**Practical 1**

**K Nearest Neighbour Algorithm**

**Code:**

#iris dataset

data=iris

print(data)

summary(data)

#normalisation formula

normalise = function(x){

return(x-sin(x))/(max(x)-min(x))

}

#normalising data

data$Sepal.Length=normalise(data$Sepal.Length)

data$Sepal.Width=normalise(data$Sepal.Width)

data$Petal.Length=normalise(data$Petal.Length)

data$Petal.Width=normalise(data$Petal.Width)

#randomize data using sample function

data=data[sample(1:nrow(data)),]

#dividing first 140 instances as training data and 10 instances as testing data

train\_data=data[1:140,]

test\_data=data[141:150,]

#removing target coloumn which is the 5th coloumn

train\_label=train\_data[,5]

train\_data=train\_data[,-5]

test\_label=test\_data[,5]

test\_data=test\_data[,-5]

#printing latest data

print(train\_data)

print(test\_data)

#installing package class for implementing k nearest neighbour algorithm

library(class)

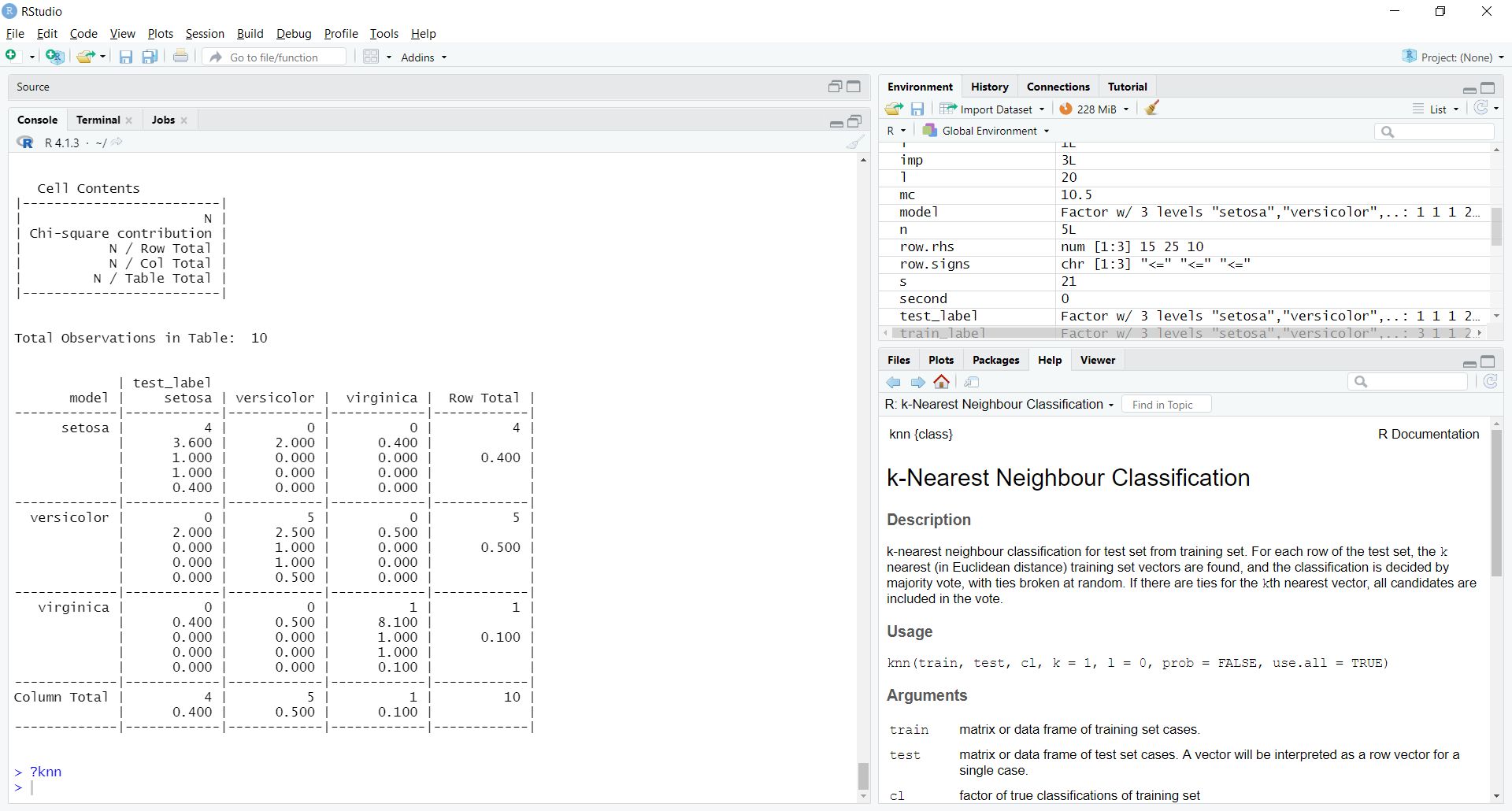
#implementation of k nearest neighbour algorithm

model=knn(train\_data,test\_data,train\_label,k=11, 6)

#installing package gmodels for cross table

library(gmodels)

CrossTable(model,test\_label)



**Practical 3**

**Naïve Bayes Algorithm**

**Code:**

library(e1071)

?naiveBayes

#working with our own dataset by creating excel sheet

#data=read.csv("C:\\Users\\OLIFIA\\OneDrive\\Desktop\\Roy sir\\Code Documentation\\Height\_Weight.csv")

#working with readymade dataset

data=iris

set.seed(125)

#randomising the data

ind=sample(1:nrow(data),size=0.85\*nrow(data),replace=FALSE)

train\_data=data[ind,]

test\_data=data[-ind,]

#assignning species coloumn to a variable

test\_data\_label=test\_data[5]

#removing species coloumn

test\_data=test\_data[,-5]

#applying naive baaayes theorem

model=naiveBayes(Species~.,train\_data)

#prediction

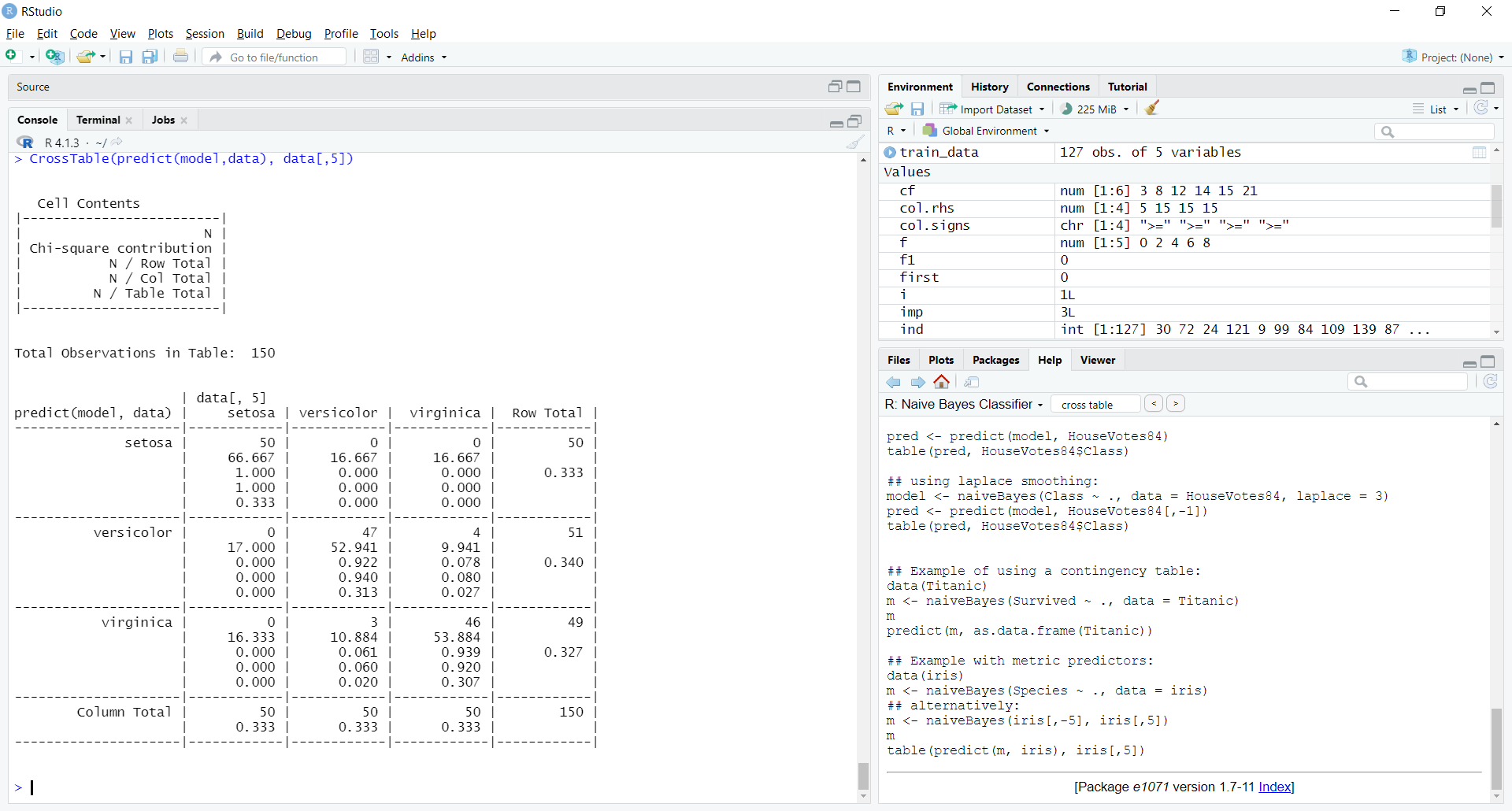
predict(model,data)

library(gmodels)

#calculating efficiency of the model

CrossTable(predict(model,data), data[,5])

**Output:**



**DECISION TREE**

**Method 1: party package**

data=iris

set.seed(125) #fixed output every time

ind=sample(1:nrow(data),size=0.95\*nrow(data),replace=FALSE) #random output everytime

train\_data=data[ind,]

test\_data=data[-ind,]

test\_label=test\_data[5,]

test\_data=test\_data[-5,]

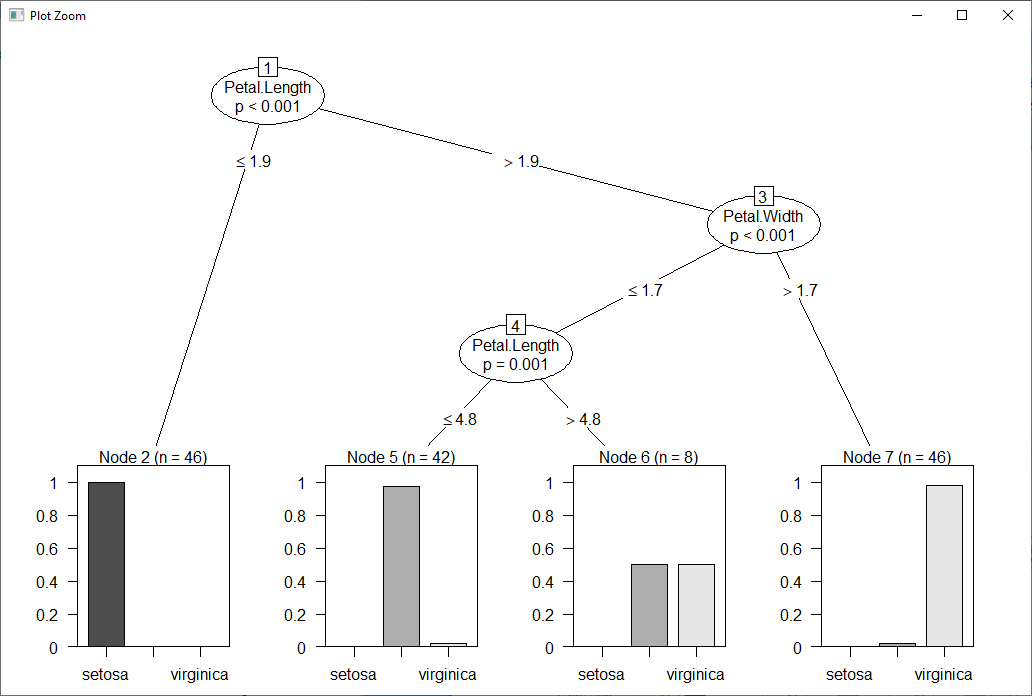
library(party) #library of package party

?ctree #document

model=ctree(Species~.,train\_data) #ctree decision tree algo

plot(model)

ans=predict(model,test\_data)



**Method 2: rpart package and rattle package**

data=iris

set.seed(125) #fixed output every time

ind=sample(1:nrow(data),size=0.95\*nrow(data),replace=FALSE) #random output everytime

train\_data=data[ind,]

test\_data=data[-ind,]

test\_label=test\_data[5,]

test\_data=test\_data[-5,]

library(rattle) #library of package rattle and it is used to plot the graph

library(rpart) #library of package rpart

?rpart #document

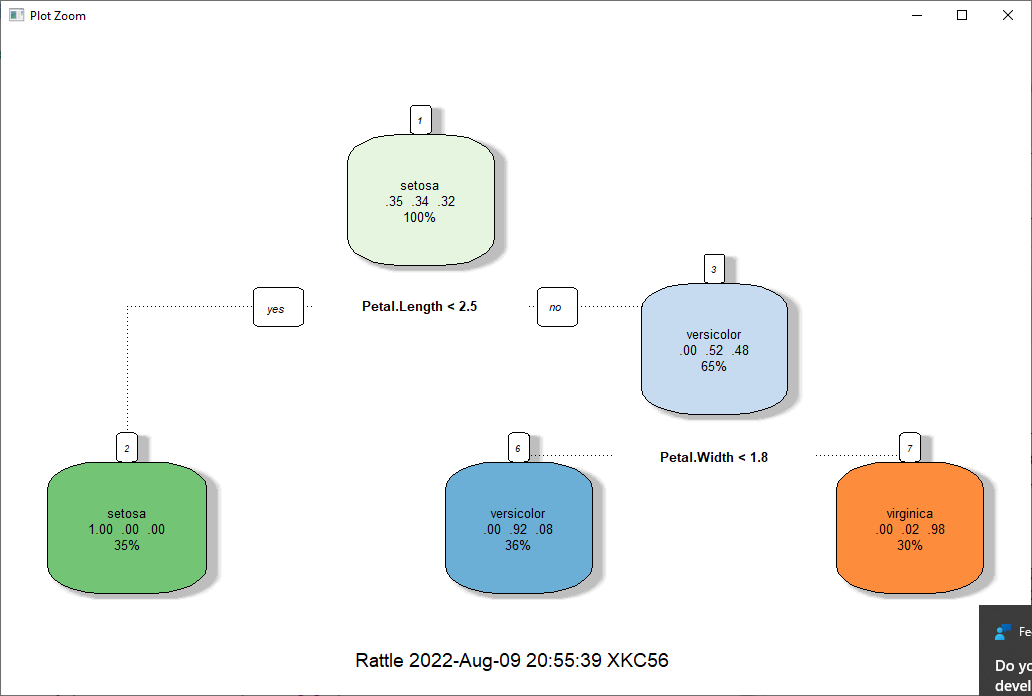
model1=rpart(Species~.,train\_data) #rpart decision tree algo

fancyRpartPlot(model1)#for plotting graph

model1=rpart(Species~.,train\_data,minsplit=5,cp=0.001)#minsplit means if it is more than 5 then split

fancyRpartPlot(model1)

ans=predict(model,test\_data)



data=iris

set.seed(125) #fixed output every time

ind=sample(1:nrow(data),size=0.95\*nrow(data),replace=FALSE) #random output everytime

train\_data=data[ind,]

test\_data=data[-ind,]

test\_label=test\_data[5,]

test\_data=test\_data[-5,]

library(rattle) #library of package rattle and it is used to plot the graph

library(rpart) #library of package rpart

?rpart #document

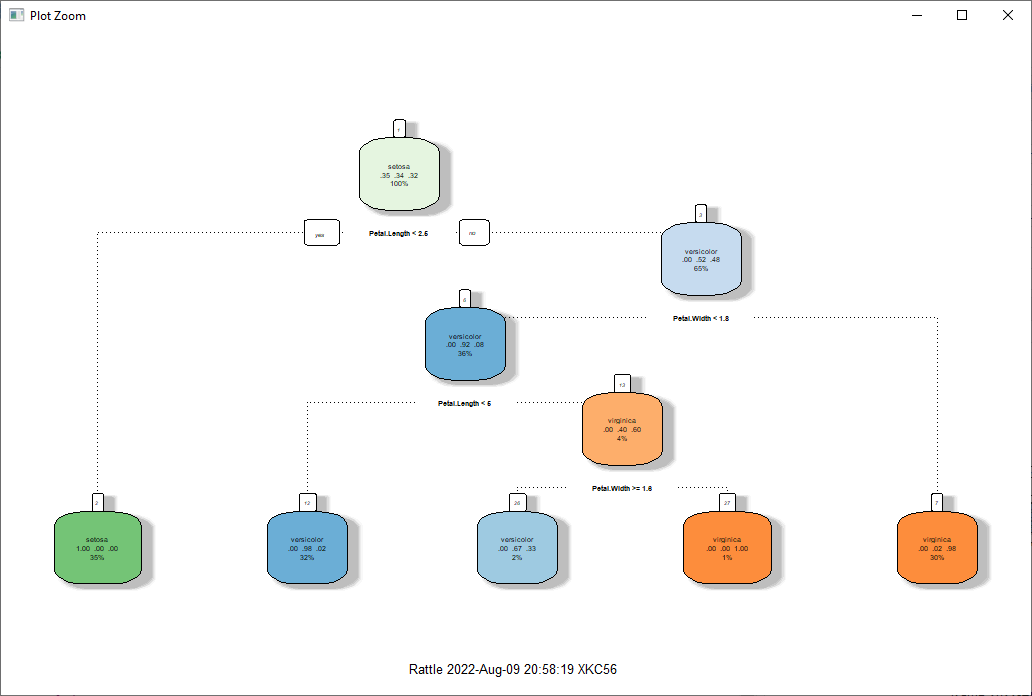
model1=rpart(Species~.,train\_data) #rpart decision tree algo

fancyRpartPlot(model1)#for plotting graph

model1=rpart(Species~.,train\_data,minsplit=5,cp=0.001)#minsplit means if it is more than 5 then split

fancyRpartPlot(model1)

ans=predict(model,test\_data)



**Practical 5**

**K Mean Clustering Algorithm**

**Code:**

#K Mean Clustering

#A(2,2) B(4,5) C(10,10) D(4,3) E(8,9) F(2,1) G(11,9) H(6,7)

#working with our own dataset

data=read.csv("C:\\Users\\OLIFIA\\OneDrive\\Desktop\\Roy sir\\Code Documentation\\Points\_data.csv")

library(cluster)

?kmeans

data=data[,-1]

set.seed(125)

#implimentation of k means clustering algo

model=kmeans(data,3)

#simple graph plotting

plot(data$X.coordinate,data$Y.coordinate)

#plotting with colors having the same cluster

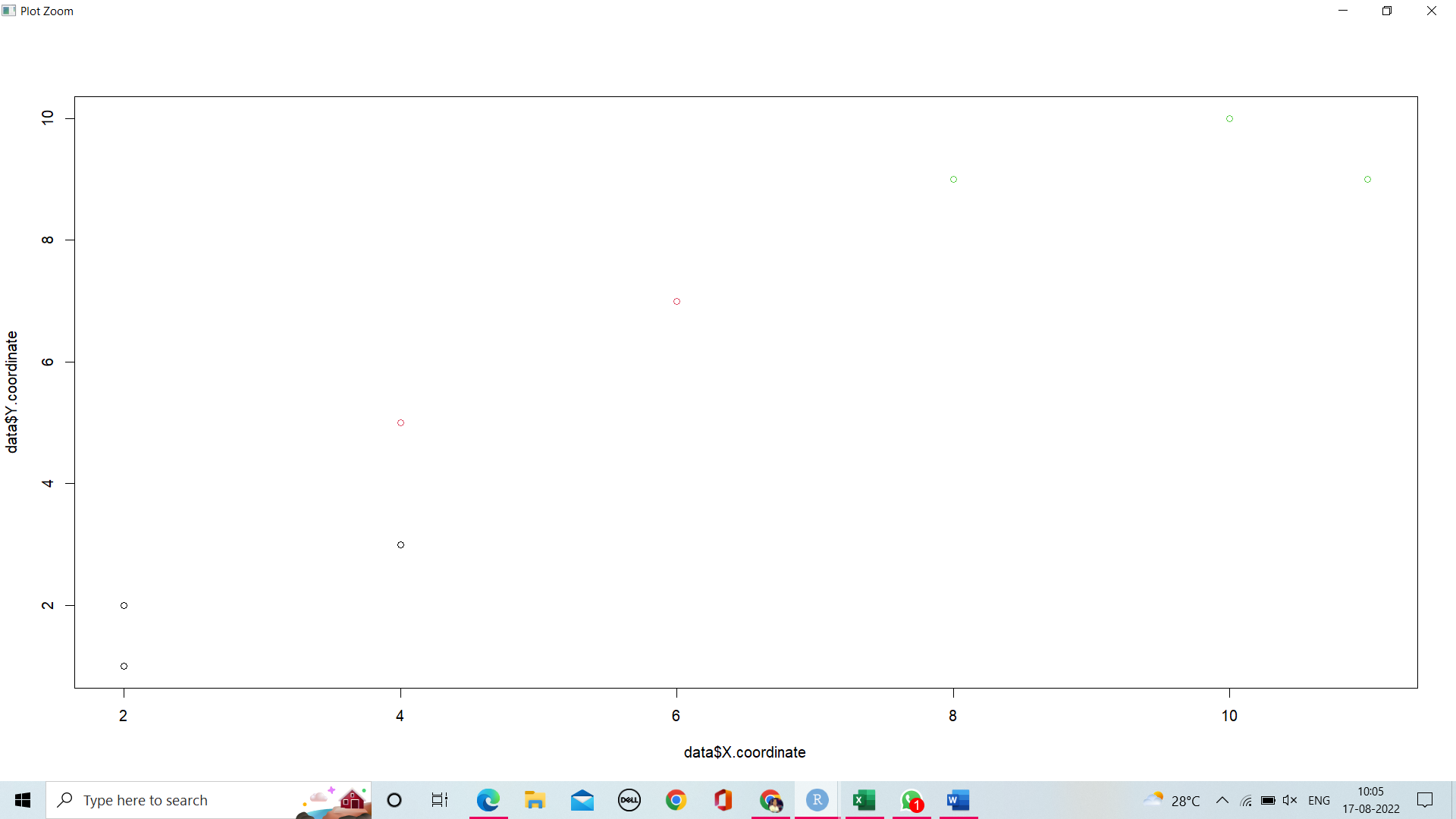
plot(data$X.coordinate, data$Y.coordinate, col=model$cluster)

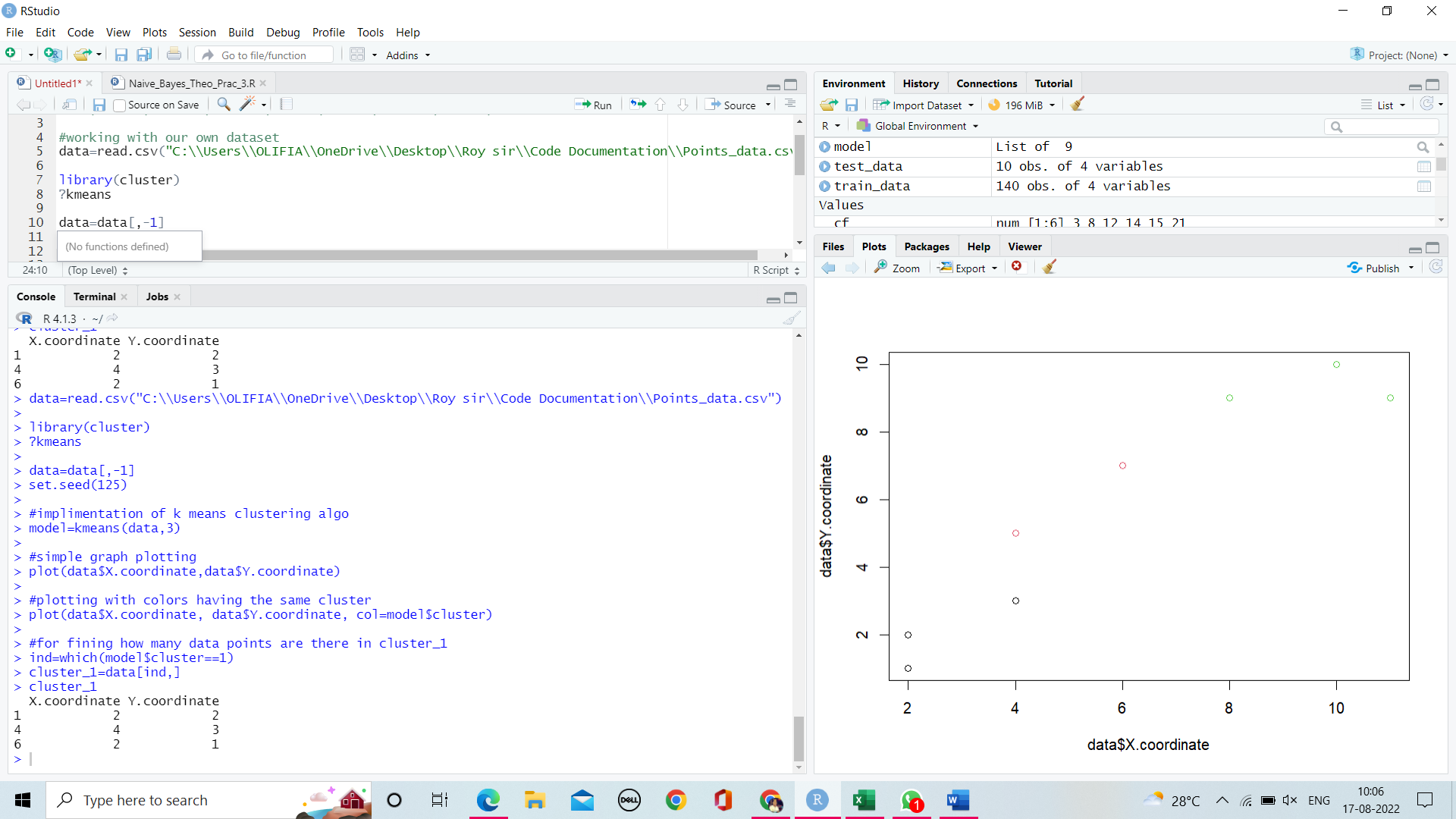
#for fining how many data points are there in cluster\_1

ind=which(model$cluster==1)

cluster\_1=data[ind,]

**Output:**





**Practical 6**

**Single Link Clustering Algorithm**

Points X Y

A 3 4

B 5 6

C 2 2

D 10 10

E 9 5

F 7 8

**Code:**

#single link algorithm

#working with data points

data=read.csv("C:\\Users\\OLIFIA\\OneDrive\\Desktop\\Roy sir\\Code Documentation\\Points\_data.csv")

#finding distance for distance vertex by using manhattan formula

dist=dist(data,method="manhattan")

#library for lingle link algo

library(cluster)

?hclust

#implementaing averange link algo

model=hclust(dist,method = "average")

#plotting dendogram

plot(model)

#this fuctions shows the position of data points in each cluster

#-1=A, -2=B, -3=C, and so on

model$merge

#this gives perfect dendogram on one bottom line

plot(model,hang = -1)

#we get a desired number of cluster eg here it gives 3 clusters out of 4 clusters

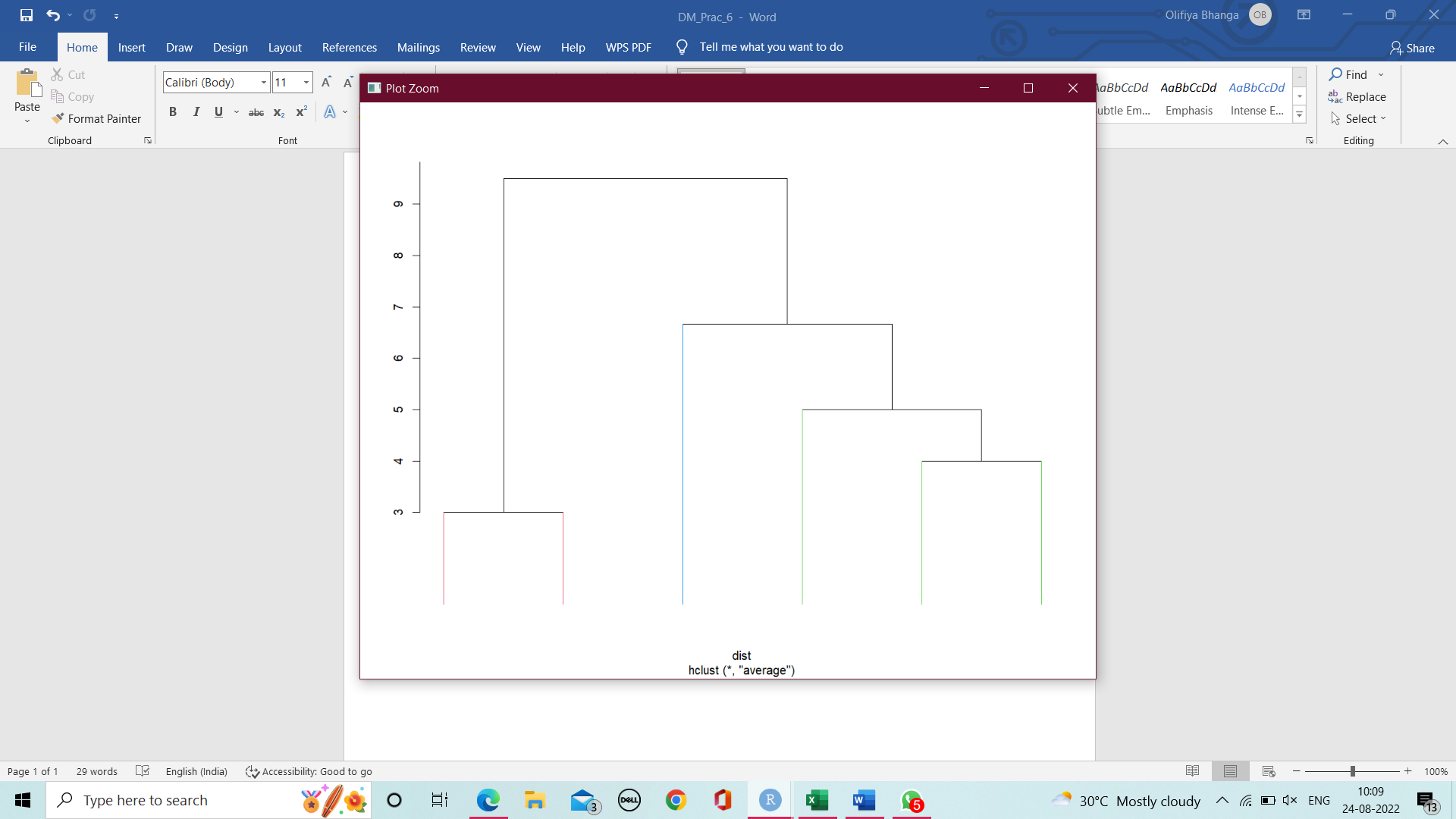
model2=cutree(model,3)

#color dendogram with desired cluster

library(sparcl)

ColorDendrogram(model,model2,branchlength = 20)

**Output:**



**Single and complete link clustering algo**

**Code:**

**#single link clustering algorithm**

data=read.csv('marks\_dataset.csv')

data=data[,-1]

dis=dist(data, method='manhattan')

library(cluster)

model=hclust(dis, method='single')

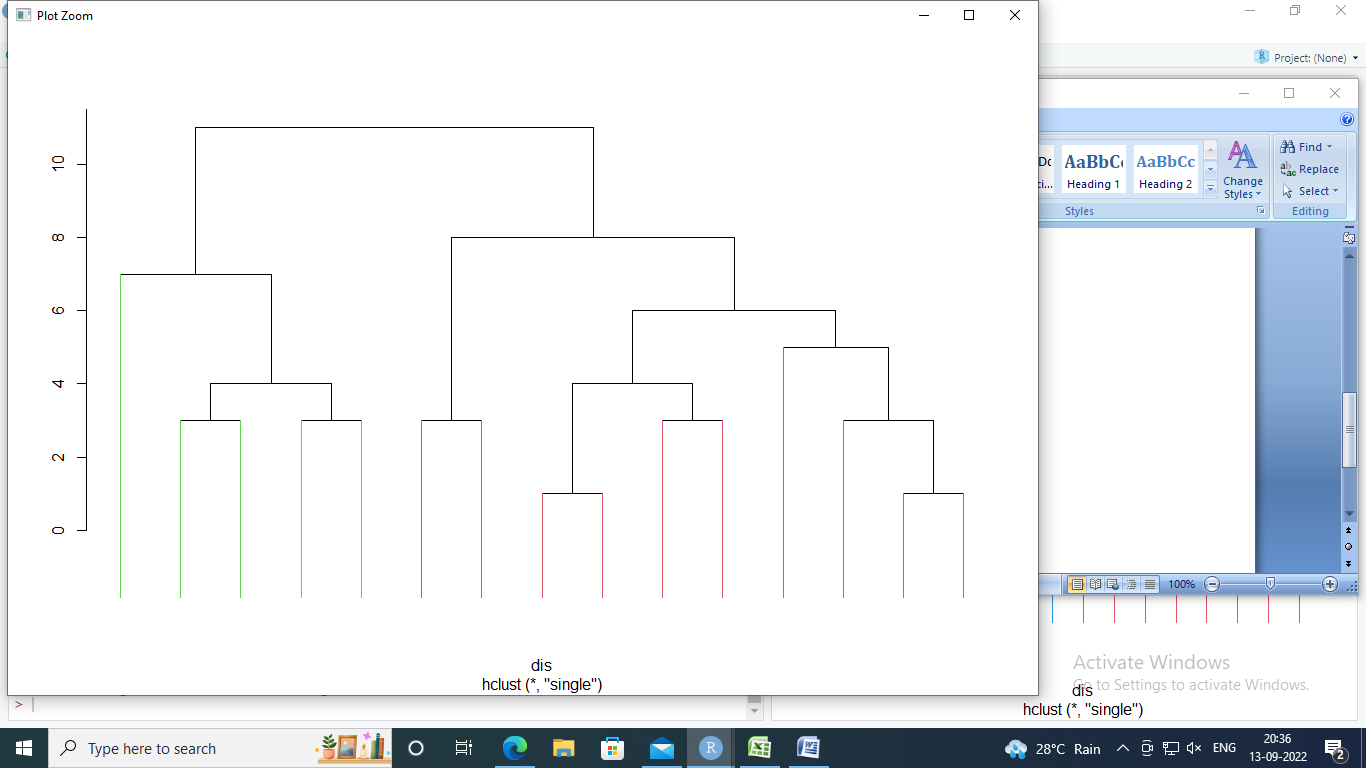
plot(model)

#cutree gives the specific number of custer

m3=cutree(model,3)

library(sparcl)

ColorDendrogram(model, m3, branchlength = 20)



**#complete link clustering algorithm**

data=read.csv('marks\_dataset.csv')

data=data[,-1]

dis=dist(data, method='manhattan')

library(cluster)

model=hclust(dis, method='complete')

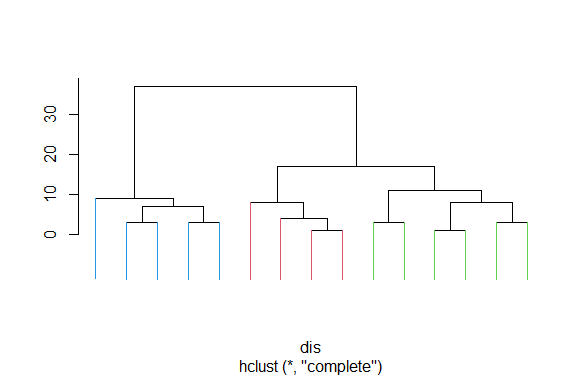
plot(model)

#cutree gives the specific number of custer

m3=cutree(model,3)

library(sparcl)

ColorDendrogram(model, m3, branchlength = 20)



**DBSCAN**

#circle wityh radius 5

x1=y1=0

for (i in 0:360) {

x1=5\*cos(i)

y1=5\*sin(i)

x=append(x,x1)

y=append(y,y1)

}

#circle wityh radius 10

for (i in 0:360) {

x1=10\*cos(i)

y1=10\*sin(i)

x=append(x,x1)

y=append(y,y1)

}

table=rbind(x,y)

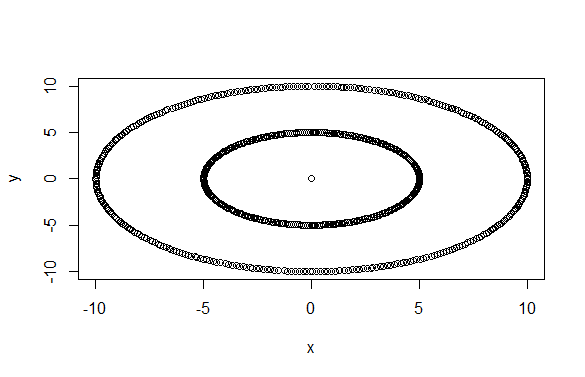
data=data.frame(x,y)

plot(data)

library(fpc)

?dbscan

model=dbscan(data,eps=1,MinPts = 3)



**Apriori Algorithm**

**Apply apriopi algo and find association rule from the following data. Minimum support count= 3 and confidence=80%.**

T1=A,B,C,D,E

T2=A,C,D

T3=B,C,D,E

T4=A,P,R

T5=P,Q,R

T6=A,C,D,E

T7=M,N

T8=A,M,N

T9=A,B,C

|  |  |  |
| --- | --- | --- |
| C4 | Count | L4 |
| ABCD | 2 | ACDE |
| ABCE | 2 | BCDE |
| ACDE | 3 |  |
| BCDE | 3 |  |
| ABDE | 2 |  |

T10=A,B,C,D,E

**Theory:-**

|  |  |  |
| --- | --- | --- |
| C1 | Count | L1 |
| A | 7 | A |
| B | 4 | B |
| C | 6 | C |
| D | 5 | D |
| E | 4 | E |
| M | 2 |  |
| N | 2 |  |
| P | 2 |  |
| Q | 1 |  |
| R | 2 |  |

|  |  |  |
| --- | --- | --- |
| C2 | Count | L2 |
| AB | 3 | AB |
| AC | 5 | AC |
| AD | 4 | AD |
| AE | 3 | AE |
| BC | 4 | BC |
| BD | 3 | BD |
| BE | 3 | BE |
| CD | 5 | CD |
| CE | 4 | CE |
| DE | 4 | DE |

|  |  |  |
| --- | --- | --- |
| C3 | Count | L3 |
| ABC | 3 | ABC |
| ABD | 2 | ACD |
| ABE | 2 | ACE |
| ACD | 4 | ADE |
| ACE | 3 | BCD |
| ADE | 3 | BCE |
| BCD | 3 | BDE |
| BCE | 3 | CDE |
| BDE | 3 |  |
| CDE | 4 |  |

|  |  |  |
| --- | --- | --- |
| C5 | Count | L5 |
| ABCDE | 2 | NULL |

L={L1 U L2 U L3 U L4 }

Consider any one large data set i.e ACDE

ACD -> E

Conf(ACD -> E)= n(ACDE) / n(ACD) = 3/4 = 0.75% < 80%

Conf(CDE ->A)= n(ACDE) / n(CDE) = 3/4 = 0.75% < 80%

BCD -> E

Conf(BCD -> E)= n(BCDE) / n(BCD) = 3/3 = 100% > 80%

AB -> C

Conf(AB -> C)= n(ABC) / n(AB) = 3/3 = 100% > 80%

**Code:-**

library(arules)

??arules

?apriori

data = list(

c("A","B","C","D","E"),

c("A","C","D"),

c("B","C","D","E"),

c("A","P","R"),

c("P","Q","R"),

c("A","C","D","E"),

c("M","N"),

c("A","M","N"),

c("A","B","C"),

c("A","B","C","D","E")

)

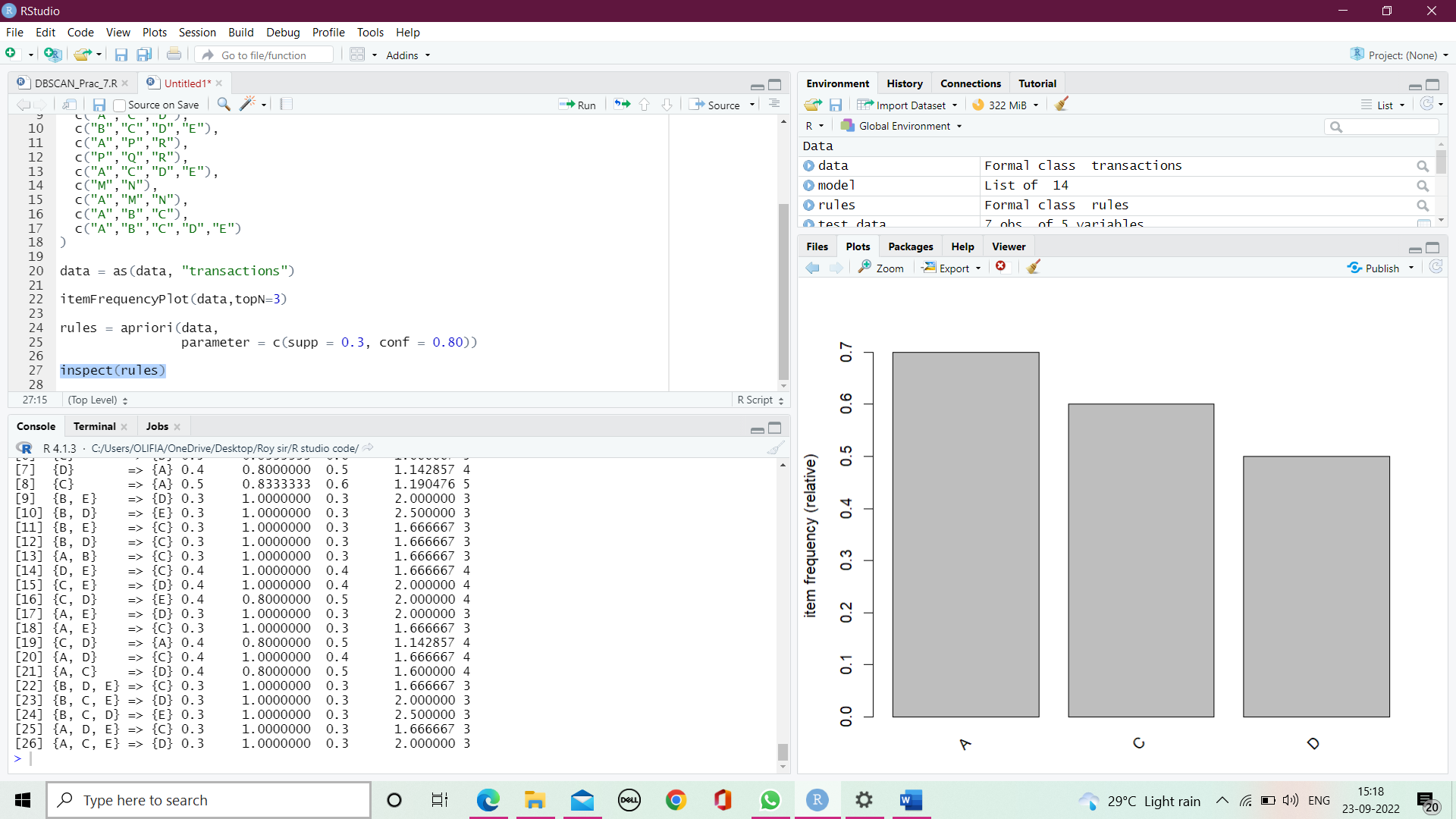
data = as(data, "transactions")

itemFrequencyPlot(data, topN=3)

rules = apriori(data,

parameter = c(supp = 0.3, conf = 0.80))

inspect(rules)



**#working with groceries dataset**

data("Groceries")

itemFrequencyPlot(data, topN=5)

rules = apriori(data,

parameter = c(supp = 0.3, conf = 0.80 ))

inspect(rules)

