DATA MINING

PRACTICAL FILE

NAME – SANDEEP YADAV

ROLL NUMBER- 18009570017/1802035

SEMESTER- VI

Program 1 : Create a file "people.txt" with the following data:

Age	Age Group	Height	Status	Years Married
21	adult	6.0	single	-1
2	child	3	married	0
18	adult	5.7	married	20
221	elderly	5	widowed	2
34	child	-7	married	3

i) Read the data from the file "people.txt".
ii) Create a rule set E that contain rules to check for the following conditions :

1. **The age should be in the range 0-150.**
2. **The age should be greater than years married.**
3. **The status should be married or single or widowed.**
4. **If age is less than 18 the age group should be child, if age is between 18 and 65 the age group should be adult, if age is more than 65 the age group should be elderly.**

iii) Check whether rule set E is violated by the data in the file people.txt.
iv) Summarize the results obtained in part(iii)
v) Visualize the results obtained in part(iii)

```
install.packages(editrules)

library(editrules)

d = read.table(file.choose(),header=TRUE)  #select people.txt

attach(d)  #to avoid using $ symbol with dataset

E <- editset(expression(

  age >= 0,

  age <= 150,
```

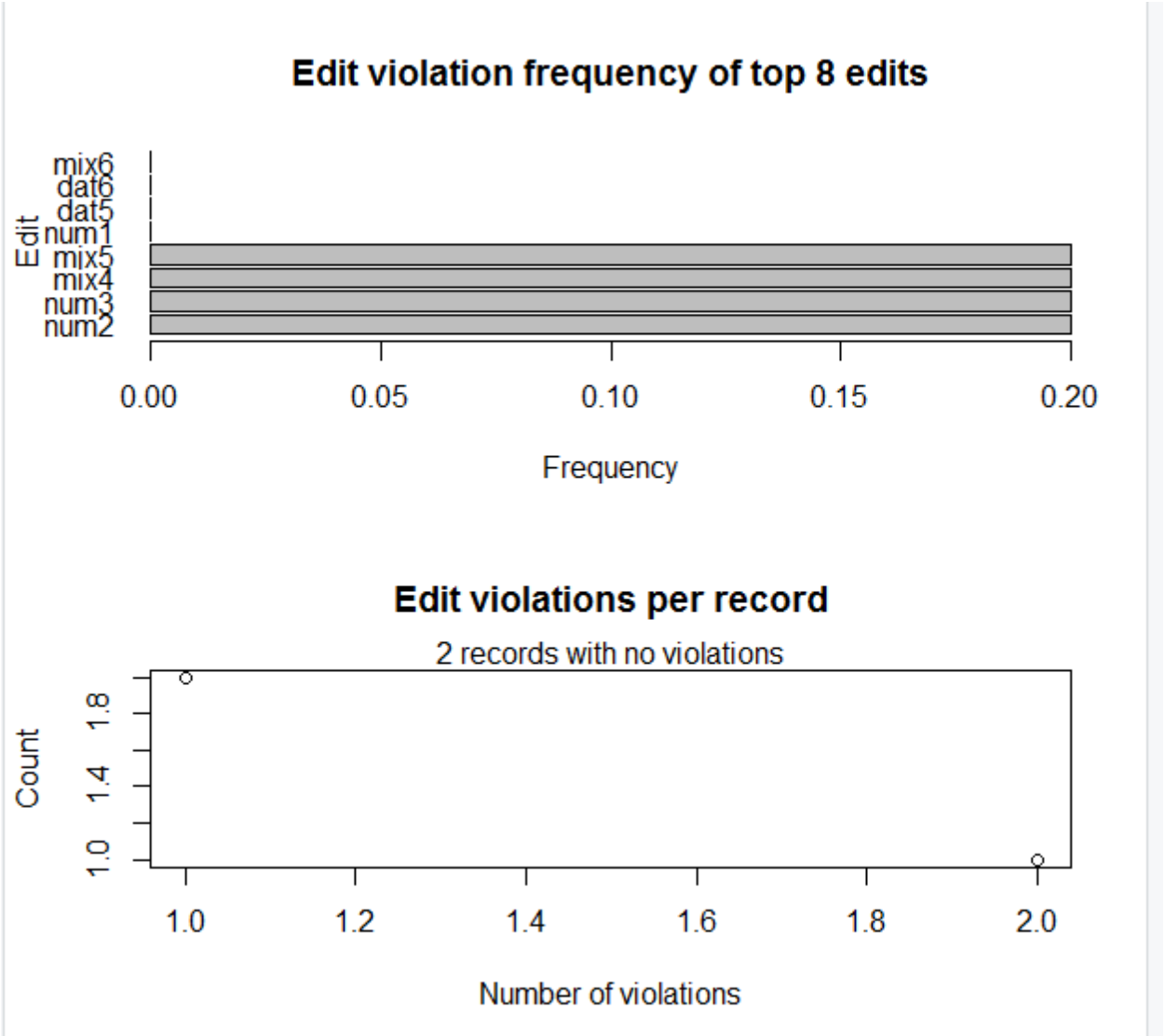
```
  age > yearsmarried,
  status %in% c('single','married','widowed'),
  if(age <= 18) agegroup %in% c('child'),
  if(age > 18 && age < 65 ) agegroup %in% c('adult'),
  if(age >= 65) agegroup %in% c('elderly')
))

sm <- violatedEdits(E,d)
summary(sm)
plot(sm)
```

TEXT FILE:-

```
age agegroup height status yearsmarried
 21 adult     6.0   single    -1
 2  child     3     married   0
 18 adult     5.7   married   20
 221 elderly   5    widowed    2
 34  child    -7    married    3
```

OUTPUT:

```
> summary(sm)
Edit violations, 5 observations, 0 completely missing (0%):

 editname freq rel
     num2    1 20%
     num3    1 20%
     mix4    1 20%
     mix5    1 20%

Edit violations per record:

 errors freq rel
      0    2 40%
      1    2 40%
      2    1 20%
> plot(sm)
>
```



Program 2: Perform the following preprocessing tasks on the dirty_iris dataset.

i) Calculate the number and percentage of observations that are complete.

ii) Replace all the special values in data with NA.

iii) Define these rules in a separate text file and read them.

(Use editfile function in R (package editrules). Use similar function in Python).

Print the resulting constraint object.

- **– Species should be one of the following values: setosa, versicolor or virginica.**
- **– All measured numerical properties of an iris should be positive.**
- **– The petal length of an iris is atleast 2 times its petal width.**
- **– The sepal length of an iris cannot exceed 30cm.**
- **– The sepals of an iris are longer than its petals.**

iv) Determine how often each rule is broken (violatedEdits). Also summarize and plot the result.

v) Find outliers in sepal length using boxplot and boxplot.stats

```
x = read.csv(file.choose())  #select dirty_iris.csv


#replace special values with NA

x[,-5] = lapply(x[,-5], function(y) as.numeric(as.character(y)))


#total number of complete observations

c = sum(complete.cases(x))

cat("Number of complete observations : ", c, "\n")


#percentage of complete observations

cat("Number of complete observations : ", c/(dim(x)[1])*100, "\n\n")

x = na.omit(x)    #delete records with NAs
```

```
library(editrules)
edit2 <- editfile(file.choose())  #select rules2.txt
sm <- violatedEdits(edit2,x)
summary(sm)
plot(sm)

boxplot(iris$Sepal.Length)
boxplot.stats(iris$Sepal.Length)
```

CSV FILE

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
@		1.3	0.2	setosa
4.6	3.1	1.5	0.2	ABC
5	3.6	1.4	0.2	setosa
5.4		1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
_	3.4	1.5	0.2	XYZ
4.4	2.9	1.4	0.2	setosa
4.9	@	1.5	0.1	setosa
$	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa

4.8	3	1.4	0.1	setosa
4.3	3	1.1	0.1	setosa
5.8	4	1.2	0.2	setosa
5.7	4.4	-1.5	0.4	setosa
5.4	3.9	-1.3	0.4	setosa
5.1	3.5	6	0.3	setosa
48	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa
44	3.4	0.2	0.2	setosa
5.1	3.7	1.5	0.4	setosa
4.6	3.6	1	0.2	setosa
5.1	3.3	1.7	0.5	setosa
4.8	3.4	1.9	0.2	setosa
5	3	1.6	0.2	setosa
5	3.4	1.6	0.4	setosa
5.2	3.5	1.5	0.2	setosa
5.2	3.4	1.4	0.2	setosa
4.7	3.2	1.6	0.2	setosa
4.8	3.1	1.6	0.2	setosa
5.4	3.4	1.5	0.4	setosa
5.2	4.1	1.5	0.1	setosa
5.5	4.2	1.4	0.2	setosa

4.9	3.1	1.5	0.1	setosa
5	3.2	1.2	0.2	setosa
5.5	3.5	1.3	0.2	setosa
4.9	3.1	1.5	0.1	setosa
4.4	3	1.3	0.2	setosa
5.1	3.4	1.5	0.2	setosa
5	3.5	1.3	0.3	setosa
4.5	2.3	1.3	0.3	setosa
4.4	3.2	1.3	0.2	setosa
5	3.5	1.6	0.6	setosa
5.1	3.8	1.9	0.4	setosa
4.8	3	1.4	0.3	setosa
5.1	3.8	1.6	0.2	setosa
4.6	3.2	1.4	0.2	setosa
5.3	3.7	1.5	0.2	setosa
5	3.3	1.4	0.2	setosa
7	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor
5.7	2.8	4.5	1.3	versicolor

6.3	3.3	4.7	1.6	versicolor
4.9	2.4	3.3	1	versicolor
6.6	2.9	4.6	1.3	versicolor
5.2	2.7	3.9	1.4	versicolor
5	2	3.5	1	versicolor
5.9	3	4.2	1.5	versicolor
6	2.2	4	1	versicolor
6.1	2.9	4.7	1.4	versicolor
5.6	2.9	3.6	1.3	versicolor
6.7	3.1	4.4	1.4	versicolor
5.6	3	4.5	1.5	versicolor
5.8	2.7	4.1	1	versicolor
6.2	2.2	4.5	1.5	versicolor
5.6	2.5	3.9	1.1	versicolor
5.9	3.2	4.8	1.8	versicolor
6.1	2.8	4	1.3	versicolor
6.3	2.5	4.9	1.5	versicolor
6.1	2.8	4.7	1.2	versicolor
6.4	2.9	4.3	1.3	versicolor
6.6	3	4.4	1.4	versicolor
6.8	2.8	4.8	1.4	versicolor
6.7	3	5	1.7	versicolor

6	2.9	4.5	1.5	versicolor
5.7	2.6	3.5	1	versicolor
5.5	2.4	3.8	1.1	versicolor
5.5	2.4	3.7	1	versicolor
5.8	2.7	3.9	1.2	versicolor
6	2.7	5.1	1.6	versicolor
5.4	3	4.5	1.5	versicolor
6	3.4	4.5	1.6	versicolor
6.7	3.1	4.7	1.5	versicolor
6.3	2.3	4.4	1.3	versicolor
5.6	3	4.1	1.3	versicolor
5.5	2.5	4	1.3	versicolor
5.5	2.6	4.4	1.2	versicolor
6.1	3	4.6	1.4	versicolor
5.8	2.6	4	1.2	versicolor
5	2.3	3.3	1	versicolor
5.6	2.7	4.2	1.3	versicolor
5.7	3	4.2	1.2	versicolor
5.7	2.9	4.2	1.3	versicolor
6.2	2.9	4.3	1.3	versicolor
5.1	2.5	3	1.1	versicolor
5.7	2.8	4.1	1.3	versicolor

6.3	3.3	6	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3	5.9	2.1	virginica
6.3	2.9	5.6	1.8	virginica
6.5	3	5.8	2.2	virginica
7.6	3	6.6	2.1	virginica
4.9	2.5	4.5	1.7	virginica
7.3	2.9	6.3	1.8	virginica
6.7	2.5	5.8	1.8	virginica
7.2	3.6	6.1	2.5	virginica
6.5	3.2	5.1	2	virginica
6.4	2.7	5.3	1.9	virginica
6.8	3	5.5	2.1	virginica
5.7	2.5	5	2	virginica
5.8	2.8	5.1	2.4	virginica
6.4	3.2	5.3	2.3	virginica
6.5	3	5.5	1.8	virginica
7.7	3.8	6.7	2.2	virginica
7.7	2.6	6.9	2.3	virginica
6	2.2	5	1.5	virginica
6.9	3.2	5.7	2.3	virginica
5.6	2.8	4.9	2	virginica

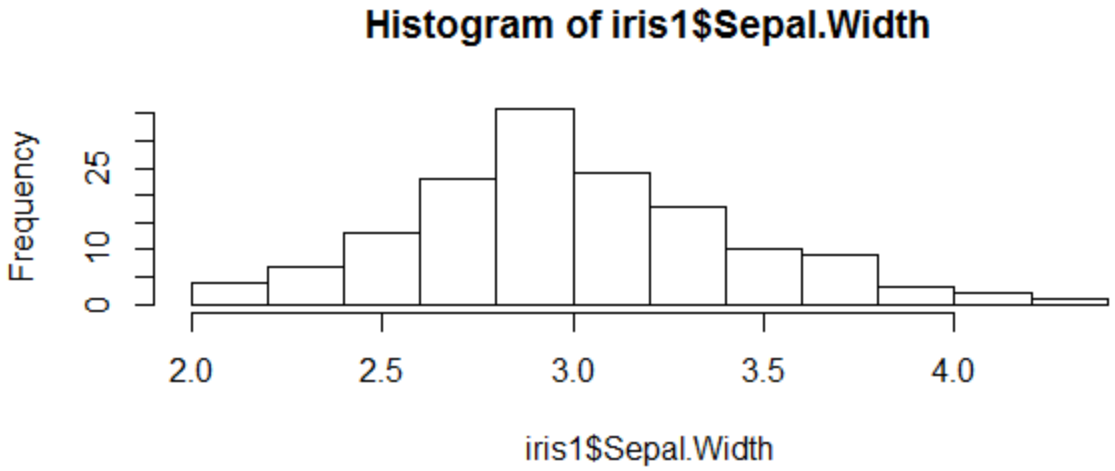
7.7	2.8	6.7	2	virginica
6.3	2.7	4.9	1.8	virginica
6.7	3.3	5.7	2.1	virginica
7.2	3.2	6	1.8	virginica
6.2	2.8	4.8	1.8	virginica
6.1	3	4.9	1.8	virginica
6.4	2.8	5.6	2.1	virginica
7.2	3	5.8	1.6	virginica
7.4	2.8	6.1	1.9	virginica
7.9	3.8	6.4	2	virginica
6.4	2.8	5.6	2.2	virginica
6.3	2.8	5.1	1.5	virginica
6.1	2.6	5.6	1.4	virginica
7.7	3	6.1	2.3	virginica
6.3	3.4	5.6	2.4	virginica
6.4	3.1	5.5	1.8	virginica
6	3	4.8	1.8	virginica
6.9	3.1	5.4	2.1	virginica
6.7	3.1	5.6	2.4	virginica
6.9	3.1	5.1	2.3	virginica
5.8	2.7	5.1	1.9	virginica
6.8	3.2	5.9	2.3	virginica

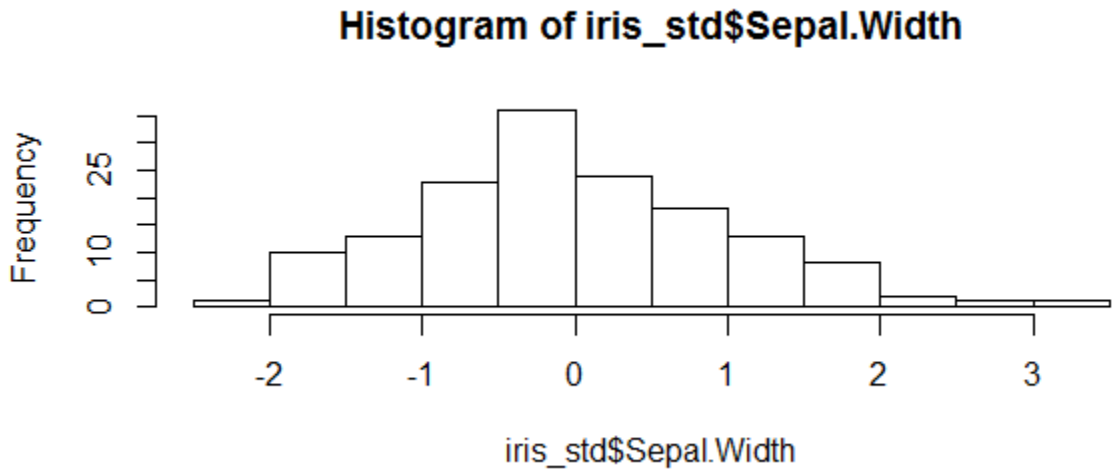
6.7	3.3	5.7	2.5	virginica
6.7	3	5.2	2.3	virginica
6.3	2.5	5	1.9	virginica
6.5	3	5.2	2	virginica
6.2	3.4	5.4	2.3	virginica
5.9	3	5.1	1.8	virginica

Program 3: Load the data from wine dataset. Check whether all attributes are standardized or not (mean is 0 and standard deviation is 1). If not, standardize the attributes. Do the same with Iris dataset.

```
iris1 <- iris[,-5]

summary(iris1)

s <- sapply(iris1,sd)

s

hist(iris1$Sepal.Width)  #check bell shape curve for all attributes

data_std <- function(x) { (x-mean(x))/sd(x) }

iris_std <- data.frame(sapply(iris[,-5],data_std))

summary(iris_std)

s <- sapply(iris_std,sd)

s

hist(iris_std$Sepal.Width)  #check bell shape curve for all attributes
```

OUTPUT:


Histogram of iris1$Sepal.Width
Frequency
0
10
25
2.0
2.5
3.0
3.5
4.0
iris1$Sepal.Width


Histogram of iris_std$Sepal.Width
Frequency
0
10
25
-2
-1
0
1
2
3
iris_std$Sepal.Width

```
> iris1 <- iris[,-5]
> summary(iris1)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> s <- sapply(iris1,sd)
> s
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.8280661    0.4358663    1.7652982    0.7622377
> hist(iris1$Sepal.Width)  #check bell shape curve for all attributes
> data_std <- function(x) { (x-mean(x))/sd(x) }
> iris_std <- data.frame(sapply(iris[,-5],data_std))
> summary(iris_std)
  Sepal.Length      Sepal.Width       Petal.Length      Petal.Width
 Min.   :-1.86378   Min.   :-2.4258   Min.   :-1.5623   Min.   :-1.4422
 1st Qu.:-0.89767   1st Qu.:-0.5904   1st Qu.:-1.2225   1st Qu.:-1.1799
 Median :-0.05233   Median :-0.1315   Median : 0.3354   Median : 0.1321
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.67225   3rd Qu.: 0.5567   3rd Qu.: 0.7602   3rd Qu.: 0.7880
 Max.   : 2.48370   Max.   : 3.0805   Max.   : 1.7799   Max.   : 1.7064
> s <- sapply(iris_std,sd)
> s
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
           1            1            1            1
> hist(iris_std$Sepal.Width)  #check bell shape curve for all attributes
>
```

Program 4: Run Apriori algorithm to find frequent itemsets and association rules

4.1 Use minimum support as 50% and minimum confidence as 75%

4.2 Use minimum support as 60% and minimum confidence as 60%

```
library(arules)

data(Groceries)


#SUPPORT=0.1% CONFIDENCE=80%

itemFrequencyPlot(Groceries, topN=20, type="absolute")

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

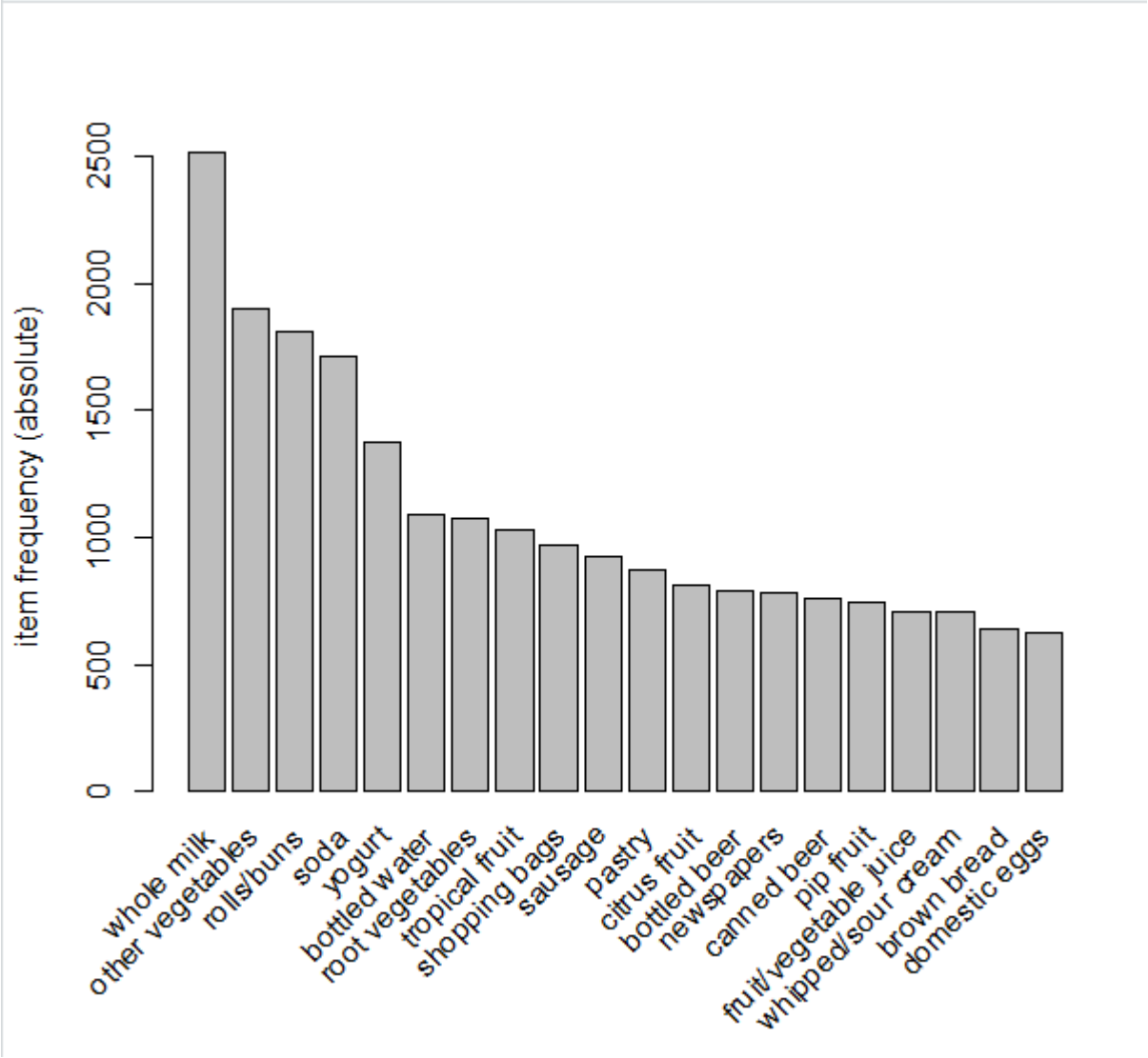
```
inspect(head(rules))

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

inspect(head(rules))



#SUPPORT=0.5% CONFIDENCE=70%

itemFrequencyPlot(Groceries, topN=20, type="absolute")

rules <- apriori(Groceries, parameter=list(sup=0.005,conf=0.7))

inspect(head(rules))

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

inspect(head(rules))
```

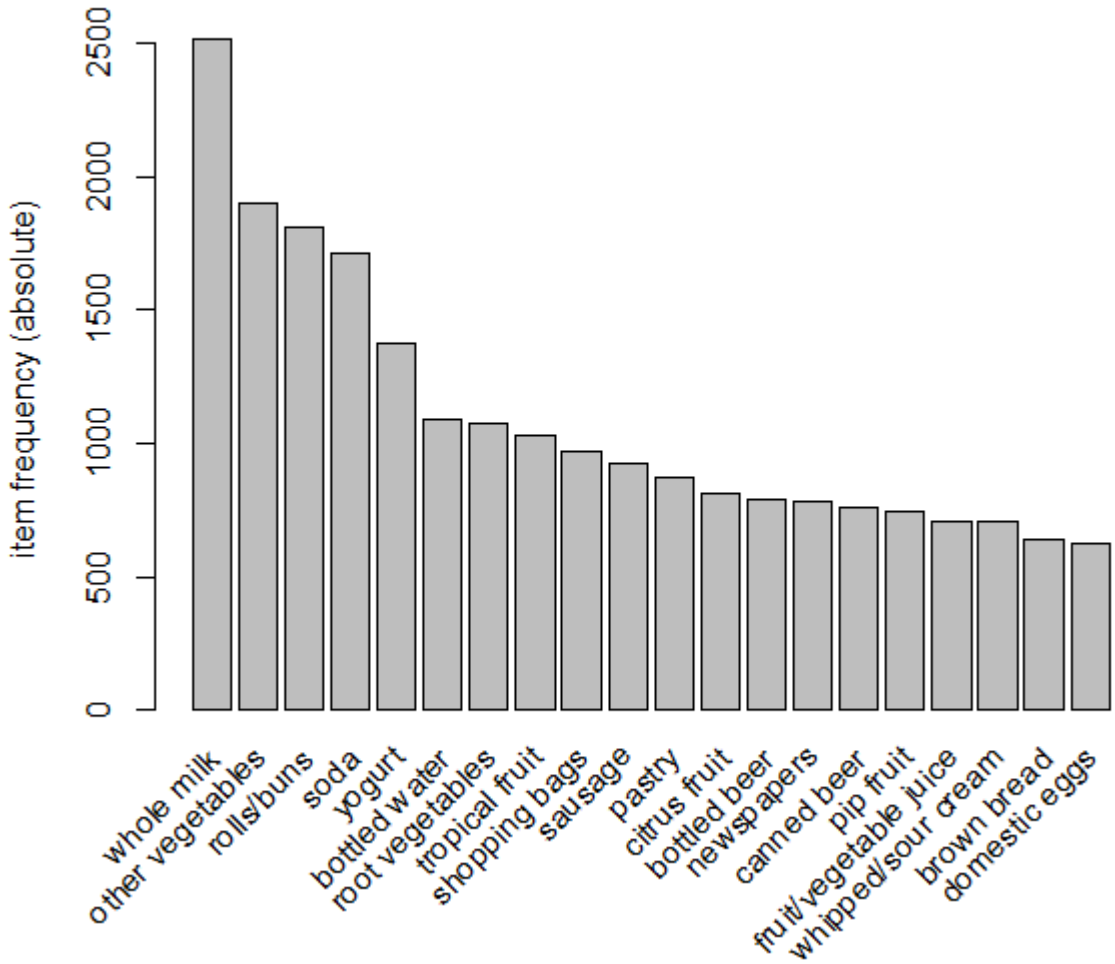
OUTPUT:

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5   0.001      1
 maxlen target   ext
     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.02s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 done [0.07s].
writing ... [410 rule(s)] done [0.01s].
creating S4 object  ... done [0.01s].
> inspect(head(rules))
    lhs                              rhs            support     confidence
[1] {liquor,red/blush wine}       => {bottled beer} 0.001931876 0.9047619
[2] {curd,cereals}                => {whole milk}   0.001016777 0.9090909
```

```
[3] {yogurt,cereals}                 => {whole milk}   0.001728521 0.8095238
[4] {butter,jam}                     => {whole milk}   0.001016777 0.8333333
[5] {soups,bottled beer}             => {whole milk}   0.001118454 0.9166667
[6] {napkins,house keeping products} => {whole milk}   0.001321810 0.8125000
    lift      count
[1] 11.235269 19
[2]  3.557863 10
[3]  3.168192 17
[4]  3.261374 10
[5]  3.587512 11
[6]  3.179840 13
> rules <- sort(rules, by="confidence", decreasing = TRUE)
> inspect(head(rules))
    lhs                     rhs                    support confidence
lift count
[1] {rice,
     sugar}              => {whole milk}       0.001220132          1
3.913649    12
[2] {canned fish,
     hygiene articles}   => {whole milk}       0.001118454          1
3.913649    11
[3] {root vegetables,
     butter,
     rice}               => {whole milk}       0.001016777          1
3.913649    10
[4] {root vegetables,
     whipped/sour cream,
     flour}              => {whole milk}       0.001728521          1
3.913649    17
[5] {butter,
     soft cheese,
     domestic eggs}      => {whole milk}       0.001016777          1
3.913649    10
[6] {citrus fruit,
     root vegetables,
     soft cheese}        => {other vegetables} 0.001016777          1
5.168156    10
> #SUPPORT=0.5% CONFIDENCE=70%
> itemFrequencyPlot(Groceries, topN=20, type="absolute")
> rules <- apriori(Groceries, parameter=list(sup=0.005,conf=0.7))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.7    0.1    1 none FALSE            TRUE       5   0.005      1
 maxlen target   ext
     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 49

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.02s].
sorting and recoding items ... [120 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.02s].
writing ... [1 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
> inspect(head(rules))
```

```
    lhs                                                rhs          support
[1] {tropical fruit,root vegetables,yogurt} => {whole milk} 0.00569395
    confidence lift     count
[1] 0.7        2.739554 56
> rules <- sort(rules, by="confidence", decreasing = TRUE)
> inspect(head(rules))
    lhs                                                rhs          support
[1] {tropical fruit,root vegetables,yogurt} => {whole milk} 0.00569395
    confidence lift     count
[1] 0.7        2.739554 56
```





Program 5: **Use Naive bayes, K-nearest, and Decision tree classification algorithms and build classifiers. Divide the data set in to training and test set. Compare the accuracy of the different classifiers under the following situations:**

5.1 a) Training set = 75% Test set = 25%

5.1 b) Training set = 66.6% (2/3rd of total), Test set = 33.3%

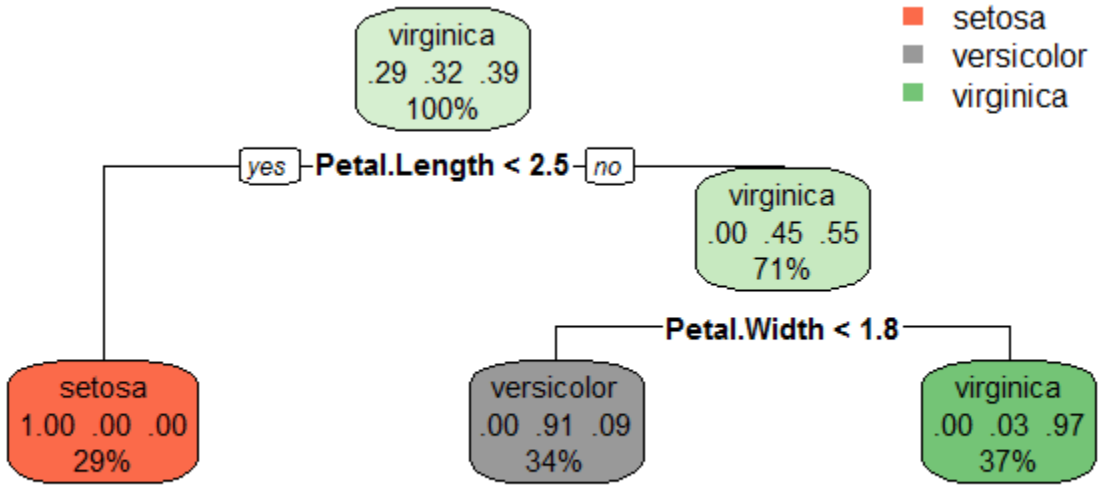
5.2 Training set is chosen by i)hold out method ii)Random subsampling iii)Cross-Validation. Compare the accuracy of the classifiers obtained.

5.3 Data is scaled to standard format.

```
#Naive Bayes
library(naivebayes)
library(caret)
set.seed(1234)
id <- sample(2,150,prob=c(0.7,0.3),replace = T)
train <- iris[id==1,]
test <- iris[id==2,]
imp_nb = naive_bayes(Species~., data=train)
pre3 <- predict(imp_nb,test)
confusionMatrix(table(pre3, test$Species))
mean(pre3==test[,5])

#KNN
library(class)
normalize <- function(x) { (x-min(x))/(max(x)-min(x)) }
iris_norm <- sapply(iris[,-5], normalize)
s <- sample(150,120)
iris_train <- iris_norm[s,]
iris_test <- iris_norm[-s,]
iris_pred <- knn(iris_train,iris_test,iris[s,5],k=13)
table(iris_pred,iris[-s,5])
mean(iris_pred==iris[-s,5])
```

```
#DECISION TREE

library(rpart)

library(rpart.plot)

s <- sample(150, 100)

train <- iris[s,]

test <- iris[-s,]

d <- rpart(Species~., train, method = "class")

rpart.plot(d)

p <- predict(d, test, type  = "class")

table(test[,5], p)

mean(p==test[,5])
```

OUTPUT:



```
> #DECISION TREE
> library(rpart)
> library(rpart.plot)
> s <- sample(150, 100)
> train <- iris[s,]
> test <- iris[-s,]
> d <- rpart(Species~., train, method = "class")
> rpart.plot(d)
> p <- predict(d, test, type  = "class")
> table(test[,5], p)
            p
             setosa versicolor virginica
  setosa         21          0         0
  versicolor      0         18         0
  virginica       0          2         9
> mean(p==test[,5])
[1] 0.96
>
```

```
> #KNN
> library(class)

Attaching package: 'class'

The following object is masked from 'package:igraph':

    knn

> normalize <- function(x) { (x-min(x))/(max(x)-min(x)) }
> iris_norm <- sapply(iris[,-5], normalize)
> s <- sample(150,120)
> iris_train <- iris_norm[s,]
> iris_test <- iris_norm[-s,]
> iris_pred <- knn(iris_train,iris_test,iris[s,5],k=13)
> table(iris_pred,iris[-s,5])

iris_pred    setosa versicolor virginica
  setosa          9          0         0
  versicolor      0          9         0
  virginica       0          2        10
> mean(iris_pred==iris[-s,5])
[1] 0.9333333
```

```
> #Naive Bayes
> library(naivebayes)
> library(caret)
Loading required package: lattice
Loading required package: ggplot2
> set.seed(1234)
> id <- sample(2,150,prob=c(0.7,0.3),replace = T)
> train <- iris[id==1,]
> test <- iris[id==2,]
> imp_nb = naive_bayes(Species~., data=train)
> pre3 <- predict(imp_nb,test)
> confusionMatrix(table(pre3, test$Species))
Confusion Matrix and Statistics


pre3         setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         12         2
  virginica       0          0        14

Overall Statistics

               Accuracy : 0.9474
                 95% CI : (0.8225, 0.9936)
    No Information Rate : 0.4211
    P-Value [Acc > NIR] : 7.335e-12

                  Kappa : 0.9202

 Mcnemar's Test P-Value : NA


Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.8750
Specificity                 1.0000            0.9231           1.0000
Pos Pred Value              1.0000            0.8571           1.0000
Neg Pred Value              1.0000            1.0000           0.9167
Prevalence                  0.2632            0.3158           0.4211
Detection Rate              0.2632            0.3158           0.3684
Detection Prevalence        0.2632            0.3684           0.3684
Balanced Accuracy           1.0000            0.9615           0.9375
> mean(pre3==test[,5])
[1] 0.9473684
```

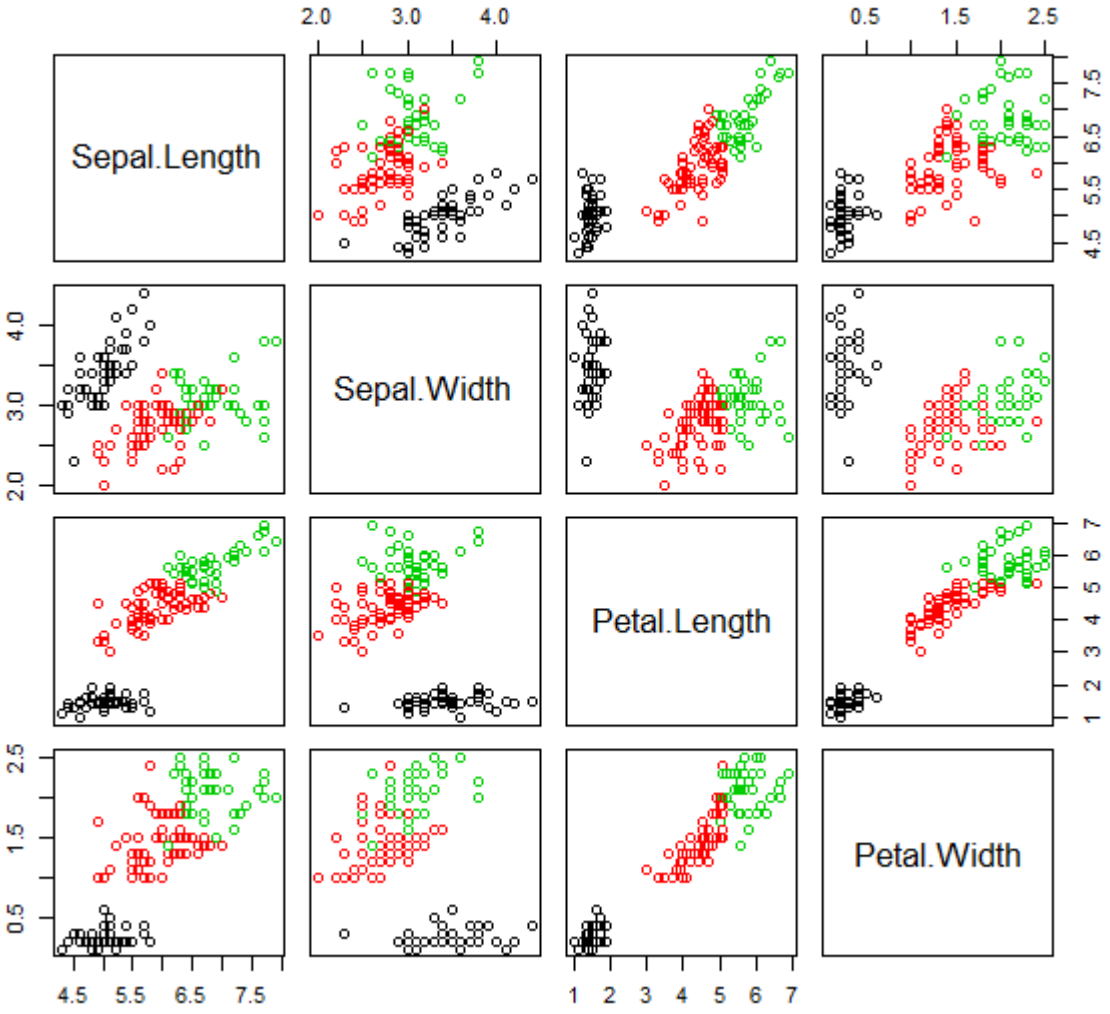
Program 6: Use Simple Kmeans, DBScan, Hierachical clustering algorithms for clustering. Compare the performance of clusters by changing the parameters involved in the algorithms.

#Kmeans

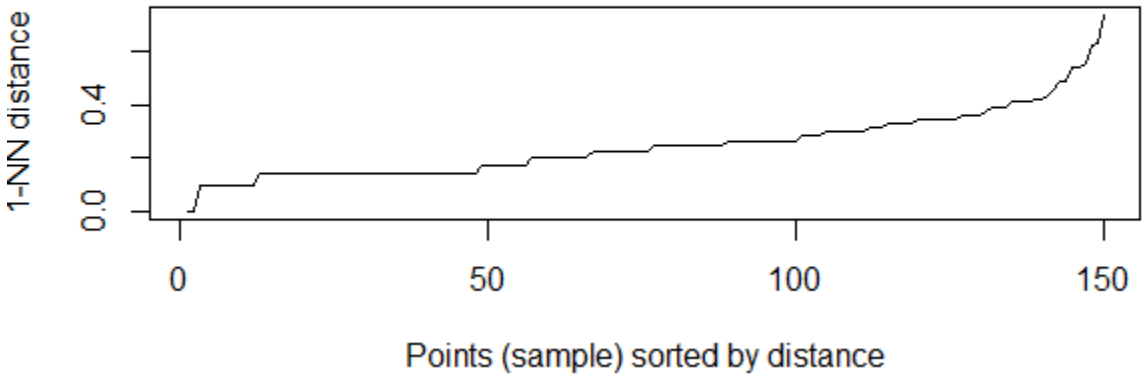
```
library(cluster)
iris1 <- iris[,1:4]
results <- kmeans(iris1,3)
results
table(iris$Species,results$cluster)
plot(iris[,-5],col=results$cluster)

#DBScan
library(dbscan)
iris_m <- iris[,1:4]
kNNdistplot(iris_m,k=1)
abline(h=0.4,col="red")
db <- dbscan(iris_m,0.4,4)
db
hullplot(iris_m,db$cluster)
table(iris$Species,db$cluster)

#Hierarchical Clustering
library(cluster)
hc_complete <- hclust(dist(iris),method = "complete")
hc_single <- hclust(dist(iris),method = "single")
plot(hc_complete, main = "Hierarchical Clustering Complete", cex = 0.9)
```

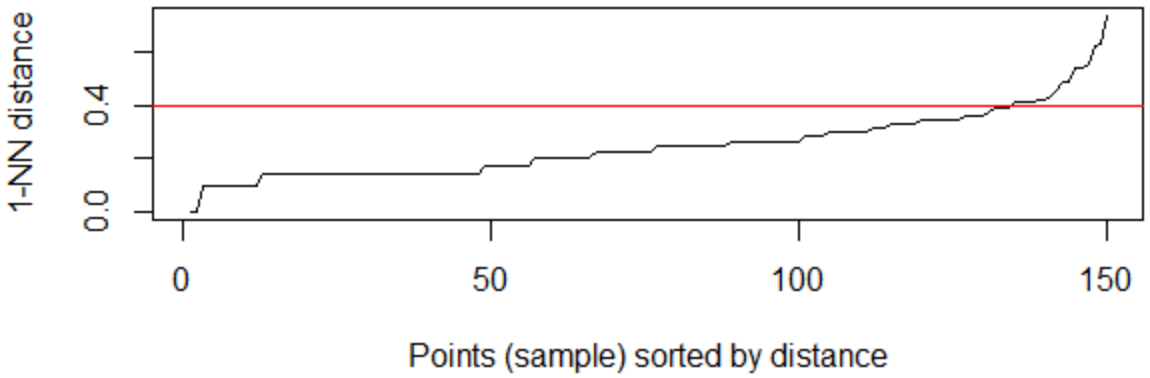
plot(hc_single, main = "Hierarchical Clustering Single", cex = 0.9)

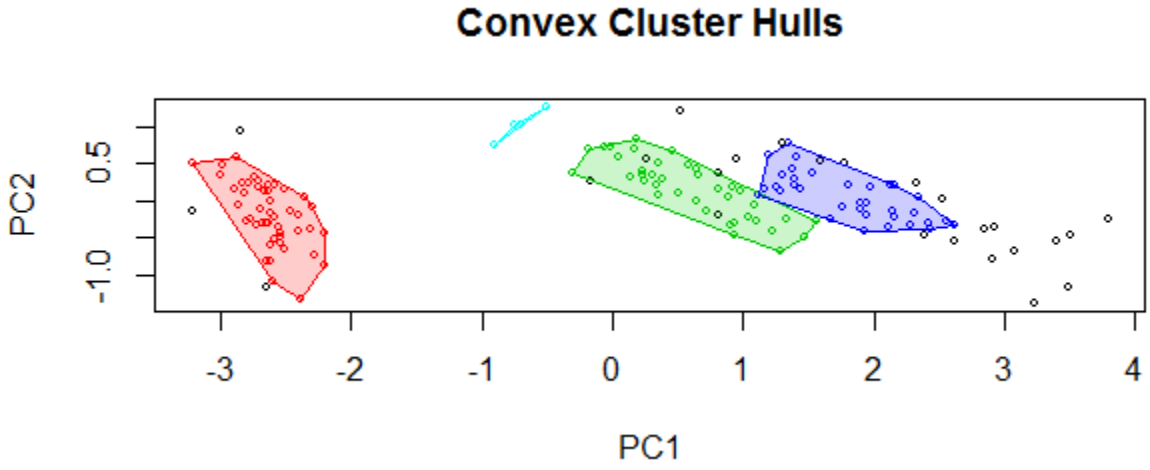
OUTPUT:



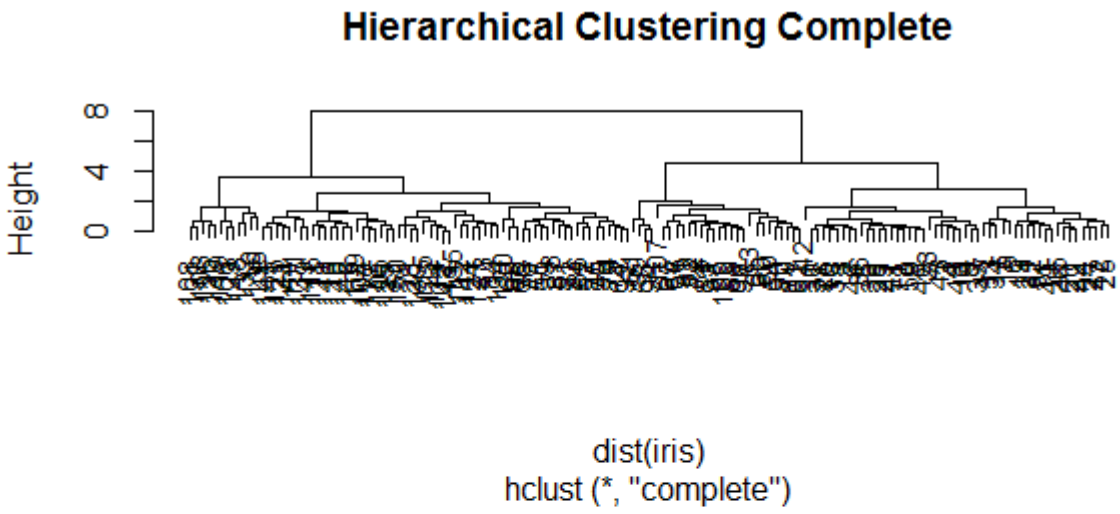
```
> #Kmeans
> library(cluster)
> iris1 <- iris[,1:4]
> results <- kmeans(iris1,3)
> results
K-means clustering with 3 clusters of sizes 50, 62, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [41] 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2
 [81] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2
[121] 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3 3 3 2 3 3 3 2 3 3 2

Within cluster sum of squares by cluster:
[1] 15.15100 39.82097 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
> table(iris$Species,results$cluster)

              1  2  3
  setosa     50  0  0
  versicolor  0 48  2
  virginica   0 14 36
> plot(iris[,-5],col=results$cluster)
>
```

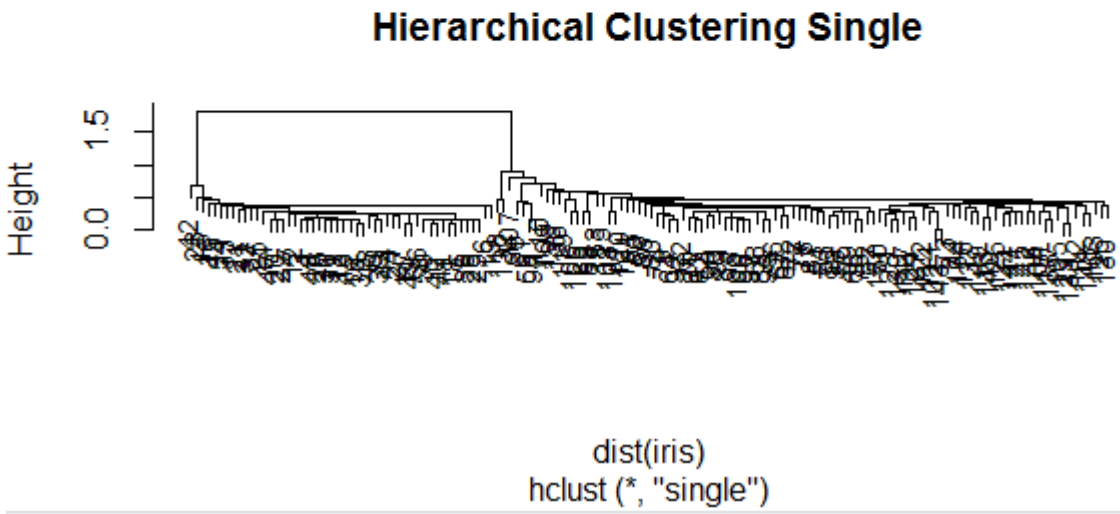



1-NN distance
0.0
0.4
0
50
100
150
Points (sample) sorted by distance


Convex Cluster Hulls
PC2
-1.0
0.5
-3
-2
-1
0
1
2
3
4
PC1

```
> #DBScan
> library(dbscan)
> iris_m <- iris[,1:4]
> kNNdistplot(iris_m,k=1)
> abline(h=0.4,col="red")
> db <- dbscan(iris_m,0.4,4)
> db
DBSCAN clustering for 150 objects.
Parameters: eps = 0.4, minPts = 4
The clustering contains 4 cluster(s) and 25 noise points.

 0  1  2  3  4
25 47 38 36  4

Available fields: cluster, eps, minPts
> hullplot(iris_m,db$cluster)
> table(iris$Species,db$cluster)

              0  1  2  3  4
  setosa      3 47  0  0  0
  versicolor  5  0 38  3  4
  virginica  17  0  0 33  0
>
```






```
> #Hierarchical Clustering
> library(cluster)
> hc_complete <- hclust(dist(iris),method = "complete")
Warning message:
In dist(iris) : NAs introduced by coercion
> hc_single <- hclust(dist(iris),method = "single")
Warning message:
In dist(iris) : NAs introduced by coercion
> plot(hc_complete, main = "Hierarchical Clustering Complete", cex = 0.9)
> plot(hc_single, main = "Hierarchical Clustering Single", cex = 0.9)
>
```