MULTIVARIATE ANALYSIS - ASSIGNMENT - 01

MADHUSUDHAN ASHWATH - 19203116

2/29/2020

Question 1

Load the Glass Data into working directory and few modifications for futher

```
glass = read.csv("C:/Users/Win10/Desktop/Glass.csv")

#Setting seed to my student number
set.seed(19203116)

#Delete a row from the dataset by randomly generating a integer between 0 to n
glass = glass[-floor(runif(1,min=0,max = nrow(glass))),]

#number of observations
o = nrow(glass)
```

1(a) i

```
#Remove the group column and convert the dataset into matrix
mat = as.matrix(glass[,-1])
mat[,1]
```

```
##      1      2      3      4      5      6      7      8      9     10     11
## 13.904 14.194 14.668 14.800 14.078 13.600 12.942 15.656 13.935 17.174 14.004
##     12     13     14     15     16     17     18     19     20     21     22
## 13.440 14.744 13.984 17.050 14.992 13.976 14.389  6.084  6.558  4.808  6.826
##     23     24     25     26     27     28     29     30     31     32     33
##  5.164  3.516  8.142  5.274  7.148  6.238  6.042  5.936  6.272  5.332  7.306
##     34     35     36     37     38     39     40     41     42     43     44
## 14.930 16.128 16.256 12.976 15.496 15.912 16.016 14.684 14.800 11.826 15.490
##     45     46     47     48     49     50     51     52     53     54     55
## 15.752 13.188 14.016 12.900 15.570 12.952 13.154 17.024 14.190 15.692 15.818
##     56     57     58     59     60     61     62     63     64     65     66
## 16.020  1.602  1.512  1.220  0.986  1.700  1.282  1.180  5.772  5.886  5.514
##     67     68     69     70     71     72     73     74     75     76     77
##  4.910  4.812  4.102  4.208  5.526  4.066  4.350  2.010  1.206  1.504 16.956
##     78     79     80     81     82     83     84     85     86     87     88
## 13.088 15.038 14.502 12.924 14.606 17.586 15.040 14.923 14.258 13.402 14.355
##     89     90     91     92     93     94     95     96     97     98     99
## 14.207 14.093 14.782 12.576 13.172 14.164 15.164 13.582 13.812 13.620 13.662
##    100    101    102    103    104    105    106    107    108    109    110
## 15.038 15.572 11.742 13.110 15.854 15.576 14.266 13.516 14.206  9.922 16.345
##    111    112    113    114    115    116    117    118    119    120    121
## 16.550 15.370 13.788 15.820 14.400 14.926 14.030 15.050 14.202 12.138 15.516
##    122    123    124    125    126    127    128    130    131    132    133
```

```
## 14.906 17.125 13.440 11.012 11.168 11.174 16.784 15.918 15.890 16.452 17.190
##    134    135    136    137    138    139    140    141    142    143    144
## 16.476 11.590 13.178 14.774 14.084 11.552 14.974 16.654 15.033 13.950 15.954
##    145    146    147    148    149    150    151    152    153    154    155
## 15.802 14.930 15.340 16.010 13.432 14.456 15.308 12.134 15.726 14.128 16.196
##    156    157    158    159    160    161    162    163    164    165    166
## 15.556 15.654 14.374 15.388 15.328 15.438 15.114 15.202 15.928 12.606 15.170
##    167    168    169    170    171    172    173    174    175    176    177
## 13.912 12.044 15.480 14.388 16.130 15.470 15.522 15.366 11.826 12.196 12.208
##    178    179    180
## 15.816 12.446 13.890
```

```r
#Column means
cbind(t(colMeans(mat)))
```

```
##           Na2O      MgO    Al2O3     SiO2      P2O5       SO3        Cl      K2O
## [1,] 12.55841 2.345782 1.638978 66.52081 0.5102458 0.1554749 0.6150894 5.856179
##            CaO       MnO    Fe2O3        BaO       PbO
## [1,] 8.390056 0.5350503 0.401933 0.07941341 0.3819274
```

```r
mean_glass = matrix(data = 1, nrow = o) %*% cbind(t(colMeans(mat)))

#Distance between datapoint from the mean
dif = mat - mean_glass

#Covariance Matrix
S = ((o-1)^-1) * t(dif) %*% dif
rownames(S) = c()

#Diagonal Element Matrix
Dia = diag(apply(mat,2,sd))
D = solve(Dia)

#Correlation Matrix
R = D %*% S %*% D

#Correlation of data set
R_cor = cor(mat)

#Comparization of two matrix
which(which(R == R_cor) == FALSE)
```

```
## integer(0)
```

1(a)ii

```r
# Eigen vector and Eigen values of covariance matrix
eig = eigen(S)
eigenvalue = eig$values
eigenvector = as.matrix(eig$vectors)
```

```r
#First two eigen value and vector
eigenvalue[1:2]
```

```
## [1] 43.60141 18.23893
```

```r
eigenvector[,1:2]
```

```
##              [,1]          [,2]
##  [1,]  0.525068308  0.5615551989
##  [2,] -0.089857079  0.0846160764
##  [3,] -0.072954152  0.0047922331
##  [4,]  0.536957743 -0.2888751972
##  [5,] -0.083264423  0.0302480045
##  [6,]  0.003688426  0.0033942452
##  [7,]  0.017102222  0.0281959189
##  [8,] -0.134668537 -0.6735119663
##  [9,] -0.628875252  0.3528485350
## [10,] -0.033784221  0.0109827903
## [11,] -0.016340521  0.0037434795
## [12,] -0.008250770  0.0001290196
## [13,] -0.015742846 -0.1185219445
```

```r
#Check for A.V = Lamda.V for first 2 values
A_V1 = S %*% eigenvector[,1]
A_V2 = S %*% eigenvector[,2]
L1_V1= as.matrix(eigenvalue[1]*eigenvector[,1])
L2_V2= as.matrix(eigenvalue[2]*eigenvector[,2])
A_V1 == L1_V1
```

```
##        [,1]
##  [1,] FALSE
##  [2,] FALSE
##  [3,] FALSE
##  [4,] FALSE
##  [5,] FALSE
##  [6,] FALSE
##  [7,] FALSE
##  [8,] FALSE
##  [9,] FALSE
## [10,] FALSE
## [11,] FALSE
## [12,] FALSE
## [13,] FALSE
```

```r
A_V2 == L2_V2
```

```
##        [,1]
##  [1,] FALSE
##  [2,] FALSE
##  [3,] FALSE
##  [4,] FALSE
```

```
##  [5,] FALSE
##  [6,]  TRUE
##  [7,]  TRUE
##  [8,] FALSE
##  [9,] FALSE
## [10,] FALSE
## [11,] FALSE
## [12,] FALSE
## [13,] FALSE
```

1(a)iii

```r
#Two vector are orthogonal if product of the vector is zero.
t(eigenvector[,1]) %*% eigenvector[,1]
```

```
##      [,1]
## [1,]    1
```

```r
t(eigenvector[,1]) %*% eigenvector[,2]
```

```
##              [,1]
## [1,] 4.553649e-17
```

```r
t(eigenvector[,2]) %*% eigenvector[,1]
```

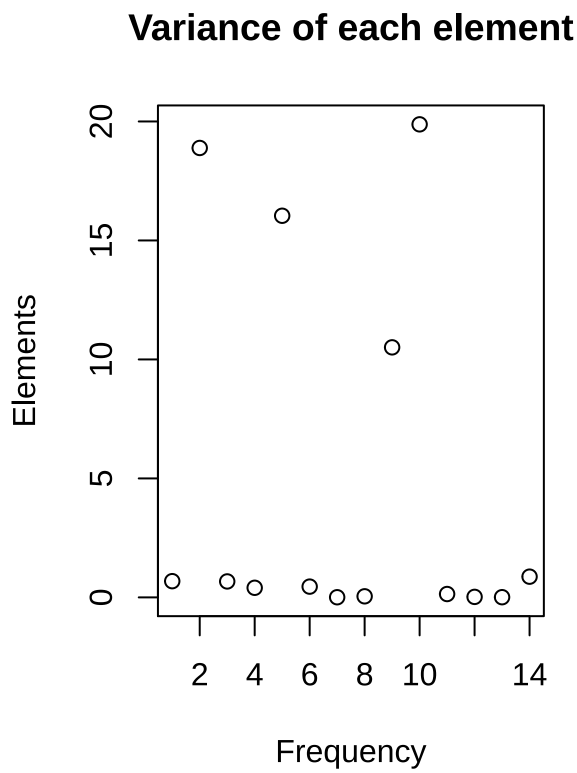
```
##              [,1]
## [1,] 4.553649e-17
```

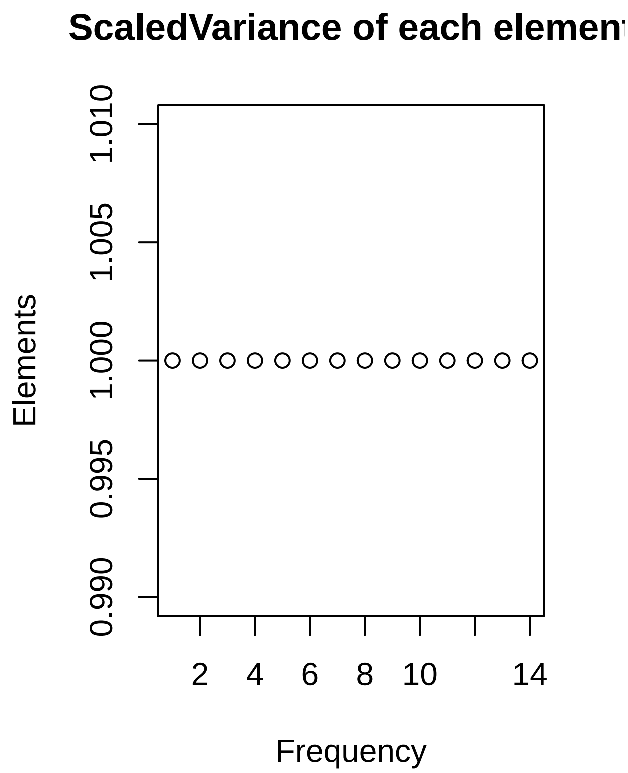
```r
t(eigenvector[,2]) %*% eigenvector[,2]
```

```
##      [,1]
## [1,]    1
```

1(a)iv

```r
#Variance of elements and Sumarised Plot
var_glass<-apply(glass,2,var)
scaled_glass<-scale(glass)
scaled_var<-apply(scaled_glass,2,var)
par( mfrow = c(1,2) )
plot(var_glass,main = "Variance of each element",xlab = "Frequency",ylab = "Elements")
plot(scaled_var,main ="ScaledVariance of each element",xlab = "Frequency",ylab = "Elements")
```





Analysis : From the above graph we can see that values of elements are higher compaired to other elements, which may influence other elements, hence we standardise all the elements before we procude the analysis

1(b)i

```
#Assigning Values to variables
ex1=5
varx1=6
ex2=8
varx2=7
cov_x1x2=2.5

#Expected value of x1-x2
exp_val<-ex2-ex1
exp_val
```

```
## [1] 3
```

```
#variance of x1 and x2
var_x1x2<-varx1+varx2-2*cov_x1x2
var_x1x2
```

```
## [1] 8
```

1(b)ii

```
#Calculate the Variance of U and V

Var_U = ((-1*-1)*varx1)+((1*1)*varx2)-(2*cov_x1x2)
Var_V = ((-2*-2)*varx1)+((1*1)*varx2)-(4*cov_x1x2)
Var_U
```

```
## [1] 8
```

```
Var_V
```

```
## [1] 21
```

```
# Calculating the Correlation by dividing the Covariance with the square root of variance product
cov_UV1<-varx2+2*varx1+(-2-1)*cov_x1x2
cor_UV<-cov_UV1/sqrt(Var_U*Var_V)
cor_UV
```

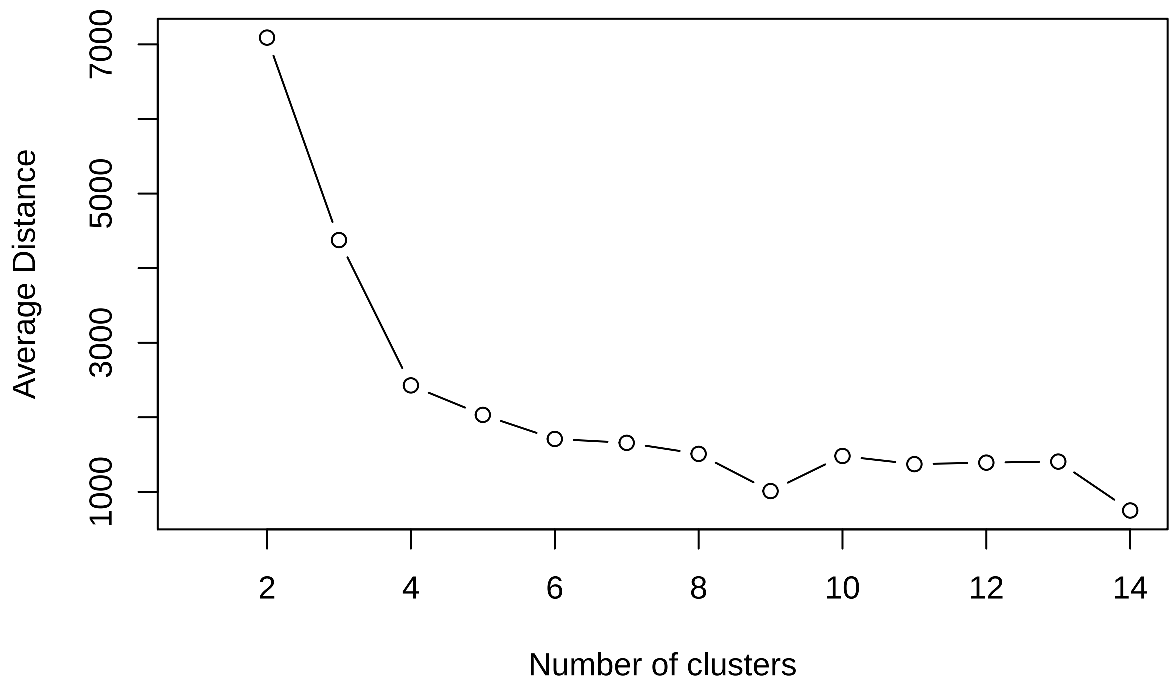
```
## [1] 0.8872443
```

Question 2

2(a)

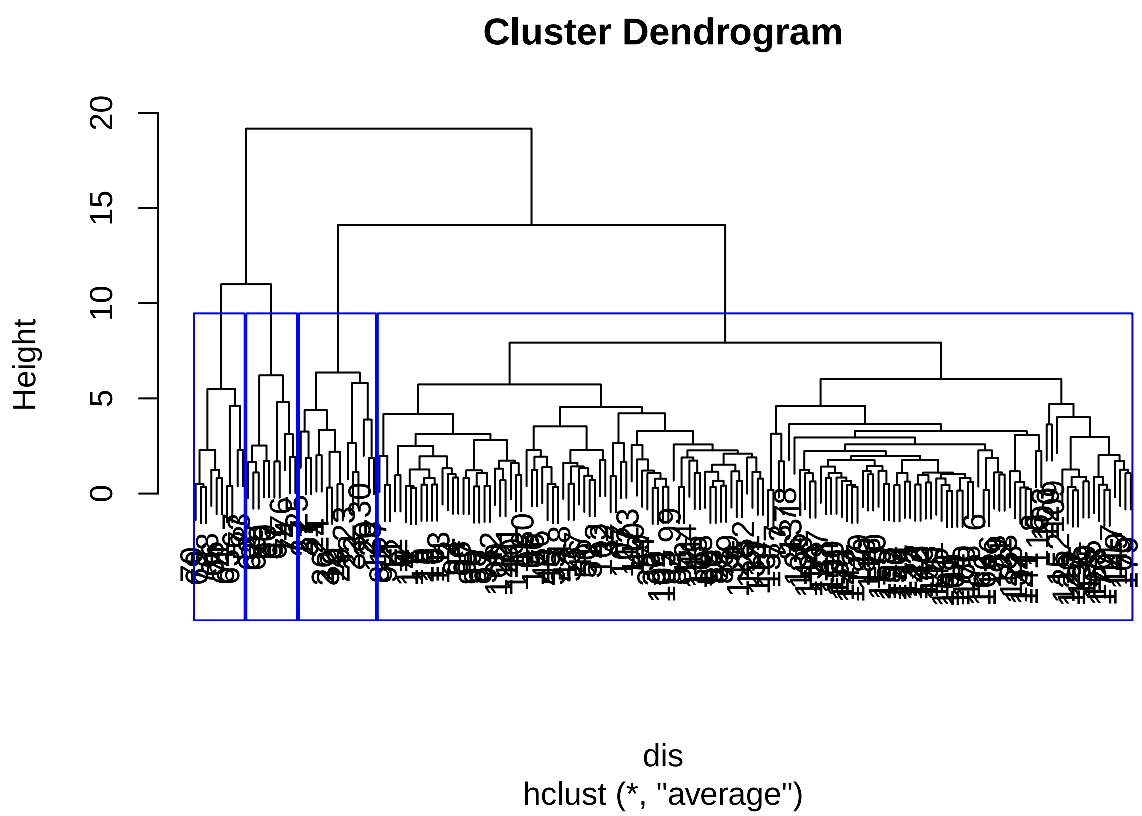
here we are using K=4 clusters, we are using 4 clusters as we observe through the elbow plot there is a sharp dip in the sum of square with the clusters at K=4

```
# set k to maximum
K = 14
# Applying Kmeans Cluster
fitk = kmeans(glass,4)

# Assign a dummy vector to wss
wss = c()

# Loop clusters for the dummy variable created and within cluster Sum of Square to the dummy variable

for(K in 2:K)
  wss[K] = sum (kmeans(glass,centers = K)$withinss)

  #Plot of SS for clustors
  plot(1:K,wss,type = "b" , xlab = "Number of clusters" , ylab = "Average Distance")
```



2(b)

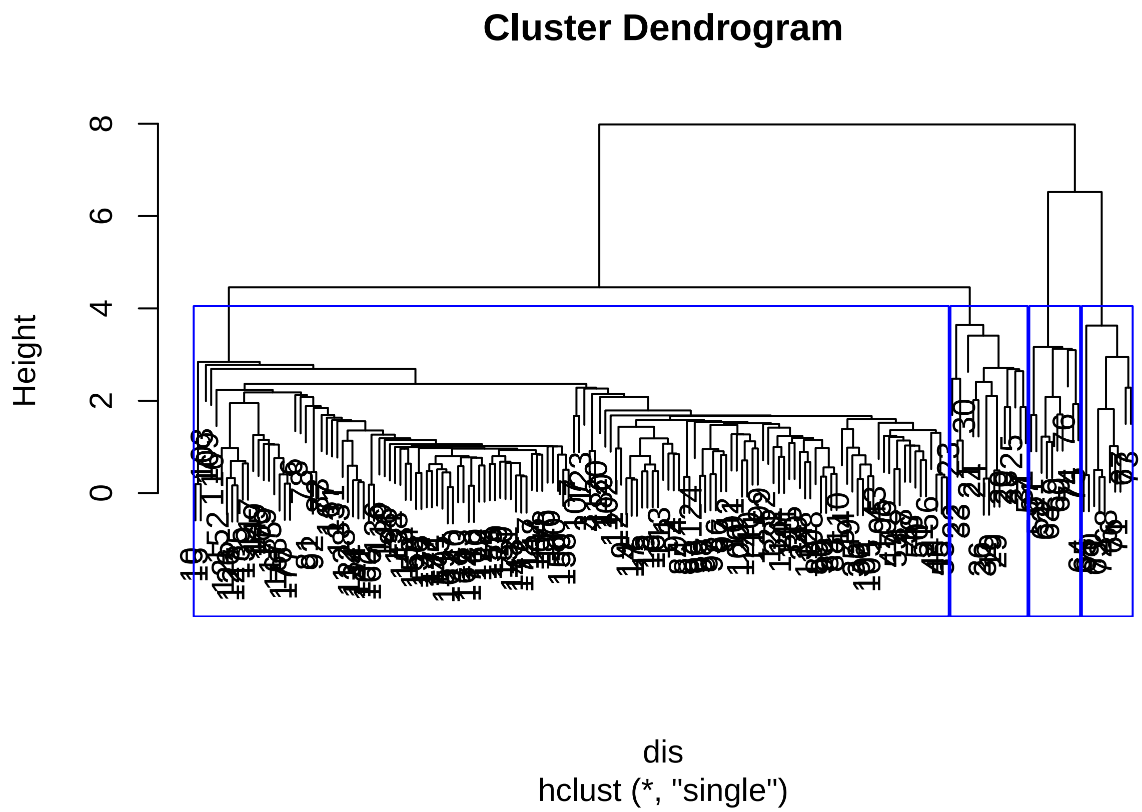
```
#Euclidean Distance between points
dis = dist(glass, method = "euclidean")
khist = 4

# Algorithm hierarchical clustering using average method
fitavg = hclust(dis,method = "average")

#Dendrogram
plot(fitavg)

#Cut the tree into 2 Clustor
groupsavg = cutree(fitavg,khist)

#Indicating the 2 clustors by drawint the border
rect.hclust(fitavg,khist,border = "blue")
```




```r
table(groupsavg)
```

```
## groupsavg
##   1   2   3   4
## 144  15  10  10
```

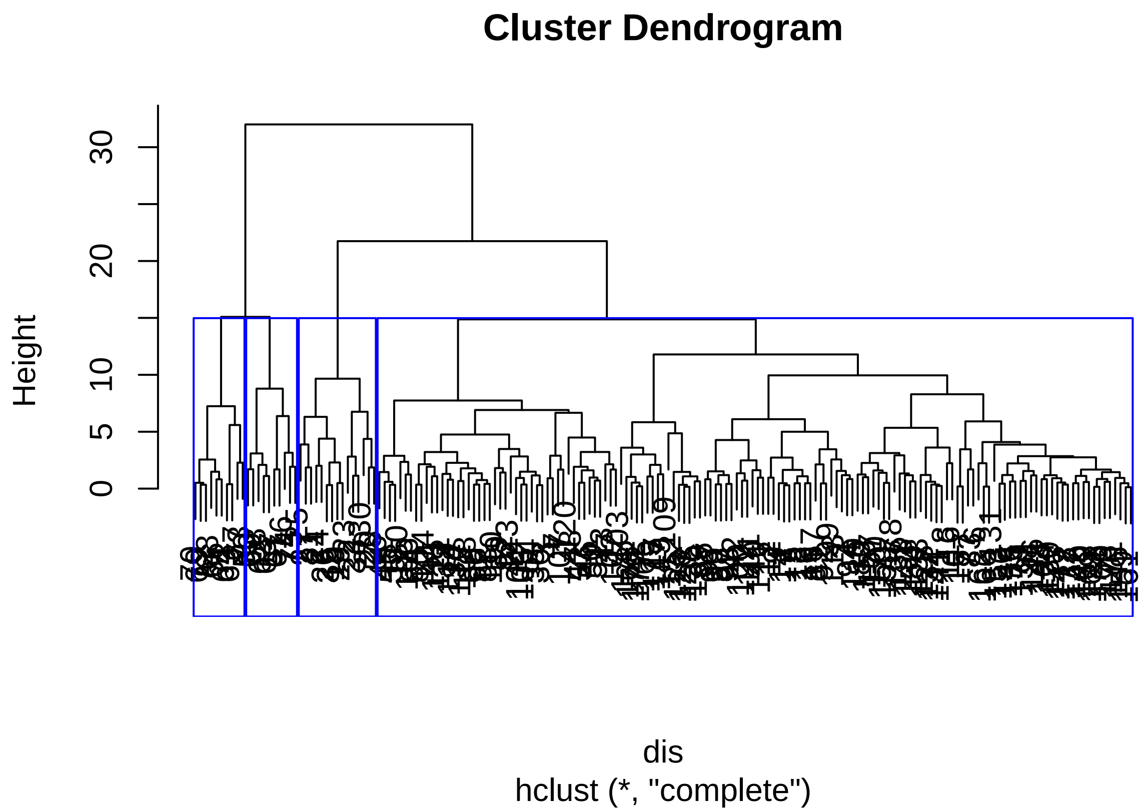
```r
# Algorithm hierarchical clustering using single method
fitsin = hclust(dis,method = "single")

#Dendrogram
plot(fitsin)

#Cut the tree into 2 Clustor
groupssin = cutree(fitsin,khist)

#Indicating the 2 clustors by drawint the border
rect.hclust(fitsin,khist,border = "blue")
```




```r
table(groupssin)
```

```
## groupssin
##   1   2   3   4
## 144  15  10  10
```

```r
# Algorithm hierarchical clustering using Complete method
fitcom = hclust(dis,method = "complete")

#Dendrogram
plot(fitcom)

#Cut the tree into 2 Clustor
groupscom = cutree(fitcom,khist)

#Indicating the 2 clustors by drawint the border
rect.hclust(fitcom,khist,border = "blue")
```




```r
table(groupscom)
```

```
## groupscom
##   1   2   3   4
## 144  15  10  10
```

Analysis : We can see from the above 3 methods, once we cut the three all the three methods give the same result. As average method suits well we go ahead with this method

2(c)

```r
#Cross Tabulation
library(e1071)
cross_tab = table(groupsavg,fitk$cluster)
classAgreement(cross_tab)
```

```
## $diag
## [1] 0.4022346
##
## $kappa
## [1] 0.06584402
##
## $rand
```

```
## [1] 0.6629213
##
## $crand
## [1] 0.3893869
```

```r
#rand value for different K values
#K=2
groupsavg = cutree(fitavg, 2)
tab_cross = table(groupsavg)
fitk2 = kmeans(glass,2)
table(groupsavg,fitk2$cluster)
```

```
##
## groupsavg   1   2
##         1   0 159
##         2  20   0
```

```r
classAgreement(table(groupsavg,fitk2$cluster))
```

```
## $diag
## [1] 0
##
## $kappa
## [1] -0.2476539
##
## $rand
## [1] 1
##
## $crand
## [1] 1
```

```r
#K=3
groupsavg = cutree(fitavg, 3)
tab_cross = table(groupsavg)
fitk3 = kmeans(glass,3)
table(groupsavg,fitk3$cluster)
```

```
##
## groupsavg  1  2  3
##         1 71 73  0
##         2 15  0  0
##         3  0  0 20
```

```r
classAgreement(table(groupsavg,fitk3$cluster))
```

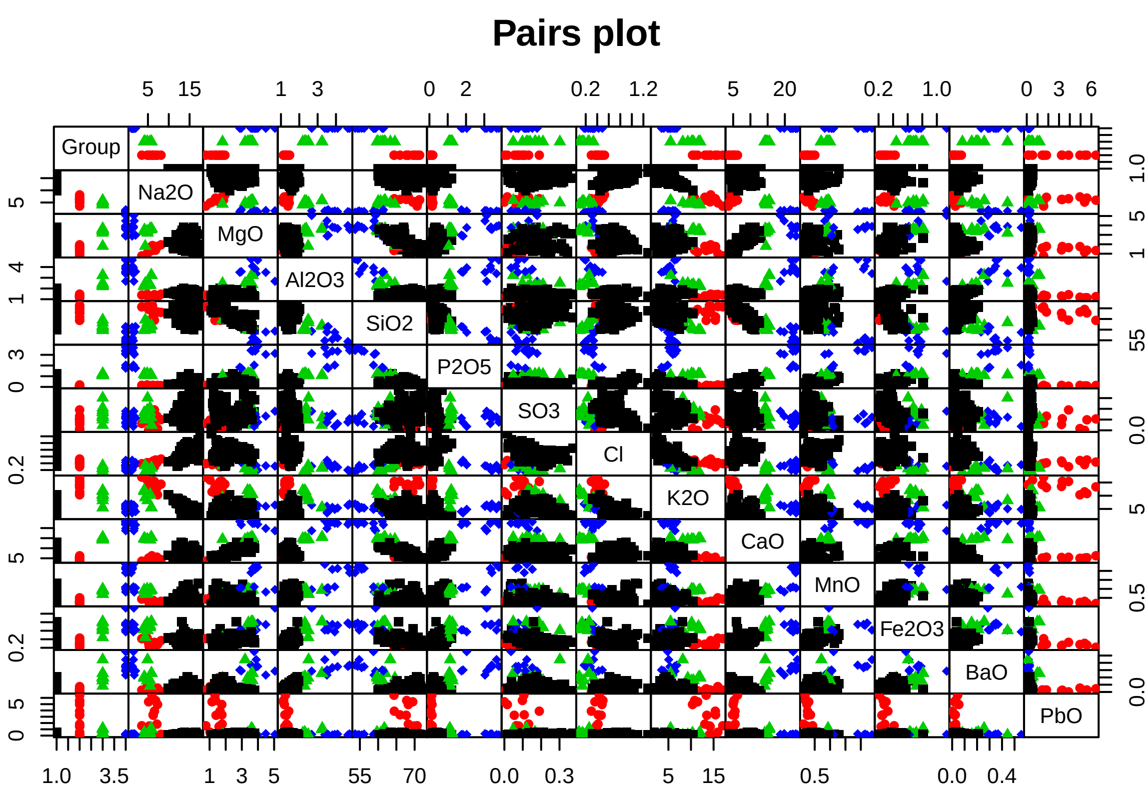
```
## $diag
## [1] 0.5083799
##
## $kappa
## [1] 0.1326946
##
```

```
## $rand
## [1] 0.6078087
##
## $crand
## [1] 0.2612423
```

If the rand value is close to 1 the data is clustered correctly Analysis : We can see that the value of rand is greater for K=2 when compared to K=3, so we can conclude K=2 clustering holds good

2(d)

```
#Pairs plot on type of vessle
#Symbols for different vessels group
symbol = c(15,16,17,18)
pairs(glass,gap =0,col = glass$Group,main ="Pairs plot",pch=symbol[glass$Group])
```



Analysis : We can see that dataset consists a large of group 1 data in the overall data, so the distribution can be a partial distribution because of that. From the distrubution of the Pb0 variable and the groups we can see that the concentration of group2 is maximum when compared to the other vessel types

Question 3

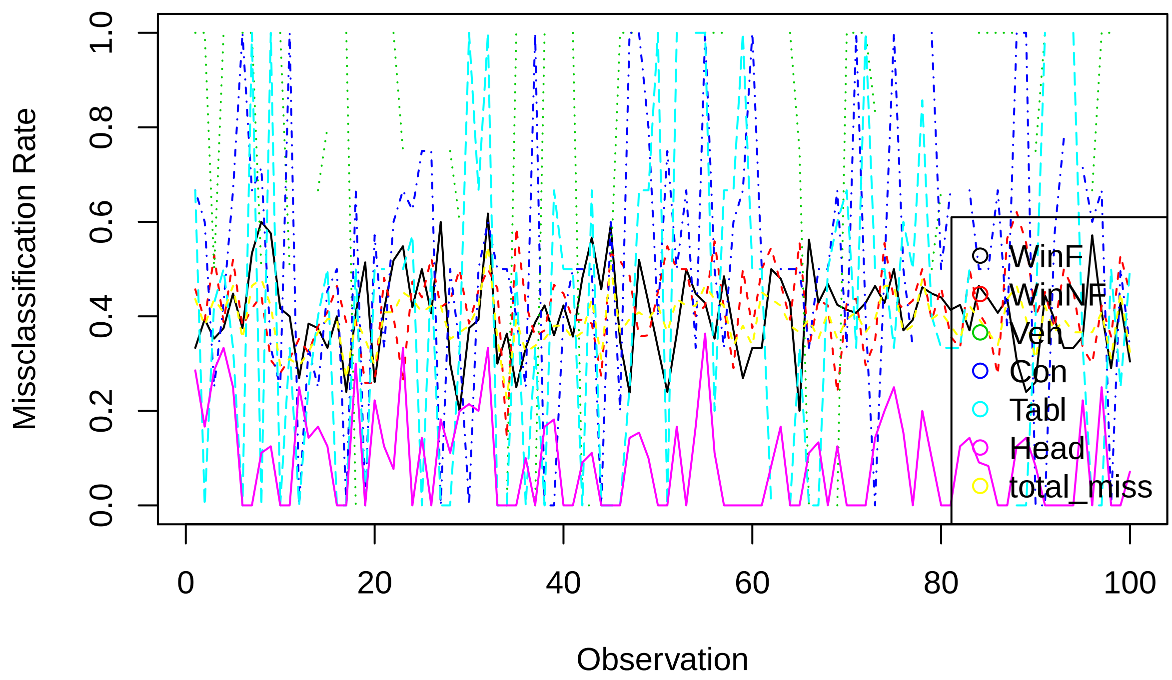
```
#Load the library
library(MASS)

#Load the dataset.
fgl_data<-fgl
nrow(fgl_data)
```

```
## [1] 214
```

```
#Delete a row from the dataset by randomly generating a integer between 0 and n
fgl_data<-fgl[-floor(runif(1,min=0,max = nrow(fgl))),]
result <- data.frame(matrix(ncol = 7, nrow = 0))
colnames(result) <- c("WinF","WinNF","Veh","Con","Tabl","Head","total_miss")

# For loop for creating 100 iterations
for(i in 1:100)
  {
# Set sample size as per provided formula
Sample_Size <- floor(nrow(fgl_data)*(2/3))

#sample the data
train_samp <- sample(seq_len(nrow(fgl_data)), size = Sample_Size)

#split the sample data into test and training sets
train_set <- fgl_data[train_samp, ]
test_set <- fgl_data[-train_samp, ]

# LDA fucntion to calculate data mean and prior values
lda_fgl<-lda(type~.,data = train_set)

#Select the prior value for each group from the output
prior<-lda_fgl$prior

#mean value for each group from the output
means <-lda_fgl$means

# number of rows in the training data
N <- nrow(train_set)

#number of groups in the data
G <- length(levels(train_set$type))

#subset of the data based on the type variable values.
fgl_data.WinF <- subset(train_set,type== "WinF")
fgl_data.WinNF <- subset(train_set,type== "WinNF")
fgl_data.Veh <- subset(train_set,type== "Veh")
fgl_data.Con <- subset(train_set,type== "Con")
fgl_data.Tabl <- subset(train_set,type== "Tabl")
fgl_data.Head <- subset(train_set,type== "Head")
```

```
#covariance of each dataset
cov_WinF <-cov(fgl_data.WinF[1:9])
cov_WinNF <-cov(fgl_data.WinNF[1:9])
cov_Veh <-cov(fgl_data.Veh[1:9])
cov_Con <-cov(fgl_data.Con[1:9])
cov_Tabl <-cov(fgl_data.Tabl[1:9])
cov_Head <-cov(fgl_data.Head[1:9])

# total variance
cov_total<-((cov_WinF*(nrow(fgl_data.WinF)-1))+
(cov_WinNF*(nrow(fgl_data.WinNF)-1))+(cov_Veh*(nrow(fgl_data.Veh)-1))+
(cov_Con*(nrow(fgl_data.Con)-1))+
(cov_Tabl*(nrow(fgl_data.Tabl)-1))+(cov_Head*(nrow(fgl_data.Head)-1)))/(N - G)

#linear discriminant function
ldf <- function(ti, priori, mu, covar1)
{

  #Checks if observation is in correct format
ti <- matrix(as.numeric(ti),ncol=1)
log(priori)-(0.5*t(mu) %*% solve(covar1) %*% mu) + (t(ti) %*% (solve(covar1) %*% mu))
}

# initialize the vectors for holding the values
dfs <-rep(0,G)
test_grp<-rep(0,1)

# Iterative loop for test data
for(v in 1:nrow(test_set))
  {
  # For loop for different groups in the data
for(g in 1:G)
  {
  #Call the ldf function and store the output
dfs[g] <- ldf(t(as.matrix(test_set[v,1:9])), prior[g], means[g,], cov_total)
}

  #Store the value for each observation class data
test_grp[v]<-levels(test_set$type)[dfs == max(dfs)]
}

#Order the data for correct tabulation of results
test_grp1<-ordered(test_grp,levels=c("WinF","WinNF","Veh","Con","Tabl","Head"))

#Tabulate the results for test data and test LDA values
tab_class<- table(test_set$type,test_grp1)

# number of instances
n = sum(tab_class)

# number of classes
nc = nrow(tab_class)
```

```r
# number of correctly classified instances per class
diag = diag(tab_class)

# number of instances per class
rowsums = apply(tab_class, 1, sum)

# number of predictions per class
colsums = apply(tab_class, 2, sum)

# distribution of instances over the actual classes
p = rowsums / n

# distribution of instances over the predicted classes
q = colsums / n

# calculate the values for missclassification per class
miss_class <- (1- diag / colsums)

# Calculate the values for total missclassification
total_miss <- 1-sum(diag(tab_class))/sum(tab_class)

# bind the miss_class and total_miss values
miss_class<-cbind(t(miss_class),total_miss)

# Attach the values to the result dataframe
result<-rbind(result,miss_class)
}

# Plot the missclassification results of all the class and total missclassification rates
matplot(rownames(result), result, type='l', xlab='Observation',ylab='Missclassification Rate', col=1:7)
legend('bottomright', legend=colnames(result),pch=1, col=1:7)
```




```r
# average overall missclassification rates
class_error_avg<- mean(result$total_miss)
print("Average overall missclassification rate :- ")
```

```
## [1] "Average overall missclassification rate :- "
```

```r
cat(class_error_avg)
```

```
## 0.3901408
```

Question 4

Question 4

We know that Baye's decision boundary between the classes k & l will be reached by using the following condition

$\Rightarrow \{x : \delta_k(x) = \delta_l(x)\}$.

The Linear discriminant function for some class k is given by

$$\Rightarrow \delta_k(x) = \log \pi_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + x^T \Sigma^{-1} \mu_k .$$

Here $\delta_k(x) = \delta_l(x)$.

$$\Rightarrow \log \pi_k + x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k = \log \pi_l + x^T \Sigma^{-1} \mu_l - \frac{1}{2}\mu_l^T \Sigma^{-1} \mu_l .$$

Consider, $\pi_k = \pi_l = \frac{1}{2}$ and $\rho = 1$.

$$\Rightarrow \log\frac{1}{2} + x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k = \log\frac{1}{2} + x^T \Sigma^{-1} \mu_l - \frac{1}{2}\mu_l^T \Sigma^{-1} \mu_l .$$

$$\Rightarrow x^T \Sigma^{-1} (\mu_k - \mu_l) = \frac{1}{2}\Sigma^{-1}(\mu_k^{2} - \mu_l^{2})$$

$$\Rightarrow x^T \cancel{\Sigma^{-1}} (\mu_k - \cancel{\mu_l}) = \frac{1}{2}\cancel{\Sigma^{-1}}(\mu_k + \mu_l)^T (\mu_k \cancel{-} \mu_l)$$

Thus $\Rightarrow x^T = \frac{1}{2}(\mu_k + \mu_l)^T \longrightarrow (1)$

we have $\rho = 1$; $x = x^T$; $(\mu_k + \mu_l)^T = \mu_k + \mu_l$.

By substituting eq (1) we have

$$x = \frac{\mu_k + \mu_l}{2}$$

or

$$x = \frac{\mu_1 + \mu_2}{2} .$$

Figure 1: Question 4