Gene expression cancer RNA-Seq

STAT 437 Project 1 Data Set Analysis

Shawn Petersen(011691731), Laura Pelayo(011750956)

Washington State University

April 05, 2022

Abstract

This research paper studies cancer prediction from gene expression data using clustering methods and classification methods

1. Introduction

In this analysis, various classification methods are applied to cancer genetic data to determine the accuracy of each method with a various number of inputs. While investigating the dataset, I found this set is used to classify cancer types based on gene expressions which has been used as a tool for scientists to advance cancer research.

The data used to test these methods is the Gene Expression Cancer RNA-Seq Data Set from UC Irvine Machine learning, https://archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq# (https://archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq#)

The data analysis for this project uses the following algorithms to build models and analyze the results:

- K-Means Clustering
- Hierarchical Clustering
- Quadratic Discriminant Analysis
- K-Nearest Neighbors

The clustering methods, k-means and hierarchical, will be applied to determine how well the sample classes can be grouped into the models. For these experiments, all five cancer classes will be present.

The classification algorithms, quadratic discriminant analysis, and k-nearest neighbors will be used to develop models that can be trained to classify the type of cancer a patient has from their gene expression testing. For the classification algorithms, two cancer classes, `BRCA` and `LUAD`, will be used, which will encompass a binary classification problem.

Data analysis techniques:

- K-Means algorithm will be performed on two subsets of the total data, one that utilizes 50 genes, and another that utilizes 250 genes
- Hierarchical clustering will be run on the 250 gene subset
- Binary classification tasks that utilize quadratic discriminant analysis will be performed on a subset that uses 3 genes and 100 genes
- K-nearest neighbor's model will be run on the 100 gene subset after splitting the data in training and test sets

The goal of utilizing multiple feature sizes is to understand how the number of features impacts model test accuracy. When the models are trained, the accuracy of the models will be determined by the use of 2x2 prediction tables, ROC curves, and model accuracies calculations.

2. Data Analysis Approach

The data set contains 801 samples from patients with five cancer types are in the gene samples: `BRCA`, `KIRC`, `COAD`, `LUAD`, and `PRAD`. For each patient, they each have 20,531 gene expressions values. Using the information we have learned the past several weeks in STAT 437 class, I applied the four algorithms outlined in the introduction to the cancer data set. The data set was sampled with different number of genes to observe the results and if a small or larger sampled affected the results.

3. Data Cleaning

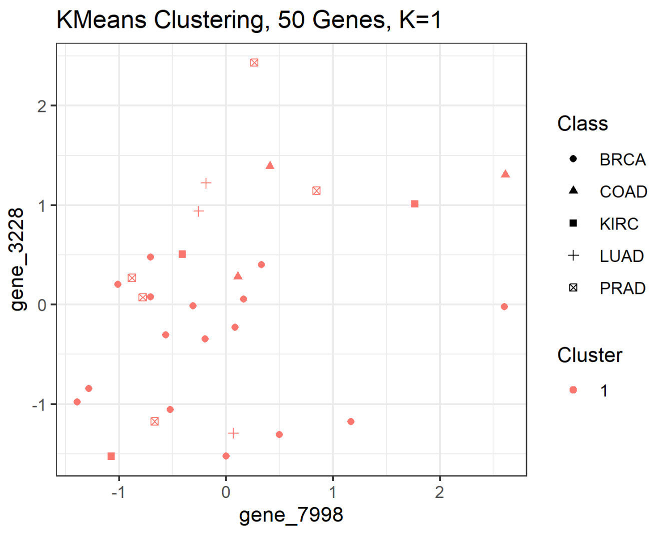
For this project, genes that contained less than 300 values of 0’s were selected for the data set. From the remaining set, 1000 genes were randomly selected for a more manageable data set to analyze with the various techniques. 30 samples were selected at random along with their cancer diagnosis labels. The data was also scaled to minimize the effects that relatively large values have on clustering metrics. One of them is Euclidean distance which calculates how dissimilar are two values in a cluster. Then two subsets of the cancer data were taken, one with 50 genes and the other with 250 genes, to compare if the number of features would improve the clustering results.

4. EDA (Clustering)

4.1 K-Means Clustering

To determine the best k value, two methods were incorporated. The first method was to compute the k-value using the gap statistic method which is implemented using the clusGap function. This technique uses the output of the clustering algorithm by comparing the change in within-cluster dispersion with the expected null distribution. Using this method to determine the best fit k for both the 50-gene experiment and 250-gene experiment, yielded a k=1.

Using a k=1, as seen in **Figure 1** and **Figure 2** below, all gene samples were grouped into the same cluster for both the 50 and 250 gene samples with no clear groupings. The 2 genes selected (gene_7998, gene_322) were from the first 2 columns of the data set.



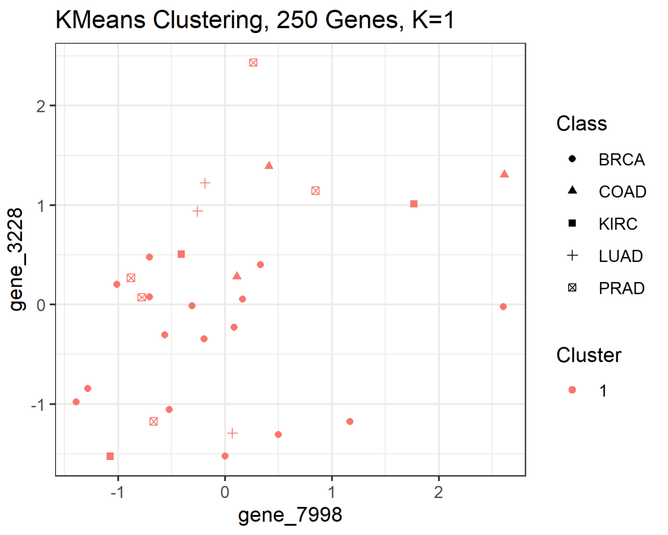
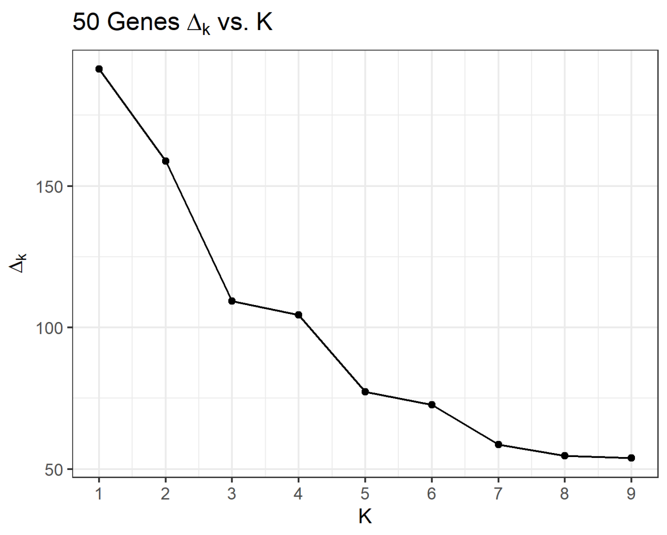


Figure 1 (left) - **Figure 2** (right)

The elbow method was used to estimate k. With this method, the k-means algorithm was with k-values from 1 to 10. With each iteration of k, the total within-cluster sum of squares was calculated and plotted. Seen below in **Figure 3** and **Figure 4**.



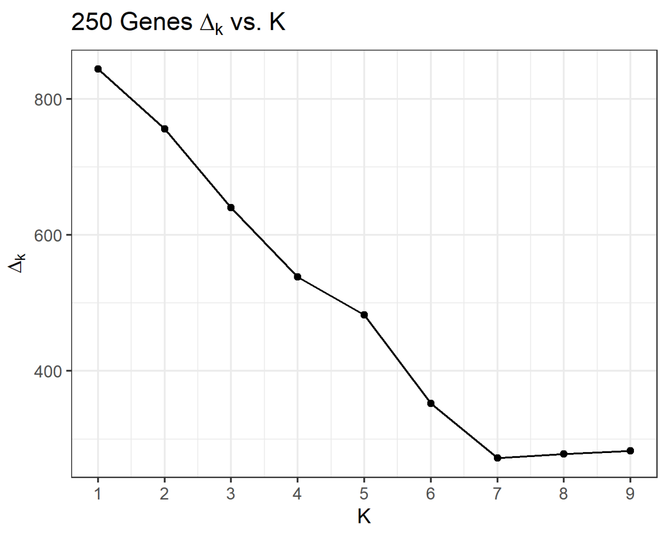
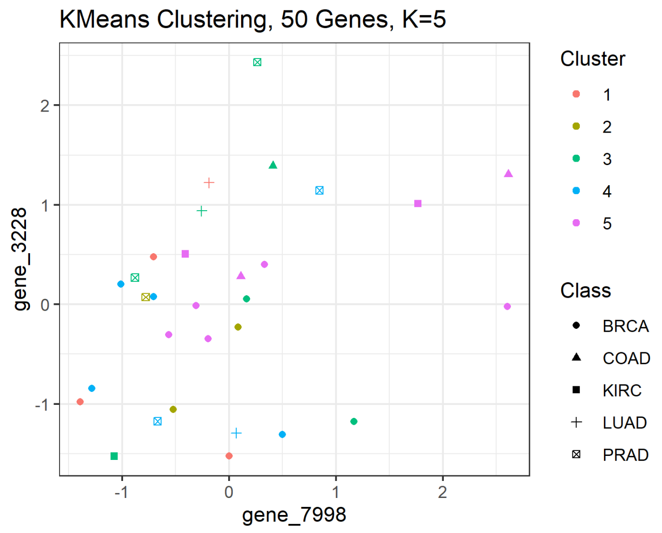


Figure 3 (left) - **Figure 4** (right)

Both the 50-gene and 250-gene plots display the majority of the benefits as k is increased are maximized when k=5. The k-means algorithm was then re-run with the new estimates for k=5. The results on both subsets(50 & 250 genes) are seen below in **Figure 5** and **Figure 6**.



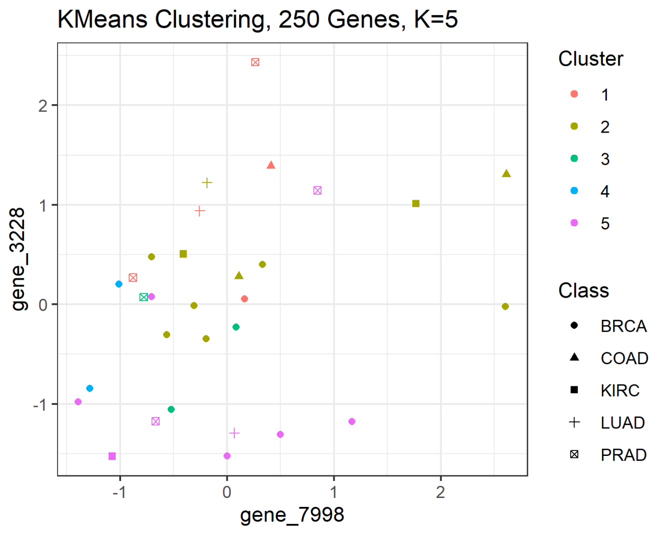


Figure 5 (left) - **Figure 6** (right)

The 50-gene and 250-gene experiments with K=5 had the same classification error, 14/30=43.66% of samples present in clusters that did not match the in-cluster class majority, as seen in **Table 1** and **Table 2** below.

	X1	X2	X3	X4	X5
BRCA	3	2	2	4	5
COAD	0	0	1	0	2
KIRC	0	0	1	0	2
LUAD	1	0	1	1	0
PRAD	0	1	2	2	0

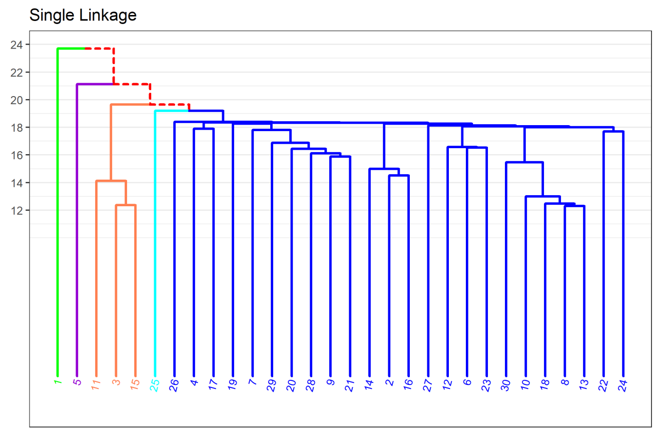
Table 1: Kmeans 50 genes K=5 classification

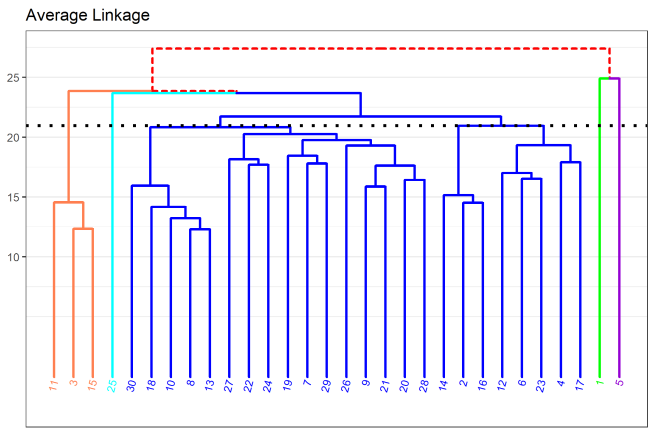
	X1	X2	X3	X4	X5
BRCA	1	6	2	2	5
COAD	1	2	0	0	0
KIRC	0	2	0	0	1
LUAD	1	1	0	0	1
PRAD	2	0	1	0	2

Table 2: Kmeans 250 genes K=5 classification

4.2 Hierarchical Clustering

Hierarchical Clustering was calculated for the 250-gene subset with single, average, and complete linkages. The cut height calculation to obtain 5 clusters was performed for the average linkage experiment. The cut height to obtain 5 clusters with average linkage was calculated to be **20.96**. The complete linkage had the best performance with 16.6%(5/30) of the samples located in a cluster not matching the within-cluster majority.





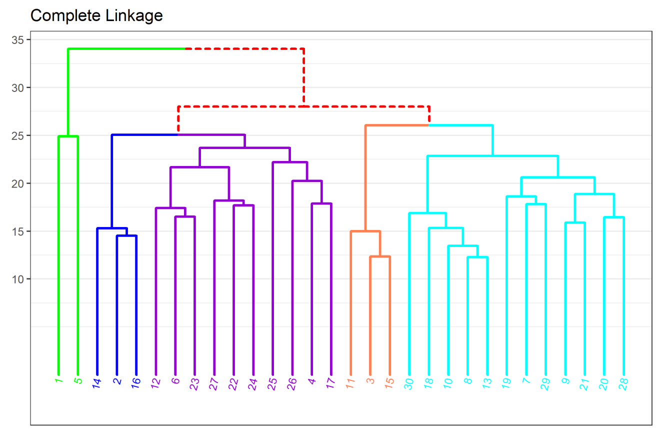


Figure 7 (left) - **Figure 8** (bottom-center) - **Figure 9** (right)

5. EDA (Classification)

5.1 Quadratic Discriminant Analysis

The data was cleaned and a subset was created for classification analysis as follows:

- Gene expressions that contained less than 300 values of 0 were part of the final data set
- 1000 genes were randomly sampled
- All samples with a class label of LUAD or BRCA were extracted along with their corresponding class labels, 441 observations

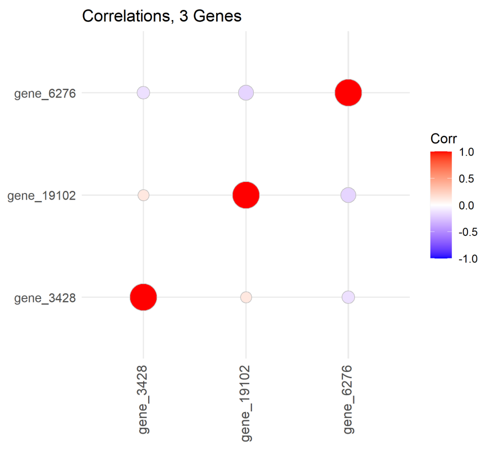
QDA was applied to the two subsets of the data, one contained 3 genes and the other containing 100 genes.

For the 3-gene experiment, 60% of the samples were selected at random for training the model with the other 40% used for testing

For the 100-gene experiment, 75% was used for training with 25% used for testing

For each model, two procedures were used, correlation and Gaussian Mixture, as part of the linear and quadratic discriminant analysis, which, requires that each observation follows a Gaussian distribution.

1. First, correlations between each pair of variables were calculated to find pairs that correlated above 0.9. Any gene pairs which correlated above this value were to be removed from the data set. Some gene pairs had a good correlation but no samples exceeded the >0.9 limits. The correlation plots for both the 3-gene experiment and the 100-gene experiment are seen below in **Figure 10** and **Figure 11**



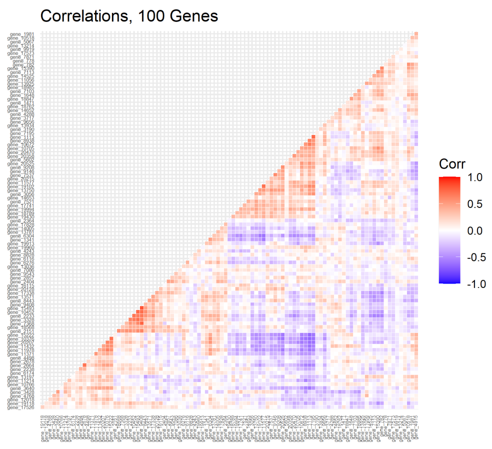
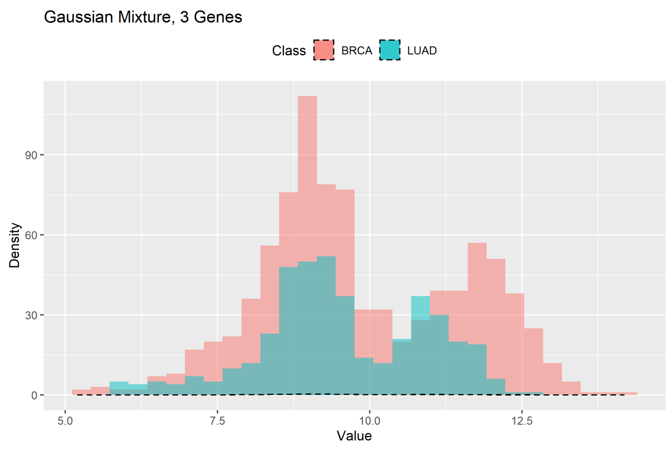


Figure 10 (left) - **Figure 11** (right)

2. Second, is the Gaussian Mixture Model assumption, which determines whether each set of class observations is normally distributed. To visualize this, density plots were created for each set of class observations. During my investigation of Gaussian Mixture models they are used for representing Normally Distributed subpopulations within an overall population.

The density plots for both classes, for both the 3-gene and 100-gene experiments, are seen below in **Figure 12** and **Figure 13**



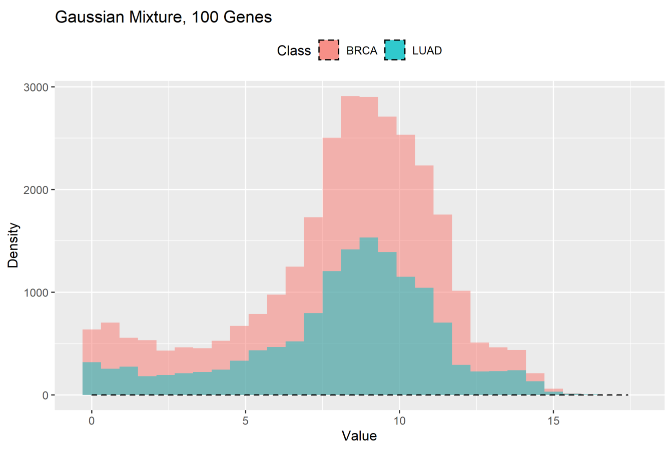


Figure 12 (left) - **Figure 13** (right)

Analysis:

1. 3-genes

- The density plot appears to be normally distributed data for both classes
- Have feature mixtures in each cancer class are bi-modal and overlapped

2. 100-genes

- The density plot does not have normally distributed data(left skewed)
- Both cancer classes exhibit highly irregular distributions, and appear to be unimodal with a significant amount of overlap between the classes

3. Final outcome

- The distributions of the class observations for the 100-gene violates the Gaussian Mixture Model assumption while the 3-gene experiment does not.

The model test accuracy showed that the ROC experiment had a 23.16% error rate, with an AUC of 0.786. The model built for the 100-gene subset performed slightly worse with an error rate of 24.32% and an AUC of 0.907.

Below are the QDA 2x2 classification error tables, **Table 3** and **Table 4**.

- 3-genes classification error = 32.38% (41/127)
- 100-genes classification error = 32.14% (27/84)

Between the 3 gene and 100 gene experiments, both have virtually the same classification errors.

	BRCA	**LUAD**
BRCA	110	17
LUAD	24	26

Table 3: QDA 3-genes classification error

	BRCA	**LUAD**
BRCA	82	0
LUAD	27	2

Table 4: QDA 100-genes classification error

5.2 K-Nearest Neighbors

In the K-nearest neighbor analysis for classification, the most accurate result is when k = 3. In **Figure 12** the Gaussian distribution when k = 3 overlaps less than in **Figure 13**, when k = 100. Both do not follow an exact Gaussian model but when k = 100, the distributions are identical, making it difficult for the k-nearest neighbor to accurately classify the genes with the cancer type.

Finally, our ROC curve when k = 3 reinforces its accuracy compared to the ROC curve when k = 100. When k = 3 we get an elbow shape curve, indicating the connection between sensitivity and specificity, which is more efficient. In **Figure 15**, when k = 100, the ROC curve is ineffective, just reinforcing what was found in the Gaussian distribution when k = 100.

Reviewing the classification error results, this classification method performed the best with an error rate of 2.7% (3/108) as seen below in **Table 5**.

	BRCA	**LUAD**
BRCA	81	2
LUAD	1	27

Table 5: KNN Predictions

6. Conclusion

Our analysis indicates that using more features does not necessarily give more accurate clustering or classification results. The clustering methods we applied to our data were k-means and hierarchical. K-means is redundant when we get k = 1 from our GAP, for both when our sample size is 50 and 250. Even when k = 5, it is hard to say both graphs make an impact in clustering analysis. K-means clustering was not a useful approach for clustering this data set. Hierarchical clustering does the best job overall for classification, specifically when we use our complete method to model our dendrogram. (Not sure what else to say about the dendrogram)

The classification methods we applied to our data were Quadratic Discriminant Analysis (QDA) and K-Nearest Neighbors (K-NN). With classification, more features did not give a more accurate result. QDA performed best with only 3 genes as features. The density plot for the 3-gene experiment was promising because there was a bimodal distribution and K-NN was more accurately able to classify. On the other hand, when QDA was performed with 100 genes as features, the distribution fails the Gaussian Mixture Model. There is simply too much of an exact overlap between the class observations. The distributions being almost identical, make it impossible for K-NN to accurately classify the genes with the cancer type.

We may think more features to describe a data, will render more accurate clustering and classification, but it may be that too many features, create too much “noise” in our data.

Final model analysis

- K-Nearest Neighbors: 2.7% Error
- K-Means Clustering: 16.66% Error
- Hierarchical Clustering: 16.66% Error
- Quadratic Discriminant Analysis: 23.16% Error

7. Appendix

7.1 Import Libraries

The first chunk of code imports the required libraries

```
library(dplyr)
library(tidyverse)
library(viridis)
library(plotly)
library(ggridges)
library(tibble)
library(cluster)
library(ggdendro)
library(ggcorrplot)
library(latex2exp)
```

7.2 Data File load

The first step of the analysis is to load the data and label files into R for future processing.

```
# Load data sets
df_data = read.csv("data/data.csv")
df_labels = read.csv("data/labels.csv")
```

7.3 Task A - Clustering

K-Means Clustering and Hierarchical Clustering techniques are applied to the data. This section prepares the data for use with the k-means algorithm will later be executed

```
set.seed(123)

# Filter columns with <=300 zero's
gexp2 = df_data[colSums(df_data == 0) <= 300]
head(gexp2)
```

	X <chr>	gene_1 <dbl>	gene_2 <dbl>	gene_3 <dbl>	gene_4 <dbl>	gene_6 <dbl>	gene_11 <dbl>	gene_12 <dbl>	gene_19 <dbl>	▸
1	sample_0	2.0172093	3.265527	5.478487	10.431999	7.175175	1.3342822	2.015391	5.619994	
2	sample_1	0.5927321	1.588421	7.586157	9.623011	6.816049	0.5878450	2.466601	11.055208	
3	sample_2	3.5117590	4.327199	6.881787	9.870730	6.972130	0.4525954	1.981122	8.210248	
4	sample_3	3.6636179	4.507649	6.659068	10.196184	7.843375	0.4348817	2.874246	8.306317	

	X <chr>	gene_1 <dbl>	gene_2 <dbl>	gene_3 <dbl>	gene_4 <dbl>	gene_6 <dbl>	gene_11 <dbl>	gene_12 <dbl>	gene_19 <dbl>
5	sample_4	2.6557411	2.821547	6.539454	9.738265	6.566967	1.2758414	2.141204	10.149150
6	sample_5	3.4678533	3.581918	6.620243	9.706829	7.758510	0.5154097	2.516797	6.842765

6 rows | 1-10 of 17255 columns

```r
sample.column.list = gexp2[,1]

# Random sample 1000 genes(columns)
gexp3 = gexp2[sample(ncol(gexp2), size = 1000), drop = F]

gexp3 = cbind(sample.column.list, gexp3)
colnames(gexp3)[1] = "X"

# Random sample 30 items from labels data set
labels1 = sample_n(df_labels, 30, replace = F)

# Select 30 samples from 'gexp3' that are in 'labels1'
gexpProj1 = gexp3 %>%
      filter(X %in% labels1$X)

head(gexpProj1)
```

	X <chr>	gene_3428 <dbl>	gene_2104 <dbl>	gene_3871 <dbl>	gene_13882 <dbl>	gene_5486 <dbl>	gene_7794 <dbl>	gene_19323 <dbl>	gene_318 <db
1	sample_29	11.022881	6.318132	6.168137	8.899420	13.48248	9.533318	10.206892	6.81867
2	sample_57	10.067797	3.690048	6.055000	9.998576	12.44193	9.805205	10.601316	6.19183
3	sample_68	6.908969	0.000000	6.155846	9.922098	12.79846	10.289304	9.885869	5.55043
4	sample_69	9.477809	1.236462	7.040575	10.262565	12.69732	9.787757	10.047069	6.02669
5	sample_74	10.743850	1.220206	5.933252	9.335272	12.78203	10.951081	11.106210	5.82911
6	sample_75	8.944572	3.322130	5.759278	12.040666	12.97517	9.794286	9.853389	6.91012

6 rows | 1-10 of 1002 columns

```r
# Scale columns to have a std=1 and mean=0
stdgexpProj1 = gexpProj1 %>% mutate_at(c(2:1001), ~(scale(.) %>% as.vector))
head(stdgexpProj1)
```

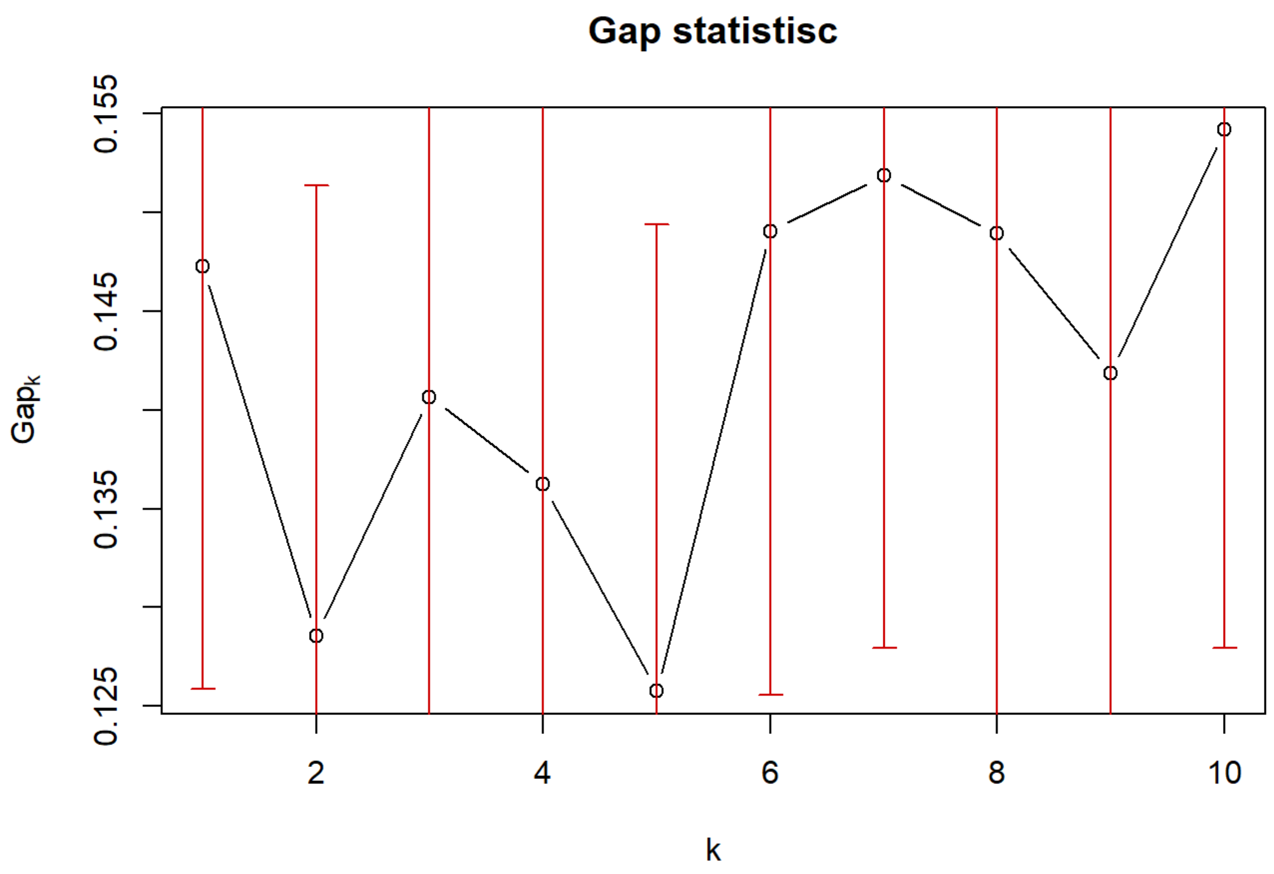
	X <chr>	gene_3428 <dbl>	gene_2104 <dbl>	gene_3871 <dbl>	gene_13882 <dbl>	gene_5486 <dbl>	gene_7794 <dbl>	gene_1 <
1	sample_29	1.6953210	1.76517717	0.436996527	-1.03046000	1.32458412	-1.0945103	0.06156

	X	gene_3428	gene_2104	gene_3871	gene_13882	gene_5486	gene_7794	gene_19
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<
2	sample_57	1.0819668	0.29748430	0.314600843	-0.08975385	-0.33911743	-0.5682849	0.99534
3	sample_68	-0.9466311	-1.76327830	0.423700454	-0.15520719	0.23092280	0.3686661	-0.69844
4	sample_69	0.7030765	-1.07275762	1.380838387	0.13617905	0.06920938	-0.6020549	-0.31681
5	sample_74	1.5161273	-1.08183597	0.182887653	-0.65743896	0.20466324	1.6495039	2.19066
6	sample_75	0.3606317	0.09201487	-0.005324217	1.65795695	0.51345808	-0.5894191	-0.77534

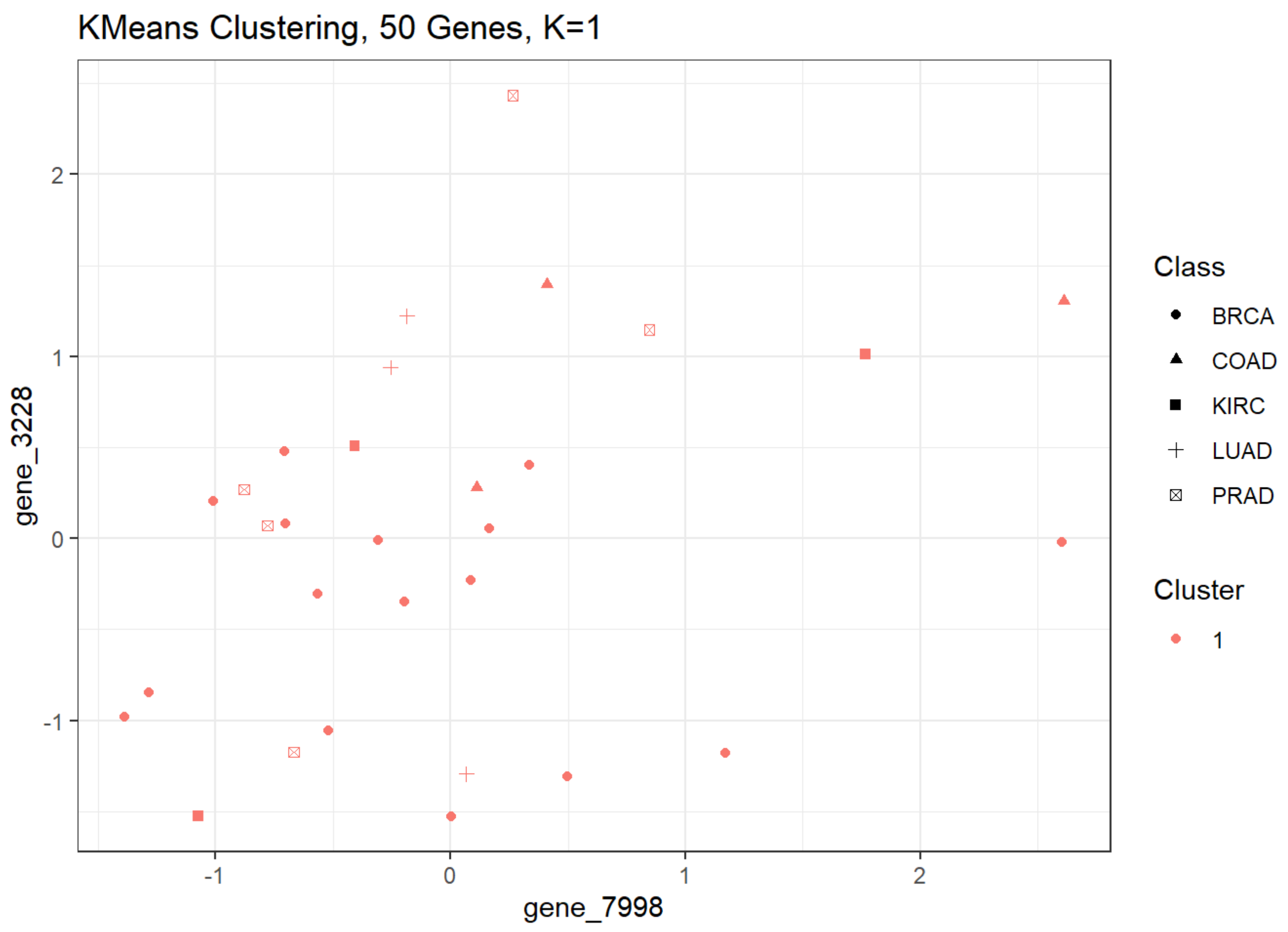
6 rows | 1-9 of 1002 columns

7.4 Task A2 (Part 1 of Task A2)

```r
set.seed((123))

# Random sample 50 genes from data set
km.gexProj1.50 = stdgexpProj1[sample(ncol(stdgexpProj1), size = 50), drop = FALSE]
stdgexProj1.column.list = stdgexpProj1[,1]



km.gexProj1.50 = cbind(stdgexProj1.column.list, km.gexProj1.50)
colnames(km.gexProj1.50)[1] = "X"

#remove NA's
km.gexProj1.50 = na.omit(km.gexProj1.50)

###GAP****STATISTICS****###
#set.seed(123)
gap.estimate.50 = clusGap(km.gexProj1.50[,2:51], FUNcluster=kmeans, K.max=10, B=200,
         iter.max=100)
k.50 =  maxSE(gap.estimate.50$Tab[,"gap"], gap.estimate.50$Tab[,"SE.sim"], method="T
         ibs2001SEmax")
plot(gap.estimate.50, main = "Gap statistisc")
```



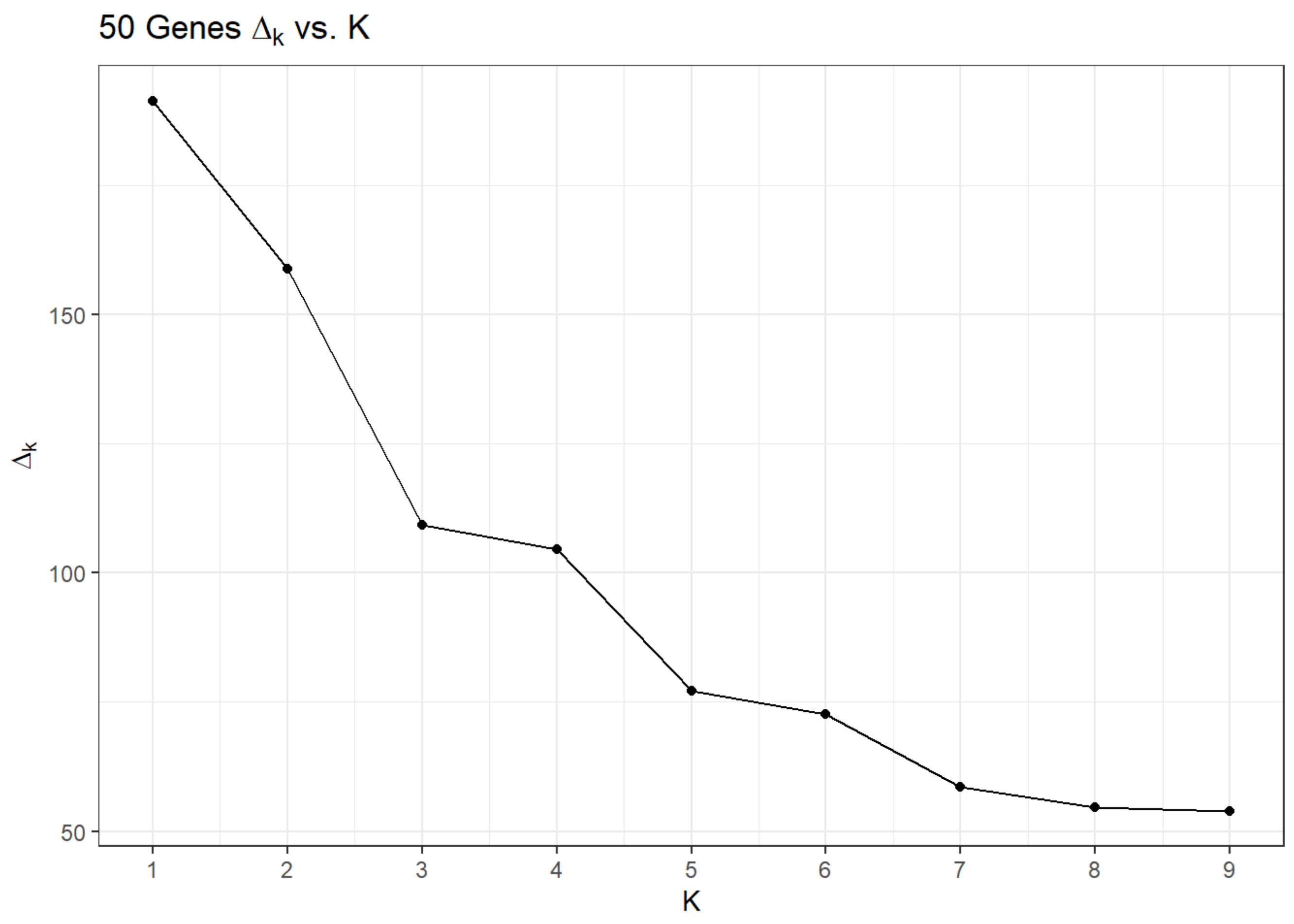
```
###****KMEANS****###
#set.seed(123)
km.50.k1 = kmeans(km.gexProj1.50[2:51], iter.max = 100, centers = k.50, nstart = 25,
         algorithm=c("Hartigan-Wong"))

#Creating the dataframe for storing clustering results
km.50.df = data.frame(km.gexProj1.50)
km.50.df$Class = as.factor(labels1$Class)
km.50.df$Cluster = as.factor(km.50.k1$cluster)

# Graphing km results 50 genes, K=1
km.plot.50.k1 = ggplot(km.50.df, aes(gene_7998, gene_3228)) +
   geom_point(aes(shape=Class, color=Cluster)) +
   theme_bw() +
   ggtitle("KMeans Clustering, 50 Genes, K=1")

km.plot.50.k1
```



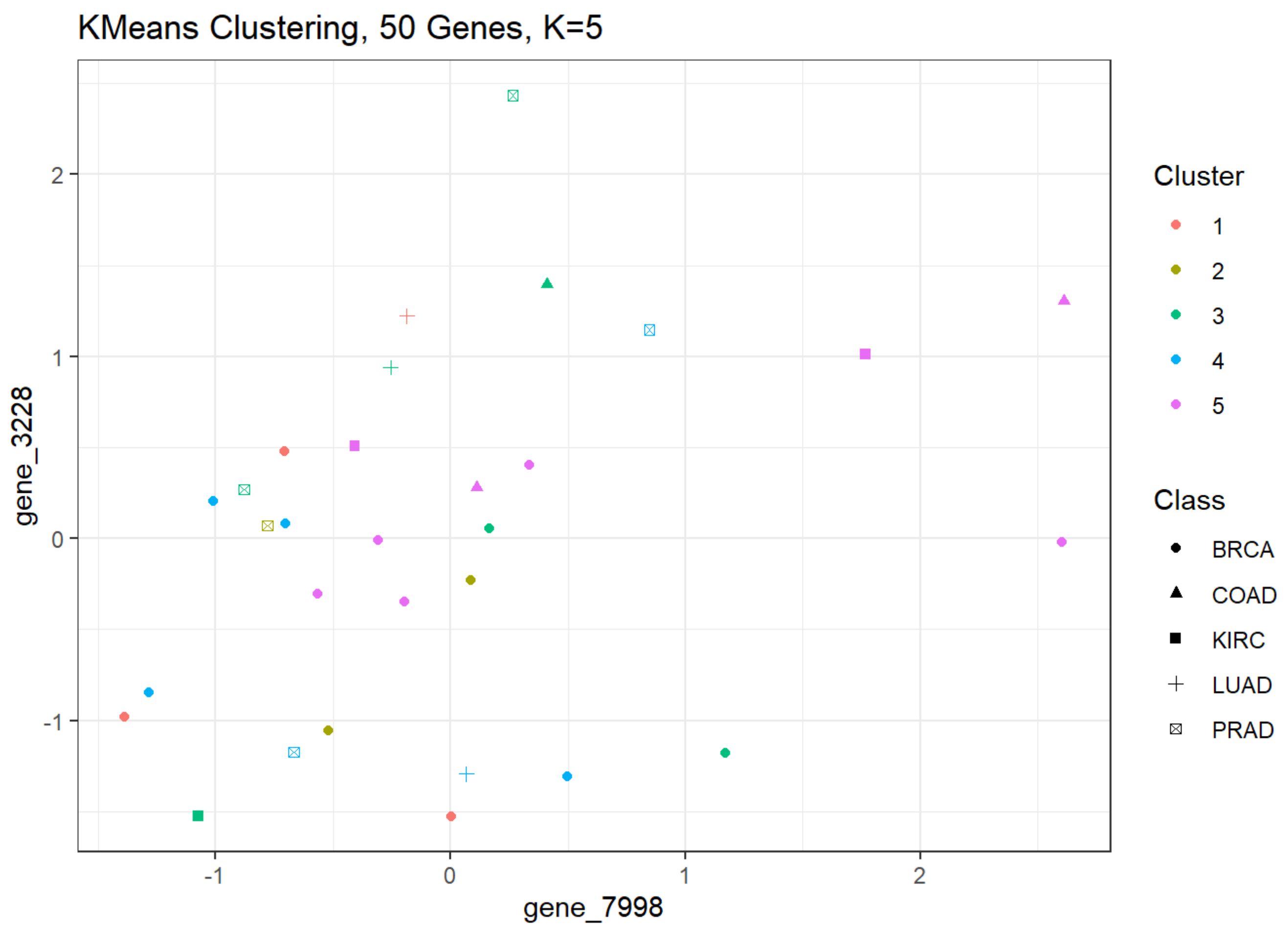
```
ggsave("figures/a2.graph1.png", device="png", width=5, height=4)

# Kmeans prediction table
kmeans.pred.table = table(labels1$Class, km.50.k1$cluster)

# Write kmeans table to .csv file
kmeans.pred.table = as.data.frame.matrix(kmeans.pred.table)
write.csv(kmeans.pred.table, file="figures/kmeans-50gene-k1-pred.csv")
```

7.5 Task A2 (Part 2 of Task A2)

```r
# Set the seed
set.seed(123)

# Running k-means with k=1:10 and recording sum of squares
sum.square.list = list()

for (k in 1:10)
{
   set.seed(123)
   km.loop.50 = kmeans(km.gexProj1.50[2:51], centers=k, iter.max=100, nstart=25, alg
         orithm=c("Hartigan-Wong"))
   sum.square.list[k] = km.loop.50$tot.withinss
}


#Calculating delta-ks
unlist.sums = unlist(sum.square.list)
k.delta = unlist.sums[1:9] - unlist.sums[2:10]
k.values = 1:9
k.df.50 = data.frame(k.values, k.delta)


# Create kmeans cluster plot with 1 center
km.loop.50.plot = ggplot(k.df.50, aes(k.values, k.delta)) +
   geom_line() +
   geom_point() +
   scale_x_continuous(breaks=1:10) +
   theme_bw() +
   labs(title=TeX("50 Genes $\\Delta_k$ vs. K"), x="K", y=TeX("$\\Delta_k$"))

km.loop.50.plot
```



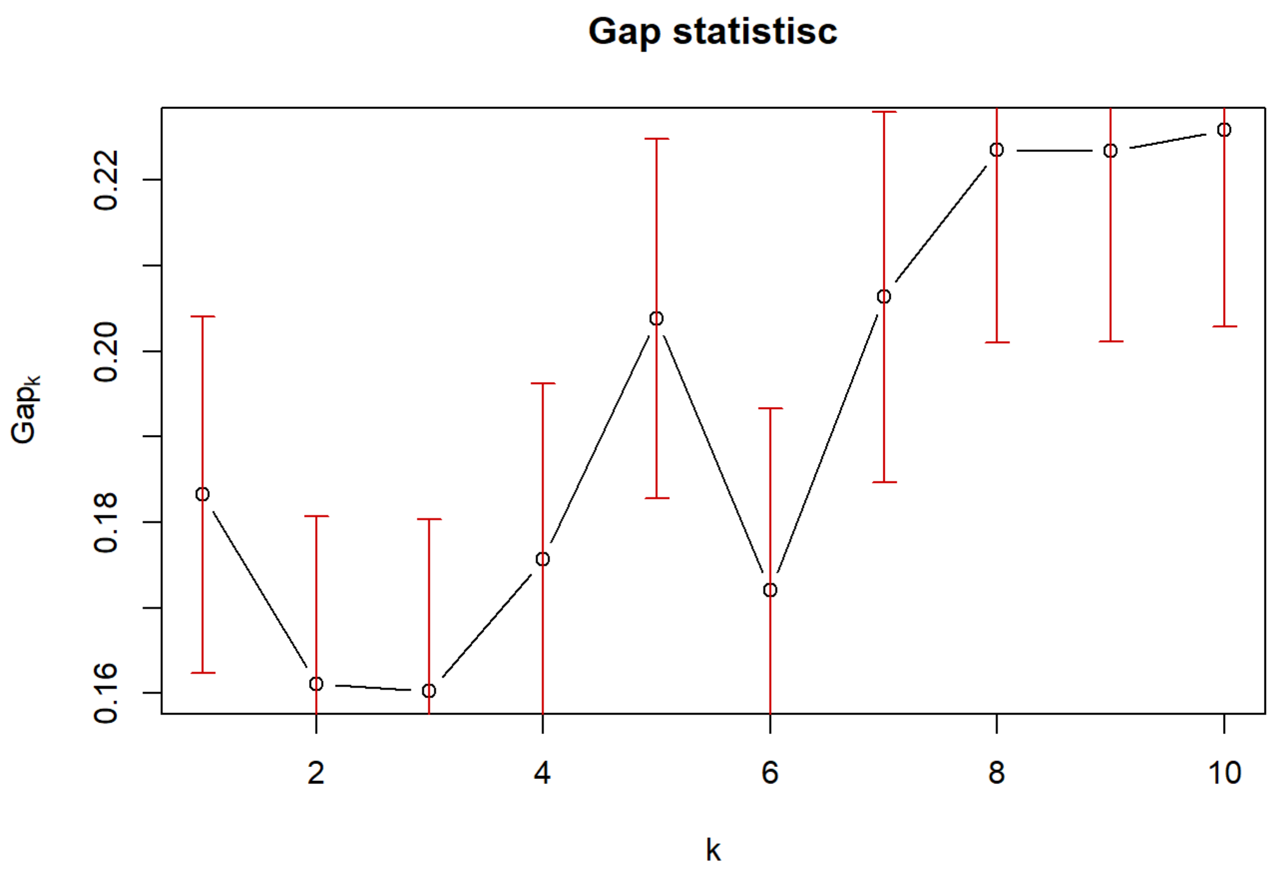
```
ggsave("figures/a2.graph2.png", device="png", width=5, height=4)

# Found a k=5 at the knee and plotting to compare to k=1 in part 1
set.seed(123)
km.50.k5 = kmeans(km.gexProj1.50[2:51], centers=5, iter.max=100, nstart=25, algorith
         m=c("Hartigan-Wong"))

# Creating dataframe for storing clustering results
km.50.k5.df = data.frame(km.gexProj1.50)
km.50.k5.df$Class = as.factor(labels1$Class)
km.50.k5.df$Cluster = as.factor(km.50.k5$cluster)

# Visualizing the results
km.plot.50.k5 = ggplot(km.50.k5.df, aes(gene_7998, gene_3228)) +
   geom_point(aes(shape=Class, color=Cluster)) +
   theme_bw() +
   ggtitle("KMeans Clustering, 50 Genes, K=5")

km.plot.50.k5
```



```
ggsave("figures/a2.graph3.png", device="png", width=5, height=4)

# Prediction table
kmeans.pred.table = table(labels1$Class, km.50.k5$cluster)


# Write kmeans table to .csv file
kmeans.pred.table = as.data.frame.matrix(kmeans.pred.table)
write.csv(kmeans.pred.table, file="figures/kmeans-50gene-k5-pred.csv")
```

7.6 Task A2 (Part 3 of Task A2)

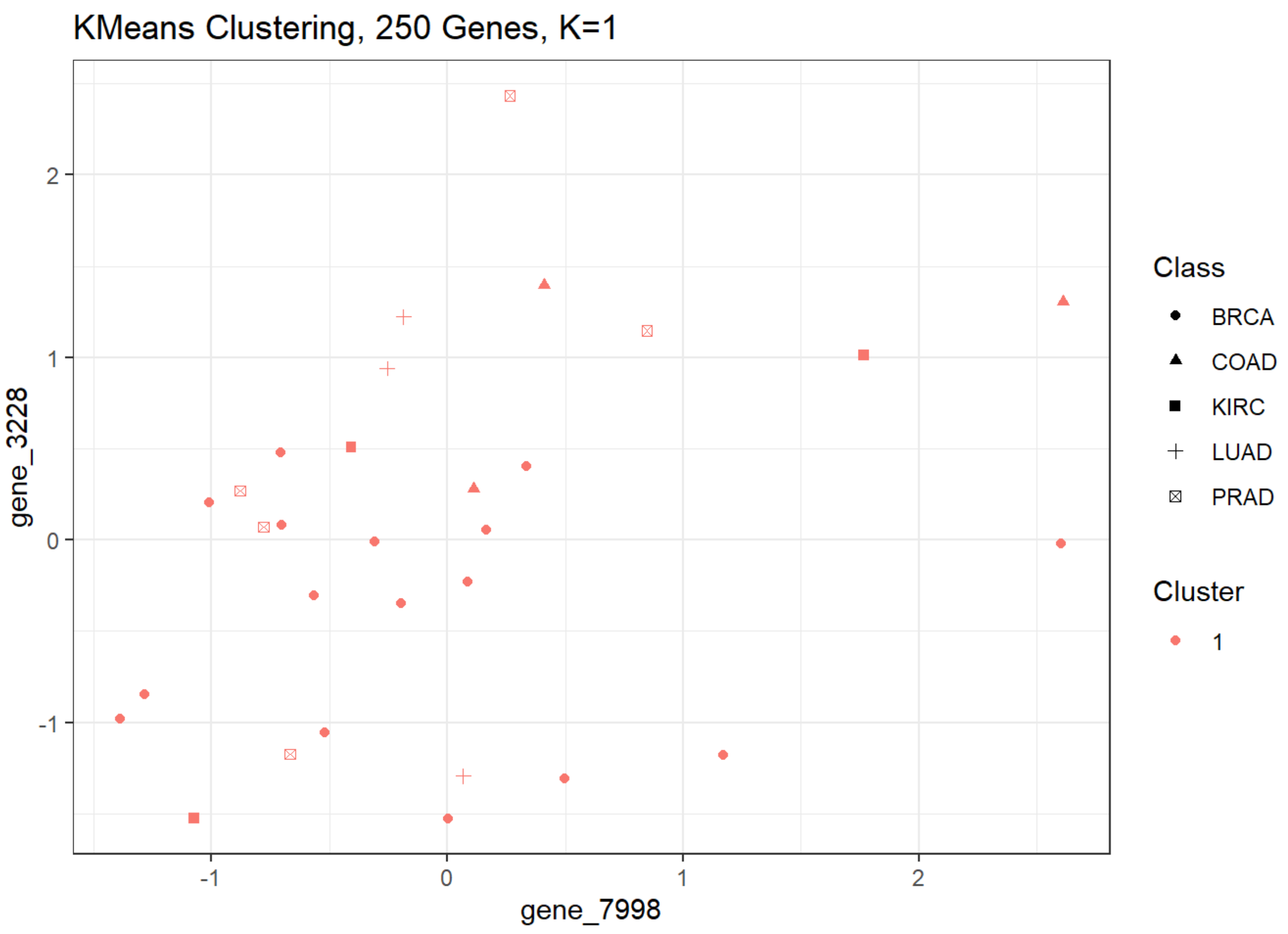
```
set.seed(123)

genes = sample(1:dim(stdgexpProj1)[[2]], 250, replace=FALSE)
km.gexProj1.250 = stdgexpProj1[,c(TRUE, genes)]

###GAP****STATISTICS****###
set.seed(123)
gap.estimate.250 = clusGap(km.gexProj1.250[,2:251], FUNcluster=kmeans, K.max=10, B=2
         00, iter.max=100)
k.250 =  maxSE(gap.estimate.50$Tab[,"gap"], gap.estimate.250$Tab[,"SE.sim"], method=
         "Tibs2001SEmax")

# Plot gap statistics
plot(gap.estimate.250, main = "Gap statistisc")
```



```
k.250
```

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

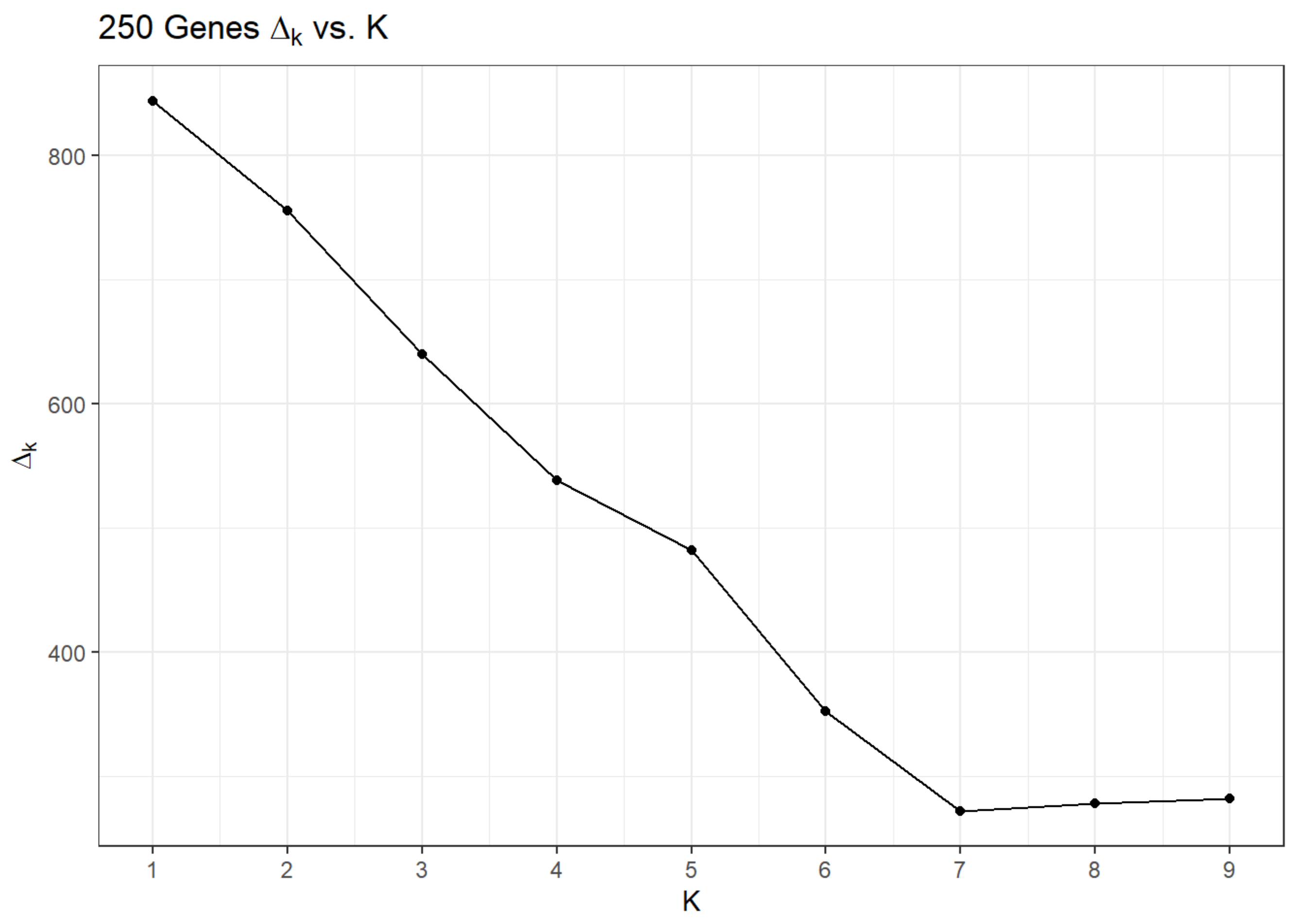
```
###****KMEANS****###
set.seed(123)
km.250.k1 = kmeans(km.gexProj1.250[2:251], iter.max = 100, centers = k.250, nstart =
         25,algorithm=c("Hartigan-Wong"))


#data frame for storing clustering results
km.250.k1.df = data.frame(km.gexProj1.250)
km.250.k1.df$Class = as.factor(labels1$Class)
km.250.k1.df$Cluster = as.factor(km.250.k1$cluster)

# Graphing k-means results for 250 genes, K=1
km.plot.250.k1 = ggplot(km.250.k1.df, aes(gene_7998, gene_3228)) +
   geom_point(aes(shape=Class, color=Cluster)) +
   theme_bw() +
   ggtitle("KMeans Clustering, 250 Genes, K=1")

km.plot.250.k1
```



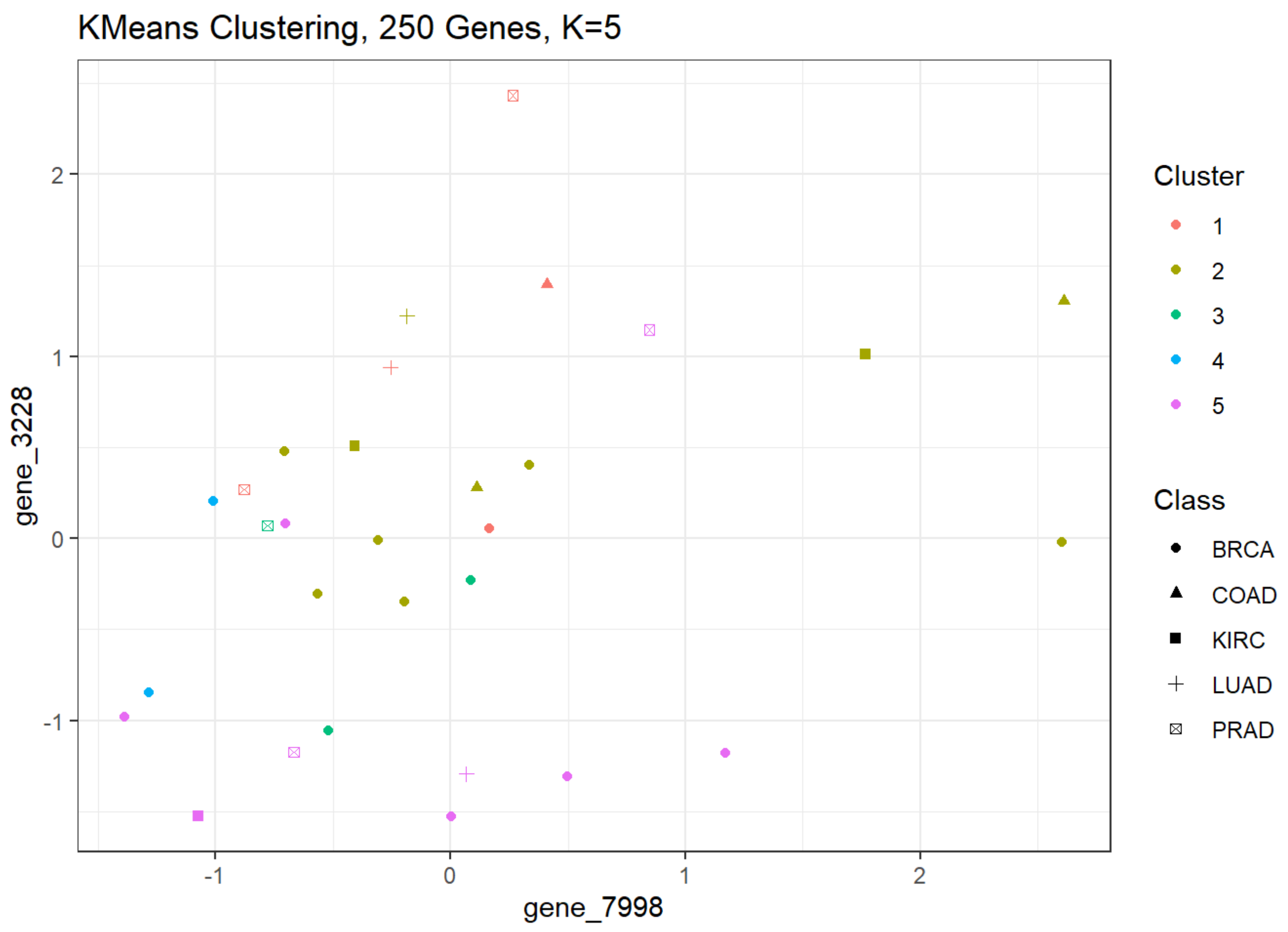
```r
ggsave("figures/a2.graph4.png", device="png", width=5, height=4)

# K-means 250 genes prediction table
kmeans.pred.table = table(labels1$Class, km.250.k1$cluster)


# Write kmeans table to .csv file
kmeans.pred.table = as.data.frame.matrix(kmeans.pred.table)
write.csv(kmeans.pred.table, file="figures/kmeans-250gene-k1-pred.csv")
```

7.7 Task A2 (Part 3 of Task A2)

```r
# Running k-means with k=1:10, calculate sum of squares
sum.square.list = list()

for (k in 1:10)
{
   set.seed(123)
   km.loop.250 = kmeans(km.gexProj1.250[2:251], centers=k, iter.max=100, nstart=25,
         algorithm=c("Hartigan-Wong"))
   sum.square.list[k] = km.loop.250$tot.withinss
}


#Calculating delta-ks
unlist.sums = unlist(sum.square.list)
k.delta = unlist.sums[1:9] - unlist.sums[2:10]
k.values = 1:9
k.df.250 = data.frame(k.values, k.delta)


# Create kmeans cluster plot with 1 center
km.loop.250.plot = ggplot(k.df.250, aes(k.values, k.delta)) +
   geom_line() +
   geom_point() +
   scale_x_continuous(breaks=1:9) +
   theme_bw() +
   labs(title=TeX("250 Genes $\\Delta_k$ vs. K"), x="K", y=TeX("$\\Delta_k$"))

km.loop.250.plot
```



```
ggsave("figures/a3.graph1.png", device="png", width=5, height=4)

# Re-running k-means with newly determined k to compare performance
set.seed(123)
km.250.k5 = kmeans(km.gexProj1.250[2:251], centers=5, iter.max=100, nstart=25, algor
         ithm=c("Hartigan-Wong"))

# Creating data frame for 250 genes
km.250.k5.df = data.frame(km.gexProj1.250)
km.250.k5.df$Class = as.factor(labels1$Class)
km.250.k5.df$Cluster = as.factor(km.250.k5$cluster)

# Visualizing the results
km.plot.250.k5 = ggplot(km.250.k5.df, aes(gene_7998, gene_3228)) +
   geom_point(aes(shape=Class, color=Cluster)) +
   theme_bw() +
   ggtitle("KMeans Clustering, 250 Genes, K=5")

km.plot.250.k5
```



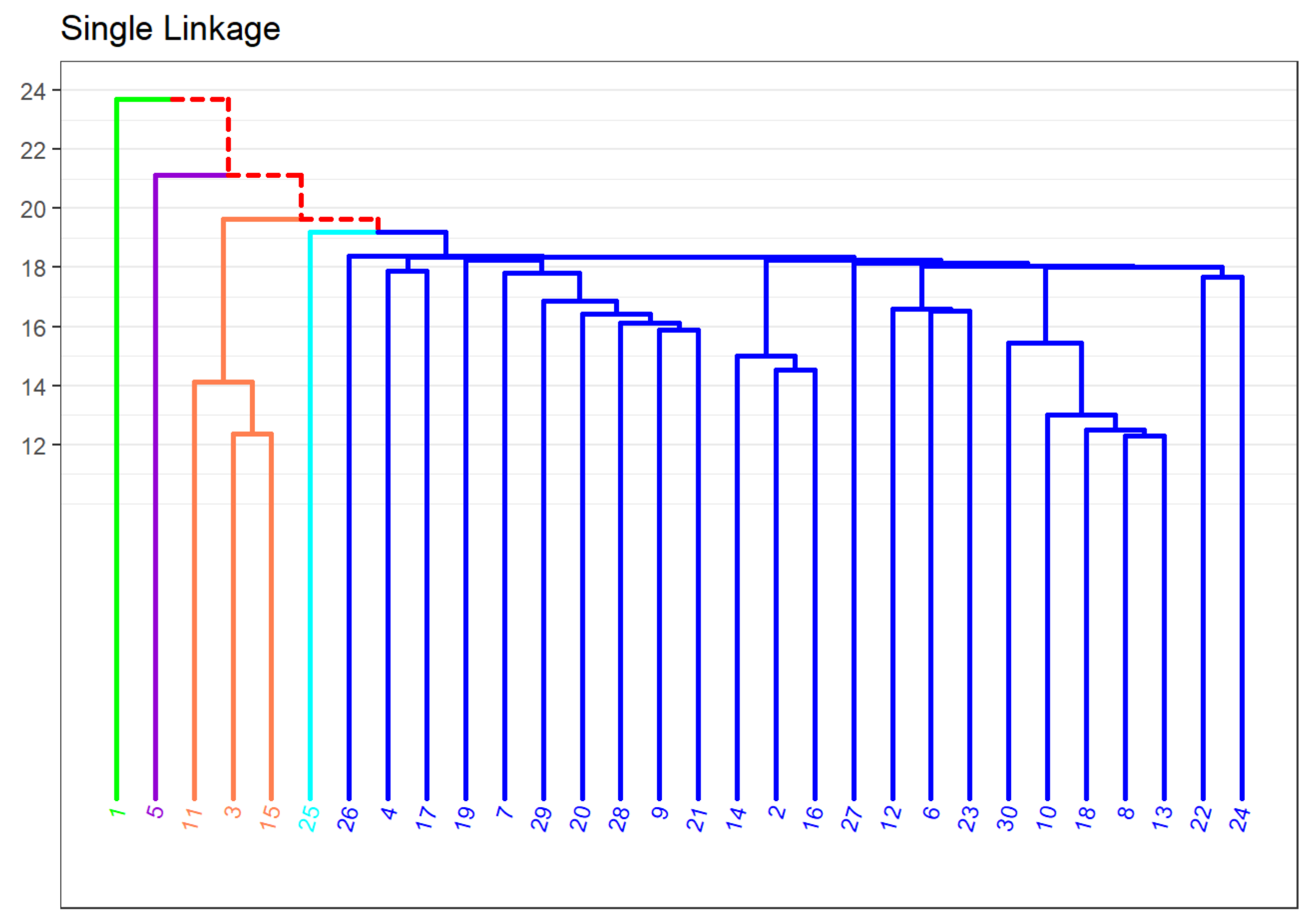
```
ggsave("figures/a3.graph2.png", device="png", width=5, height=4)

# K-means 250 genes prediction table
kmeans.pred.table = table(labels1$Class, km.250.k5$cluster)


# Write kmeans table to .csv file
kmeans.pred.table = as.data.frame.matrix(kmeans.pred.table)
write.csv(kmeans.pred.table, file="figures/kmeans-250gene-k5-pred.csv")
```

7.8 Task A3

```
set.seed(123)

# Randomly selecting 250 genes and their expressions from the data
genes = sample(1:dim(stdgexpProj1)[[2]], 250, replace=FALSE)
gexp250 = stdgexpProj1[,c(TRUE, genes)]



#Applying hierarchical clustering
hc.single = hclust(dist(gexp250), method="single")
hc.avg = hclust(dist(gexp250), method="average")
hc.complete = hclust(dist(gexp250), method="complete")

# Calculating cut height
label.groups = length(unique(labels1$Class))
hc.avg.cut = hc.avg$height[length(hc.avg$height) - label.groups]
cat("Cut height:", label.groups, "average linkage:", hc.avg.cut)
```

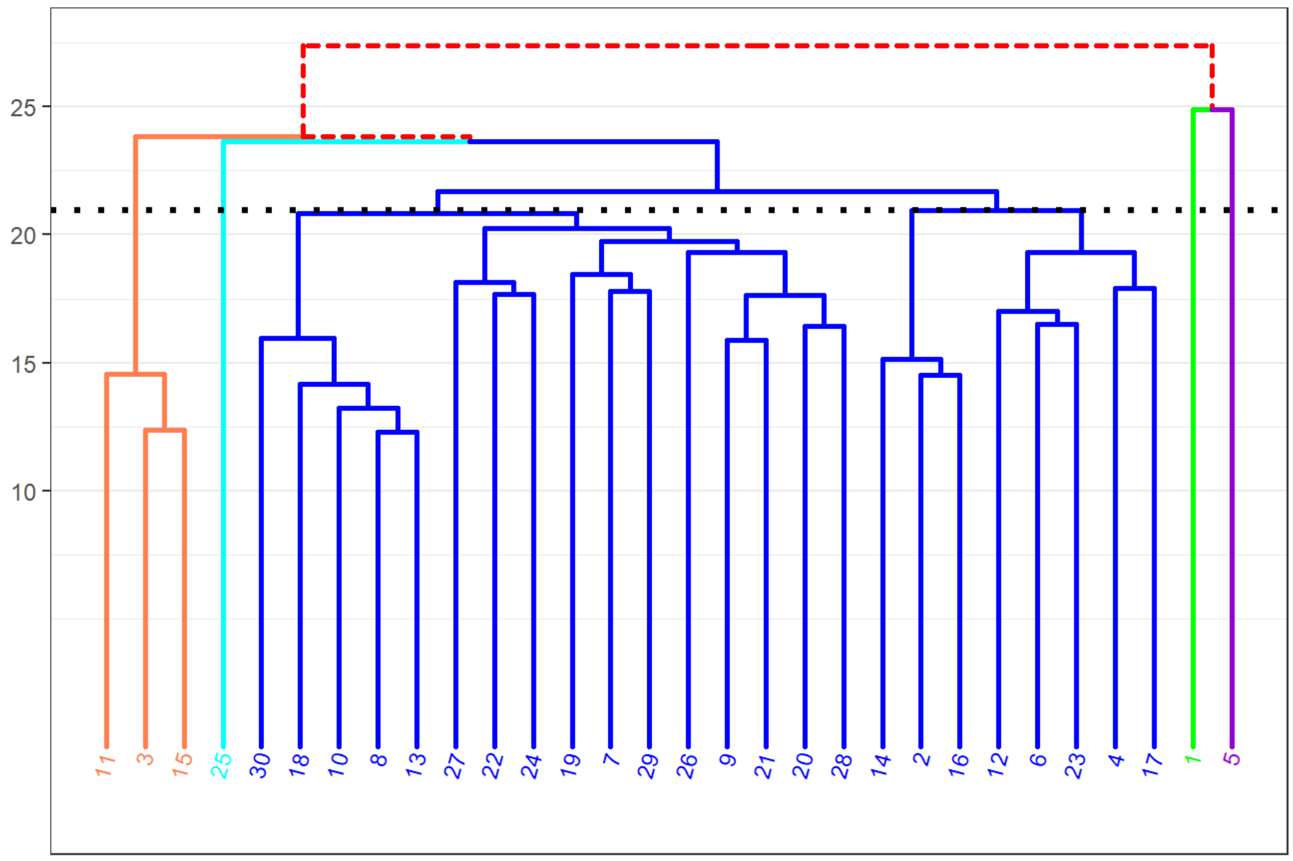
```
## Cut height: 5 average linkage: 20.94867
```

```
# include external R functions
source("Plotggdendro.r",local=T)

cols = c("red", "green", "blue","coral","darkviolet","cyan")

# Visualizing the results for all linkages
hc.single.plot = plot_ggdendro(dendro_data_k(hc.single, label.groups), direction="t
         b", title="Single Linkage",scale.color = cols)
hc.single.plot
```

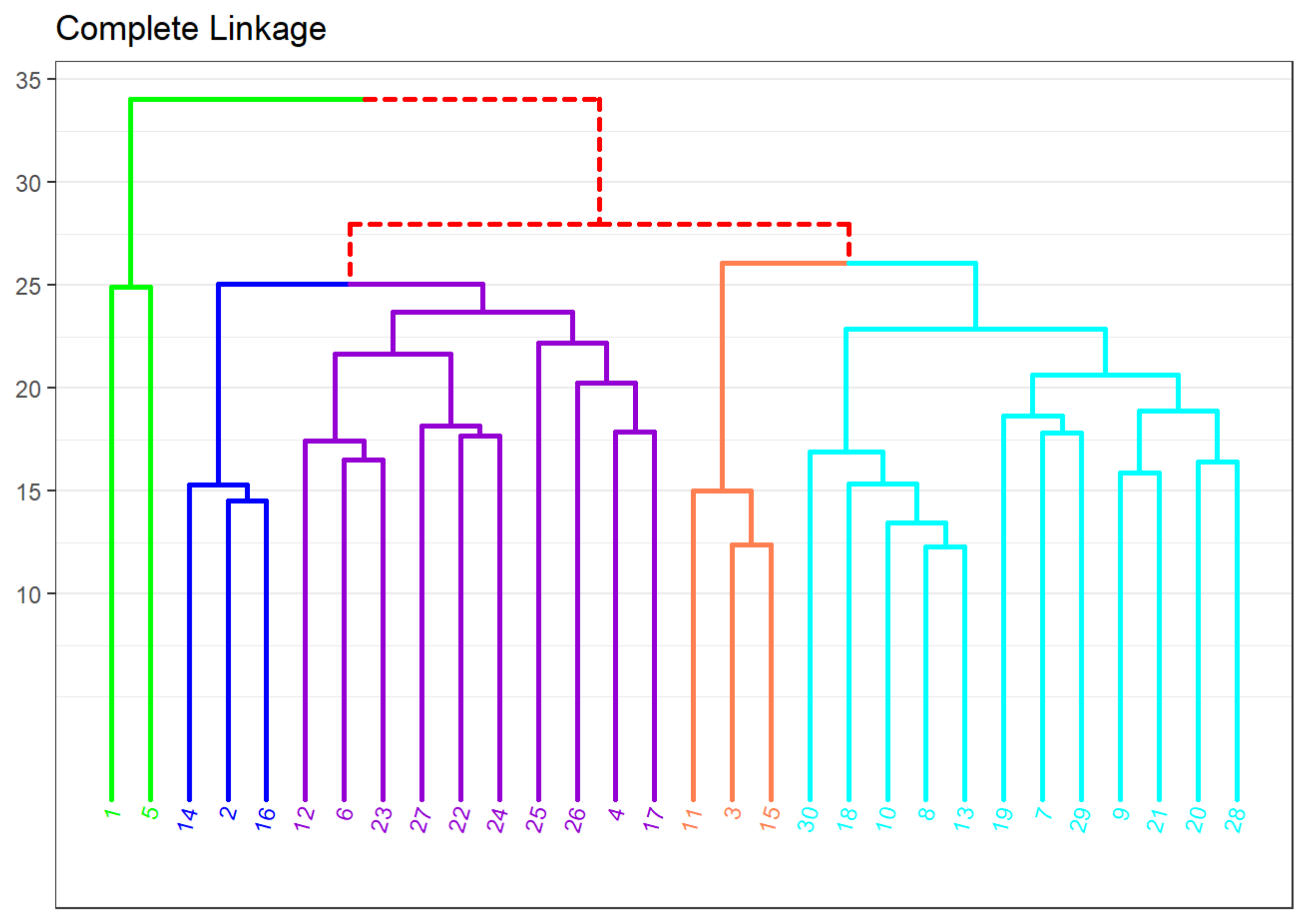


```
ggsave("figures/a3.single_dendro.png", device="png", width=7.5, height=5)


hc.avg.plot = plot_ggdendro(dendro_data_k(hc.avg, label.groups), direction="tb", hei
          ghtReferece = hc.avg.cut, title="Average Linkage",scale.color = cols)
hc.avg.plot
```

Average Linkage



```
ggsave("figures/a3.avgerage_dendro.png", device="png", width=7.5, height=5)


hc.complete.plot = plot_ggdendro(dendro_data_k(hc.complete, label.groups), direction
          ="tb", title="Complete Linkage",scale.color = cols)
hc.complete.plot
```



```r
ggsave("figures/a3.complete_dendro.png", device="png", width=7.5, height=5)


# Compare clustering methods
# vector of methods to compare
m = c( "average", "single", "complete")
names(m) = c( "average", "single", "complete")

# function to compute coefficient
ac = function(x) {
  agnes(gexp250, method = x)$ac
}
map_dbl(m, ac)
```

```
##   average    single  complete
## 0.4134613 0.2577966 0.5160520
```

8. Task B Classification

In this section QDA and KNN are applied to the data.

8.1 Task B1

```r
# Set the seed
set.seed(123)

# Filter columns with <=300 zero's
gexp2 = df_data[colSums(df_data == 0) <= 300]
head(gexp2)
```

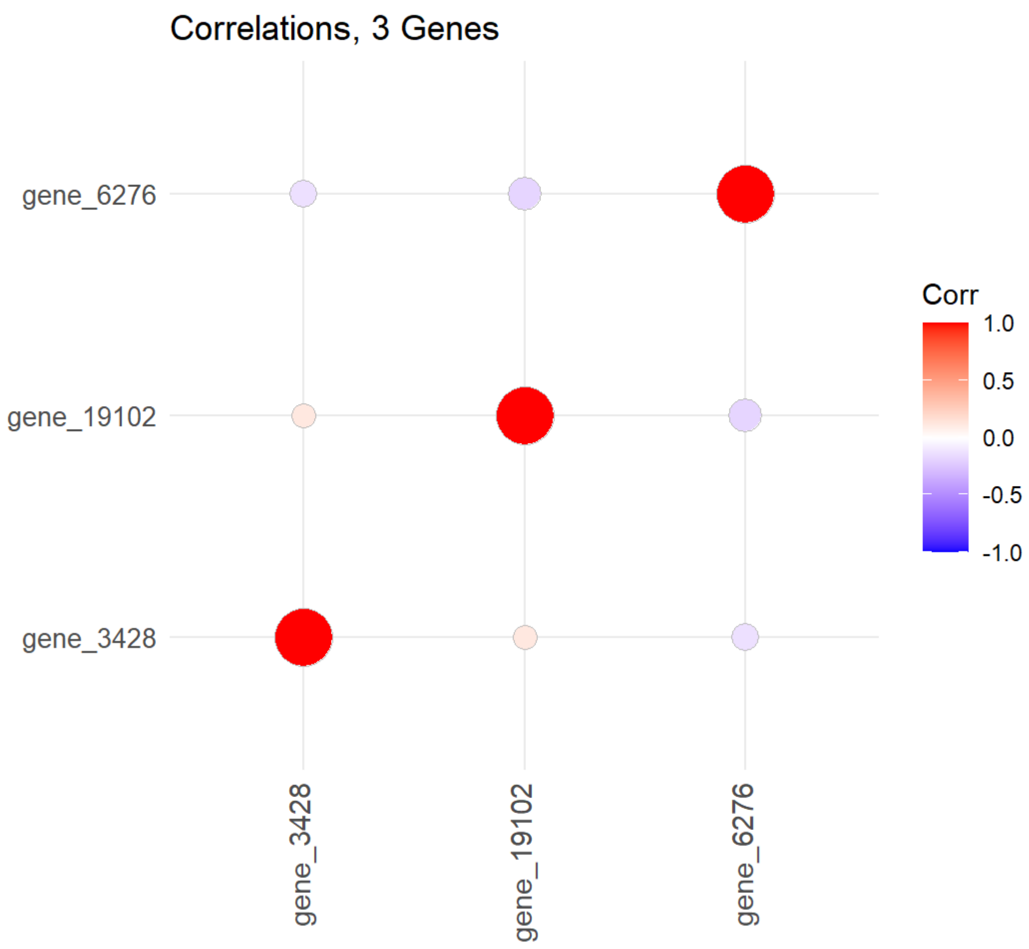
	X	gene_1	gene_2	gene_3	gene_4	gene_6	gene_11	gene_12	gene_19
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	sample_0	2.0172093	3.265527	5.478487	10.431999	7.175175	1.3342822	2.015391	5.619994
2	sample_1	0.5927321	1.588421	7.586157	9.623011	6.816049	0.5878450	2.466601	11.055208
3	sample_2	3.5117590	4.327199	6.881787	9.870730	6.972130	0.4525954	1.981122	8.210248
4	sample_3	3.6636179	4.507649	6.659068	10.196184	7.843375	0.4348817	2.874246	8.306317
5	sample_4	2.6557411	2.821547	6.539454	9.738265	6.566967	1.2758414	2.141204	10.149150
6	sample_5	3.4678533	3.581918	6.620243	9.706829	7.758510	0.5154097	2.516797	6.842765

6 rows | 1-10 of 17255 columns

```r
# Get a list of samples from the first column
sample.column.list = gexp2[,1]

# Random sample 1000 genes(columns)
gexp3 = gexp2[sample(ncol(gexp2), size = 1000), drop = FALSE]

gexp3 = cbind(sample.column.list, gexp3)
colnames(gexp3)[1] = "X"

# Selecting samples from labels for cancer types `LUAD` and `BRCA`
labels2 = df_labels %>% dplyr::select(X, Class) %>% dplyr::filter(Class %in% c("LUA
         D", "BRCA"))

# Selecting subset of labels2 from gexp3
stdgexp2 = subset(gexp3, X %in% labels2$X)

# Setting row names to sample IDs for both label and data tables
stdgexp2 = as.data.frame(stdgexp2)
rownames(stdgexp2) = stdgexp2$X1

labels2 = as.data.frame(labels2)
rownames(labels2) = labels2$X

# Removing sample IDs from `stdgexp2` and `labels2`
labels2 = labels2 %>% dplyr::select(-X)
stdgexp2 = stdgexp2 %>% dplyr::select(-X)
```

8.2 Task B2

```r
set.seed(123)

# Randomly selecting 3 genes from `stdgexp2`
genes.3 = sample(1:dim(stdgexp2)[[2]], 3-1, replace=FALSE)
stdgexp2a = stdgexp2[,c(TRUE, genes.3)]
stdgexp2a = bind_cols(stdgexp2a, labels2)

# Correlation Plot
corr.genes.3 = cor(stdgexp2a[1:3])

corr.plot.g3 = ggcorrplot(corr.genes.3, method = "circle") +
   theme(axis.text.x=element_text(size=11, angle=90, hjust=0.99, vjust=0.3), axis.te
         xt.y=element_text(size=10)) +
   labs(title="Correlations, 3 Genes", x="Variable 1", y="Variable 2")

corr.plot.g3
```


Correlations, 3 Genes
gene_6276
gene_19102
gene_3428
gene_3428
gene_19102
gene_6276
Corr
1.0
0.5
0.0
-0.5
-1.0

```
ggsave("figures/b2.corr3.png", device="png", width=7.5, height=5)

gaussian.mixture.g3 = as.data.frame(stdgexp2a)
gaussian.mixture.g3$Class = labels2$Class
gaussian.mixture.g3 = reshape2::melt(gaussian.mixture.g3, id = "Class")

{
gm.3 = ggplot(gaussian.mixture.g3) + geom_histogram(aes(x = value, fill = Class),pos
         ition="identity", alpha=0.5) +
      geom_density(aes(x = value, fill = Class),alpha = 0.6,linetype="dashed") +
      theme(legend.position="top") +
    labs(title = "Gaussian Mixture, 3 Genes",
    x = "Value", y = "Density")

gm.3
ggsave("figures/b2.gaussian3.png", device="png", width=7.5, height=5)
}

# Splitting
# 60% training data
# 40% test data
#stdgexp2a = bind_cols(stdgexp2a, labels2)

set.seed(123)
training.rows = sample(1:nrow(stdgexp2a), floor(0.60*nrow(stdgexp2a)), replace=FALSE
         )
test.rows = (1:nrow(stdgexp2a))[-training.rows]

# Create train and test dataframs
train.data = stdgexp2a[training.rows,]
test.data = stdgexp2a[test.rows,]


library(MASS)
# Quadratic discriminant analysis model
qda.train = qda(Class ~ ., data=train.data)
qda.test = predict(qda.train, newdata=test.data)

# Reporting error rate
pred.table.qda.3 = table(test.data$Class, qda.test$class)

# Write 3 gene prediction table
pred.table.qda.3 = as.data.frame.matrix(pred.table.qda.3)
write.csv(pred.table.qda.3, file="figures/gene3-predtable.csv")

cat("QDA Classification Error: ", 1 - mean(qda.test$class == test.data$Class),"\n")
```

```
## QDA Classification Error:  0.2316384
```

```
# Plotting ROC
pred.QDA.3 = ROCR::prediction(qda.test$posterior[,2], test.data$Class)
perf.ROC.3 = ROCR::performance(pred.QDA.3, "tpr", "fpr")

{
png("figures/b2.roc3.png",width=5.5, height=5, units = "in", res = 100)
roc.1= plot(perf.ROC.3, avg="threshold", spread.estimate="boxplot", colorize=TRUE, m
         ain="ROC Curve, 3 Genes")
dev.off()
}
```

```
## png
##   2
```

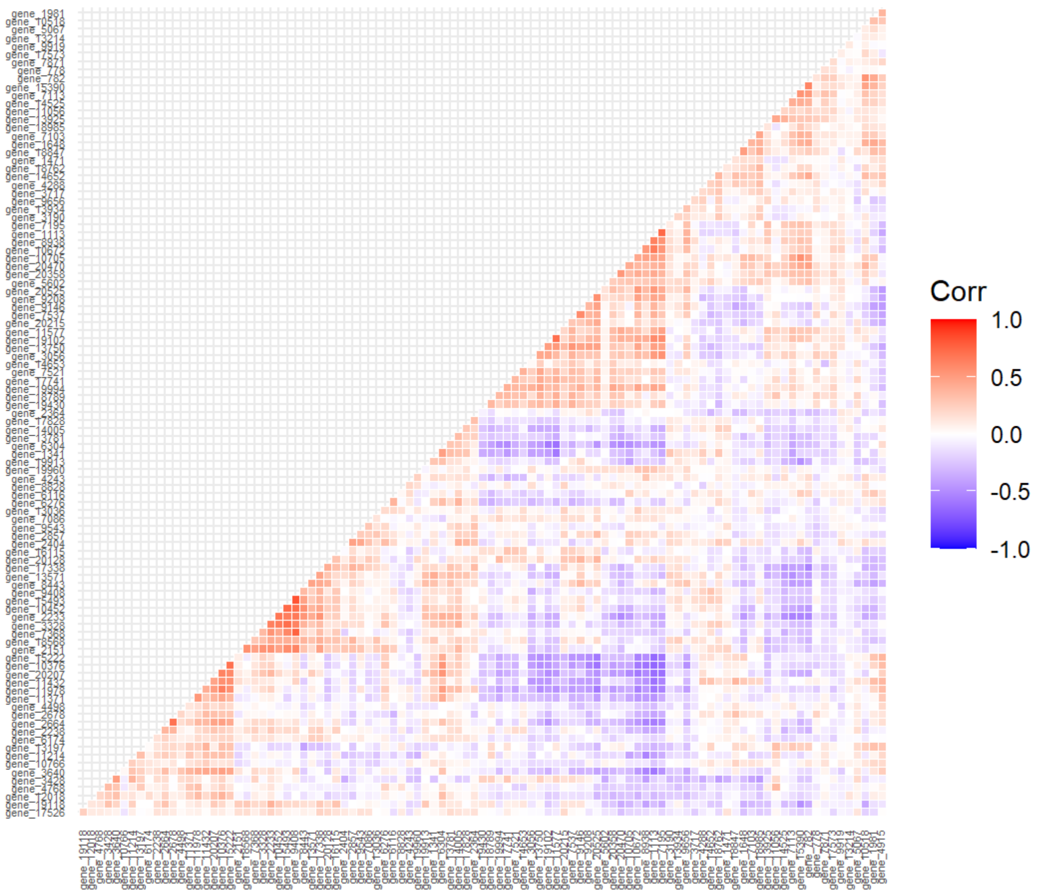
```
AUC.3 = pred.QDA.3 %>% ROCR::performance(measure = "auc") %>% .@y.values
AUC.3
```

```
## [[1]]
## [1] 0.7861417
```

8.3 Task B3

```
library(plyr)
set.seed(123)

# Randomly selecting 100 genes and their expressions from `stdgexp2`
genes.100 = sample(1:dim(stdgexp2)[[2]], 100 - 1, replace=FALSE)
stdgexp2b = stdgexp2[,c(TRUE, genes.100)]

# Correlation Plot
corr.genes.100 = cor(stdgexp2b)

corr.plot.g100 = ggcorrplot(corr.genes.100,hc.order = TRUE, type = "lower",
     outline.col = "white",insig = "blank") +
   theme(axis.text.x=element_text(size=4, angle=90, hjust=0.99, vjust=0.3), axis.tex
         t.y=element_text(size=4)) +
   labs(title="Correlations, 100 Genes", x="Variable 1", y="Variable 2")

corr.plot.g100
```

Correlations, 100 Genes



```
ggsave("figures/b2.corr100.png", device="png", width=7.5, height=5)

gaussian.mixture.g100 = as.data.frame(stdgexp2b)
gaussian.mixture.g100$Class = labels2$Class
gaussian.mixture.g100 = reshape2::melt(gaussian.mixture.g100, id = "Class")

{
gm.100 = ggplot(gaussian.mixture.g100) + geom_histogram(aes(x = value, fill = Clas
         s),position="identity", alpha=0.5) +
      geom_density(aes(x = value, fill = Class),alpha = 0.6,linetype="dashed") +
      theme(legend.position="top") +
    labs(title = "Gaussian Mixture, 100 Genes",
    x = "Value", y = "Density")

gm.100
ggsave("figures/b2.gaussian100.png", device="png", width=7.5, height=5)
}

# Splitting
# 75% training data
# 25% test data
stdgexp2b = bind_cols(stdgexp2b, labels2)

set.seed(123)
training.rows = sample(1:nrow(stdgexp2b), floor(0.75*nrow(stdgexp2b)), replace=FALSE
         )
test.rows = (1:nrow(stdgexp2b))[-training.rows]

train.data = stdgexp2b[training.rows,]
test.data = stdgexp2b[test.rows,]

library(MASS)
# Quadratic discriminant analysis model
qda.train = qda(Class ~ ., data=train.data)
qda.test = predict(qda.train, newdata=test.data)

# Reporting error rate
pred.table.qda.100 = table(test.data$Class, qda.test$class)

# Write 100 gene prediction table
pred.table.qda.100 = as.data.frame.matrix(pred.table.qda.100)
write.csv(pred.table.qda.100, file="figures/gene100-predtable.csv")

cat("QDA 100 genes Classification Error: ", 1 - mean(qda.test$class == test.data$Cla
         ss),"\n")
```

```
## QDA 100 genes Classification Error:  0.2432432
```

```
# Plotting ROC
pred.QDA.100 = ROCR::prediction(qda.test$posterior[,2], test.data$Class)
perf.ROC.100 = ROCR::performance(pred.QDA.100, "tpr", "fpr")

{
png("figures/b2.roc100.png",width=5.5, height=5, units = "in", res = 100)
roc.1= plot(perf.ROC.100, avg="threshold", spread.estimate="boxplot", colorize=TRUE,
         main="ROC Curve, 100 Genes")
dev.off()
}
```

```
## png
##   2
```

```
AUC.100 = pred.QDA.100 %>% ROCR::performance(measure = "auc") %>% .@y.values
AUC.100
```

```
## [[1]]
## [1] 0.9074853
```

8.4 Task B4

```
library(class)


# pick 100 random genes
set.seed(123)
genes = sample(1:dim(stdgexp2)[[2]], 100-1, replace=FALSE)
stdgexp2b = stdgexp2[,c(TRUE, genes)]
head(stdgexp2b)
```

	gene_3428 <dbl>	gene_19102 <dbl>	gene_6276 <dbl>	gene_10518 <dbl>	gene_11432 <dbl>	gene_17828 <dbl>	gene_17526 <dbl>	gene_13934 <dbl>
1	11.394634	10.79524	8.915646	0.0000000	13.95881	0.000000	0.3236583	11.437263
2	8.104981	11.87015	8.866163	1.0956544	14.45459	1.435949	2.9142392	10.358245
3	10.121715	11.87877	8.689631	0.6353365	14.02854	2.364404	2.5116198	11.537194
4	8.137524	11.27896	9.725654	5.6515225	14.09658	2.806881	2.4016305	9.623578
5	11.035466	10.32736	8.816068	4.7212086	15.24612	0.000000	0.0000000	10.861227
6	9.189543	11.69214	8.677794	3.1500397	14.34369	2.499399	1.6545272	10.378414

6 rows | 1-10 of 101 columns

```r
# Splitting the data into training and test sets
set.seed(123)
training.rows = sample(1:nrow(stdgexp2b), floor(0.75*nrow(stdgexp2b)), replace=FALSE
         )
test.rows = (1:nrow(stdgexp2b))[-training.rows]

train.data = as.data.frame(stdgexp2b[training.rows,])
train.labels = as.data.frame(labels2[training.rows,])
colnames(train.labels) = "Class"

test.data = as.data.frame(stdgexp2b[test.rows,])
test.labels = as.data.frame(labels2[test.rows,])

# Implementing KNN
knn.75p = knn(train.data, test.data, cl=train.labels$Class, k=3)

# Prediction table
pred.table.knn = table(knn.75p, test.labels[[1]])
pred.table.knn
```

```
##
## knn.75p BRCA LUAD
##    BRCA   81    2
##    LUAD    1   27
```

```r
# Write prediction table to file
pred.table.knn = as.data.frame.matrix(pred.table.knn)
write.csv(pred.table.knn, file="figures/knn-pred.csv")

# Classification error
cat("KNN Classification Error:", sum(1 - as.numeric(knn.75p==test.labels[[1]])) / le
         ngth(test.labels[[1]]), "\n")
```

```
## KNN Classification Error: 0.02702703
```

9. References

James, Witten, Hastie, Tibshirani. 2014. “An Introduction to Statistical Learning with Applications in R”