STAT40150 Assignment 1

Azam Jainullabudin Mohamed & 18201501

23 February 2019

```
# setting seed to roll number
set.seed(18201501)

# Importing the dataset
pottery <- read.csv(file="C:/Users/Shanila/OneDrive/Documents/R/Multivariate/PotteryDa
ta(1).csv")
head(pottery)
```

```
##   Al2O3 Fe2O3  MgO  CaO Na2O  K2O TiO2   MnO   BaO kiln
## 1  18.8  9.52 2.00 0.79 0.40 3.20 1.01 0.077 0.015    1
## 2  16.9  7.33 1.65 0.84 0.40 3.05 0.99 0.067 0.018    1
## 3  18.2  7.64 1.82 0.77 0.40 3.07 0.98 0.087 0.014    1
## 4  16.9  7.29 1.56 0.76 0.40 3.05 1.00 0.063 0.019    1
## 5  17.8  7.24 1.83 0.92 0.43 3.12 0.93 0.061 0.019    1
## 6  18.8  7.45 2.06 0.87 0.25 3.26 0.98 0.072 0.017    1
```

```
str(pottery)
```

```
## 'data.frame':    45 obs. of  10 variables:
##  $ Al2O3: num  18.8 16.9 18.2 16.9 17.8 18.8 16.5 18 15.8 14.6 ...
##  $ Fe2O3: num  9.52 7.33 7.64 7.29 7.24 7.45 7.05 7.42 7.15 6.87 ...
##  $ MgO  : num  2 1.65 1.82 1.56 1.83 2.06 1.81 2.06 1.62 1.67 ...
##  $ CaO  : num  0.79 0.84 0.77 0.76 0.92 0.87 1.73 1 0.71 0.76 ...
##  $ Na2O : num  0.4 0.4 0.4 0.4 0.43 0.25 0.33 0.28 0.38 0.33 ...
##  $ K2O  : num  3.2 3.05 3.07 3.05 3.12 3.26 3.2 3.37 3.25 3.06 ...
##  $ TiO2 : num  1.01 0.99 0.98 1 0.93 0.98 0.95 0.96 0.93 0.91 ...
##  $ MnO  : num  0.077 0.067 0.087 0.063 0.061 0.072 0.066 0.072 0.062 0.055 ...
##  $ BaO  : num  0.015 0.018 0.014 0.019 0.019 0.017 0.019 0.017 0.017 0.012 ...
##  $ kiln : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
# Generating a Random number
random_sample = sample(1:45, 1)
# random_sample

# Removing a random row, removing row 10
pot = pottery[-random_sample,]
# str(pot)

# Remove the last column "kiln"
potter = pot[,-ncol(pot)]
# str(potter)

# >>>>>>>>> 1a. i. COMPUTING THE COVARIANCE MATRIX <<<<<<<<<<
# Calculating the covariance with Data as data frame
covariance_mat = cov(potter)
covariance_mat
```

```
##              Al2O3        Fe2O3           MgO           CaO          Na2O
## Al2O3  7.054862579 -0.867522199 -3.2137991543  0.2557367865 -0.0022568710
## Fe2O3 -0.867522199  5.913571459  1.6404097252  0.7462195560  0.2971320296
## MgO   -3.213799154  1.640409725  2.8692966173 -0.1239878436  0.0552309725
## CaO    0.255736786  0.746219556 -0.1239878436  0.2081950846  0.0418961416
## Na2O  -0.002256871  0.297132030  0.0552309725  0.0418961416  0.0322580867
## K2O   -1.372624736  1.282536469  1.2769791755  0.0281069239  0.0520234144
## TiO2   0.330995772 -0.061830021 -0.2061551797  0.0122101480  0.0009346723
## MnO   -0.071733192  0.076961078  0.0641464905  0.0032151268  0.0045953171
## BaO    0.001841015  0.001660507  0.0002096617  0.0002823784  0.0001805391
##                K2O          TiO2           MnO          BaO
## Al2O3 -1.3726247357  3.309958e-01 -7.173319e-02 1.841015e-03
## Fe2O3  1.2825364693 -6.183002e-02  7.696108e-02 1.660507e-03
## MgO    1.2769791755 -2.061552e-01  6.414649e-02 2.096617e-04
## CaO    0.0281069239  1.221015e-02  3.215127e-03 2.823784e-04
## Na2O   0.0520234144  9.346723e-04  4.595317e-03 1.805391e-04
## K2O    0.7327274313 -9.172262e-02  3.445940e-02 3.048943e-04
## TiO2  -0.0917226216  3.225497e-02 -4.606871e-03 9.657505e-05
## MnO    0.0344593975 -4.606871e-03  2.234818e-03 2.864905e-05
## BaO    0.0003048943  9.657505e-05  2.864905e-05 7.756871e-06
```

```
# >>>>>>>>> 1a. ii. CALCULATING THE FIRST TWO EIGEN VALUE AND EIGEN <<<<<<<
# >>>>>>>>> VECTORS OF COVARIANCE MATRIX <<<<<<<
# Calculating only the eigen values
eigen_values = eigen(covariance_mat, only.values = TRUE)
# eigen_values

# Calculating both eigen value and eigen vectors
eigen = eigen(covariance_mat)
eigen
```

```
## eigen() decomposition
## $values
## [1] 1.003451e+01 5.614466e+00 1.011876e+00 8.936741e-02 6.427223e-02
## [6] 1.762003e-02 1.295777e-02 3.289054e-04 5.254004e-06
##
## $vectors
##                [,1]          [,2]          [,3]          [,4]         [,5]
##  [1,]  7.400875e-01  0.4893328426 -0.4601574924 -0.0004533019 -0.021381008
##  [2,] -4.185688e-01  0.8545398410  0.2287912156  0.0727952926  0.181541027
##  [3,] -4.703614e-01 -0.0320910794 -0.7889986697  0.3822997314 -0.087091667
##  [4,] -7.279717e-03  0.1426580920  0.1791334484  0.1452418558 -0.957436667
##  [5,] -1.643787e-02  0.0467901531  0.0185697458  0.0271023678  0.009494384
##  [6,] -2.320400e-01  0.0795121651 -0.2832870613 -0.9087051442 -0.171137337
##  [7,]  3.889614e-02  0.0197436110  0.0249384222  0.0301273575  0.113493112
##  [8,] -1.233503e-02  0.0056891895 -0.0091224946 -0.0080637059  0.007477402
##  [9,]  4.947834e-05  0.0004253579 -0.0006551836 -0.0003164522 -0.001024231
##               [,6]         [,7]         [,8]          [,9]
##  [1,]  0.010768610 -0.022408961 -0.001782549 -0.0005266146
##  [2,]  0.062328348 -0.006957841  0.005156420  0.0005550737
##  [3,] -0.012067601  0.035548054  0.007996962 -0.0003293191
##  [4,] -0.031940218  0.092524986 -0.011968176 -0.0015577040
##  [5,] -0.926381599 -0.363408726  0.078198008 -0.0026950782
##  [6,] -0.044248617  0.049437740  0.012175815 -0.0006385931
##  [7,] -0.361511226  0.923196004 -0.026247845 -0.0041029449
##  [8,] -0.062978002 -0.053123744 -0.995944852 -0.0302789751
##  [9,] -0.006000585  0.001379992 -0.030079375  0.9995276674
```

```
eigen_values=eigen$values
eigen_vectors=eigen$vectors
eigen_vectors[,1]
```

```
## [1]  7.400875e-01 -4.185688e-01 -4.703614e-01 -7.279717e-03 -1.643787e-02
## [6] -2.320400e-01  3.889614e-02 -1.233503e-02  4.947834e-05
```

```
eigenvalue_vector = eigen(covariance_mat)
eigenvalue_vector
```

```
## eigen() decomposition
## $values
## [1] 1.003451e+01 5.614466e+00 1.011876e+00 8.936741e-02 6.427223e-02
## [6] 1.762003e-02 1.295777e-02 3.289054e-04 5.254004e-06
##
## $vectors
##                [,1]          [,2]          [,3]          [,4]         [,5]
##  [1,]  7.400875e-01  0.4893328426 -0.4601574924 -0.0004533019 -0.021381008
##  [2,] -4.185688e-01  0.8545398410  0.2287912156  0.0727952926  0.181541027
##  [3,] -4.703614e-01 -0.0320910794 -0.7889986697  0.3822997314 -0.087091667
##  [4,] -7.279717e-03  0.1426580920  0.1791334484  0.1452418558 -0.957436667
##  [5,] -1.643787e-02  0.0467901531  0.0185697458  0.0271023678  0.009494384
##  [6,] -2.320400e-01  0.0795121651 -0.2832870613 -0.9087051442 -0.171137337
##  [7,]  3.889614e-02  0.0197436110  0.0249384222  0.0301273575  0.113493112
##  [8,] -1.233503e-02  0.0056891895 -0.0091224946 -0.0080637059  0.007477402
##  [9,]  4.947834e-05  0.0004253579 -0.0006551836 -0.0003164522 -0.001024231
##               [,6]         [,7]         [,8]          [,9]
##  [1,]  0.010768610 -0.022408961 -0.001782549 -0.0005266146
##  [2,]  0.062328348 -0.006957841  0.005156420  0.0005550737
##  [3,] -0.012067601  0.035548054  0.007996962 -0.0003293191
##  [4,] -0.031940218  0.092524986 -0.011968176 -0.0015577040
##  [5,] -0.926381599 -0.363408726  0.078198008 -0.0026950782
##  [6,] -0.044248617  0.049437740  0.012175815 -0.0006385931
##  [7,] -0.361511226  0.923196004 -0.026247845 -0.0041029449
##  [8,] -0.062978002 -0.053123744 -0.995944852 -0.0302789751
##  [9,] -0.006000585  0.001379992 -0.030079375  0.9995276674
```

```
# Extracting the first two eigen vectors
eigenvalue_vector[[2]][,c(1,2)]
```

```
##                [,1]          [,2]
##  [1,]  7.400875e-01  0.4893328426
##  [2,] -4.185688e-01  0.8545398410
##  [3,] -4.703614e-01 -0.0320910794
##  [4,] -7.279717e-03  0.1426580920
##  [5,] -1.643787e-02  0.0467901531
##  [6,] -2.320400e-01  0.0795121651
##  [7,]  3.889614e-02  0.0197436110
##  [8,] -1.233503e-02  0.0056891895
##  [9,]  4.947834e-05  0.0004253579
```

```r
first_eigenvalue = as.matrix(eigenvalue_vector[[1]][1])
second_eigenvalue = as.matrix(eigenvalue_vector[[1]][2])
first_eigenvector <- as.matrix(eigen_vectors[,1])
second_eigenvector <- as.matrix(eigen_vectors[,2])
round(t(covariance_mat %*% first_eigenvector)) == round(first_eigenvalue %*% t(first_e
igenvector))
```

```
##      Al2O3 Fe2O3  MgO  CaO Na2O  K2O TiO2  MnO  BaO
## [1,]  TRUE  TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```r
round(t(covariance_mat %*% second_eigenvector)) == round(second_eigenvalue %*% t(secon
d_eigenvector))
```

```
##      Al2O3 Fe2O3  MgO  CaO Na2O  K2O TiO2  MnO  BaO
## [1,]  TRUE  TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```r
# 1 (a). (iii)
# Proving Orthonormality
# Conditions for Orthonoramality - Orthogonality and Magnitude of the value to be uni
t vector
first_eigenvector <- as.matrix(eigenvalue_vector[[2]][1,])
second_eigenvector <- as.matrix(eigenvalue_vector[[2]][2,])

# Orthogonality check (transpose(x) * x == 0)
orthogonality = t(first_eigenvector) %*% second_eigenvector
round(orthogonality)
```

```
##      [,1]
## [1,]    0
```

```r
# Calculating the magnitude value
# First eigenvector
mag_eigen_vector_1 = sum((eigenvalue_vector[[2]][,1])^2)
mag_eigen_vector_1
```

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

```r
# Second eigenvector
mag_eigen_vector_2 = sum((eigenvalue_vector[[2]][,2])^2)
mag_eigen_vector_2
```

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

The above matrix is Orthonormal as both eigen vectors satisfy the condition of Orthogonality and Magnitude of both vectors is found to be unit vector.

```
# 1 (a). (iv)
```

It is at times beneficial to scale the data before analysis. Scaling is done for mean 0 and SD 1 for each variable in data, the reason for scaling is to reduce the variable with large variance and to make all the explanatory variables to contributing to the output. Since, all the variable in our dataset is not significantly contribute to the output, I would advise, it is necessary to standardize our dataset

```
# Standardizing the data using scale function
scaling = scale(potter)
# scaling
```

Calculating the expected mean, variance and correlation

```
# >>>>>>>> 1b. CALCULATION OF EXPECTED VALUE AND COVARIANCE <<<<<<<<<<
expect_x1=10; var_x1=11
expect_x2=8; var_x2=14
covar_x1x2=2

# 1b (i)
# Expect value of x1 - x2
expect_x1_sub_x2 = expect_x1 - expect_x2
expect_x1_sub_x2
```

```
## [1] 2
```

```
# Variance of x1 - x2
var_x1_sub_x2 = var_x1 + var_x2 - 2*covar_x1x2
var_x1_sub_x2
```

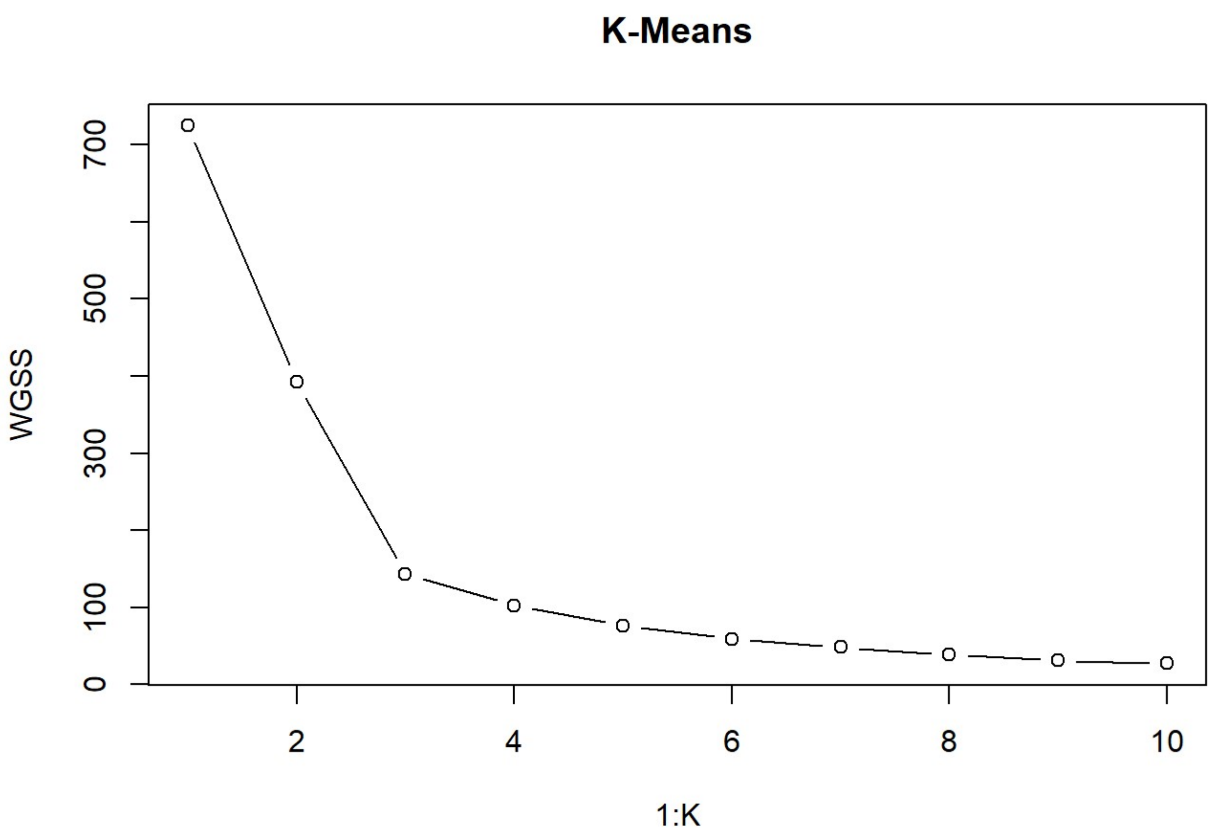
```
## [1] 21
```

```
# 1b (ii)
expect_x1_expect_x2 = expect_x1 *expect_x2 + covar_x1x2
covar_UV = var_x1 + expect_x1^2 - 3 * expect_x1_expect_x2 + 2 *(var_x2+expect_x2^2) +
12
var_x1_sub_x2 = var_x1 + var_x2 - 2*covar_x1x2
var_x1_sub_2x2 = var_x1 + 4*var_x2 - 4*covar_x1x2
corr_UV = covar_UV / (sqrt(var_x1_sub_x2)*sqrt(var_x1_sub_2x2))
corr_UV
```

```
## [1] 0.9375151
```

k-means clustering

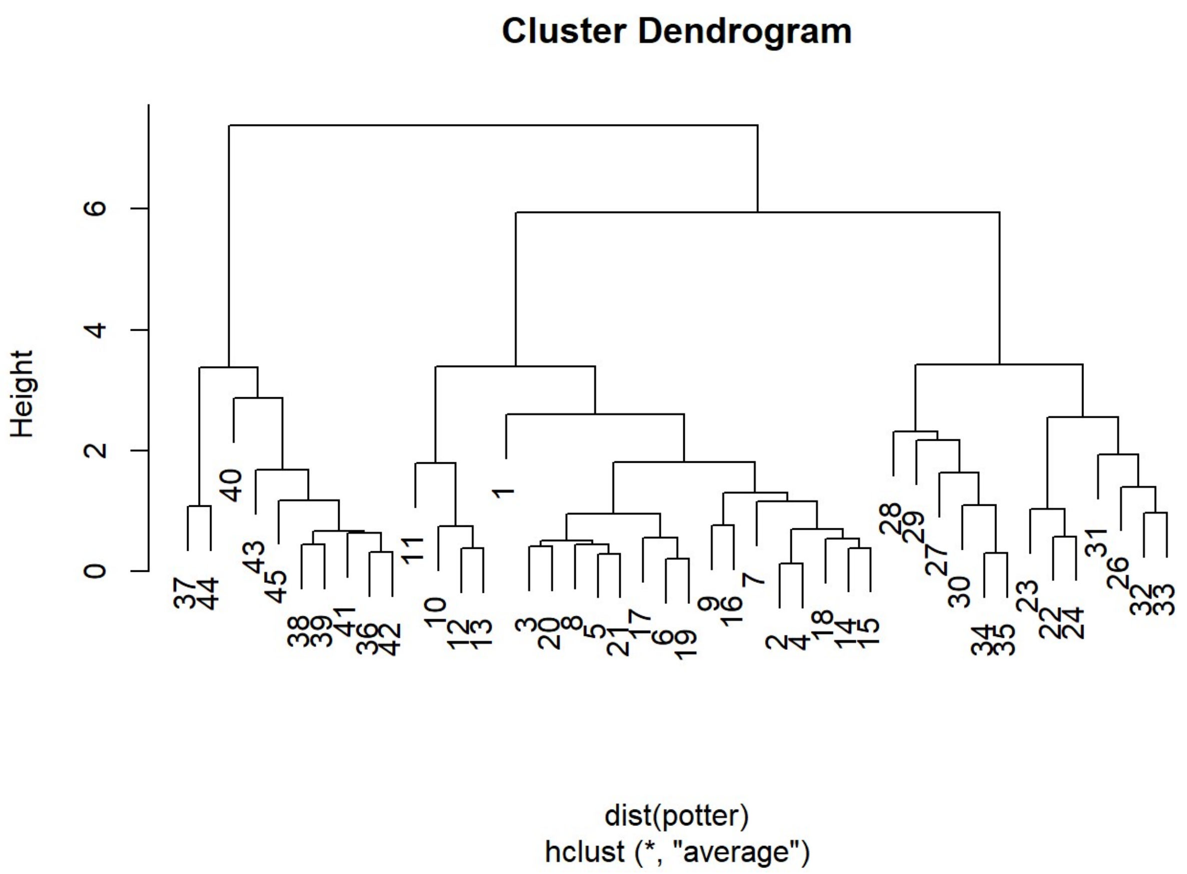
```
# >>>>>>>>>>>>> 2a. K-MEANS CLUSTERING <<<<<<<<<<<<<<<<<<<<<<
# Maximum range of K values
K=10
WGSS <- rep(NA, K)

# Calculating the Weighted sum of squares
for(k in 1:K) {
  WGSS[k] <- kmeans(potter, centers=k, nstart=1000)$tot.withinss
}

# Elbow plot to identify the best k-means clustering value
plot(1:K, WGSS, type='b', main="K-Means")
```



Based on my inference using k-means clustering, there were 3 cluster present in the pottery dataset. The reason behind this decision is plotting through the elbow plot. In which, the weighted sum of squares was plotted against the range of k values (1< k < 10). Made use of the kmeans function with arguments center value from 1 to 10 and nstart value of 1000 (with 1000 random samples). As seen in the above figure, there found a major drop in k value of 3. Hence, 3 cluster was chosen to be the ideal one.

Hierarchical clustering

```
# >>>>>>>>>> 2b. HIERARCHICAL CLUSTERING <<<<<<<<<<<<<<<<
hcluster = hclust(dist(potter), method = "average")
plot(hcluster)
```



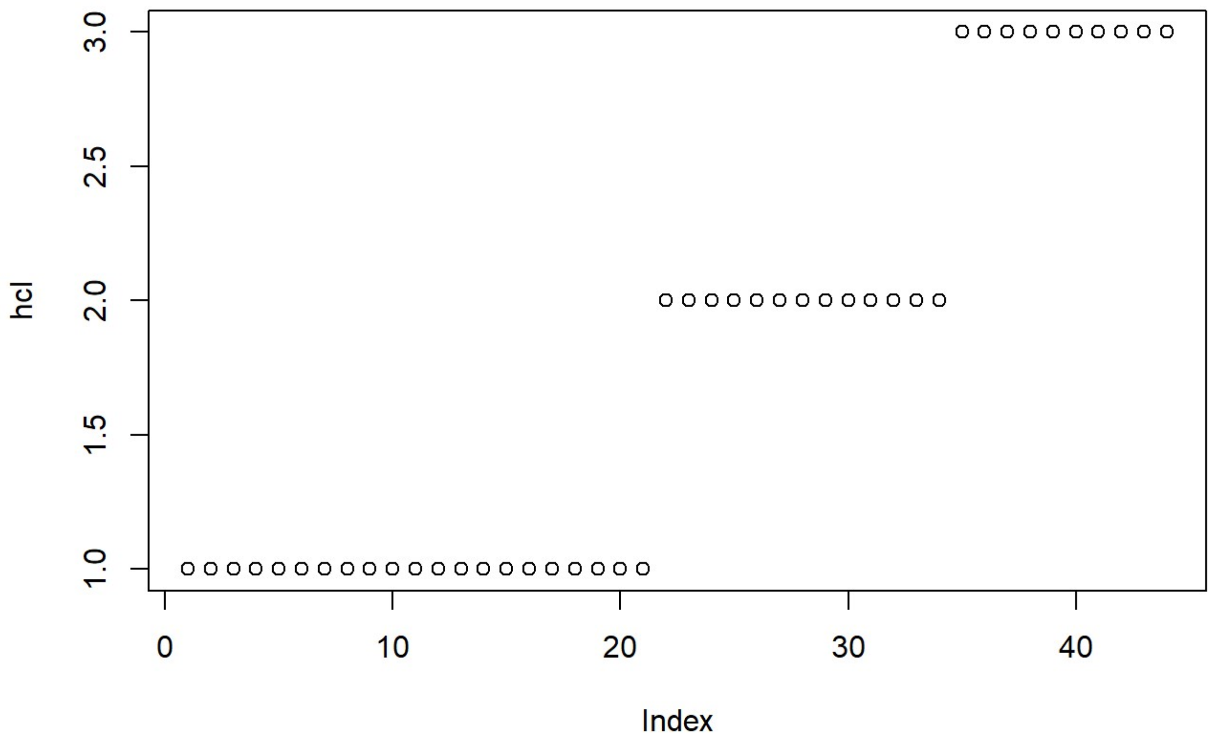
```
# Cutting dendogram
hcl = cutree(hcluster, k=3)
table(hcl)
```

```
## hcl
##  1  2  3
## 21 13 10
```

```
kcl = kmeans(potter, centers = 3)
table(hcl, potter[,1])
```

```
##
## hcl 10.1 10.9 11.1 11.6 11.8 12.4 13.1 13.4 13.7 13.8 14.4 14.6 14.8 15.8
##   1    0    0    0    0    0    0    0    0    1    0    0    2    1    2
##   2    1    1    1    2    1    1    1    1    0    2    1    1    0    0
##   3    0    0    0    0    0    0    0    0    0    0    0    0    1    1
##
## hcl 16.5 16.7 16.8 16.9 17.1 17.7 17.8 18 18.2 18.3 18.6 18.8 18.9 19.1
##   1    1    0    1    3    1    0    2  2    1    0    1    2    1    0
##   2    0    0    0    0    0    0    0  0    0    0    0    0    0    0
##   3    0    1    0    0    0    1    0  2    0    2    0    0    0    1
##
## hcl 20.8
##   1    0
##   2    0
##   3    1
```

```
plot(hcl)
```



Clustered the 45 pots using hierarchical clustering and average linking method using the function hclust with arguments dist(pot) and method=“average” for average linking. On plotting the dendrogram, it was evident of having 3 cluster. On cutting the dendrogram at desired number of cluster say k=3 as per our observation. After using cutree got a clear picture of number of pots in each cluster in our case it was 21 pots, 14 pots and 10 pots in cluster 1,2 and 3 respectively.

```
# >>>>>>>>>>>>>>>> 2c. cOMPARING THE CLUSTERING SOLUTION <<<<<<<<<<<<<
hcl = cutree(hcluster, k=3)
kcl = kmeans(potter, centers = 3, nstart = 10)
tab <- table(hcl, kcl$cluster)
tab
```

```
##
## hcl  1  2  3
##   1  0  0 21
##   2 13  0  0
##   3  0 10  0
```

```
# comparison using class agreement
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.5.2
```

```
n <- classAgreement(tab)

# To find the best mapping between the clusters.
matchClasses(tab, method="exact")
```

```
## Direct agreement: 3 of 3 pairs
## Cases in matched pairs: 100 %
```

```
## 1 2 3
## 3 1 2
```

```
library(mclust)
```

```
## Warning: package 'mclust' was built under R version 3.5.2
```

```
## Package 'mclust' version 5.4.2
## Type 'citation("mclust")' for citing this R package in publications.
```

```
# Clustering
mod1 <- Mclust(tab)
summary(mod1)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EII (spherical, equal volume) model with 2 components:
##
##  log.likelihood n df       BIC       ICL
##       -26.84952 3  8 -62.48793 -62.48793
##
## Clustering table:
## 1 2
## 1 2
```

The clustering solutions were compared to many ways namely, cross tabulation, class agreement, math classes and mclust. In our case, for 3 clusters all the methods revealed there were no misclassfication between k-means and hierarchical clustering. On comparing the clustering solution obtained using k-means clustering and hierarchical clustering with average linkage, it is found to be direct agreement between the clustering solution using classAgreement function. Moreover, the rand index and adjusted rand index returned value 1 that evidenced 100% agreement between the both clustering solutions and the same was revealed through the mclust function.

```
#  >>>>>>>>>>>>>>>> 2d <<<<<<<<<<<<<<<<<<<<<<<<<
# Clustering relationship after binding the kiln column back
# Repeated the same above procedure
# class agreement
tab <- table(hcl, kcl$cluster, pot$kiln)
tab
```

```
## , ,  = 1
##
##
## hcl  1  2  3
##   1  0  0 21
##   2  0  0  0
##   3  0  0  0
##
## , ,  = 2
##
##
## hcl  1  2  3
##   1  0  0  0
##   2 11  0  0
##   3  0  0  0
##
## , ,  = 3
##
##
## hcl  1  2  3
##   1  0  0  0
##   2  2  0  0
##   3  0  0  0
##
## , ,  = 4
##
##
## hcl  1  2  3
##   1  0  0  0
##   2  0  0  0
##   3  0  5  0
##
## , ,  = 5
##
##
## hcl  1  2  3
##   1  0  0  0
##   2  0  0  0
##   3  0  5  0
```

```
data = data.frame(hcl, kcl$cluster, pot$kiln)
# data

cor(data)
```

```
##                    hcl kcl.cluster   pot.kiln
## hcl          1.0000000  -0.6266127  0.9571311
## kcl.cluster -0.6266127   1.0000000 -0.4522672
## pot.kiln     0.9571311  -0.4522672  1.0000000
```

```
# >>>>>>>>>>>> Question 3 <<<<<<<
```

On comparing the clustering solution with the kiln variable offered an individual stepwise cross tabulation classification as follows. Revealing the number of pots in the cluster for each values of kiln. On discussing about the concerns in reproducing the clustering solution, there was no concerns in reproducing the clustering after appending the “kiln” variable. Moreover, the rand index value remains 1 and class agreement retains the same good agreement even after reproducing the clustering solutions. Additionally, verifying with the correlation between the clustering and kiln variable, where kil variable has perfectly high correlation with hierarchical clustering and high correlation with k-means clustering. Therefore the column “kiln” have no influence on the rand index value and class agreement of the clustering solution whether “kiln” variable is appended or removed.

```
# Import the dataset "Pima"
pima <- read.csv("C:/Users/Shanila/OneDrive/Documents/R/Multivariate/Pima.csv")

# Listing the first 5 observations of the dataset
head(pima)
```

```
##   npreg glu bp skin  bmi   ped age type
## 1     5  86 68   28 30.2 0.364  24   No
## 2     7 195 70   33 25.1 0.163  55  Yes
## 3     5  77 82   41 35.8 0.156  35   No
## 4     0 165 76   43 47.9 0.259  26   No
## 5     0 107 60   25 26.4 0.133  23   No
## 6     5  97 76   27 35.6 0.378  52  Yes
```

```
# Listing the structure of the dataset
str(pima)
```

```
## 'data.frame':    527 obs. of  8 variables:
##  $ npreg: int  5 7 5 0 0 5 3 1 3 2 ...
##  $ glu  : int  86 195 77 165 107 97 83 193 142 128 ...
##  $ bp   : int  68 70 82 76 60 76 58 50 80 78 ...
##  $ skin : int  28 33 41 43 25 27 31 16 15 37 ...
##  $ bmi  : num  30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...
##  $ ped  : num  0.364 0.163 0.156 0.259 0.133 ...
##  $ age  : int  24 55 35 26 23 52 25 24 63 31 ...
##  $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
```

```
# Making the object accessible by variable
attach(pima)

# Importing the MASS package for making use of Qaudratic discriminant function
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.5.2
```

```
# Generate a random number from the observation
random_sample = sample(1:527, 1)
random_sample
```

```
## [1] 314
```

```
# Delete the observation with the random number generated
dat = pima[-random_sample,]
# dat

# Using the Quadratic discriminant function on the dataset
qda.res <- qda(type ~ npreg + glu + bp + skin + bmi + ped + age, data=dat)

# Summary statistics of the model
summary(qda.res)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   14     -none- numeric
## scaling 98     -none- numeric
## ldet     2     -none- numeric
## lev      2     -none- character
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
# Attributes of the model
attributes(qda.res)
```

```
## $names
##  [1] "prior"   "counts"  "means"   "scaling" "ldet"    "lev"     "N"      
##  [8] "call"    "terms"   "xlevels"
## 
## $class
## [1] "qda"
```

```
# Prior value of the model
qda.res$prior
```

```
##        No       Yes 
## 0.6653992 0.3346008
```

```
# Means of the model
qda.res$means
```

```
##        npreg      glu       bp     skin     bmi       ped      age
## No  2.905714 110.1657 69.92286 27.22286 31.4500 0.4466657 29.17714
## Yes 4.676136 142.9659 74.70455 32.98864 35.7733 0.6178011 36.37500
```

```
# Finding the Posterior values using built it function qda
qda.res.cv <- qda(type ~ npreg + glu + bp + skin + bmi + ped + age, CV=TRUE, data=dat)
attributes(qda.res.cv)
```

```
## $names
## [1] "class"     "posterior" "terms"     "call"      "xlevels"
```

```
qda.res.cv$posterior[1,]
```

```
##         No        Yes 
## 0.96331478 0.03668522
```

```
# Calculating the overall misclassification rate
table(qda.res.cv$class, dat$type)
```

```
##      
##        No Yes
##   No  296  75
##   Yes  54 101
```

```
# Manual calculations of qda
# Observation count
N <- nrow(dat)
N
```

```
## [1] 526
```

```
# Group count
G <- length(levels(dat$type))
G
```

```
## [1] 2
```

```
# Observations with type "Yes"
pima_yes <- subset(dat[,-8], type =="Yes")
# pima_yes

# Number of Observations with type "Yes"
length(which(dat$type == "Yes"))
```

```
## [1] 176
```

```
# Observations with type "No"
pima_no <- subset(dat[,-8], dat$type =="No")
# pima_no

# Number of Observations with type "No"
length(which(dat$type == "No"))
```

```
## [1] 350
```

```
# Covariance of group "Yes"
cov_yes <- cov(pima_yes[1:7])
# cov_yes

# Covariance of group "No"
cov_no <- cov(pima_no[1:7])
# cov_no

# Constructing function to calculate quadratic discriminant function for group "Yes"
/ "No"
qdf <- function(x, prior, mu, covar)
{
  x <- matrix(as.numeric(x), ncol=1)
  log(prior) - (0.5*log(det(covar))) - (0.5*t(x-mu)%*%solve(covar)%*%(x-mu))
}

# Function definition for calculating the qdf for two group types
id = 1
qdfs <- rep(0, G)
for(g in 1:G)
{
  # Group type "NO"
  if (g == 1){
    qdfs[g] <- qdf(dat[id,1:7], qda.res$prior[g], qda.res$mean[g,], cov_no)
  }
  # Group type "YES"
  else if (g ==2){
    qdfs[g] <- qdf(dat[id,1:7], qda.res$prior[g], qda.res$mean[g,], cov_yes)
  }
}

# Calucalating the means, prior manually
prior = rep(0, G)
prior[1] = length(which(dat$type == "No"))/N
prior[2] = length(which(dat$type == "Yes"))/N
prior
```

```
## [1] 0.6653992 0.3346008
```

```
# Calculating the mean manaully
mean_no = apply(pima_no, 2, mean)
mean_yes = apply(pima_yes, 2, mean)
```

```
#  >>>>>>>>>>>>>>>>>>>>>>>>>>>> 3a <<<<<<<<<<<<<<<<<<<<<<<<<<<
# New member of the Pima Indian Population
new_member = data.frame(npreg =7, glu=187, bp=50, skin=33, bmi=33.9, ped=0.826, age=3
0)

# QDA Function to verify the new member of the Pima Indian Population
qdfs <- rep(0, G)
for(g in 1:G)
{
  # Group type "NO"
  if (g == 1){
    qdfs[g] <- qdf(new_member, qda.res$prior[g], qda.res$mean[g,], cov_no)
  }
  # Group type "YES"
  else if (g ==2){
    qdfs[g] <- qdf(new_member, qda.res$prior[g], qda.res$mean[g,], cov_yes)
  }
}

# To Identify the type of the new member of the Pima Indian Population
levels(dat$type)[qdfs == max(qdfs)]
```

```
## [1] NA NA
```

The Quadratic discrimination function for the new member for which we need to create a function with arguments x (with the categorical values removed from the new member), prior value, mu value and co-variance values (for group “Yes” and group “No”) as per the study criteria. From the figure above, the first condition calculates the quadratic discrimination analysis function (QDA) for Pima Indian population who do not have diabetes (Type = “No”), whereas the second function calculates the QDA for Pima Indian population who have diabetes (Type = “Yes”). On interpreting the new member of Pima Indian population mentioned in the figure above, it found from QDF, that the new member have diabetes (shows “Yes” to the variable Type). The reason for arriving into this result is after substituting the explanatory variables of the new member of PIP on both the equations held under the condition g == 1 and g == 2, we arrived with the values, -19.67576 and -18.15183. From, the result since the value of group 1 is maximum among the both the new member was classified as type “Yes”. Hence, the new member was categorized as the person with diabetes.

```r
# >>>>>>>>>>>>>>>>>> 3b <<<<<<<<<<<<<<<<<<<<<<<<
# Constructing the test set values
test_set = data.frame(npreg =c(2,9,10,5,1), glu=c(88,170,101,121,93),
                      bp=c(58,74,76,72,70), skin=c(26,31,48,23,31),
                      bmi=c(28.5,44.0,32.9,26.2,30.4),
                      ped=c(0.766,0.403,0.171,0.245,0.315), age=c(22,43,63,30,23),
                      type=c("No", "Yes", "No", "No", "No"))

# Function definition for calculating the qdf for two group types
# Validation for the existing observations
test <- c(1:5)
flag  = 1
predicted_value = rep(0, 5)
for(id in test){
  qdfs <- rep(0, G)
  for(g in 1:G)
  {
    if (g == 1){
      # Test data
      qdfs[g] <- qdf(test_set[id,1:7], qda.res$prior[g], qda.res$mean[g,], cov_no)
    }
    else if (g ==2){
      # Test data
      qdfs[g] <- qdf(test_set[id,1:7], qda.res$prior[g], qda.res$mean[g,], cov_yes)
      print(qdfs)
      print(levels(dat$type)[qdfs == max(qdfs)])
      predicted_value[flag] = levels(dat$type)[qdfs == max(qdfs)]
      flag = flag + 1
    }
  }
}
```

```
## [1] -13.18389       NaN
## [1] NA NA
```

```
## Warning in predicted_value[flag] <- levels(dat$type)[qdfs == max(qdfs)]:
## number of items to replace is not a multiple of replacement length
```

```
## [1] -19.87016       NaN
## [1] NA NA
```

```
## Warning in predicted_value[flag] <- levels(dat$type)[qdfs == max(qdfs)]:
## number of items to replace is not a multiple of replacement length
## [1] -20.96155       NaN
## [1] NA NA
```

```
## Warning in predicted_value[flag] <- levels(dat$type)[qdfs == max(qdfs)]:
## number of items to replace is not a multiple of replacement length
```

```
## [1] -12.76906       NaN
## [1] NA NA
```

```
## Warning in predicted_value[flag] <- levels(dat$type)[qdfs == max(qdfs)]:
## number of items to replace is not a multiple of replacement length
```

```
## [1] -12.52663       NaN
## [1] NA NA
```

```
## Warning in predicted_value[flag] <- levels(dat$type)[qdfs == max(qdfs)]:
## number of items to replace is not a multiple of replacement length
```

```
# Predicted values by the QDF function
print(predicted_value)
```

```
## [1] NA NA NA NA NA
```

```
# Comparing the true known test set observations with the test set group type predicte
d the model
qda_predict = predict(qda.res, test_set)
qda_predict
```

```
## $class
## [1] No  Yes Yes No  No
## Levels: No Yes
##
## $posterior
##           No        Yes
## 1 0.97913602 0.02086398
## 2 0.04726247 0.95273753
## 3 0.44632554 0.55367446
## 4 0.94434161 0.05565839
## 5 0.98284093 0.01715907
```

```
# Tabulation of predicted values using predict function and test set data
cross_validation = table(qda_predict$class, test_set$type)
cross_validation
```

```
## 
##        No Yes
##   No   3   0
##   Yes  1   1
```

```
# Tabulation of qda function predicted values and test set data
tab = table(predicted_value, test_set$type)
tab
```

```
## < table of extent 0 x 2 >
```

```
# Calculating the misclassification cross_validation
misclassification = (cross_validation[1,2]+cross_validation[2,1])/
  (cross_validation[1,2]+cross_validation[2,1]+cross_validation[1,1]+cross_validation
[2,2])
misclassification
```

```
## [1] 0.2
```

Out of the 5 members of the test set, except the second member rest four were not found to suffer from diabetes. Tried predicting whether or not the five members of test data have diabetes using the previously fitted Quadratic discriminant Analysis model (QDF). Firstly, created a data frame with the test set data consisting of 5 observation as mentioned in the figure above. Secondly, Made use of the QDA function defined to predict the type value, to check whether the members shows “Yes” / “No” to diabetes.

After validating the test set observation in the QDA function, it was found that there was visible that the third member/ observation was misclassified from the given test set. The predicted values using QDA function and the actual values were cross tabulation and misclassification was observed. On calculating the misclassification rate it was found to be 0.2, proving it evident that there was 20% misclassification.

Parallely tried verifying using the build in predict function in R. The misclassification was same with both user defined QDA function and with the predict function.

AZAM JAINULLABUDIN MOHAMED

18201501

4) Given: Prior probability.

$\pi_A = \frac{N_A}{N}$ $\pi_B = \frac{N_B}{N}$

To prove:

$\exists\ \delta_B(\underline{x}) > \delta_A(\underline{x})$ — ①

Eventually,

$$\exists\ \underline{x}^T \Sigma^{-1} (\underline{\mu}_B - \underline{\mu}_A) > \frac{1}{2} \underline{\mu}_B^T \Sigma^{-1} \underline{\mu}_B - \frac{1}{2} \underline{\mu}_A^T \Sigma^{-1} \underline{\mu}_A + \log\left(\frac{N_A}{N}\right) - \log\left(\frac{N_B}{N}\right)$$

solution:

WKT, Parameters of the class g, which gives largest linear discriminant function is the class to which the point $\underline{x}$ is most likely to belong.

As we know

$$\delta_g(\underline{x}) = \log \pi_g + \underline{x}^T \Sigma^{-1} \underline{\mu}_g - \frac{1}{2} \underline{\mu}_g^T \Sigma^{-1} \underline{\mu}_g \quad -②$$

Proof:

~~Sol~~ substituting ② in ①

$$\log\left(\frac{N_B}{N}\right) + \underline{\underline{x}^T \Sigma^{-1} \underline{\mu}_B} - \frac{1}{2} \underline{\mu}_B^T \Sigma^{-1} \underline{\mu}_B > \log\left(\frac{N_A}{N}\right) + \underline{\underline{x}^T \Sigma^{-1} \underline{\mu}_A} - \frac{1}{2} \underline{\mu}_A^T \Sigma^{-1} \underline{\mu}_A$$

$$\underline{x}^T \Sigma^{-1} \underline{\mu}_B - \underline{x}^T \Sigma^{-1} \underline{\mu}_A > \log\left(\frac{N_A}{N}\right) - \log\left(\frac{N_B}{N}\right) - \frac{1}{2} \underline{\mu}_A^T \Sigma \underline{\mu}_A + \frac{1}{2} \underline{\mu}_B^T \Sigma^{-1} \underline{\mu}_B$$

$$\therefore \underline{x}^T \Sigma^{-1} (\underline{\mu}_B - \underline{\mu}_A) > \frac{1}{2} \underline{\mu}_B^T \Sigma \underline{\mu}_B - \frac{1}{2} \underline{\mu}_A^T \Sigma \underline{\mu}_A + \log\left(\frac{N_A}{N}\right) - \log\left(\frac{N_B}{N}\right).$$

$\therefore$ Hence proved.

```
# setting seed to roll number
set.seed(18201501)

# Importing the dataset
pottery <- read.csv(file="C:/Users/Shanila/OneDrive/Documents/R/Multivariate/PotteryData(1).csv")
head(pottery)
str(pottery)

# Generating a Random number
random_sample = sample(1:45, 1)
# random_sample

# Removing a random row, removing row 10
pot = pottery[-random_sample,]
# str(pot)

# Remove the last column "kiln"
potter = pot[,-ncol(pot)]
# str(potter)

# >>>>>>>>> 1a. i. COMPUTING THE COVARIANCE MATRIX <<<<<<<<<<
# Calculating the covariance with Data as data frame
covariance_mat = cov(potter)
covariance_mat

# >>>>>>>>> 1a. ii. CALCULATING THE FIRST TWO EIGEN VALUE AND EIGEN <<<<<<<
# >>>>>>>>> VECTORS OF COVARIANCE MATRIX <<<<<<<
# Calculating only the eigen values
eigen_values = eigen(covariance_mat, only.values = TRUE)
# eigen_values

# Calculating both eigen value and eigen vectors
eigenvalue_vector = eigen(covariance_mat)
eigenvalue_vector

#Calculating both eigen value and eigen vectors
eigen = eigen(covariance_mat)
eigen

eigen_values=eigen$values
eigen_vectors=eigen$vectors
eigen_vectors[,1]

eigenvalue_vector = eigen(covariance_mat)
eigenvalue_vector
# Extracting the first two eigen vectors
eigenvalue_vector[[2]][,c(1,2)]
first_eigenvalue = as.matrix(eigenvalue_vector[[1]][1])
second_eigenvalue = as.matrix(eigenvalue_vector[[1]][2])
first_eigenvector <- as.matrix(eigen_vectors[,1])
second_eigenvector <- as.matrix(eigen_vectors[,2])
round(t(covariance_mat %*% first_eigenvector)) == round(first_eigenvalue %*% t(first_eigenvector))
round(t(covariance_mat %*% second_eigenvector)) == round(second_eigenvalue %*% t(second_eigenvector))

# 1 (a). (iii)
# Proving Orthonormality
# Conditions for Orthonoramality - Orthogonality and Magnitude of the value to be unit vector
first_eigenvector <- as.matrix(eigenvalue_vector[[2]][1,])
```

```
second_eigenvector <- as.matrix(eigenvalue_vector[[2]][2,])

# Orthogonality check (transpose(x) * x == 0)
orthogonality = t(first_eigenvector) %*% second_eigenvector
round(orthogonality)

# Calculating the magnitude value
# First eigenvector
mag_eigen_vector_1 = sum((eigenvalue_vector[[2]][,1])^2)
mag_eigen_vector_1
# Second eigenvector
mag_eigen_vector_2 = sum((eigenvalue_vector[[2]][,2])^2)
mag_eigen_vector_2

# 1 (a). (iv)
# Standardizing the data using scale function
scaling = scale(potter)
# scaling

## Calculating the expected mean, variance and correlation
# >>>>>>>> 1b. CALCULATION OF EXPECTED VALUE AND COVARIANCE <<<<<<<<<<
expect_x1=10; var_x1=11
expect_x2=8; var_x2=14
covar_x1x2=2

# 1b (i)
# Expect value of x1 - x2
expect_x1_sub_x2 = expect_x1 - expect_x2
expect_x1_sub_x2

# Variance of x1 - x2
var_x1_sub_x2 = var_x1 + var_x2 - 2*covar_x1x2
var_x1_sub_x2

# 1b (ii)
expect_x1_expect_x2 = expect_x1 *expect_x2 + covar_x1x2
covar_UV = var_x1 + expect_x1^2 - 3 * expect_x1_expect_x2 + 2 *(var_x2+expect_x2^2) + 12
var_x1_sub_x2 = var_x1 + var_x2 - 2*covar_x1x2
var_x1_sub_2x2 = var_x1 + 4*var_x2 - 4*covar_x1x2
corr_UV = covar_UV / (sqrt(var_x1_sub_x2)*sqrt(var_x1_sub_2x2))
corr_UV

## k-means clustering
# >>>>>>>>>>>>>> 2a. K-MEANS CLUSTERING <<<<<<<<<<<<<<<<<<<<<<<
# Maximum range of K values
K=10
WGSS <- rep(NA, K)

# Calculating the Weighted sum of squares
for(k in 1:K) {
  WGSS[k] <- kmeans(potter, centers=k, nstart=1000)$tot.withinss
}

# Elbow plot to identify the best k-means clustering value
plot(1:K, WGSS, type='b', main="K-Means")

## Hierarchical clustering
# >>>>>>>>>>> 2b. HIERARCHICAL CLUSTERING <<<<<<<<<<<<<<<<
```

```
hcluster = hclust(dist(potter), method = "average")
plot(hcluster)

# Cutting dendogram
hcl = cutree(hcluster, k=3)
table(hcl)
kcl = kmeans(potter, centers = 3)
table(hcl, potter[,1])
plot(hcl)

# >>>>>>>>>>>>>>>> 2c. cOMPARING THE CLUSTERING SOLUTION <<<<<<<<<<<<<
hcl = cutree(hcluster, k=3)
kcl = kmeans(potter, centers = 3, nstart = 10)
tab <- table(hcl, kcl$cluster)
tab

# comparison using class agreement
library(e1071)
n <- classAgreement(tab)

# To find the best mapping between the clusters.
matchClasses(tab, method="exact")

library(mclust)
# Clustering
mod1 <- Mclust(tab)
summary(mod1)

#  >>>>>>>>>>>>>>>> 2d <<<<<<<<<<<<<<<<<<<<<<<<<
# Clustering relationship after binding the kiln column back
# Repeated the same above procedure
# class agreement
tab <- table(hcl, kcl$cluster, pot$kiln)
tab

data = data.frame(hcl, kcl$cluster, pot$kiln)
# data

cor(data)

# >>>>>>>>>>>> Question 3 <<<<<<
# Import the dataset "Pima"
pima <- read.csv("C:/Users/Shanila/OneDrive/Documents/R/Multivariate/Pima.csv")

# Listing the first 5 observations of the dataset
head(pima)

# Listing the structure of the dataset
str(pima)

# Making the object accessible by variable
attach(pima)

# Importing the MASS package for making use of Qaudratic discriminant function
library(MASS)

# Generate a random number from the observation
random_sample = sample(1:527, 1)
```

```
random_sample

# Delete the observation with the random number generated
dat = pima[-random_sample,]
# dat

# Using the Quadratic discriminant function on the dataset
qda.res <- qda(type ~ npreg + glu + bp + skin + bmi + ped + age, data=dat)

# Summary statistics of the model
summary(qda.res)

# Attributes of the model
attributes(qda.res)

# Prior value of the model
qda.res$prior

# Means of the model
qda.res$means

# Finding the Posterior values using built it function qda
qda.res.cv <- qda(type ~ npreg + glu + bp + skin + bmi + ped + age, CV=TRUE, data=dat)
attributes(qda.res.cv)
qda.res.cv$posterior[1,]

# Calculating the overall misclassification rate
table(qda.res.cv$class, dat$type)

# Manual calculations of qda
# Observation count
N <- nrow(dat)
N

# Group count
G <- length(levels(dat$type))
G

# Observations with type "Yes"
pima_yes <- subset(dat[,-8], type =="Yes")
pima_yes

# Number of Observations with type "Yes"
length(which(dat$type == "Yes"))

# Observations with type "No"
pima_no <- subset(dat[,-8], dat$type =="No")
pima_no

# Number of Observations with type "No"
length(which(dat$type == "No"))

# Covariance of group "Yes"
cov_yes <- cov(pima_yes[1:7])
cov_yes

# Covariance of group "No"
cov_no <- cov(pima_no[1:7])
```

```
cov_no

# Constructing function to calculate quadratic discriminant function for group "Yes" / "No"
qdf <- function(x, prior, mu, covar)
{
  x <- matrix(as.numeric(x), ncol=1)
  log(prior) - (0.5*log(det(covar))) - (0.5*t(x-mu)%*%solve(covar)%*%(x-mu))
}

# Function definition for calculating the qdf for two group types
id = 1
qdfs <- rep(0, G)
for(g in 1:G)
{
  # Group type "NO"
  if (g == 1){
    qdfs[g] <- qdf(dat[id,1:7], qda.res$prior[g], qda.res$mean[g,], cov_no)
  }
  # Group type "YES"
  else if (g ==2){
    qdfs[g] <- qdf(dat[id,1:7], qda.res$prior[g], qda.res$mean[g,], cov_yes)
  }
}

# Calucalating the means, prior manually
prior = rep(0, G)
prior[1] = length(which(dat$type == "No"))/N
prior[2] = length(which(dat$type == "Yes"))/N
prior

# Calculating the mean manaully
mean_no = apply(pima_no, 2, mean)
mean_yes = apply(pima_yes, 2, mean)

#  >>>>>>>>>>>>>>>>>>>>>>>>>>>>> 3a <<<<<<<<<<<<<<<<<<<<<<<<<<<<
# New member of the Pima Indian Population
new_member = data.frame(npreg =7, glu=187, bp=50, skin=33, bmi=33.9, ped=0.826, age=30)

# QDA Function to verify the new member of the Pima Indian Population
qdfs <- rep(0, G)
for(g in 1:G)
{
  # Group type "NO"
  if (g == 1){
    qdfs[g] <- qdf(new_member, qda.res$prior[g], qda.res$mean[g,], cov_no)
  }
  # Group type "YES"
  else if (g ==2){
    qdfs[g] <- qdf(new_member, qda.res$prior[g], qda.res$mean[g,], cov_yes)
  }
}

# To Identify the type of the new member of the Pima Indian Population
levels(dat$type)[qdfs == max(qdfs)]

# >>>>>>>>>>>>>>>>>>> 3b <<<<<<<<<<<<<<<<<<<<<<<<<<
# Constructing the test set values
test_set = data.frame(npreg =c(2,9,10,5,1), glu=c(88,170,101,121,93),
           bp=c(58,74,76,72,70), skin=c(26,31,48,23,31),
           bmi=c(28.5,44.0,32.9,26.2,30.4),
           ped=c(0.766,0.403,0.171,0.245,0.315), age=c(22,43,63,30,23),
           type=c("No", "Yes", "No", "No", "No"))

# Function definition for calculating the qdf for two group types
# Validation for the existing observations
test <- c(1:5)
flag  = 1
predicted_value = rep(0, 5)
for(id in test){
  qdfs <- rep(0, G)
  for(g in 1:G)
  {
    if (g == 1){
      # Test data
      qdfs[g] <- qdf(test_set[id,1:7], qda.res$prior[g], qda.res$mean[g,], cov_no)
    }
    else if (g ==2){
      # Test data
      qdfs[g] <- qdf(test_set[id,1:7], qda.res$prior[g], qda.res$mean[g,], cov_yes)
      print(qdfs)
      print(levels(dat$type)[qdfs == max(qdfs)])
      predicted_value[flag] = levels(dat$type)[qdfs == max(qdfs)]
      flag = flag + 1
    }
  }
}

# Predicted values by the QDF function
print(predicted_value)


# Comparing the true known test set observations with the test set group type predicted the model
qda_predict = predict(qda.res, test_set)
qda_predict

# Tabulation of predicted values using predict function and test set data
cross_validation = table(qda_predict$class, test_set$type)
cross_validation

# Tabulation of qda function predicted values and test set data
tab = table(predicted_value, test_set$type)
tab

# Calculating the misclassification cross_validation
misclassification = (cross_validation[1,2]+cross_validation[2,1])/
  (cross_validation[1,2]+cross_validation[2,1]+cross_validation[1,1]+cross_validation[2,2])
misclassification
```