

Human Mitochondrial SNP / Mutations Patterns

Vignesh J Muralidharan

September 30, 2018

1. INTRODUCTION

This study focuses on mitochondria data extracted from the 1000 Genome Project dataset; a repository of 1024 individuals' genetic information displayed in terms of "mutations". Each entry (i.e. person) is classified with a genetic 'grouping'. Along with this grouping are 2711 predictor variables that each represent a genetic sequence. For each genetic sequence there is either no mutation, represented as a zero, or there is a mutation, represented as a one. In this sense, a mutation occurs if the particular entry's genetic sequence varies distinctly from the average. If so, it is classified as a mutation.

This is a classification problem in which it is desired to be able to predict an individual's group based on their mutations present. Interestingly, the mutations may lead to distinct groupings that can describe the population better than or as well as the original grouping method used. This cluster based analysis could provide useful insights into the actual grouping of mutations. A combination of supervised and unsupervised learning will thus be used in the present study.

```
library(tidyverse) ; library(factoextra)
library(cluster) ; library(NbClust)
library(fpc) ; library(dendroextras)
library(dendextend) ; library(mclust)
library(dbSCAN) ; library(dplyr)
mito=read.csv("https://raw.githubusercontent.com/vigneshjmurali/Statistical-Predictive-Mo
delling/master/Datasets/Mt1t.mutate.csv")
mito<-mito[-c(1:3),]
dim(mito)
## [1] 1074 2712
#IMPUTATION - MISSING VALUES WITH MEDIAN
for (i in 2:ncol(mito)){
  mito[is.na(mito[,i]),i]<-median(mito[,i],na.rm = TRUE)
}
#REMOVING COLUMNS WHOSE VARIANCE IS EQUAL TO ZERO
mito1=as.matrix(sapply(mito[-1], as.numeric))
mito2<-as.data.frame(mito1[,apply(mito1,2,var,na.rm=TRUE) !=0])
mito2=cbind(mito$Group,mito1)
colnames(mito2)[1]<-"Group"
dim(mito2)
## [1] 1074 2712
```

2. PRINCIPAL COMPONENTS ANALYSIS

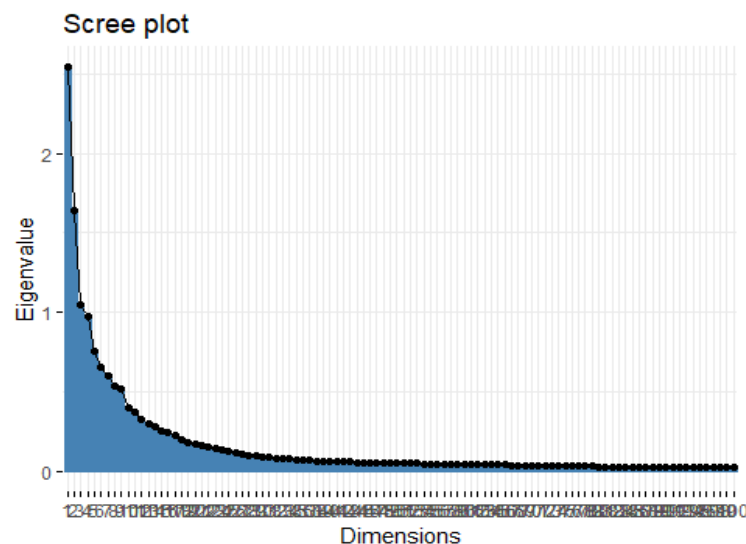
```
par(mfrow=c(1,2))
mito.s=scale(mito2)
mito.pca=prcomp(mito2[, -1],scale=FALSE)
# The rotation measure provides the principal component loading
# Each column of rotation matrix contains the principal component loading vector
mito.pca$rotation[1:5,1:5]
```

```
##          PC1          PC2          PC3          PC4          PC5
## X1  0.0000780871 -4.404682e-05 -9.757529e-05  0.0001029323 -5.138898e-05
## X2 -0.0008179848  2.450072e-03 -7.109211e-04 -0.0013039466 -1.965245e-04
## X3 -0.0004295638  1.258802e-03 -3.017082e-04 -0.0005888065  1.314967e-04
## X4 -0.0004020731  1.252719e-03 -7.331386e-05 -0.0007134918  3.795829e-04
## X5 -0.0004020731  1.252719e-03 -7.331386e-05 -0.0007134918  3.795829e-04
# Standard deviation of each principal component and computing variance
mito.sd=mito.pca$sdev
mito.var=mito.pca$sdev^2
mito.var[1:10]
## [1] 2.5420599 1.6401542 1.0478262 0.9772971 0.7598770 0.6524412 0.5995810
## [8] 0.5350683 0.5202540 0.4035791
# Proportion of variance
pve=mito.var/sum(mito.var)
which.max(cumsum(pve)[cumsum(pve)<0.95])
## [1] 372
which.max(cumsum(pve)[cumsum(pve)<0.98])
## [1] 555
## This tells us we need to keep 372 PC's to retain 95% of our total variance and further
## 555 for 98%. This is a rather large number and tells us that many of our 2712 groupings are
## necessary. Despite this, a quick look at only three PC's suggests that groupings are well
## defined. This indicates that later clustering and classification may work successfully
## with as few as three PC's.
```

3. CLUSTERING METHODS

Clustering is a form of unsupervised learning in which no labelled grouping or classification exists previously for the data, but it is wished to understand how the data is structured. This study will use the following clustering methods: (1) K-means, (2) Fuzzy k-means, (3) h-clust, (4) NbClust, (5) Mclust

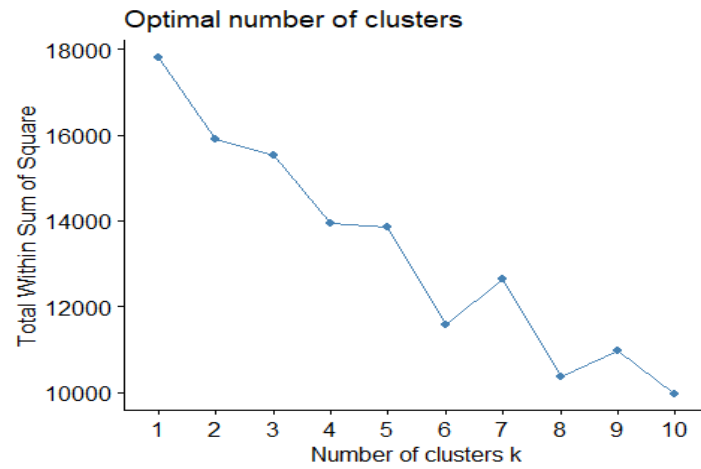
```
# Number of components to achieve account for 80% of the total variance
# Selecting the principle components of first 100 PC1 : PC100
cumsum(pve[100])
## [1] 0.00117194
fviz_screplot(mito.pca,ncp=100,choice="eigenvalue")
```



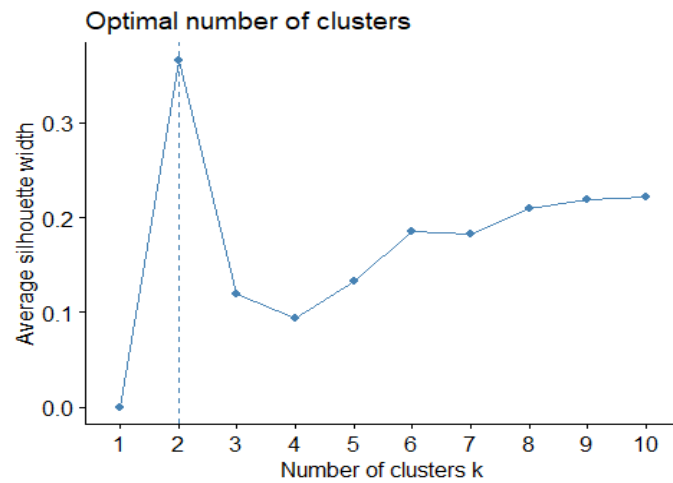
```
plot(cumsum(pve),xlab="Principal Component",
     ylab="Cumulative Proportion of variance explained",ylim=c(0,1),type='b')
```



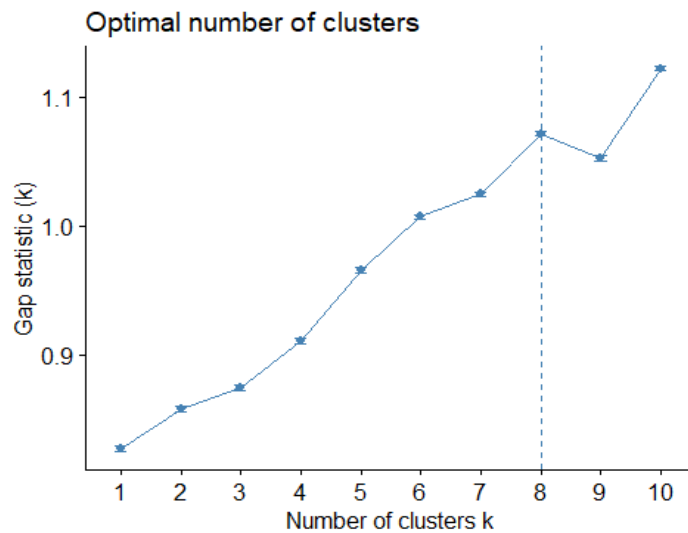
```
mitonew.s=scale(mitonew)
# OPTIMAL NUMBER OF CLUSTERS - (1) WSS, (2) Silhouette, (3) Gap_stat, (4) NbClust
# For PC 1 to PC 100
set.seed(10)
fviz_nbclust(mitonew,kmeans,method="wss") # Using elbow method - wss
```



```
fviz_nbclust(mitonew,kmeans,method="silhouette") #Using silhouette method
```



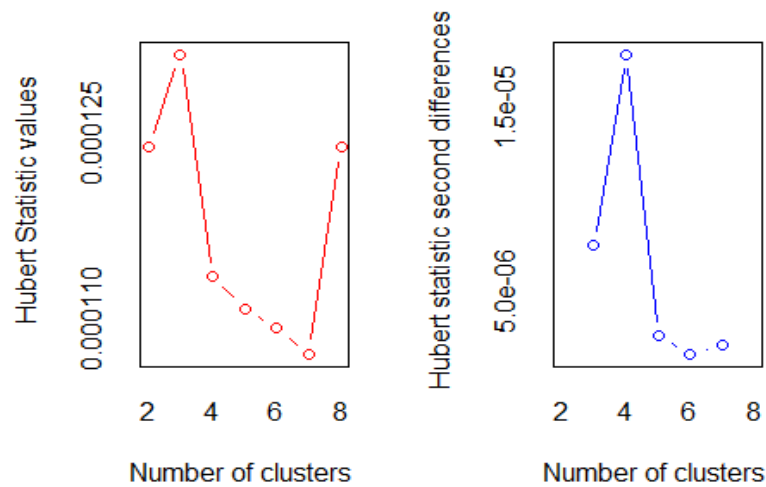
```
fviz_nbclust(mitonew,kmeans,method="gap_stat") #Using gap_stat method
```



```

mito.nbclust<-mitonew %>% #Using NbClust
  scale() %>%
  NbClust(distance="euclidean",min.nc=2,max.nc=8,method="complete",index="all")

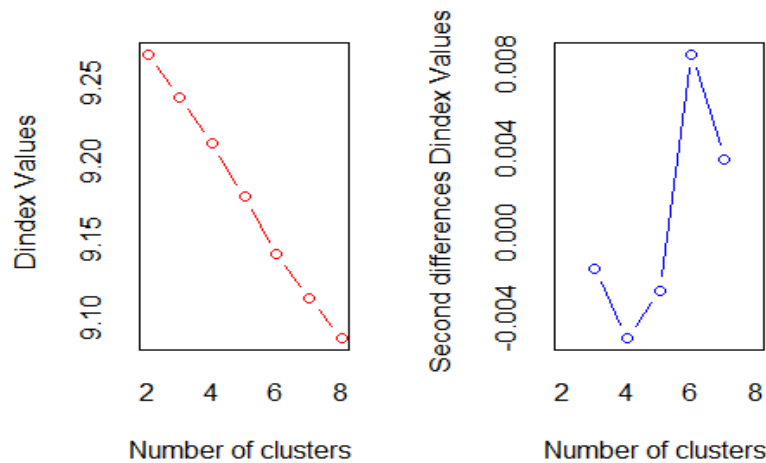
```



```

## *** : The Hubert index is a graphical method of determining the number of clusters.
##       In the plot of Hubert index, we seek a significant knee that corresponds to a
##       significant increase of the value of the measure i.e the significant peak in Hubert
##       index second differences plot.

```

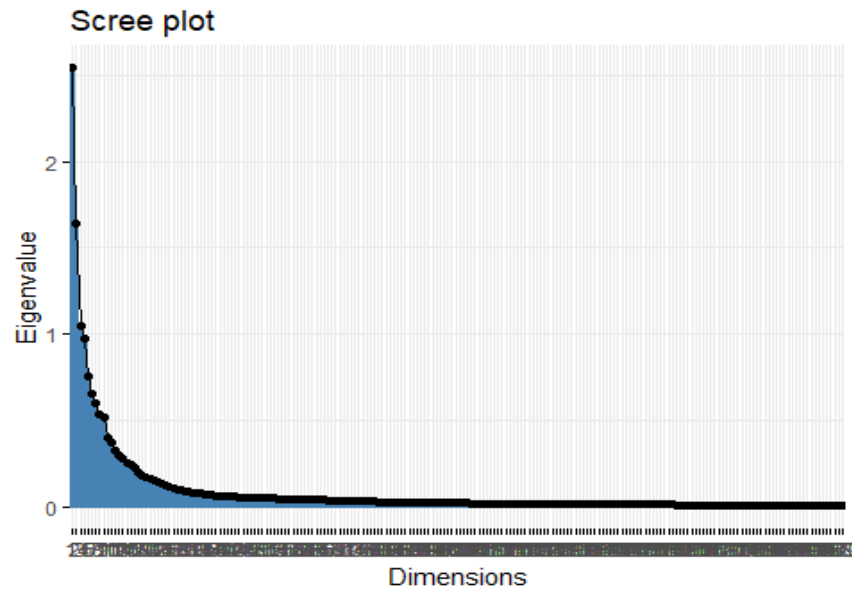


```

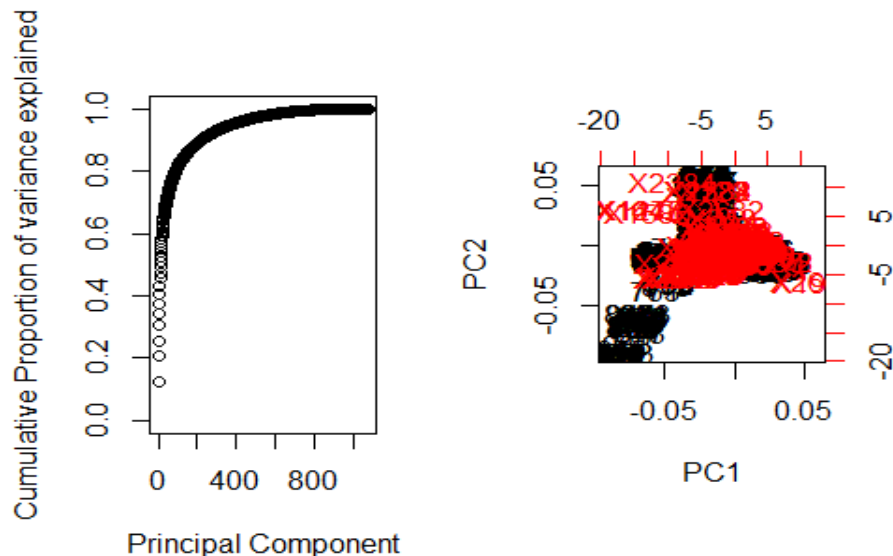
## *** : The D index is a graphical method of determining the number of clusters.
##       In the plot of D index, we seek a significant knee (the significant peak in Dindex
##       second differences plot) that corresponds to a significant increase of the value of
##       the measure.
## *****
## * Among all indices:
## * 12 proposed 2 as the best number of clusters
## * 2 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 5 proposed 6 as the best number of clusters
## * 3 proposed 8 as the best number of clusters

```

```
##          ***** Conclusion *****
## * According to the majority rule, the best number of clusters is 2
##
## *****
# Number of components to achieve account for 80% of the total variance
# Selecting the principle components of first 200 PC1 : PC200
# Proportion of variance
pve=mito.var/sum(mito.var)
# Number of components to achieve account for 80% of the total variance
cumsum(pve[200])
## [1] 0.0005113816
fviz_screplot(mito.pca,ncp=200,choice="eigenvalue")
```



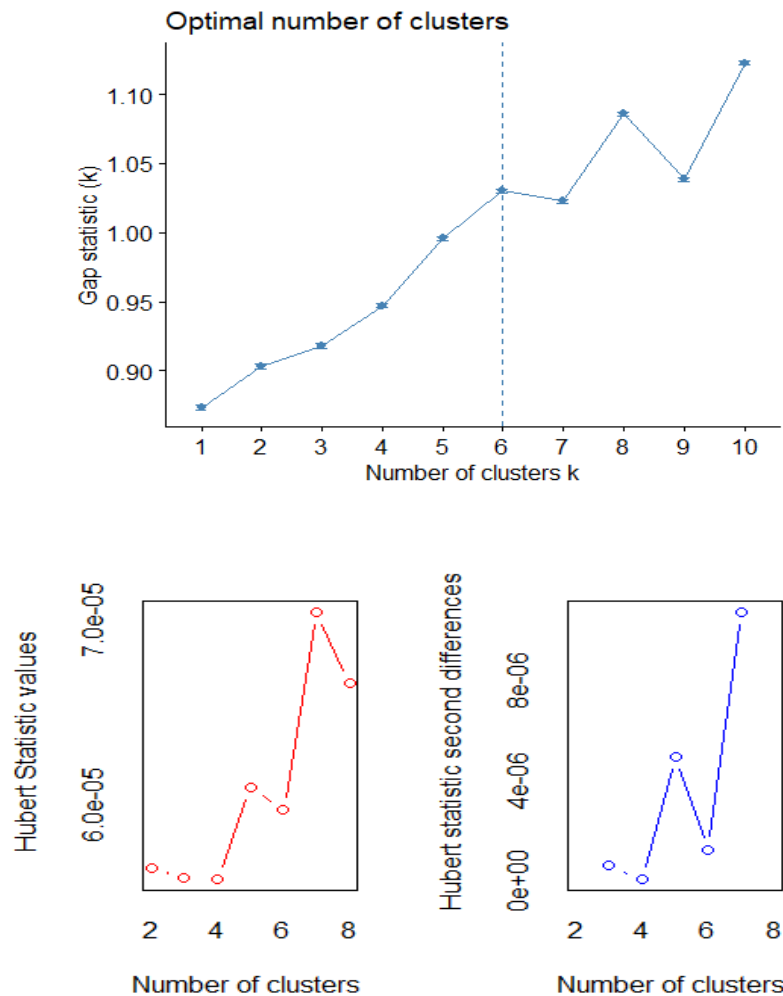
```
plot(cumsum(pve),xlab="Principal Component",
      ylab="Cumulative Proportion of variance explained",ylim=c(0,1),type='b')
biplot(mito.pca,arrow.len=0) # Arrow head length is suppressed to get rid of the errors of indeterminate angle
```



```

mitoClasses1<- factor(mito$Group)
plot(main="Different Groups",mito.pca$x[,1:200],col=mitoClasses1)
# Choosing the principle components as new variables based on the total variance
mitonew1=mito.pca$x[,1:200]
mitonew1.s=scale(mitonew1)
# OPTIMAL NUMBER OF CLUSTERS - (1) WSS, (2) Silhouette, (3) Gap_stat, (4) NbClust
# For PC 1 to PC 200
set.seed(11)
fviz_nbclust(mitonew1,kmeans,method="wss") # Using elbow method - wss
fviz_nbclust(mitonew1,kmeans,method="silhouette")#Using silhouette method
fviz_nbclust(mitonew1,kmeans,method="gap_stat")#Using gap_stat method
mito.nbclust<-mitonew1 %>% #Using NbClust
  scale() %>%
  NbClust(distance="euclidean",mi.nc=2,max.nc=8,method="complete",index="all")

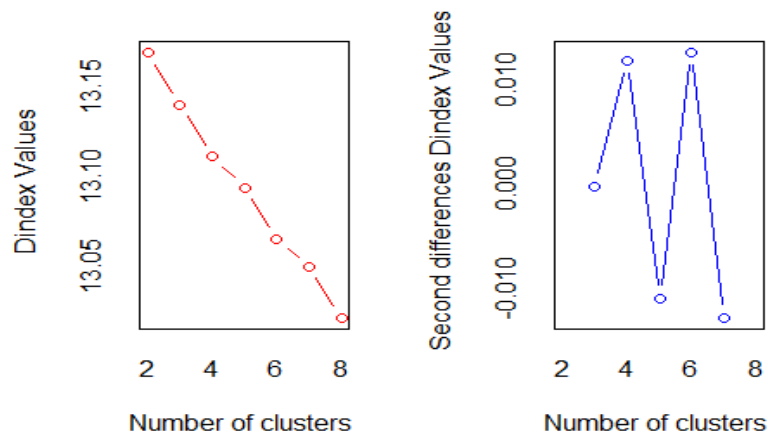
```



```

## *** : The Hubert index is a graphical method of determining the number of clusters.
##       In the plot of Hubert index, we seek a significant knee that corresponds to a
##       significant increase of the value of the measure i.e the significant peak in Hubert
##       index second differences plot.
## *** : The D index is a graphical method of determining the number of clusters.
##       In the plot of D index, we seek a significant knee (the significant peak in Dindex
##       second differences plot) that corresponds to a significant increase of the value of
##       the measure.

```



```
## *****
## * Among all indices:
## * 11 proposed 2 as the best number of clusters
## * 1 proposed 3 as the best number of clusters
## * 4 proposed 4 as the best number of clusters
## * 3 proposed 5 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 3 proposed 8 as the best number of clusters
##          ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
## *****
```

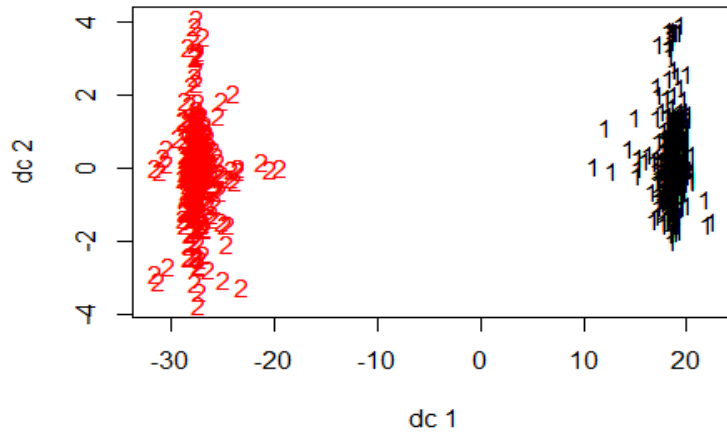
HERE WSS, SILHOUETTE, GAP_STATISTIC AND NBCLUST WERE USED TO FIND THE OPTIMAL NUMBER OF CLUSTERS.

K-MEANS CLUSTERING - PERFORMED WITH K=2, K=8 , K=6 FOR BOTH FIRST 100 AND 200 PC's

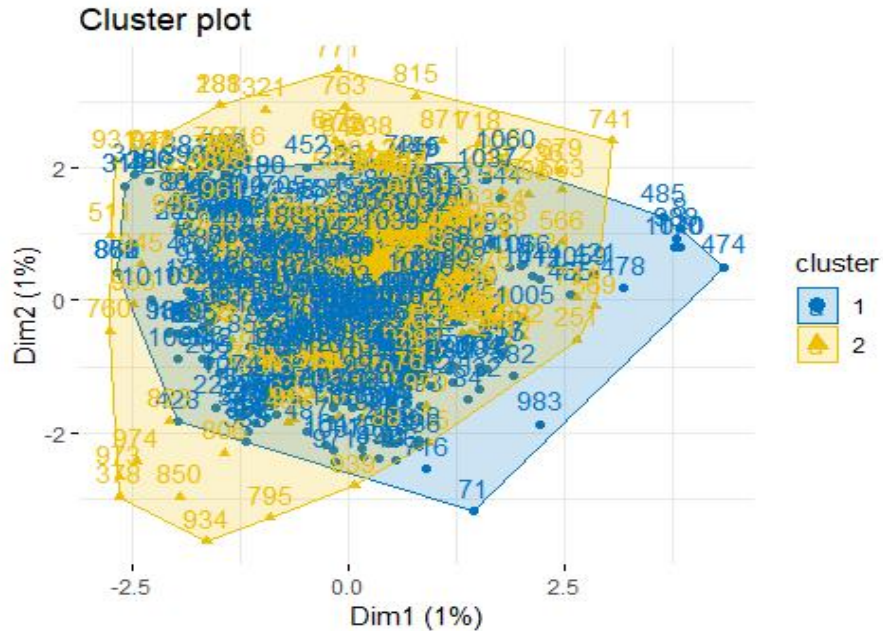
4. FUZZY K-MEANS

Fuzzy k-means is similar in concept to the original k-means with the exception that it does not assign a particular category to the nearest neighbors, but rather assigns a weight based on distance to all points.

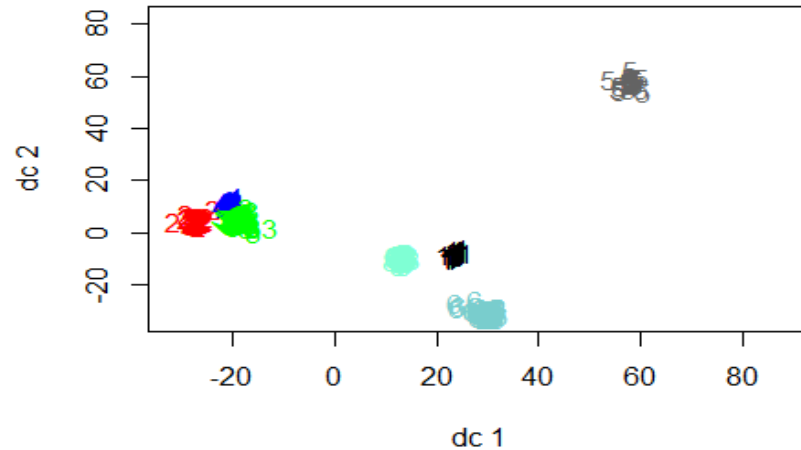
```
#mitnew - k=2 & 8
set.seed(10)
km_100_2.fit=kmeans(mitnew,2,nstart=50)
attributes(km_100_2.fit)
## $names
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
##
## $class
## [1] "kmeans"
km_100_2.fit$size
## [1] 637 437
km_100_2.fit$tot.withinss
## [1] 15378.58
plotcluster(mitnew,km_100_2.fit$cluster)
```

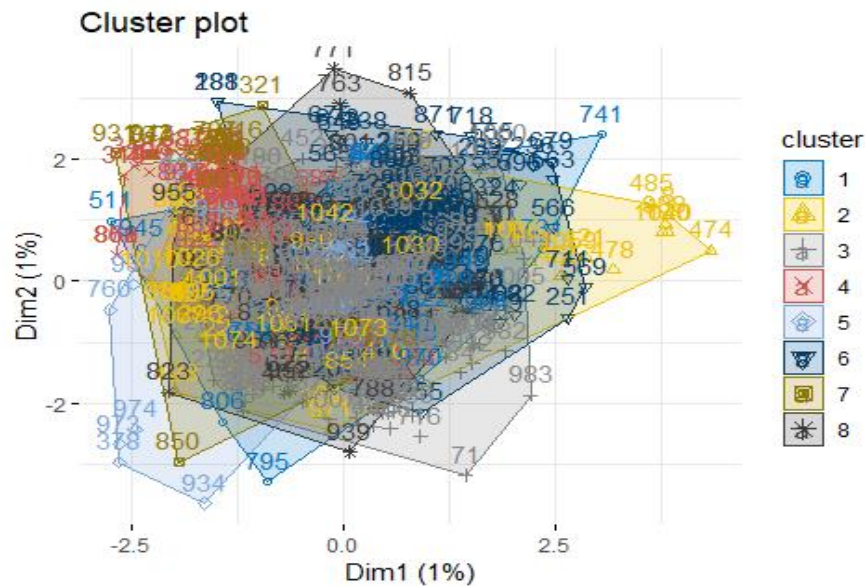
```
fviz_cluster(km_100_2.fit,data=mitonew,ellipse.type="convex",palette="jco",ggtheme=theme_minimal())
```



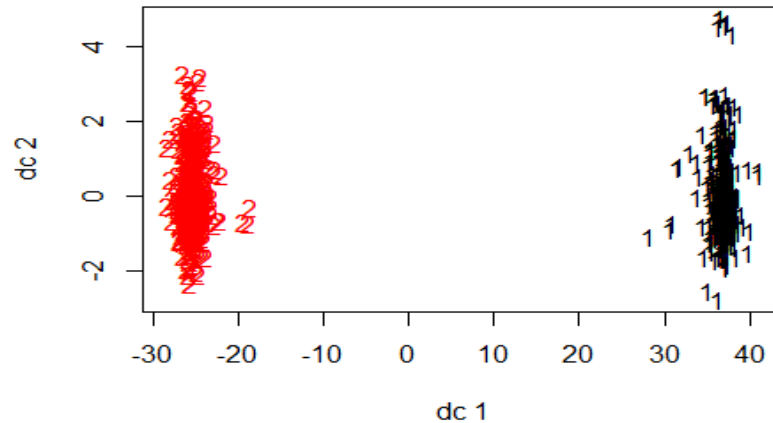
```
set.seed(5)
km_100_8.fit=kmeans(mitonew,8,nstart=50)
attributes(km_100_8.fit)
## $names
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
##
## $class
## [1] "kmeans"
km_100_8.fit$size
## [1] 63 107 443 87 39 193 23 119
km_100_8.fit$tot.withinss
## [1] 9637.872
plotcluster(mitonew,km_100_8.fit$cluster)
```



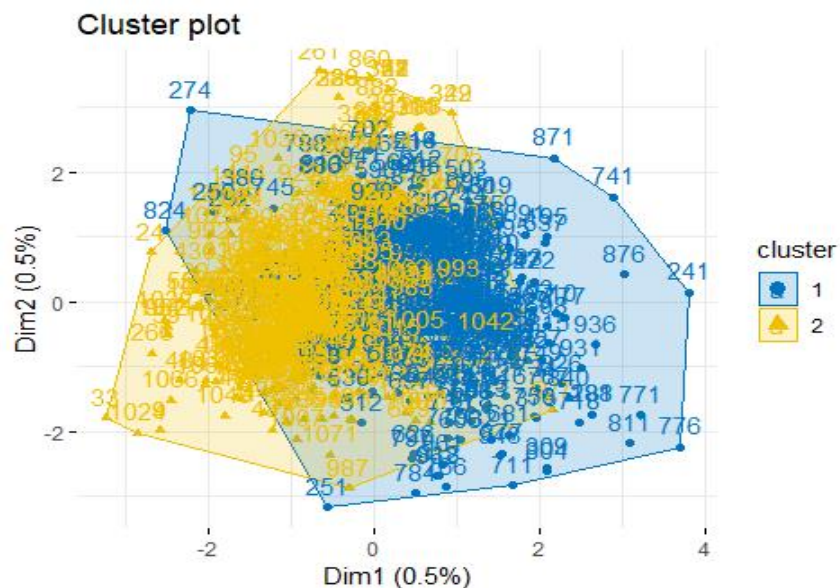
```
fviz_cluster(km_100_8.fit, data=mitonew, ellipse.type="convex", palette="jco", ggtheme=theme_minimal())
```



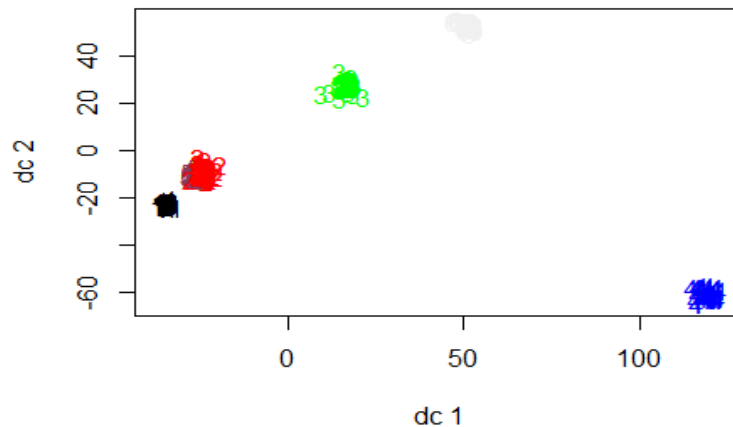
```
#mitnew1 - k=2 & 6
set.seed(9)
km_200_2.fit=kmeans(mitonew1,2,nstart=50)
attributes(km_100_2.fit)
## $names
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
##
## $class
## [1] "kmeans"
km_200_2.fit$size
## [1] 437 637
km_200_2.fit$tot.withinss
## [1] 17022.93
plotcluster(mitonew1,km_200_2.fit$cluster)
```



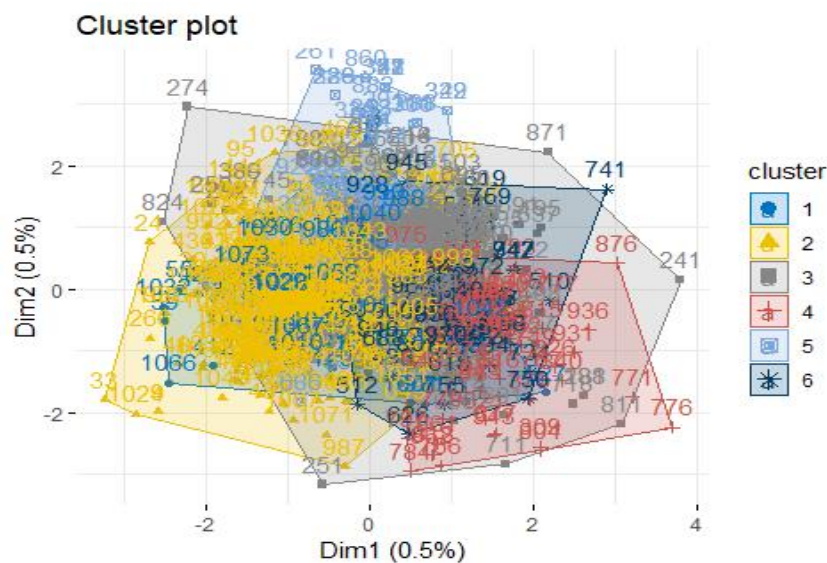
```
fviz_cluster(km_200_2.fit, data=mitonew1, ellipse.type="convex", palette="jco", ggtheme=theme_minimal())
```



```
set.seed(8)
km_200_6.fit=kmeans(mitonew1,6,nstart=50)
attributes(km_200_6.fit)
## $names
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
##
## $class
## [1] "kmeans"
km_200_6.fit$size
## [1] 107 443 303 71 87 63
km_200_6.fit$tot.withinss
## [1] 12601.02
plotcluster(mitonew1,km_200_6.fit$cluster)
```



```
fviz_cluster(km_200_6.fit,data=mitonew1,ellipse.type="convex",palette="jco",ggtheme=theme_minimal())
```

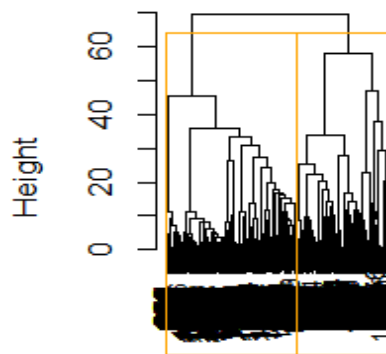


5. HIERARCHIAL CLUSTERING

Hierarchical clustering is an alternative method of clustering that does not need to preselect the number of groups that are to be produced. It uses a tree-based representation of the data known as dendrogram.

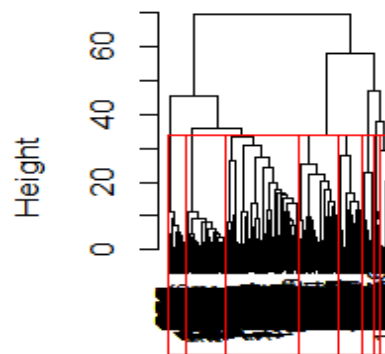
```
par(mfrow=c(1,2))
#Hierarchial Clustering with K=2
mito.hc.ward=hclust(dist(mitonew,method="euclidean"),method="ward.D2")
#Dendrogram
plot(mito.hc.ward, main="Complete Linkage",xlab="",cex=.9)
#Drawing dendrogram with red borders around the clusters
rect.hclust(mito.hc.ward,k=2,border="orange")
#Hierarchial Clustering with K=8
mito.hc.ward=hclust(dist(mitonew,method="euclidean"),method="ward.D2")
#Dendrogram
plot(mito.hc.ward, main="Complete Linkage",xlab="",cex=.9)
#Drawing dendrogram with red borders around the clusters
rect.hclust(mito.hc.ward,k=8,border="red")
```

Complete Linkage



`hclust (*, "ward.D2")`

Complete Linkage



`hclust (*, "ward.D2")`

#2D representation of the segmentation

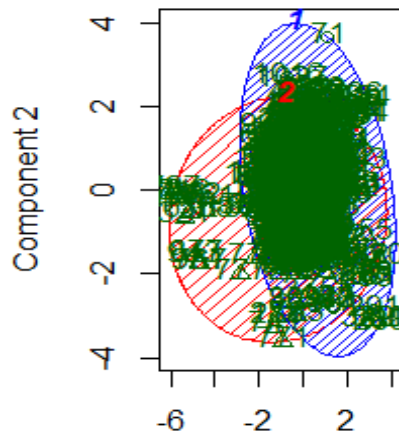
`groups2=cutree(mito.hc.ward,2)#Cut Tree into 2 clusters`

`clusplot(mitonew,groups2,color=TRUE,shade = TRUE,labels=2,lines=0, main='Group segments')`

`groups8=cutree(mito.hc.ward,8)#Cut Tree into 8 clusters`

`clusplot(mitonew,groups8,color=TRUE,shade = TRUE,labels=2,lines=0, main='Group segments')`

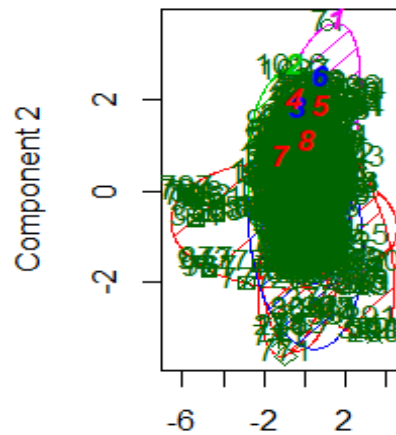
Group segments



Component 1

These two components

Group segments

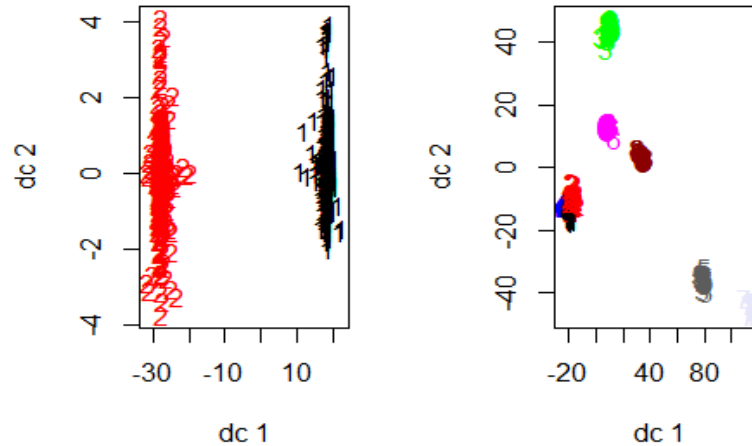


Component 1

These two components

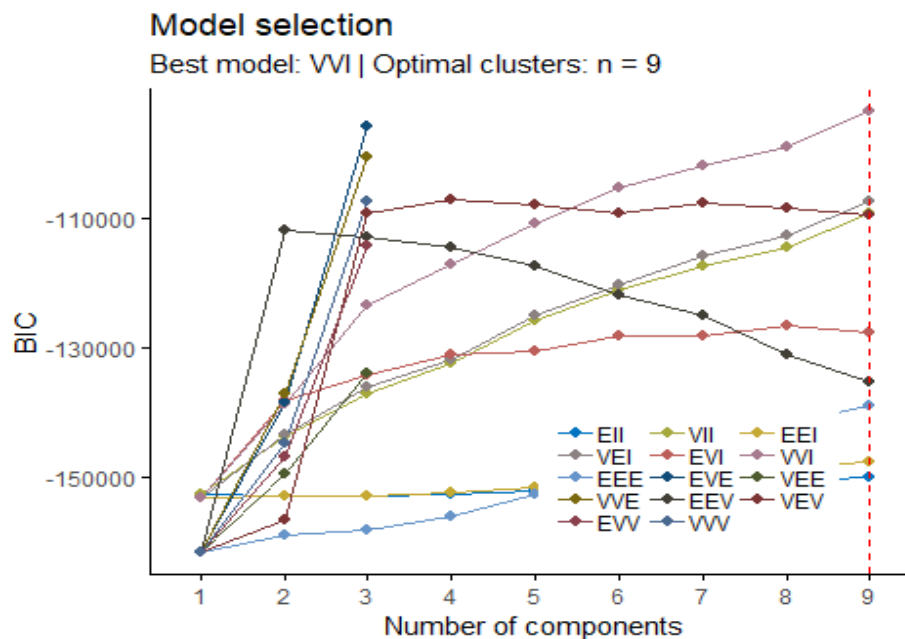
`plotcluster(mitonew, groups2)#Centroid plot against 1st 2 discriminant functions`

`plotcluster(mitonew,groups8)#Centroid plot against 1st 2 discriminant functions`

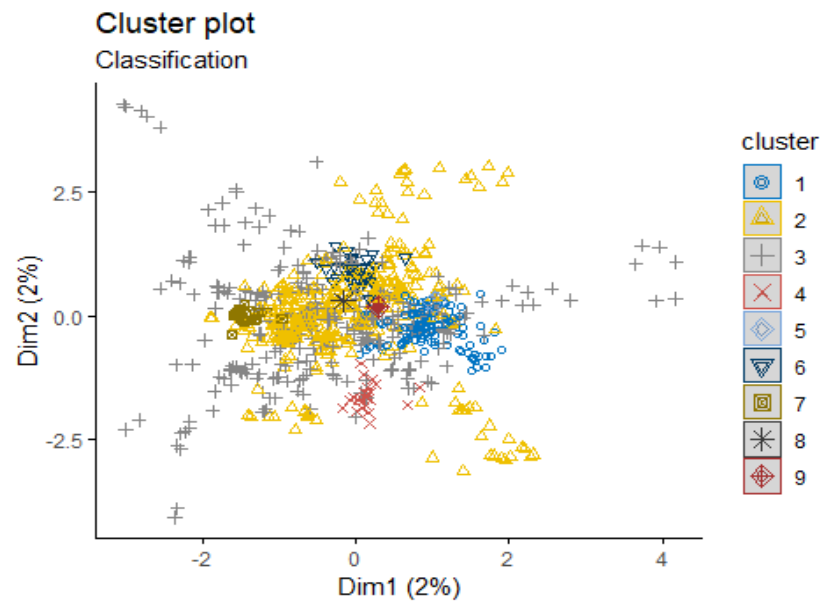


6. MODEL BASED CLUSTERING

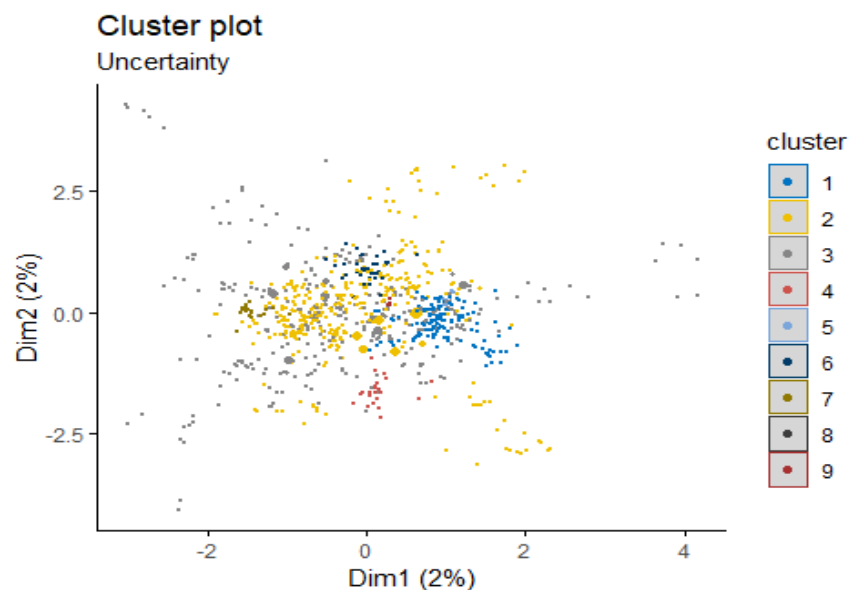
```
par(mfrow=c(1,3))
mito.fit<-Mclust(mitnew.s[,0:50])
summary(mito.fit); mito.fit$modelName ; mito.fit$G
## -----
## Gaussian finite mixture model fitted by EM algorithm
## -----
## Mclust VVI (diagonal, varying volume and shape) model with 9 components:
##
## log.likelihood    n df      BIC      ICL
##      -43603.22 1074 908 -93543.5 -93555.69
## Clustering table:
##   1  2  3  4  5  6  7  8  9
## 170 407 335  26  21  63  25  10  17
## [1] "VVI"
## [1] 9
fviz_mclust(mito.fit,"BIC",palette="jco")
```



```
fviz_mclust(mito.fit,"classification",geom="point",pointsize=1.5, palette="jco")
## Warning: Computation failed in `stat_ellipse()`:  
## the leading minor of order 2 is not positive definite
```



```
fviz_mclust(mito.fit,"uncertainty", palette="jco")
## Warning: Computation failed in `stat_ellipse()`:  
## the leading minor of order 2 is not positive definite
```



USING MODEL BASED CLUSTERING , THE RESULT IS THE OPTIMAL CLUSTER IS K=9

7. DENSITY BASED CLUSTERING

```
#par(mfrow=c(1,2))
#dbscan::kNNdistplot(mitonew,k=2)
#abline(h=2, lty=2,col="red")
#set.seed(123)
#DETERMINING THE OPTIMAL EPS VALUE
#mito.db<-fpc::dbscan(mitonew,eps=1,50)
#mito.db
#fviz_cluster(mito.db,data=mitonew, stand=FALSE, ellipse=TRUE,show.clust.cent = TRUE,
#              geon="point",palette="default", ggtheme=theme_minimal())
```

AFTER TRYING THESE CLUSTERING METHODS, IT SEEMS THAT THE CLUSTERING RESULTS ARE QUITE DIFFERENT K-MEANS = 8 IS BETTER FIT. Hierarchical Clustering: k=2 or k=8 is reasonably ok Fuzzy clustering: can't tell which number of clusters is better Model Based Clustering: k=9(close to k=8). From the result above, together with the results get from using Elbow method,Silhouette method,Gap statistic and NbClust to find the optimal number of clusters, I chose k=8 as the optimal number of clusters to generate the new groupings

8. GENERATING NEW GROUPINGS

this will create the csv file separately to perform the classification in python with the groupings

```
mito.group<-data.frame(mitonew,km_100_8.fit$cluster)
#mito.group2<-data.frame(mitonew,km_100_2.fit$cluster)
colnames(mito.group)[101]<-"Group"
mito.group$Group<-factor(as.character(mito.group$Group))
#write.csv(mito.group,"mitogroup.csv")
```