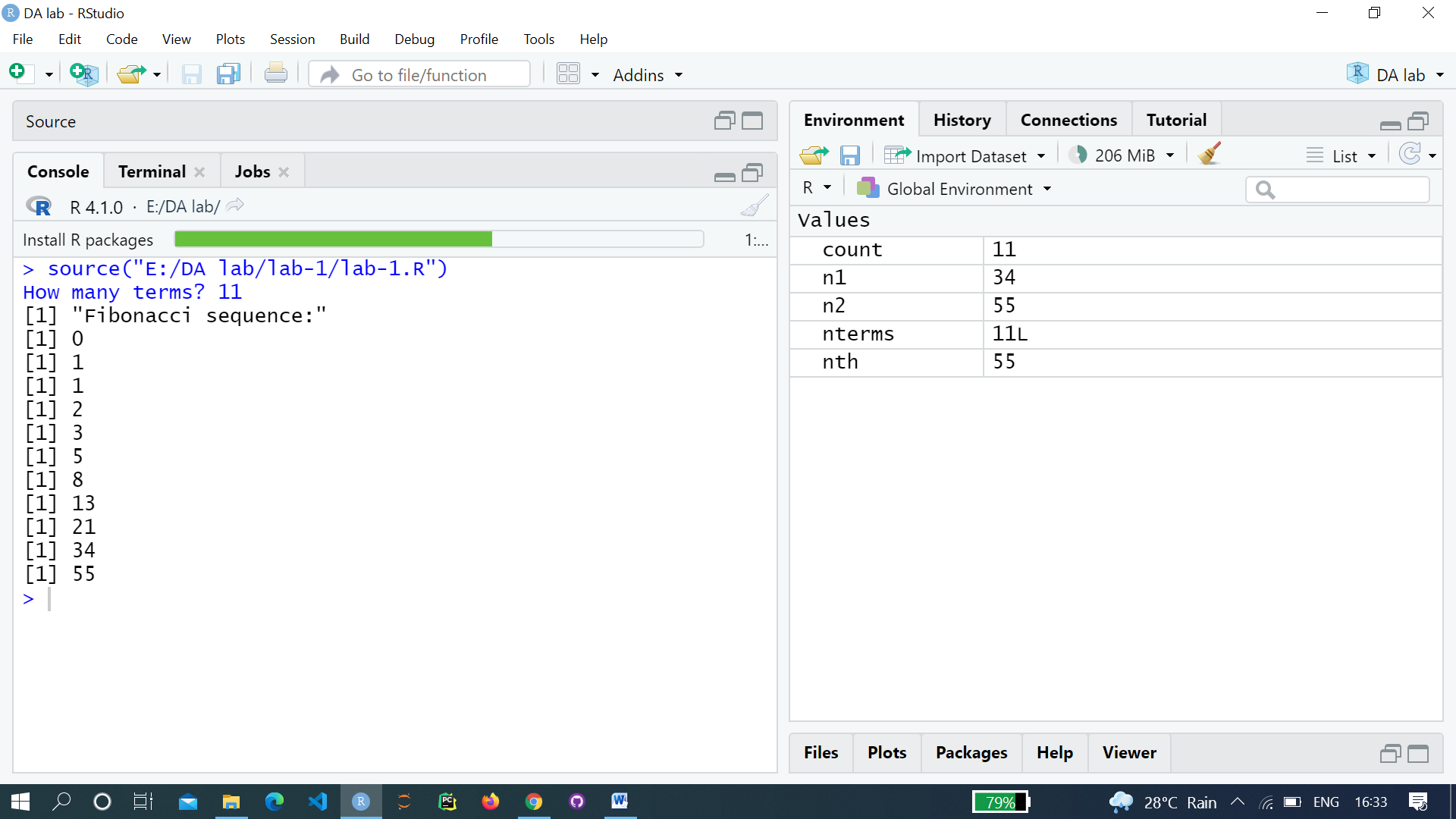
count = count + 1

}

}

}

**Output:**



**Program 2**

**Object: Write a program to find Mean, Median and Mode.**

**Mean:**

# Create a vector.

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

# Find mean.

result.mean<- mean(x)

print(result.mean)

# Find mean dropping NA values.

result.mean<- mean(x,na.rm = TRUE)

print(result.mean)

**Median:**

# Create the vector.

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

# Find the median.

median.result<- median(x)

print(median.result)

**Mode:**

# Create the function.

getmode<- function(v) {

uniqv<- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

# Create the vector with numbers.

v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)

# Calculate the mode using the user function.

result<- getmode(v)

print(result)

# Create the vector with characters.

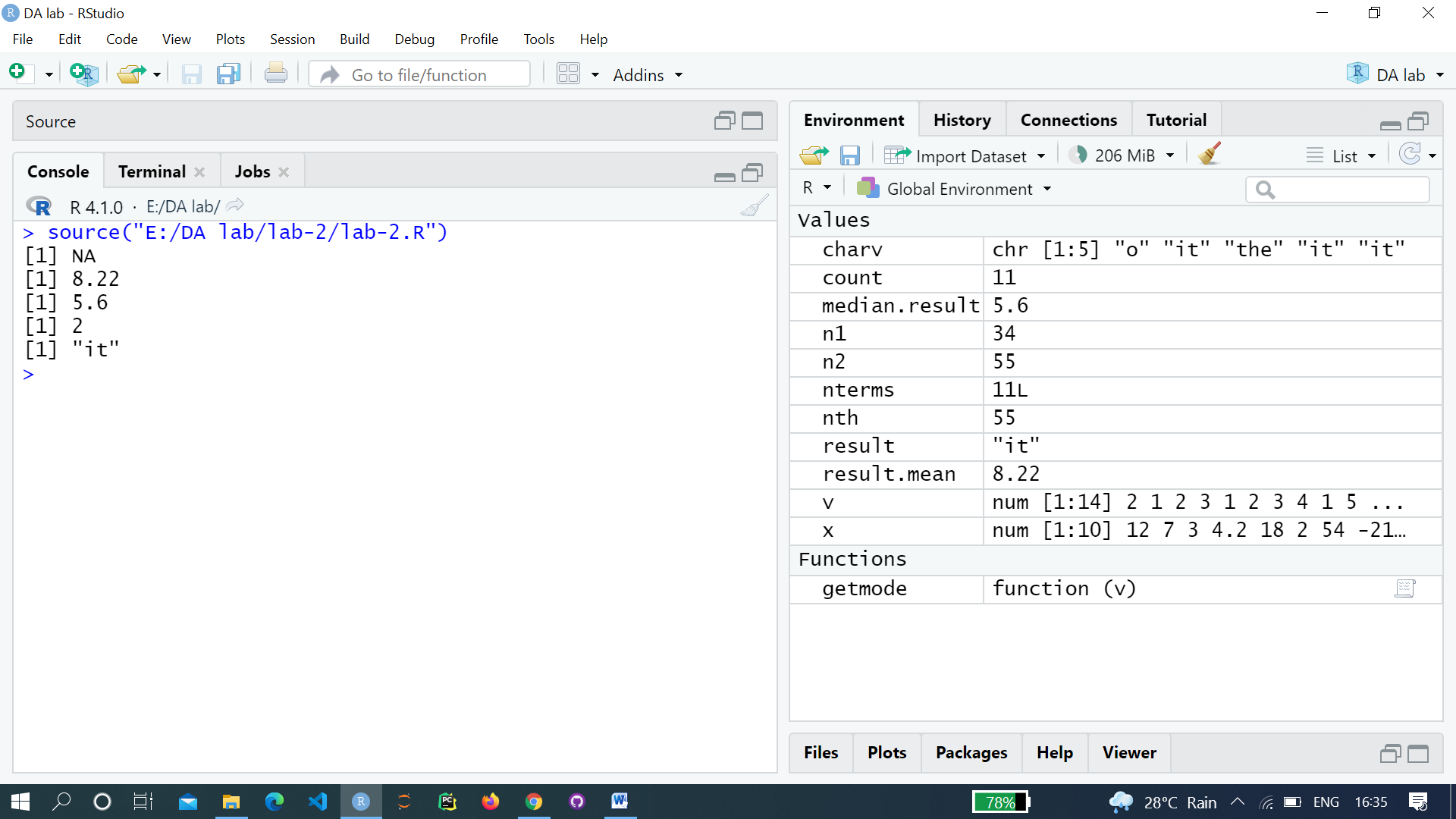
charv<- c("o","it","the","it","it")

# Calculate the mode using the user function.

result<- getmode(charv)

print(result)

**Output:**



**Program 3**

**Object: Write a program in R for Linear Regression of Salary**

**Source: Salary.csv**

**Years experienced Salary**

1.1 39343.00

1.3 46205.00

1.5 37731.00

2.0 43525.00

2.2 39891.00

2.9 56642.00

3.0 60150.00

3.2 54445.00

3.2 64445.00

3.7 57189.00

# Simple Linear Regression

# Importing the dataset

dataset = read.csv('salary.csv')

# Splitting the dataset into the

# Training set and Test set

install.packages('caTools')

library(caTools)

split = sample.split(dataset$Salary, SplitRatio = 0.7)

trainingset = subset(dataset, split == TRUE)

testset = subset(dataset, split == FALSE)

# Fitting Simple Linear Regression to the Training set

lm.r= lm(formula = Salary ~ YearsExperience, data = trainingset)

coef(lm.r)

# Predicting the Test set results

ypred = predict(lm.r, newdata = testset)

install.packages("ggplot2")

library(ggplot2)

# Visualising the Training set results

ggplot() + geom\_point(aes(x = trainingset$YearsExperience,

y = trainingset$Salary), colour = 'red') +

geom\_line(aes(x = trainingset$YearsExperience,

y = predict(lm.r, newdata = trainingset)), colour = 'blue') +

ggtitle('Salary vs Experience (Training set)') +

xlab('Years of experience') +

ylab('Salary')

# Visualising the Test set results

ggplot() +

geom\_point(aes(x = testset$YearsExperience, y = testset$Salary),

colour = 'red') +

geom\_line(aes(x = trainingset$YearsExperience,

y = predict(lm.r, newdata = trainingset)),

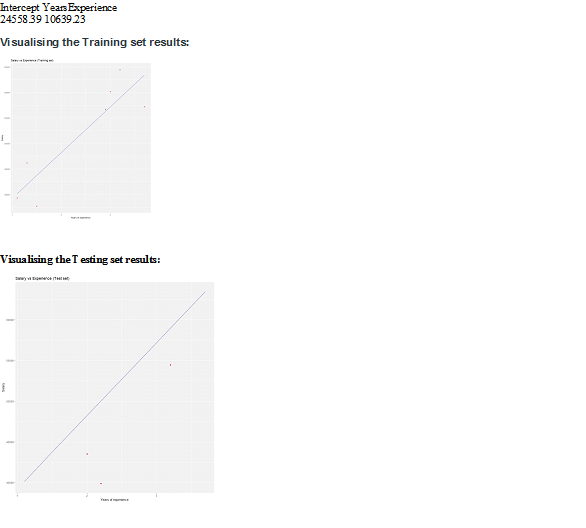
colour = 'blue') +

ggtitle('Salary vs Experience (Test set)') +

xlab('Years of experience') +

ylab('Salary')

**Output:**

****

**Program 4**

**Object: Write a program in R for logistic regression**

# Installing the package

install.packages("dplyr")

# Loading package

library(dplyr)

# Summary of dataset in package

summary(mtcars)

# 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)

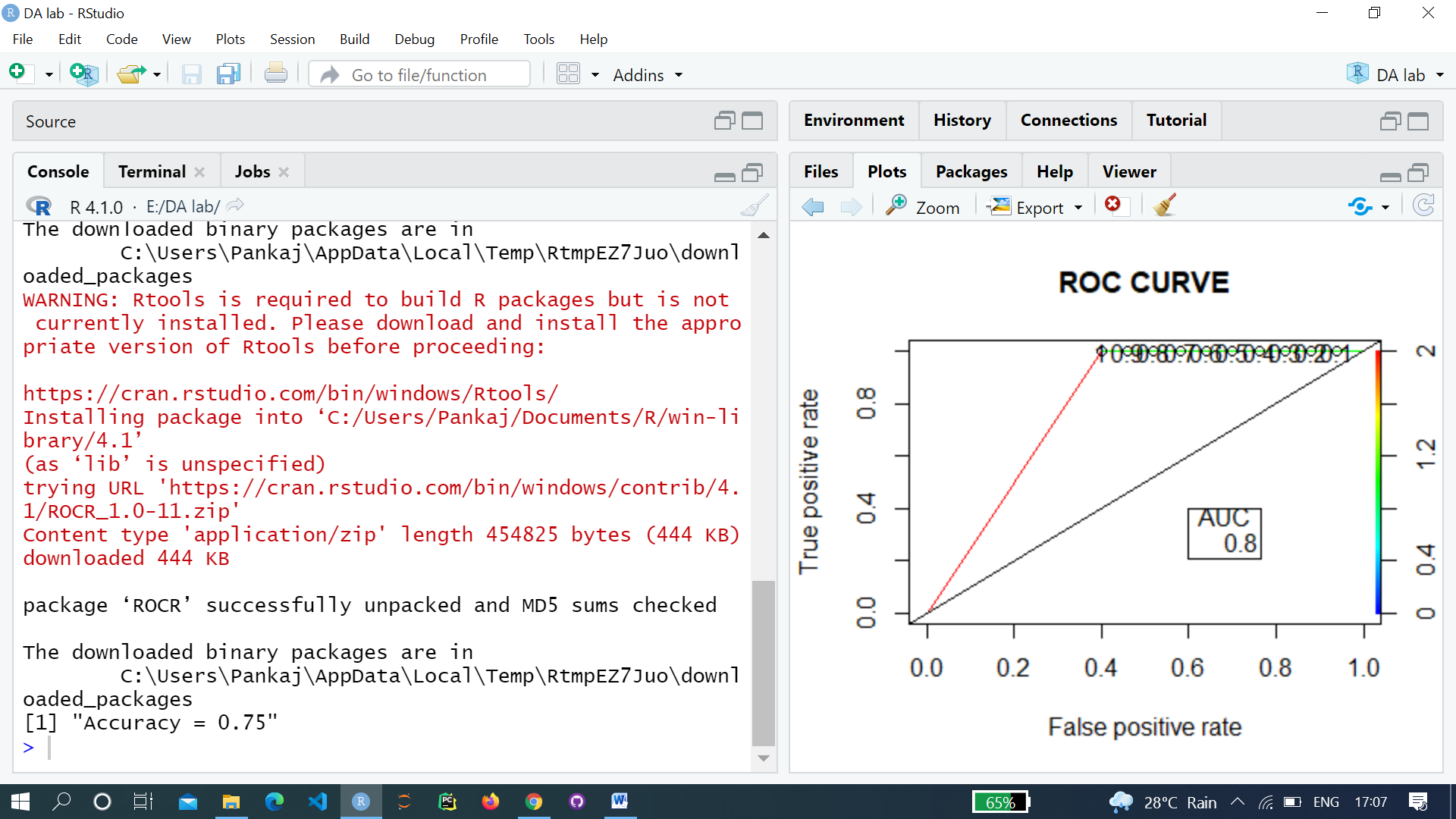
plot(ROCPer, colorize = TRUE,print.cutoffs.at = seq(0.1, by = 0.1),main = "ROC CURVE")

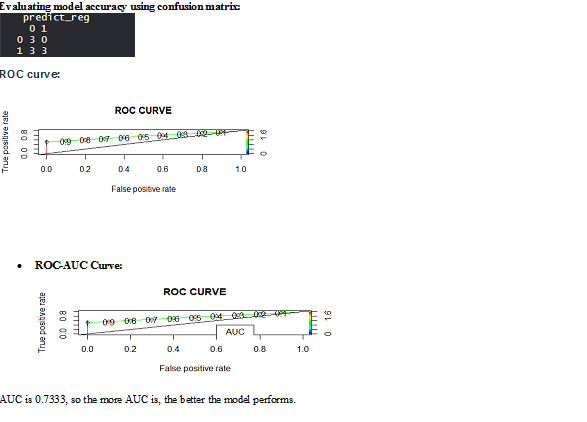
abline(a = 0, b = 1)

auc<- round(auc, 4)

legend(.6, .4, auc, title = "AUC", cex = 1)

**Output:**



****

**Program 5**

**Object: Program in R to perform matrix addition, subtraction, multiplication, and division.**

# Create two 2x3 matrixes.

m1 = matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)

print("Matrix-1:")

print(m1)

m2 = matrix(c(0, 1, 2, 3, 0, 2), nrow = 2)

print("Matrix-2:")

print(m2)

result = m1 + m2

print("Result of addition")

print(result)

result = m1 - m2

print("Result of subtraction")

print(result)

result = m1 \* m2

print("Result of multiplication")

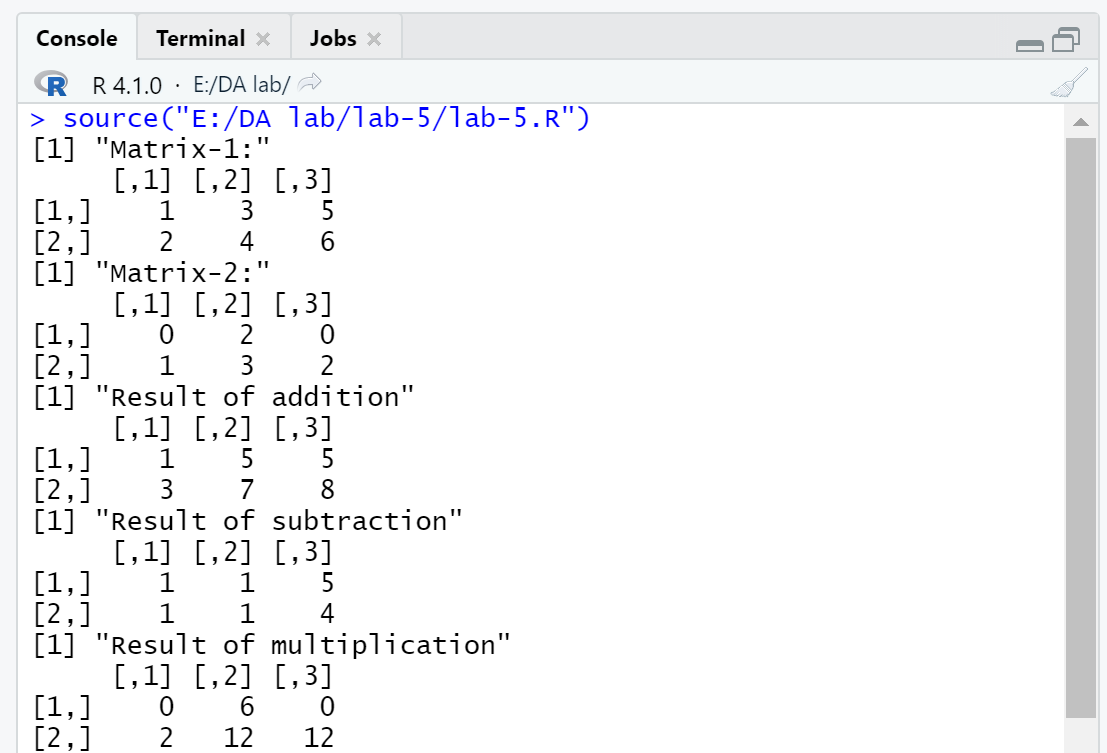
print(result)

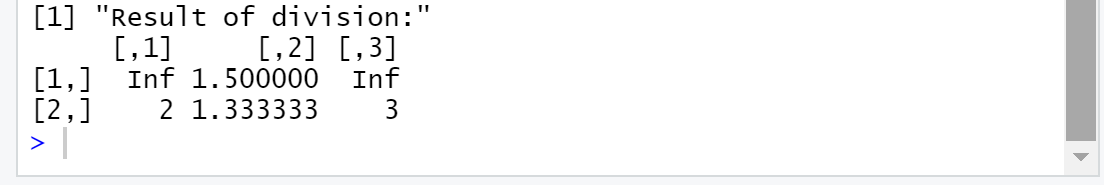
result = m1 / m2

print("Result of division:")

print(result)

**Output:**

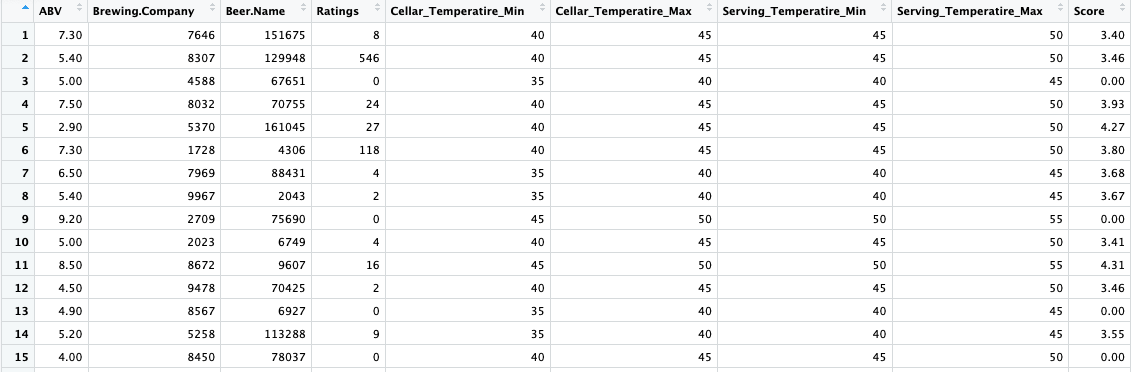
****

****

**Program 6**

**Object: Program in R for dimensionality reduction using PCA**

dataset = read.csv('Beer\_dataset.csv')



library(caTools)

set.seed(123)

split = sample.split(dataset$Score, SplitRatio = 0.80)

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

# Multiple Linear Regressor

mlr = lm(formula = Score ~ . , data = training\_set )

print(mlr)

predy = predict(mlr, newdata = test\_set)

actuals\_and\_preds<- data.frame(cbind(actuals=test\_set$Score, predicteds = predy))

min\_max\_accuracy<- mean(apply(actuals\_and\_preds, 1, min) / apply(actuals\_and\_preds, 1, max))print(min\_max\_accuracy)

Output:

0.7055808

install.packages("caret") # Execute Once

library(caret)

install.packages("e1071") # Execute Once

library(e1071)

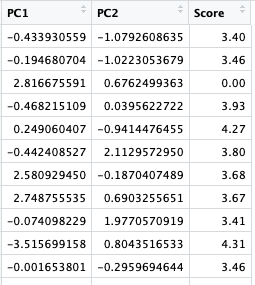
pca = preProcess(training\_set[-9], method = 'pca', pcaComp = 2)

training\_set.pca = predict(pca, training\_set)

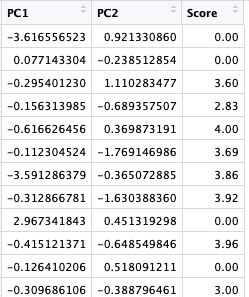
test\_set.pca = predict(pca, test\_set)

training\_set.pca = training\_set.pca[c(2,3,1)] test\_set.pca = test\_set.pca[c(2,3,1)]

**training\_set.pca:**



**test\_set.pca:**



mlr\_pca = lm(formula = Score ~ . , data = training\_set.pca)print(mlr\_pca)

predy\_pca = predict(mlr\_pca, newdata = test\_set.pca)

actuals\_and\_preds\_pca<- data.frame(cbind(actuals= test\_set.pca$Score, predicteds = predy\_pca))

min\_max\_accuracy\_pca<- mean(apply(actuals\_and\_preds\_pca, 1, min) / apply(actuals\_and\_preds\_pca, 1, max))print(min\_max\_accuracy\_pca)

**Output:**

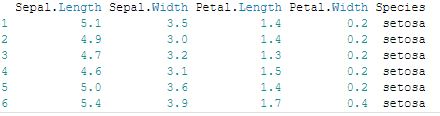
0.7301324

**Program 7**

**Object: To perform K-Means clustering operation and visualization of iris dataset**

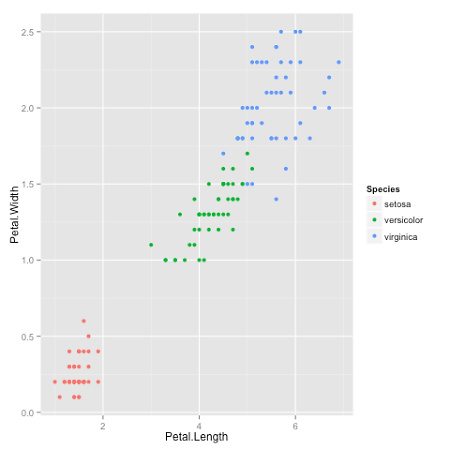
library(datasets)

head(iris)



library(ggplot2)

ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom\_point()



set.seed(20)

irisCluster<- kmeans(iris[, 3:4], 3, nstart = 20)

irisCluster

K-means clustering with 3 clusters of sizes 46, 54, 50

Cluster means:

Petal.LengthPetal.Width

1 5.626087 2.047826

2 4.292593 1.359259

3 1.462000 0.246000

Clustering vector:

[1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

[35] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

[69] 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1

[103] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 2 2 1 1 1 1 1 1 1 1

[137] 1 1 2 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:

[1] 15.16348 14.22741 2.02200

(between\_SS / total\_SS= 94.3 %)

Available components:

[1] "cluster" "centers" "totss" "withinss"

[5] "tot.withinss" "betweenss" "size" "iter"

[9] "ifault"

table(irisCluster$cluster, iris$Species)

setosaversicolorvirginica

1 0 2 44

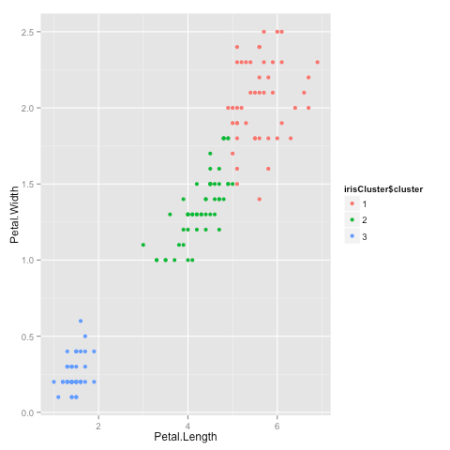
2 0 48 6

3 50 0 0

irisCluster$cluster<- as.factor(irisCluster$cluster)

ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom\_point()

Graph:



**Program 8**

**Object: Program to perform Apriori Algorithm In R**

**Load required library**

library(arules)

library(arulesViz)

library(RColorBrewer)

**Import the dataset**

data("Groceries")

**Applying apriori() function**

rules<- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.2))

**Applying inspect() function**

inspect(rules[1:10])

**Applying itemFrequencyPlot() function**

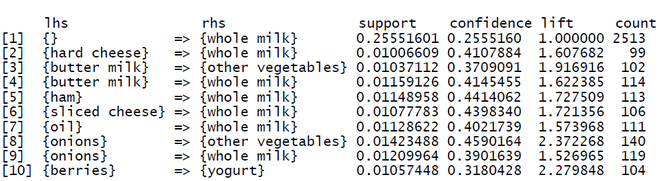
arules::itemFrequencyPlot(Groceries, topN = 20,

col = brewer.pal(8, 'Pastel2'),

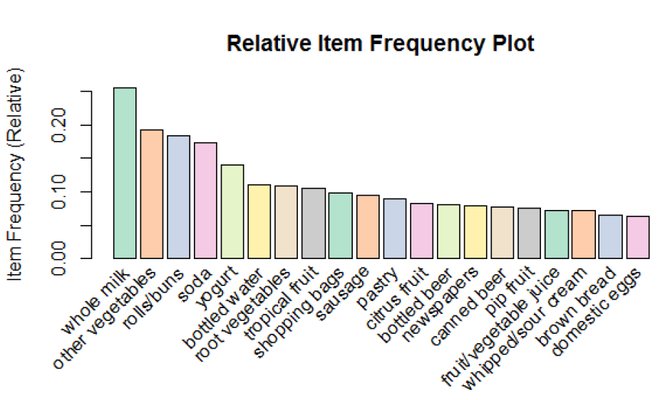
main = 'Relative Item Frequency Plot',

type = "relative",

ylab = "Item Frequency (Relative)")



Graph:



**Program 9**

**Object: To perform KNN classifier in R**

# Loading data

data(iris)

# Structure

str(iris)

# Installing Packages

install.packages("e1071")

install.packages("caTools")

install.packages("class")

# Loading package

library(e1071)

library(caTools)

library(class)

# Loading data

data(iris)

head(iris)

# Splitting data into train

# and test data

split<- sample.split(iris, SplitRatio = 0.7)

train\_cl<- subset(iris, split == "TRUE")

test\_cl<- subset(iris, split == "FALSE")

# Feature Scaling

train\_scale<- scale(train\_cl[, 1:4])

test\_scale<- scale(test\_cl[, 1:4])

# Fitting KNN Model

# to training dataset

classifier\_knn<- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

k = 1)

classifier\_knn

# Confusiin Matrix

cm <- table(test\_cl$Species, classifier\_knn)

cm

# Model Evaluation - Choosing K

# Calculate out of Sample error

misClassError<- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError))

# K = 3

classifier\_knn<- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

k = 3)

misClassError<- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError))

# K = 5

classifier\_knn<- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

k = 5)

misClassError<- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError))

# K = 7

classifier\_knn<- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

k = 7)

misClassError<- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError))

# K = 15

classifier\_knn<- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

k = 15)

misClassError<- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError))

# K = 19

classifier\_knn<- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

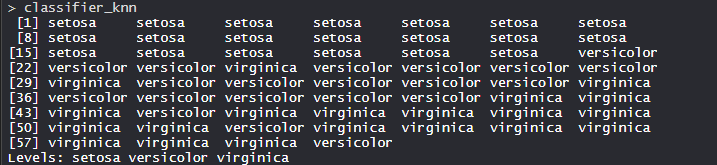
k = 19)

misClassError<- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError)

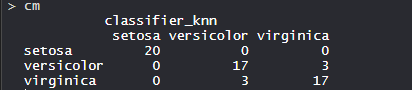
OUTPUT:

* **Model classifier\_knn(k=1):**



The KNN model is fitted with a train, test, and k value. Also, the Classifier Species feature is fitted in the model.

* **Confusion Matrix:**



So, 20 Setosa are correctly classified as Setosa. Out of 20 Versicolor, 17 Versicolor are correctly classified as Versicolor and 3 are classified as virginica. Out of 20 virginica, 17 virginica are correctly classified as virginica and 3 are classified as Versicolor.

* **Model Evaluation:**  
  **(k=1)**  
  https://media.geeksforgeeks.org/wp-content/uploads/20200616031144/Capture547.png

The model achieved 90% accuracy with k is 1.

**(K=3)**  
https://media.geeksforgeeks.org/wp-content/uploads/20200616031520/Capture_32.png

The model achieved 88.33% accuracy with k is 3 which is lower than when k was 1.

**(K=5)**  
https://media.geeksforgeeks.org/wp-content/uploads/20200616031544/Capture_5.png

The model achieved 91.66% accuracy with k is 5 which is more than when k was 1 and 3.

**(K=7)**  
https://media.geeksforgeeks.org/wp-content/uploads/20200616031701/Capture_7.png

The model achieved 93.33% accuracy with k is 7 which is more than when k was 1, 3, and 5.

**(K=15)**  
https://media.geeksforgeeks.org/wp-content/uploads/20200616031732/Capture_151.png

The model achieved 95% accuracy with k is 15 which is more than when k was 1, 3, 5, and 7.

**(K=19)**  
https://media.geeksforgeeks.org/wp-content/uploads/20200616031748/Capture_191.png

The model achieved 95% accuracy with k is 19 which is more than when k was 1, 3, 5, and 7. Its same accuracy when k was 15 which means now increasing k values doesn’t affect the accuracy.

**Program 10**

**Object: Program to perform Time series analysis in R**

# Get the data points in form of a R vector.

rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)

# Convert it to a time series object.

rainfall.timeseries<- ts(rainfall,start = c(2012,1),frequency = 12)

# Print the timeseries data.

print(rainfall.timeseries)

# Give the chart file a name.

png(file = "rainfall.png")

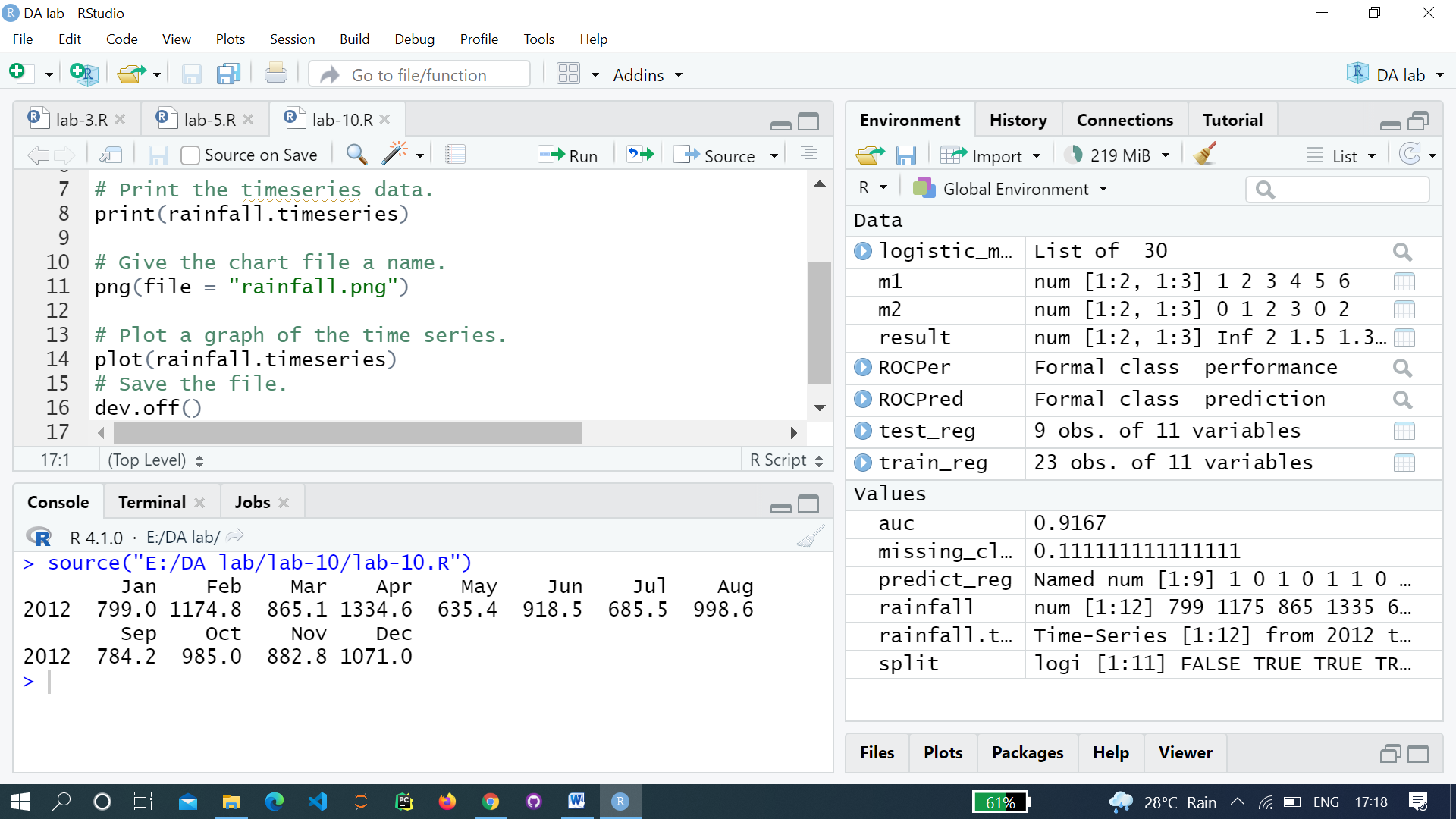
# Plot a graph of the time series.

plot(rainfall.timeseries)

# Save the file.

dev.off()

**Output:**



**Graph:**

