BINF 702 - Spring 2016 - Final Project

Cluster Analysis on Khan datasets using K-means VS Hierarchical methods

Rohan Patil and Momina Tariq*

May 9, 2016

Abstract

R has an incredible variety of functions for cluster analysis. Most common methods include hierarchical clustering and K-means clustering. Hierarchical clustering is used for relationship discovery and defines the cluster distance between two clusters to be the maximum distance between their individual components, whereas K-means requires the analyst to specify the number of clusters to extract. Our goal line is to compare the performance of various clustering methods on Khan Gene Dataset and find the most suitable one for biological significance.

*Graduate students, Department of Bioinformatics and Computational Biology, George Mason University

Introduction

The research topic of our final project is based on the following research question and our hypothesis based on it. *Research Question:* Which clustering method is the most perferable for analyzing gene data? why? *Hypothesis:* K-means clustering is most commonly used and hence we hypothesize that it would be the most preferred one. However, we would reproduce our own clusters using both Hierarchical and K-means clustering methods, then validate cluster solutions, compare their performances and analyze which one is more preferable.

This report of our research project is devided in following sections to describe the project work very well. *Contents:* (1) Background and objectives, (2) Computational methods, (3) Results and Discussion, (4) Conclusions and a brief description of how the conclusions of your analyses could be tested using biochemical or genetic techniques.

The list of references has been given at the end of this report.

1 Background and Objectives

- ***Khan Data Set:***

  Khan data set uses cDNA microarrays containing 6567 clones of which 3789 were known genes and 2778 were ESTs to study the expression of genes in of four types of small round blue cell tumours of childhood (SRBCT). These were neuroblastoma (NB), rhabdomyosarcoma (RMS), Burkitt lymphoma, a subset of non-Hodgkin lymphoma (BL), and the Ewing family of tumours (EWS). Gene expression profiles from both tumor biopsy and cell line samples were obtained and are contained in this dataset. This interesting data set offers two

interesting items

- Train: data.frame of 2308 rows and 63 columns. The training data set of 63 arrays and 2308 gene expression values
- Test: data.frame of 2308 rows and 20 columns. The test data set of 20 arrays and 2308 genes expression values

For each tissue sample, gene expression measurements are available. The data set consists of training data, xtrain and ytrain, and testing data, xtest and ytest

```
names(Khan)  # List all components

## [1] "xtrain" "xtest"  "ytrain" "ytest"

## Dimentions of each component:
dim(Khan$xtrain)  # Subjects VS Gene Expression Measurements

## [1]   63 2308

dim(Khan$xtest)  # Subjects VS Gene Expression Measurements

## [1]   20 2308

## Levels with # Subjects:
table(Khan$ytrain)

##
##  1  2  3  4
##  8 23 12 20
```

```
table(Khan$ytest)


##

## 1 2 3 4

## 3 6 6 5
```

- ***Cluster Analysis:***

  Much of the history of cluster analysis is concerned with developing algorithms that were not too computer intensive, since early computers were not nearly as powerful as they are today. Accordingly, computational shortcuts have traditionally been used in many cluster analysis algorithms. These algorithms have proven to be very useful, and can be found in most computer software.

  More recently, many of these older methods have been revisited and updated to reflect the fact that certain computations that once would have overwhelmed the available computers can now be performed routinely. In R, a number of these updated versions of cluster analysis algorithms are available through the cluster library, providing us with a large selection of methods to perform cluster analysis, and the possibility of comparing the old methods with the new to see if they really provide an advantage.

  A problem which often arises in Bioinformatics is to find genes which have similar expression patterns. Genes that express similarly under certain conditions may have similar functions. In general, cluster analysis also known as unsupervised learning consists of several methods for discovering a subset of data points, such as genes which form a group under some observable similarity criteria, such as gene expression. These methods can be divided

into k-means clustering and hierarchical clustering. k-means cluster analysis is one of the oldest methods of cluster analysis and is available in R through the k-means function. The first step, and certainly not a trivial one, when using k-means cluster analysis is to specify the number of clusters (k) that will be formed in the final solution. The process begins by choosing k observations to serve as centers for the clusters. Then, the distance from each of the other observations is calculated for each of the k clusters, and observations are put in the cluster to which they are the closest. After each observation has been put in a cluster, the center of the clusters is recalculated, and every observation is checked to see if it might be closer to a different cluster, now that the centers have been recalculated. The process continues until no observations switch clusters. Looking on the good side, the k-means technique is fast, and doesn't require calculating all of the distances between each observation and every other observation. It can be written to efficiently deal with very large data sets, so it may be useful in cases where other methods fail.

Another class of clustering methods, known as hierarchical agglomerative clustering methods, starts out by putting each observation into its own separate cluster. It then examines all the distances between all the observations and pairs together the two closest ones to form a new cluster. This is a simple operation, since hierarchical methods require a distance matrix, and it represents exactly what we want - the distances between individual observations. So finding the first cluster to form simply means looking for the smallest number in the distance matrix and joining the two observations that the distance correspnds to into a new cluster. Now there is one less cluster than there are observations. Agglomerative Hierarchical cluster analysis is provided in R through the hclust function.

In this report we tend to apply both these clustering methods on Khan data set to find similar genes and to figure out which method is optimum in order to reach our objective.

2 Computational Methods

In our project we have used the following methods on Khan Data.

2.1 Data Preperation

Before clustering of data, first step we did was data preparation. This step is done to remove or estimate missing data and rescale variables for comparability. A generic function called na.omit() was applied on our data set for this purpose.

In order to sandardize our data, another generic function called scale() was applied on our data set. This function by default centers and/or scales the columns of a numeric matrix. We also did principal component analysis to summarize our set with a smaller number of representative variables that collectively explain most of the variability in our original set.

Euclidian distance is also calculated as a measure of closeness. It is used as a dissimilarity measure during clustering analysis. Other dissimilarity measures can also be used, for example, correlation-based distance which considers two observations to be similar if their features are highly correlated, but Euclidean distance is by far the most preferred one. The Euclidean distance can be calculated using the dist() function.

```
## For Khan train data
naTrain = na.omit(Khan$xtrain)  # Omit Missing Values
stdTrain = scale(naTrain)  # Data Standardization
distTrain = dist(stdTrain)  # Euclidian Distance
## For Khan test data
naTest = na.omit(Khan$xtest)  # Omit Missing Values
stdTest = scale(naTest)  # Data Standardization
distTest = dist(stdTest)  # Euclidian Distance

## Perform principal components analysis using prcomp with scale=TRUE.
pr.out.train = prcomp(Khan$xtrain, scale. = TRUE)  # for training set
pr.out.test = prcomp(Khan$xtest, scale. = TRUE)  # for test set
```

2.2 Hierarchical Clustering

Next step we do is, hierarchical clustering. In this type of clustering we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a dendrogram, dendrogram that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to n. To create a dendogram from a cluster solution, simply pass it to the plot function. There are advantages and disadvantages to each of these clustering approaches, which we highlight later in this report. The term hierarchical refers to the fact that clusters obtained by cutting the dendrogram at a given height are necessarily nested within the

clusters obtained by cutting the dendrogram at any greater height.

The hierarchical clustering dendrogram is obtained via an extremely simple algorithm.We begin by defining some sort of dissimilarity measure between each pair of observations. The concept of dissimilarity between a pair of observations needs to be extended to a pair of groups of observations. This extension is achieved by developing the notion of linkage, which defines the dissimilarity between two groups of observations. The four most common types of linkage—complete, average, single, and centroid. Average, complete, and single linkage are most popular among statisticians. We also used "average" linkage while doing hierarchical clustering to find mean of intercluster dissimilarity.

```
## Perform hierarchical clustering using average linkage clustering.
hc.avg.train = hclust(distTrain, method ="average")
hc.avg.test = hclust(distTest, method ="average")
## Re-perform hierarchical clustering and cut the tree into 4 clusters.
hc.out.train = hclust(distTrain)
hc.cutree.train = cutree(hc.out.train, 4)
table(hc.cutree.train)  # For Khan train data

## hc.cutree.train
##  1  2  3  4
## 34  8 19  2

hc.out.test = hclust(distTest)
hc.cutree.test = cutree(hc.out.test, 4)
```

```
table(hc.cutree.test)   # For Khan test data


## hc.cutree.test

##  1  2  3  4

## 10  2  5  3
```

2.3 K-Means Clustering

Our third step was to apply K-means clustering on our data set. K-means clustering is a simple and elegant approach for partitioning data set into K distinct, non-overlapping clusters. It is an unsupervised learning algorithm that tries to cluster data based on their similarity. Unsupervised learning means that there is no outcome to be predicted, and the algorithm just tries to find patterns in the data. To perform K-means clustering, we first specified the desired number of clusters K; then the K-means algorithm assigns each observation to exactly one of the K clusters. For our analysis k (i.e. number of clusters) was set to 4.

```
# Perform k-means clustering on all data with k=4.

set.seed(0)

## Khan train data

km.out.train = kmeans(stdTrain, 4)

km.clusters.train = km.out.train$cluster

## Khan test data

km.out.test = kmeans(stdTest, 4)

km.clusters.test = km.out.test$cluster
```

```
# Re-perform k-means clustering on only 1st 4 Principal Components.
set.seed(0)
km1st4.out.train = kmeans(stdTrain[,1:4], 4)
km1st4.out.test = kmeans(stdTrain[,1:4], 4)
```

2.4 Model-Based Clustering

Model based approaches assume a variety of data models and apply maximum likelihood estimation and Bayes criteria to identify the most likely model and number of clusters. Specifically, the Mclust( )function in the mclust package selects the optimal model according to BIC for EM initialized by hierarchical clustering for parameterized Gaussian mixture models.

mclust is a contributed R package for model-based clustering, classification, and density estimation based on finite normal mixture modelling. It provides functions for parameter estimation via the EM algorithm for normal mixture models with a variety of covariance structures, and functions for simulation from these models. Also included are functions that combine model-based hierarchical clustering, EM for mixture estimation and the Bayesian Information Criterion (BIC) in comprehensive strategies for clustering, density estimation and discriminant analysis. Additional functionalities are available for displaying and visualizing fitted models along with clustering, classification, and density estimation results.

We do model based clustering on both Khan Training and test data set.

```
# For Khan train data

mc.train = Mclust(Khan$xtrain,Khan$ytrain)

summary(mc.train)


## ----------------------------------------------------

## Gaussian finite mixture model fitted by EM algorithm

## ----------------------------------------------------

##

## Mclust EEI (diagonal, equal volume and shape) model with 4 components:

##

##  log.likelihood  n    df     BIC     ICL

##       -108678.4 63 11543 -265181 -265181

##

## Clustering table:

##  1  2  3  4

## 17 22 16  8


MDA = MclustDA(Khan$xtrain[,1:4], Khan$ytrain)
```

```
# For Khan test data

mc.test = Mclust(Khan$xtest,Khan$ytest)

summary(mc.test)


## ----------------------------------------------------
```

```
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEI (diagonal, equal shape) model with 3 components:
##
##  log.likelihood  n   df       BIC       ICL
##        -32808.4 20 9236 -93285.38 -93285.38
##
## Clustering table:
##  1  2  3
##  5 10  5

# MclustDA(Khan$xtest[,1:4], Khan$ytest)
```

2.5 Plotting Cluster Solutions

```
## Provide a pairs plot on the first 2 principal components.
Cols=function(vec){ cols = rainbow(length(unique(vec)))
  return(cols[as.numeric(as.factor(vec))])}
par(mfrow=c(1,2))
plot(pr.out.train$x[,1:2], col=Cols(Khan$ytrain),
     main="Fig.1.1: Plotting 1st 2 Principal Components of training set")
```

```
plot(pr.out.test$x[,1:2], col=Cols(Khan$ytest),
     main="Fig.1.2: Plotting 1st 2 Principal Components of test set")
```

Fig.1.1: Plotting 1st 2 Principal Components of training

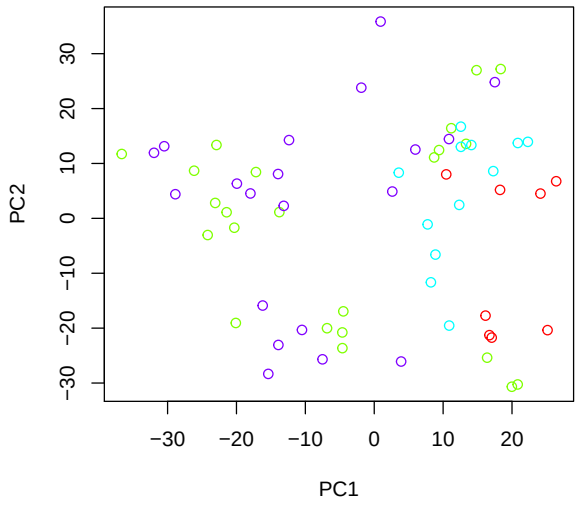
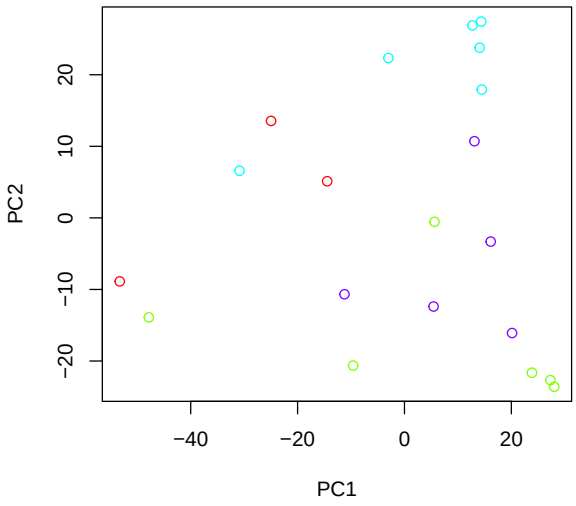


Fig.1.2: Plotting 1st 2 Principal Components of test s



We can clearly observe structures along with some outliers in this PCA of 1st 2 principal components. For training set there are comparatively more stuctures and less outliers compared to test set.

```
## PVE
pve=100*pr.out.train$sdev^2/sum(pr.out.train$sdev^2)
par(mfrow=c(1,2))
plot(pve, type="o", ylab="PVE", xlab="Principal Component",
     col =" blue ", main="Fig.1.3: the PVE of each principal component")
plot(cumsum(pve), type="o", ylab="Cumulative PVE", xlab="Principal Component
     col =" brown3 ", main="Fig.1.4: cumulative PVE of the principal compone
```

Fig.1.3: the PVE of each principal component

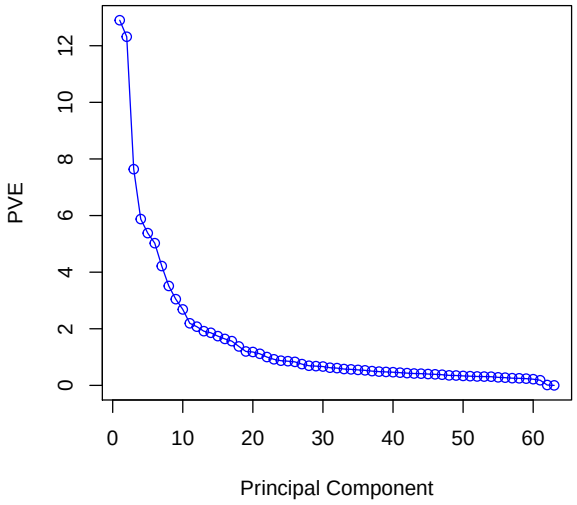
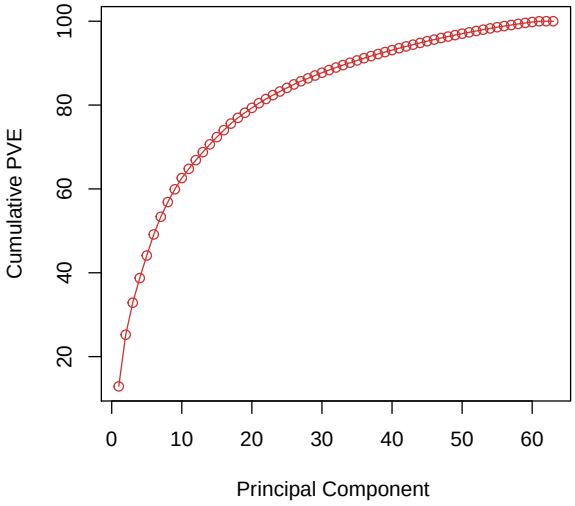


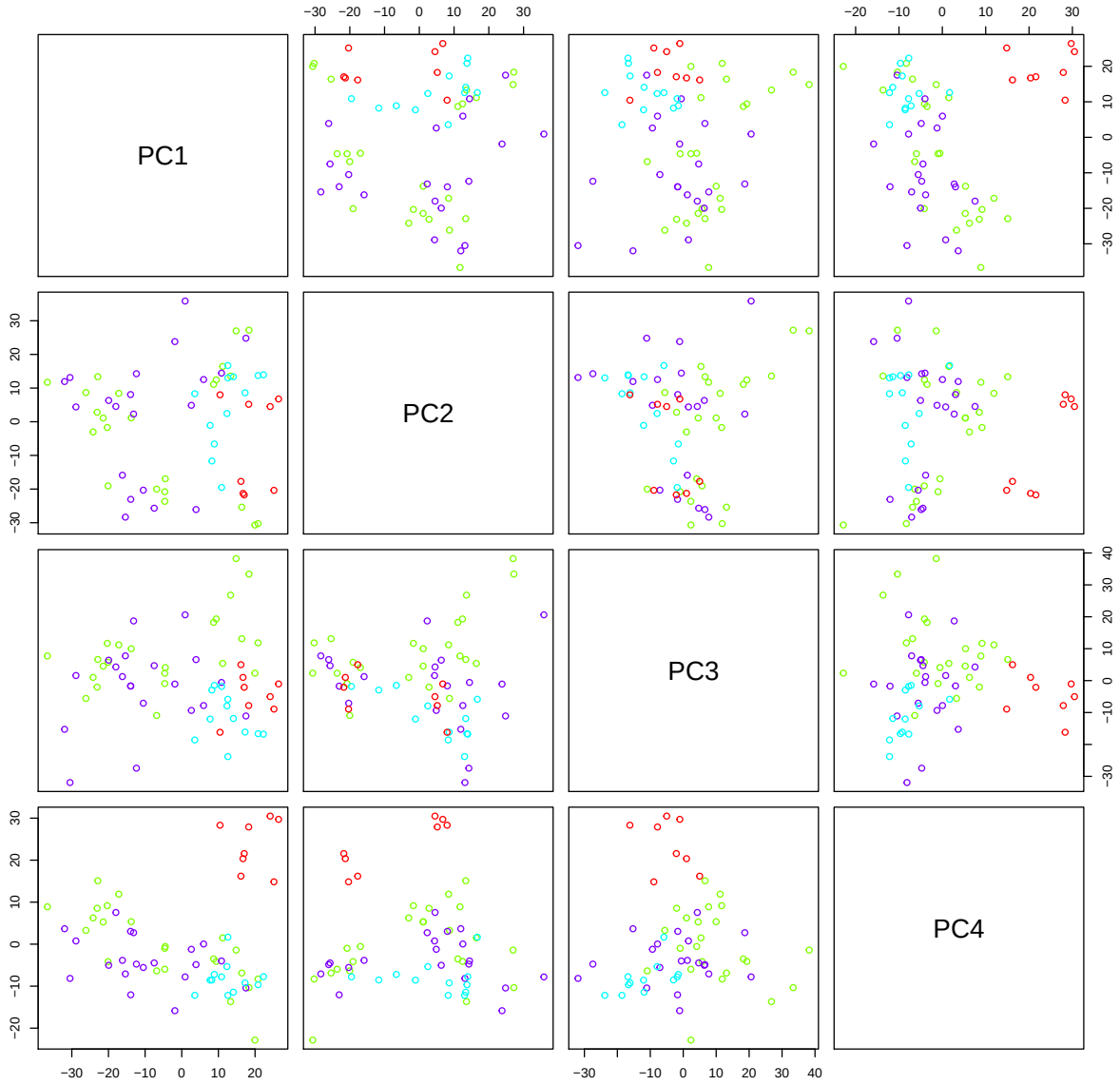
Fig.1.4: cumulative PVE of the principal component



In PVE and cumulative PVE plots, elbows are obsetved after 10 principal components when the percent variation drops off, such that addition prinicipal components do not really add a significant amount of variance.

```
## Pairs plot for 1st 4 Principal Components of training set
pairs(pr.out.train$x[,1:4], col=Cols(Khan$ytrain),
      main="Fig.1.5: Pairs plot on the first 4 Principal Components")
```

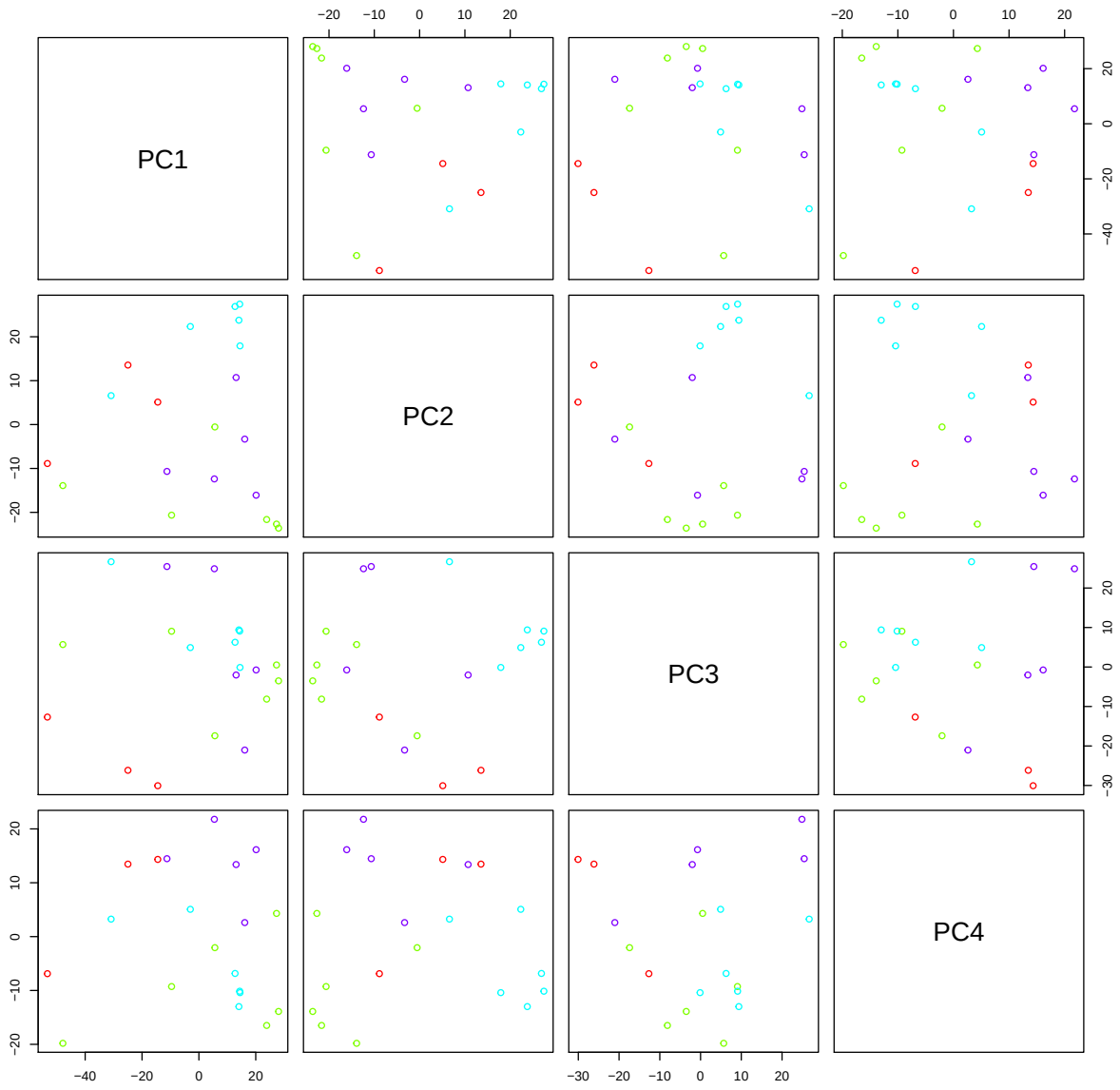
Fig.1.5: Pairs plot on the first 4 Principal Components



We can observe correlations here in pairs plot of 4 principal components of training set. Hence, there will be clusters.

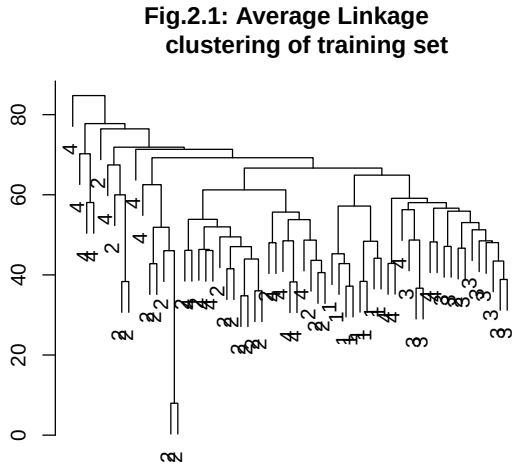
```
## Pairs plot for 1st 4 Principal Components of test set
pairs(pr.out.test$x[,1:4], col=Cols(Khan$ytest),
      main="Fig.1.6: Pairs plot on the first 4 Principal Components")
```

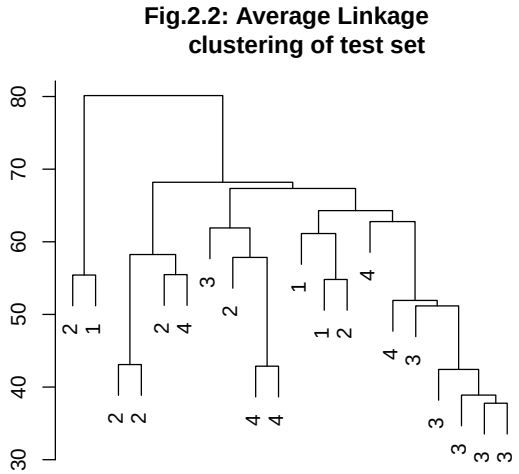
Fig.1.6: Pairs plot on the first 4 Principal Components




```
## Plot hierarchical clustering with average linkage.

par(mfrow=c(1,2))

plot(hc.avg.train, labels=Khan$ytrain, main="Fig.2.1: Average Linkage
     clustering of training set", xlab="", sub="", ylab ="")
```

```
plot(hc.avg.test, labels=Khan$ytest, main="Fig.2.2: Average Linkage
     clustering of test set", xlab="", sub="", ylab="")
```





We can observe groups by hierarchical clustering in both average linkage trees as well as in cluster dendograms on the next page.

```
## Plot hierarchical clustering with a line cutting the tree of 4 groups.
par(mfrow=c(1,2))
plot(hc.out.train, labels = Khan$ytrain, main="Fig.2.3: Cluster Dendogram
     with a line cutting the tree of training set"); # abline(h=89.5, col =
rect.hclust(hc.out.train, k=4, border="red")
plot(hc.out.test, labels = Khan$ytest,main="Fig.2.4: Cluster Dendogram
     with a line cutting the tree of test set"); # abline(h=73.5, col ="red
rect.hclust(hc.out.test, k=4, border="red")
```

Fig.2.3: Cluster Dendogram with a line cutting the tree of training set

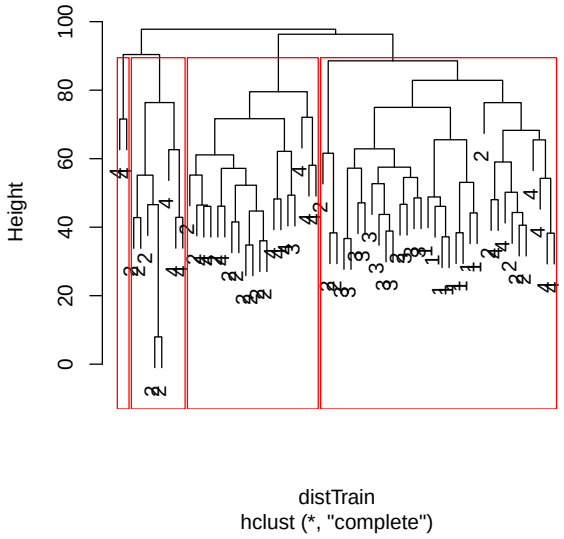
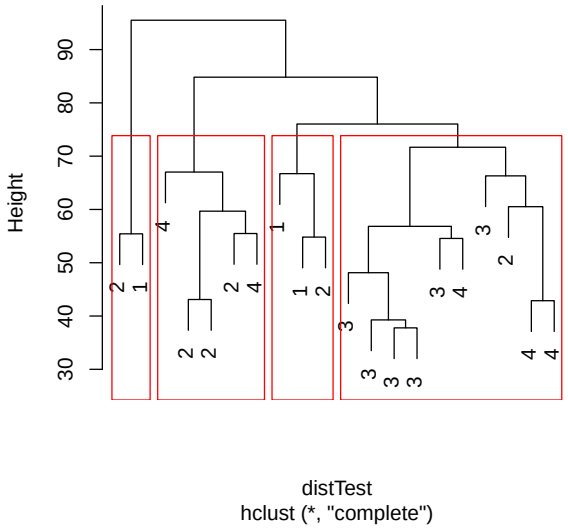


Fig.2.4: Cluster Dendogram with a line cutting the tree of test set




```
# Plot the Khan data observations on all and 1st 4 principal components.

par(mfrow=c(2,2))

plot(km.clusters.train, col=Cols(Khan$ytrain), pch=19,

     main="Fig.3.1: Training set observations on all principal components")

text(km.clusters.train, row.names(km.clusters.train), pos=3)


plot(km.clusters.test, col=Cols(Khan$ytest), pch=19,

     main="Fig.3.2: Test set observations on all principal components")

text(km.clusters.test, row.names(km.clusters.test), pos=3)


plot(km1st4.out.train$cluster, col=Cols(Khan$ytrain), pch=19,

     main="Fig.3.3: Training set observations on 1st 4 principal components"

text(km1st4.out.train$cluster, row.names(km1st4.out.train$cluster), pos=3)
```

```
plot(km1st4.out.test$cluster, col=Cols(Khan$ytest), pch=19,
     main="Fig.3.4: Test set observations on 1st 4 principal components")
text(km1st4.out.test$cluster, row.names(km1st4.out.test$cluster), pos=3)
```

Fig.3.1: Training set observations on all principal component

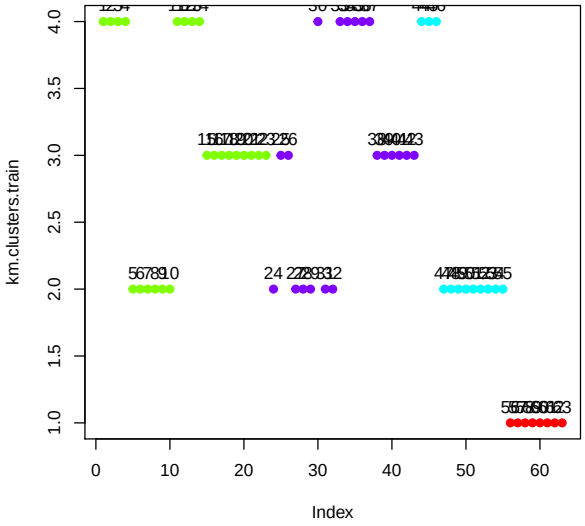


Fig.3.2: Test set observations on all principal components

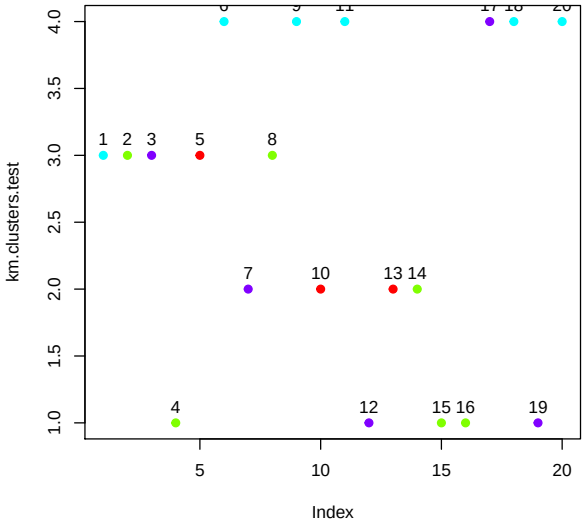


Fig.3.3: Training set observations on 1st 4 principal componer

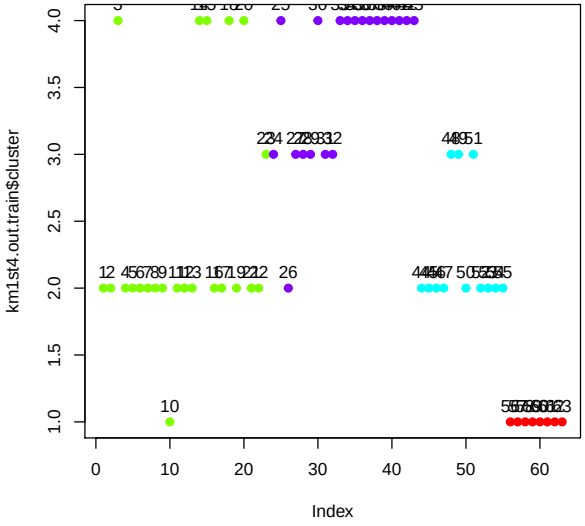
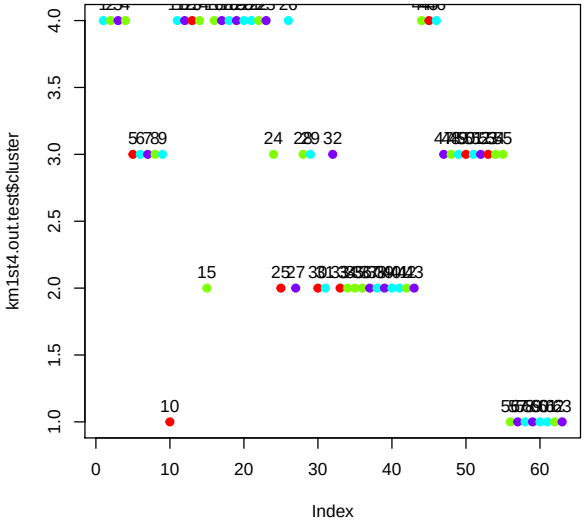
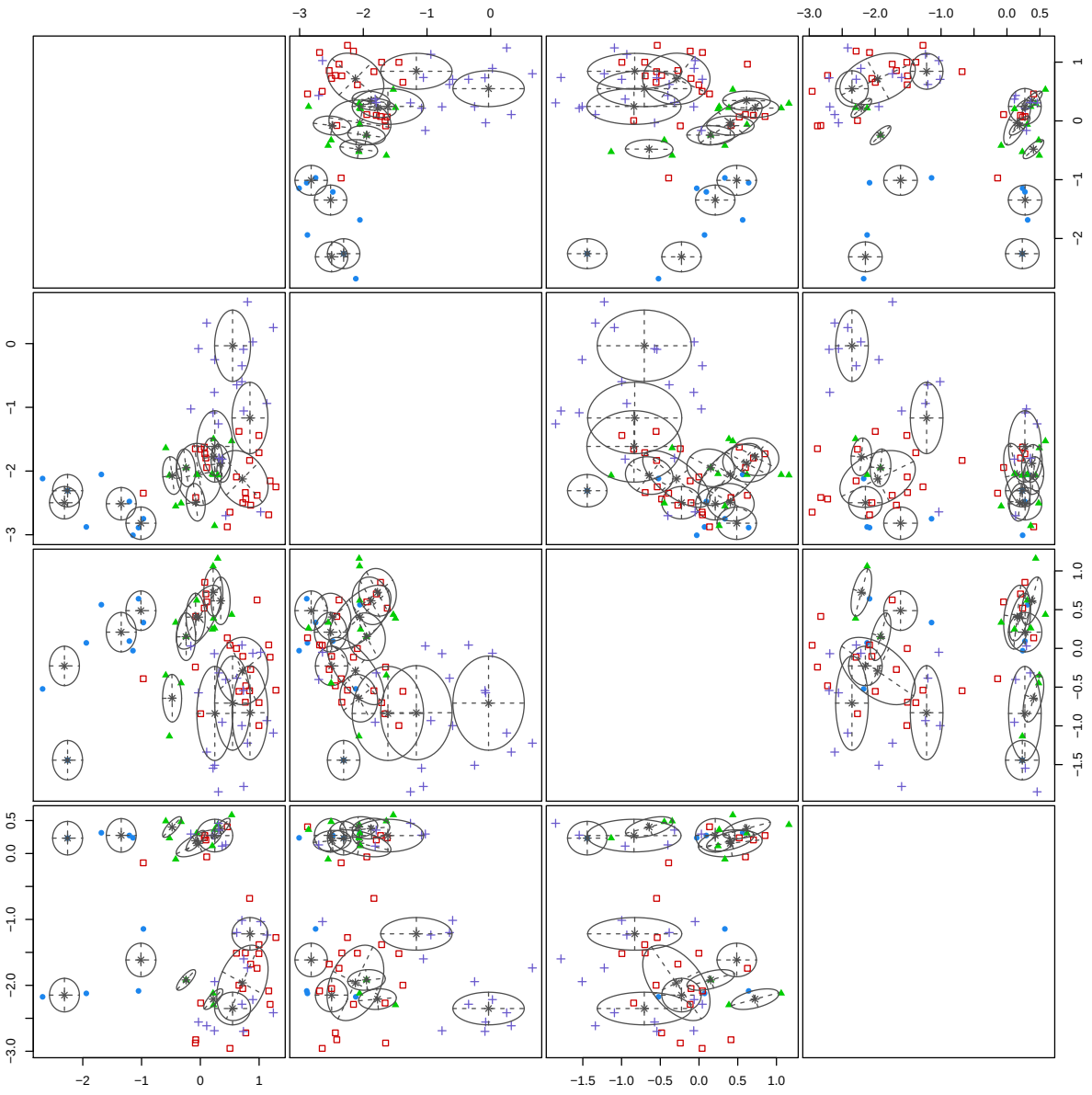


Fig.3.4: Test set observations on 1st 4 principal components



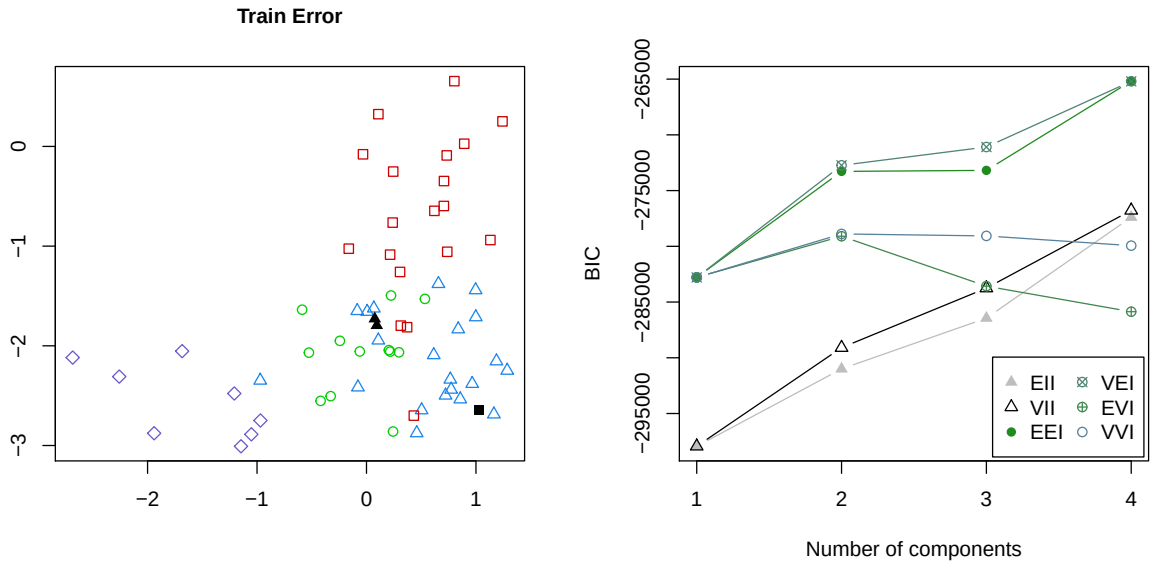
Also we can observe here groups by K-means clustering and represented in different colors for different class level.

```
## Plot Model-based clusters for training

plot(MDA, what = "scatterplot", main="Fig.4.1: Plot of Model-based clusters
```



Model based clustering showed groups by circles, each with different radius w.r.t. euclidian distances.

```
## Plot Model-based clusters for error and BIC

par(mfrow=c(1,2));plot(MDA, what = "error");plot(mc.train, what = "BIC")
```



2.6 Validating Cluster Solutions

Validate performance of Hierarchical Cluster Solution as well as K-means Cluster Solution.

```
## Provide a table of clustering results based on cut of 4 groups.

# For Khan train data

table(hc.cutree.train, Khan$ytrain)

##

## hc.cutree.train  1  2  3  4

##                1  8  9 11  6

##                2  0  5  0  3
```

```
##               3  0  9  1  9
##               4  0  0  0  2

table(hc.cutree.train == Khan$ytrain)

##
## FALSE  TRUE
##    47    16

mean(hc.cutree.train == Khan$ytrain)

## [1] 0.2539683

# For Khan test data
table(hc.cutree.test, Khan$ytest)

##
## hc.cutree.test 1 2 3 4
##              1 0 1 6 3
##              2 1 1 0 0
##              3 0 3 0 2
##              4 2 1 0 0

table(hc.cutree.test == Khan$ytest)

##
## FALSE  TRUE
##    19     1
```

```
mean(hc.cutree.test == Khan$ytest)

## [1] 0.05
```

So using hierarchical clustering, 25.39% of the training observations are correctly classified and 5% of the test observations are correctly classified.

```
# Compare results to the Khan ytrain class levels using the R table command
# For Khan train data
table(km.clusters.train, Khan$ytrain)

##
## km.clusters.train 1 2 3 4
##                 1 8 0 0 0
##                 2 0 6 9 6
##                 3 0 9 0 8
##                 4 0 8 3 6

table(km.clusters.train == Khan$ytrain)

##
## FALSE  TRUE
##    43    20

mean(km.clusters.train == Khan$ytrain)

## [1] 0.3174603
```

```
# For Khan test data
table(km.clusters.test, Khan$ytest)

##
## km.clusters.test 1 2 3 4
##                1 0 3 0 2
##                2 2 1 0 1
##                3 1 2 1 1
##                4 0 0 5 1

table(km.clusters.test == Khan$ytest)

##
## FALSE  TRUE
##    17     3

mean(km.clusters.test == Khan$ytest)

## [1] 0.15
```

So using K-means clustering, 31.75% of the training observations are correctly classified and 15% of the test observations are correctly classified.

3 Results and Discussions

Comparing performances of K-means and Hierarchical clustering

```
table(km.clusters.train, hc.cutree.train)
```

```
##                  hc.cutree.train
## km.clusters.train  1  2  3  4
##                 1  8  0  0  0
##                 2  9  8  4  0
##                 3  0  0 15  2
##                 4 17  0  0  0
```

```
table(km.clusters.train == hc.cutree.train)
```

```
##
## FALSE  TRUE
##    32    31
```

```
mean(km.clusters.train == hc.cutree.train)
```

```
## [1] 0.4920635
```

```
table(km.clusters.test, hc.cutree.test)
```

```
##                 hc.cutree.test
## km.clusters.test 1 2 3 4
```

```
##                 1 1 0 4 0

##                 2 0 0 1 3

##                 3 3 2 0 0

##                 4 6 0 0 0
```

```
table(km.clusters.test == hc.cutree.test)
```

```
##

## FALSE  TRUE

##    19     1
```

```
mean(km.clusters.test == hc.cutree.test)
```

```
## [1] 0.05
```

We see that the four clusters obtained using these 2 mthods are somewhat different. K-means clustering contains only 49.2% portion of training observations and just 5% of test observations assigned to Hierarchical clustering.

4 Conclusions and Brief Description

- It seems from the comparison of their performances that K-means clustering could do better classification than Hierarchical clustering. Athough the classification accuracy wasn't so great in either cases due to the moderate size of dataset, the clusters were successfully obtained in both cases. And this could be improved with bigger sizes of datasets or more classes.

- One down side of k-means is that, if you rearrange your data, it's very possible that you'll get a different solution every time you change the ordering of your data. This makes the procedure somewhat unattractive if you don't know exactly how many clusters you should have in the first place.

- **How the conclusions of our analyses could be tested using biochemical or genetic techniques?**

  - Monitoring global gene-expression levels by cDNA microarrays provides an additional tool for elucidating tumor biology as well as the potential for molecular diagnostic classification of cancer. Currently, classification and clustering tools using gene-expression data have not been rigorously tested for diagnostic classification of more than two categories.

  - Other approaches that share the parametric nature of artificial neural networks and have been utilized to classify gene-expression profiles include Support Vector Machines. Thus far, these other methods have not been fully explored to extract the genes or features that are most important for the classification performance and which also will be of interest to cancer biologists.

  - Our method identifies genes related to tumor histogenesis, but includes genes that may not normally be expressed in the corresponding mature tissue.

References

- Games, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). An Introduction to Statistical Learning with applications in R, www.StatLearning.com, Springer-Verlag, New York.
- Anderberg, M. R. (2014). Cluster analysis for applications: probability and mathematical statistics: a series of monographs and textbooks (Vol. 19). Academic press.
- Hair, J. F., Black, W. C., Babin, B. J., Anderson, R. E., and Tatham, R. L. (2006). Multivariate data analysis (Vol. 6). Upper Saddle River, NJ: Pearson Prentice Hall.
- Suzuki, R., and Shimodaira, H. (2006). Pvclust: an R package for assessing the uncertainty in hierarchical clustering. Bioinformatics, 22(12), 1540-1542.
- Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(7), 881-892.

Sources

- *This data were originally reported in:* Khan J, Wei J, Ringner M, Saal L, Ladanyi M, Westermann F, Berthold F, Schwab M, Antonescu C, Peterson C, and Meltzer P. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nature Medicine, v.7, pp.673-679, 2001.
- *The data were also used in:* Tibshirani RJ, Hastie T, Narasimhan B, and G. Chu. Diagnosis of Multiple Cancer Types by Shrunken Centroids of Gene Expression. Proceedings of the National Academy of Sciences of the United States of America, v.99(10), pp.6567-6572, May 14, 2002.