Stat 437 Project 1

Pat McCornack (11516300)

General rule and information

You must show your work in order to get points. Please prepare your report according to the rubrics on projects that are given in the syllabus. In particular, please note that your need to submit codes that would have been used for your data analysis. Your report can be in .doc, .docx, .html or .pdf format.

The project will assess your skills in K-means clustering,Hierarchical clustering, Nearest-neighbor classifier, and discriminant analysis for classification, for which visualization techniques you have learnt will be used to illustrate your findings.

Data set and its description

Please download the data set "TCGA-PANCAN-HiSeq-801x20531.tar.gz" from the website https://archive.ics.uci.edu/ml/machine-learning-databases/00401/. A brief description of the data set is given at https://archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq.

You need to decompress the data file since it is a .tar.gz file. Once uncompressed, the data files are "labels.csv" that contains the cancer type for each sample, and "data.csv" that contains the "gene expression profile" (i.e., expression measurements of a set of genes) for each sample. Here each sample is for a subject and is stored in a row of "data.csv". In fact, the data set contains the gene expression profiles for 801 subjects, each with a cancer type, where each gene expression profile contains the gene expressions for the same set of 20531 genes. The cancer types are: "BRCA", "KIRC", "COAD", "LUAD" and "PRAD". In both files "labels.csv" and "data.csv", each row name records which sample a label or observation is for.

```r
library(MASS)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to bec
```

```r
library(ggplot2)
library(cluster)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
library(class)


df <- read.csv('./data.csv')
labels <- read.csv('./labels.csv')
```

Task A. Clustering

For this task, you need to apply k-means and hierarchical clustering to cluster observations into their associated cancer types, and report your findings scientifically and professionally._

Your laptop may not have sufficient computational power to implement k-means and hierarchical clustering on the whole data set, and genes whose expressions are zero for most of the subjects may not be so informative of a cancer type.

Please use `set.seed(123)` for random sampling via the command `sample`, random initialization of `kmeans`, implementing the gap statistic, and any other process where artificial randomization is needed.

(**Task A1**) Complete the following data processing steps:

- Filter out genes (from "data.csv") whose expressions are zero for at least 300 subjects, and save the filtered data as R object "gexp2".
- Use the command `sample` to randomly select 1000 genes and their expressions from "gexp2", and save the resulting data as R object "gexp3".
- Use the command `sample` to randomly select 30 samples and their labels from the file "labels.csv", and save them as R object "labels1". For these samples, select the corresponding samples from "gexp3" and save them as R object "gexpProj1".
- Use the command `scale` to standard the gene expressions for each gene in "gexpProj1", so that they have sample standard deviation 1. Save the standardized data as R object "stdgexpProj1".

After subsetting the data, some of the genes had expressions of 0 for every sample. After scaling, these features were assigned NaN and had to be removed. The result is a 30 X 811 dataframe (or 810 genes).

```r
set.seed(123)

# Remove columns with at least 300 rows where the expression is 0
gexp2 <- df[,which(colSums(df[,-1] == 0) >= 300)]

# Randomly select 1000 genes
feature_ind <- sample(seq(2, dim(gexp2)[2]), 1000, replace = FALSE)
gexp3 <- gexp2 %>% select(1, feature_ind)
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(feature_ind)
##
##   # Now:
##   data %>% select(all_of(feature_ind))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
# Randomly select 30 samples/labels
observation_ind <- sample(seq(1, dim(labels)[1]), 30)
labels1 <- labels[observation_ind,]
gexpProj1 <- gexp3[observation_ind,]

# Remove columns with NaNs
stdgexpProj1 <- data.frame(X = gexpProj1$X, scale(gexpProj1[,-1])) %>%
  select_if(~ !any(is.na(.)))
```

(**Task A2**)

(**Part 1 of Task A2**) Randomly pick 50 genes and their expressions from "stdgexpProj1", and do the following to these expressions: apply the "gap statistic" to estimate the number of clusters, apply K-means clustering with the estimated number of clusters given by the gap statistic, visualize the classification results using techniques given by "LectureNotes3_notes.pdf.pdf", and provide a summary on classification errors. You may use the command `table` and "labels1" to obtain classification errors. Note that the cluster numbering given by `kmeans` will usually be coded as follows:

```
#   Class  label
#     PRAD  5
#     LUAD  4
#     BRCA  1
#     KIRC  3
#     COAD  2
```

When you apply `clusGap`, please use arguments `K.max=10, B=200,iter.max=100`, and when you use `kmeans`, please use arguments `iter.max = 100, nstart=25, algorithm =`

```
c("Hartigan-Wong").
```

The clusGap method returned an estimated value of 1 cluster in the set, which is clearly wrong. I was unable to resolve the issue, so I manually assigned k = 5 in order to continue with the assignment.

```r
set.seed(123)
# Subset to 50 genes
feature_ind <- sample(seq(1, dim(stdgexpProj1)[2]), 50, replace = FALSE)
gene_subset <- stdgexpProj1[,feature_ind]
gene_subset$X <- stdgexpProj1$X

# Apply the gap statistic
gap <- select(gene_subset, -X) %>% clusGap(kmeans, K.max = 10, B = 200, iter.max = 100)

k = maxSE(gap$Tab[,"gap"], gap$Tab[,"SE.sim"],
          method = "Tibs2001SEmax")
```

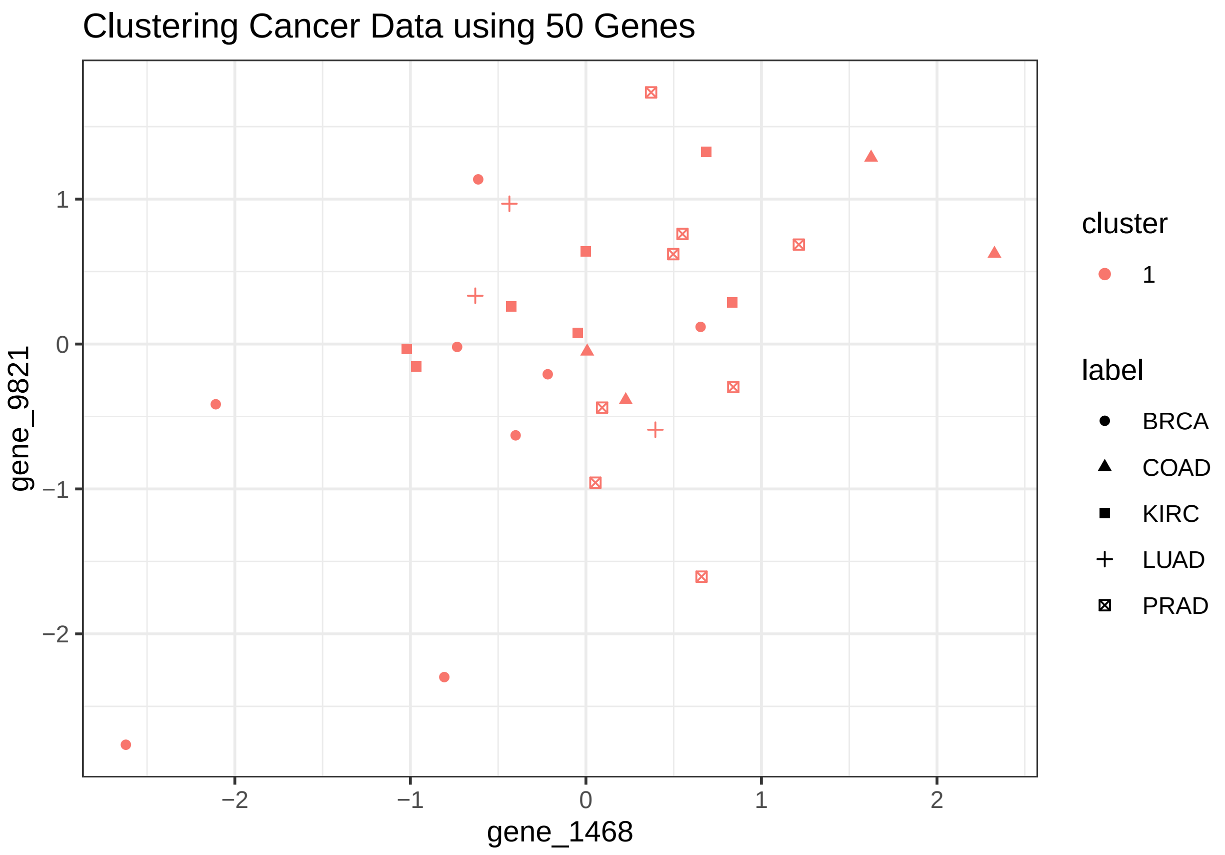
Cluster Results

Note the following performs the analysis with the estimated value of k = 1 to follow the assignment rule, then performs the analysis with k = 5 to be able get interpretable results.

The clustering tended to group three cancer types (BRCA, COAD, and LUAD) into cluster 4 - visualized as blue in the plot. While these groups were grouped together, the within group variation of cluster assignment is little to none. Notably, clustering did a very good job of separating out KIRC. All KIRC observations were grouped in cluster 3, and no other cancer types were included in this group. The clustering also separated PRAD out well, with 75% of PRAD observations grouped into cluster 1 with no other cancer types present in the group. Few observations were assigned to either cluster 2 or cluster 5, with the two observation being assigned to cluster 2 being different cancer types.

Overall the kmeans clustering was effective in separating out PRAD and KIRC, but results for the types were not as useful.

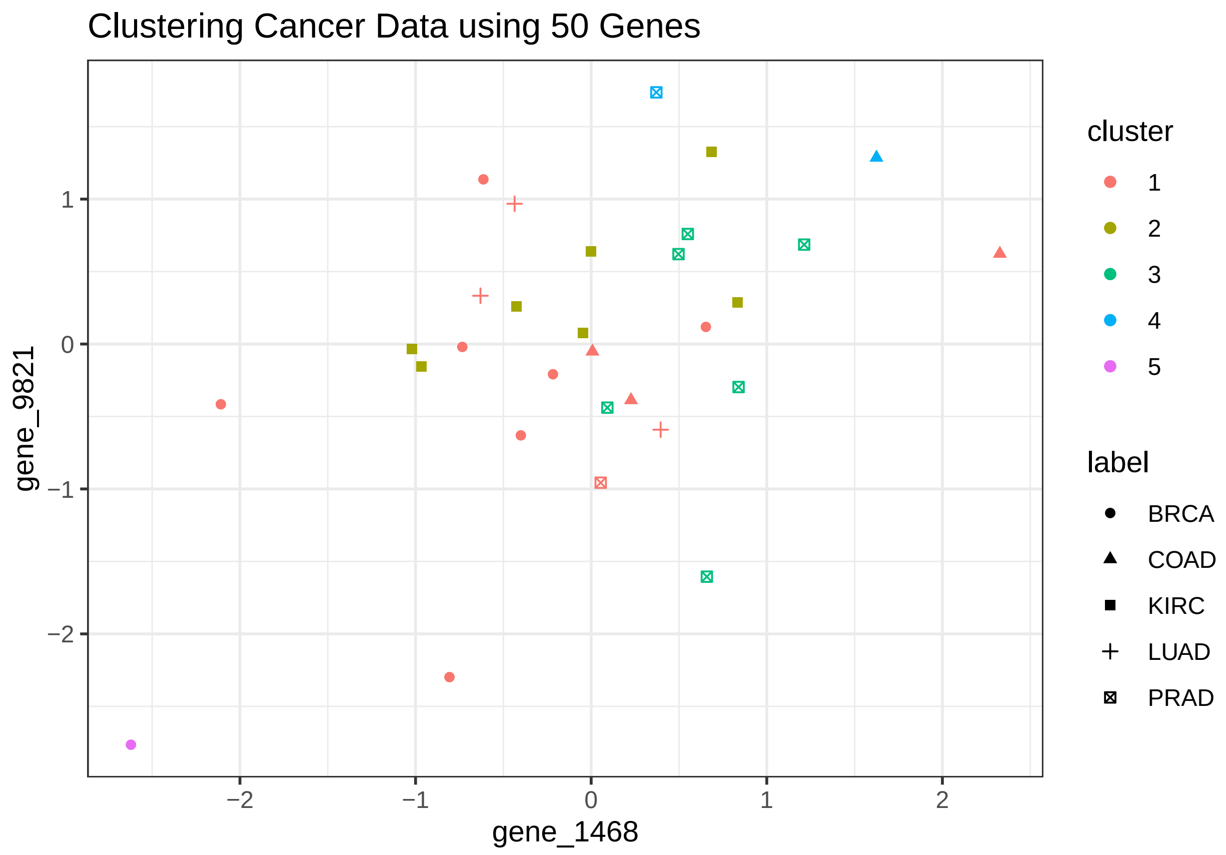
```r
set.seed(123)
## Cluster with k = 1
# Perform Kmeans
clIdx <- select(gene_subset, -X) %>% kmeans(k, iter.max = 100, nstart = 25, algorithm = c("Har
gene_subset$cluster <- factor(clIdx$cluster)
gene_subset$label <- labels1$Class

# Visualize in a plot
ggplot(gene_subset, aes(gene_1468, gene_9821)) +
  geom_point(aes(shape = label, color = cluster)) +
  ggtitle("Clustering Cancer Data using 50 Genes") +
  theme_bw()
```



```r
# Obtain classification errors
clus_error <- table(gene_subset$label, gene_subset$cluster)
clus_error
```

```
##
##          1
##   BRCA 8
##   COAD 4
##   KIRC 7
##   LUAD 3
##   PRAD 8
```

```r
## Cluster with k = 5
k = 5
# Perform Kmeans
clIdx <- select(gene_subset, -X, -label, -cluster) %>% kmeans(k, iter.max = 100, nstart = 25, a
gene_subset$cluster <- factor(clIdx$cluster)
gene_subset$label <- labels1$Class

# Visualize in a plot
ggplot(gene_subset, aes(gene_1468, gene_9821)) +
  geom_point(aes(shape = label, color = cluster)) +
  ggtitle("Clustering Cancer Data using 50 Genes") +
  theme_bw()
```



```r
# Obtain classification errors
clus_error <- table(gene_subset$label, gene_subset$cluster)
clus_error
```

```
##
##        1 2 3 4 5
##   BRCA 7 0 0 0 1
##   COAD 3 0 0 1 0
##   KIRC 0 7 0 0 0
##   LUAD 3 0 0 0 0
##   PRAD 1 0 6 1 0
```

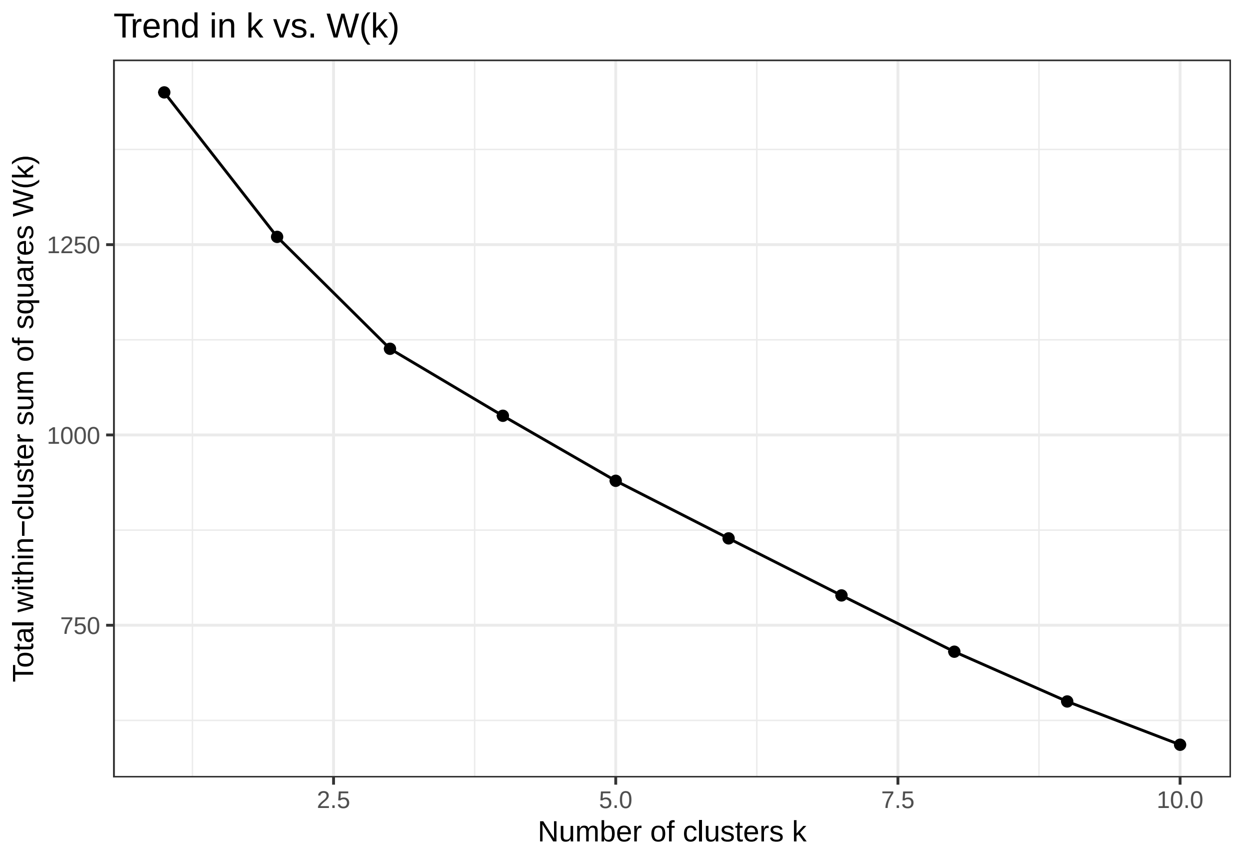
(**Part 2 of of Task A2**) Upon implementing `kmeans` with k as the number of clusters, we will obtain the "total within-cluster sum of squares" $W(k)$ from the output `tot.withinss` of `kmeans`. If we try a sequence of $k = 1, 2, 3, ..., 10$, then we get $W(k)$ for each k between 1 and 10. Let us look at the difference $\Delta_k = W(k) - W(k+1)$ for k ranging from 1 to 9. The K^* for which

$$\{\Delta_k : k < K^*\} \gg \{\Delta_k : k \geq K^*\}$$

is an estimate of the true number K of clusters in the data, where $\gg$ means "much larger". Apply this method to obtain an estimate of K for the data you created in **Part 1**, and provide a plot of $W(k)$ against k for each k between 1 and 10. Compare this estimate with the estimate obtained in **Part 1** given by the gap statistic, comment on the accuracy of the two estimates, and explain why they are different.

The method specified estimates the true number of clusters to be 3, as that is the “elbow” in the figure where the curve begins to flatten out. This agrees with the clustering results, which placed most observations into three clusters. It is also closer to the true number of 5 distinct classes than the clusGap estimate of k = 1.

```r
set.seed(123)

# Obtain W(k) for each k
clus_ss <- c()
for(k in seq(1, 10)){
  clsIdx <- select(gene_subset, -X, -label, -cluster) %>% kmeans(k, iter.max = 100, nstart = 25
  clus_ss[k] <- clsIdx$tot.withinss
}
clus_ss <- data.frame(k = seq(1:10), ss = clus_ss)

# Obtain differences in W(k) between values of k
delta = c()
for(k in seq(1,9)){
  delta[k] <- clus_ss$ss[k] - clus_ss$ss[k+1]
}
delta[10] <- NA
clus_ss$delta <- delta
clus_ss <- clus_ss %>% mutate(ratio = delta / k)

# Plot W(k) against k
ggplot(clus_ss, aes(k, ss)) +
  geom_point() +
  geom_line() +
  xlab('Number of clusters k') +
  ylab('Total within-cluster sum of squares W(k)') +
  ggtitle('Trend in k vs. W(k)') +
  theme_bw()
```



(**Part 3 of of Task A2**) Randomly pick 250 genes and their expressions from “stdgexpProj1”, and for these expressions, do the analysis in **Part 1** and **Part 2**. Report your findings, compare your findings with those from **Part 1** and **Part 2**; if there are differences between these findings, explain why. Regard using more genes as using more features, does using more features necessarily give more accurate clustering or classification results?

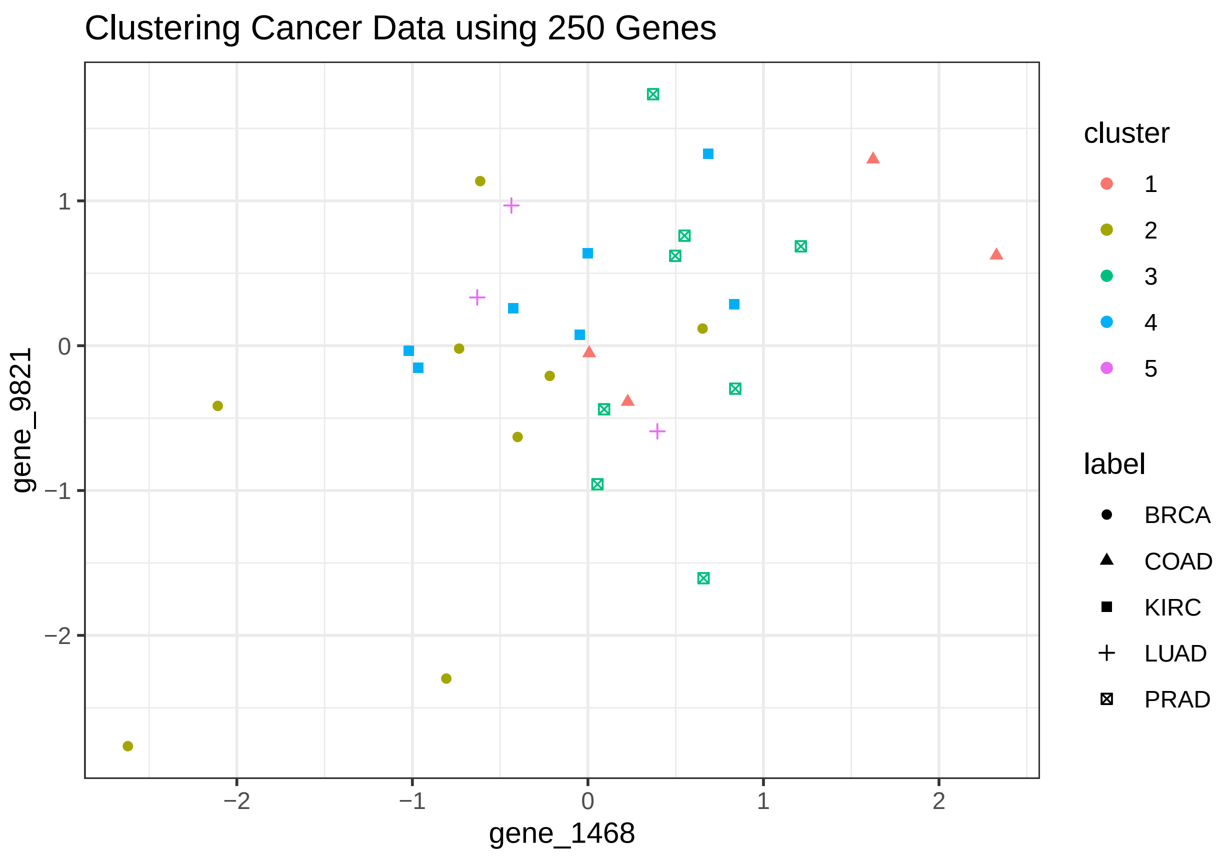
Using `clusGap` returned an estimated number of clusters of k = 1. This is clearly wrong, but I can’t find the issue in the code. To compensate I manually assigned the number of clusters as k = 5.

Using k = 5, the clustering obtained 100% accuracy. Every type of cancer was grouped into a distinct cluster, with no types spread across clusters. In this case, the higher dimensionality of the feature space separated the neighborhoods of the cancer types - making separating them into unique clusters achievable.

Note that it is not always the case that more features will yield better results. More features may sometimes degrade performance. This can be related to the ‘curse of dimensionality’ where as the feature space expands the distance between observations can sometimes homogenize - making separating them impossible.

```
set.seed(123)
# Subset to 250 genes
feature_ind <- sample(seq(1, dim(stdgexpProj1)[2]), 250, replace = FALSE)
gene_subset <- stdgexpProj1[,feature_ind]
gene_subset$X <- stdgexpProj1$X
```

```r
# Apply the gap statistic
gap <- select(gene_subset, -X) %>% clusGap(kmeans, K.max = 10, B = 200, iter.max = 100)

k = maxSE(gap$Tab[,"gap"], gap$Tab[,"SE.sim"],
          method = "Tibs2001SEmax")

# For practical purposes
k = 5

# Perform Kmeans
clIdx <- select(gene_subset, -X) %>% kmeans(k, iter.max = 100, nstart = 25, algorithm = c("Har
gene_subset$cluster <- factor(clIdx$cluster)
gene_subset$label <- labels1$Class

# Visualize in a plot
ggplot(gene_subset, aes(gene_1468, gene_9821)) +
  geom_point(aes(shape = label, color = cluster)) +
  ggtitle("Clustering Cancer Data using 250 Genes") +
  theme_bw()
```

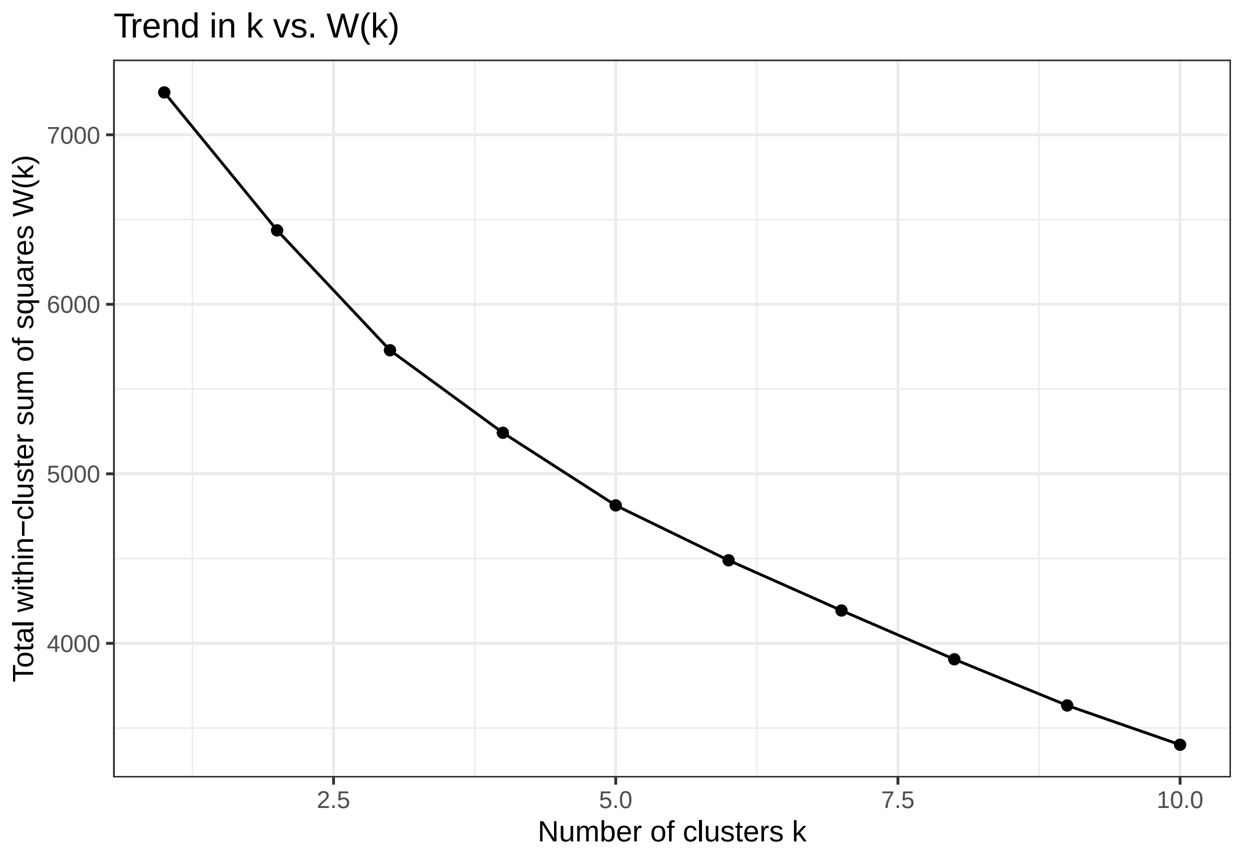


```r
# Obtain classification errors
clus_error <- table(gene_subset$label, gene_subset$cluster)
clus_error
```

```
##
##          1 2 3 4 5
##     BRCA 0 8 0 0 0
##     COAD 4 0 0 0 0
##     KIRC 0 0 0 7 0
##     LUAD 0 0 0 0 3
##     PRAD 0 0 8 0 0
```

With more features the specified method estimates the number of clusters at the correct number of 5. This is again due to the added dimensions creating more distinct neighborhoods for each cancer type. With the neighborhoods well separated, the rate of change in within neighborhood sum of squares as the number of clusters increases will be minimized at the true number of clusters.

```r
set.seed(123)
## Part 2

# Obtain W(k) for each k
clus_ss <- c()
for(k in seq(1, 10)){
  clsIdx <- select(gene_subset, -X, -label, -cluster) %>% kmeans(k, iter.max = 100, nstart = 25
  clus_ss[k] <- clsIdx$tot.withinss
}
clus_ss <- data.frame(k = seq(1:10), ss = clus_ss)

# Obtain differences in W(k) between the number of classes
delta = c()
for(k in seq(1,9)){
  delta[k] <- clus_ss$ss[k] - clus_ss$ss[k+1]
}
delta[10] <- NA
clus_ss$delta <- delta
clus_ss <- clus_ss %>% mutate(ratio = delta / k)

# Plot W(k) against k
ggplot(clus_ss, aes(k, ss)) +
  geom_point() +
  geom_line() +
  xlab('Number of clusters k') +
  ylab('Total within-cluster sum of squares W(k)') +
  ggtitle('Trend in k vs. W(k)') +
  theme_bw()
```



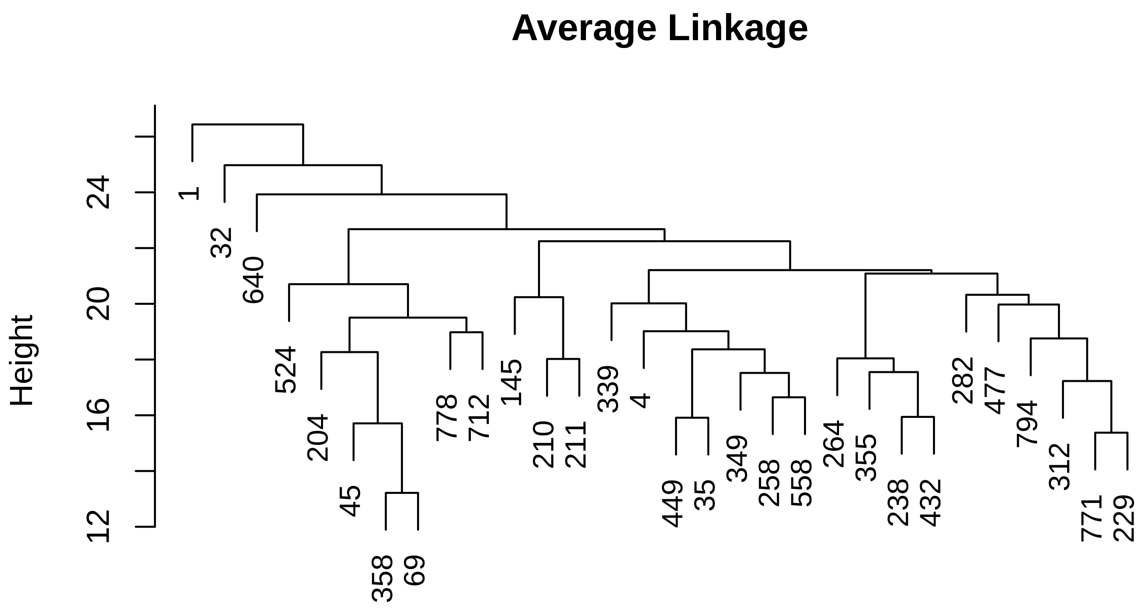
(**Task A3**) Randomly pick 250 genes and their expressions from "stdgexpProj1", and for these expressions, do the following: respectively apply hierarchical clustering with average linkage, single linkage, and complete linkage to cluster subjects into groups, and create a dendrogram. For the dendrogram obtained from average linkage, find the height at which cutting the dendrogram gives the same number of groups in "labels1", and comment on the clustering results obtained at this height by comparing them to the truth contained in "labels1".

Cutting the tree created using average linkage at a height of 23 produces 5 clusters.

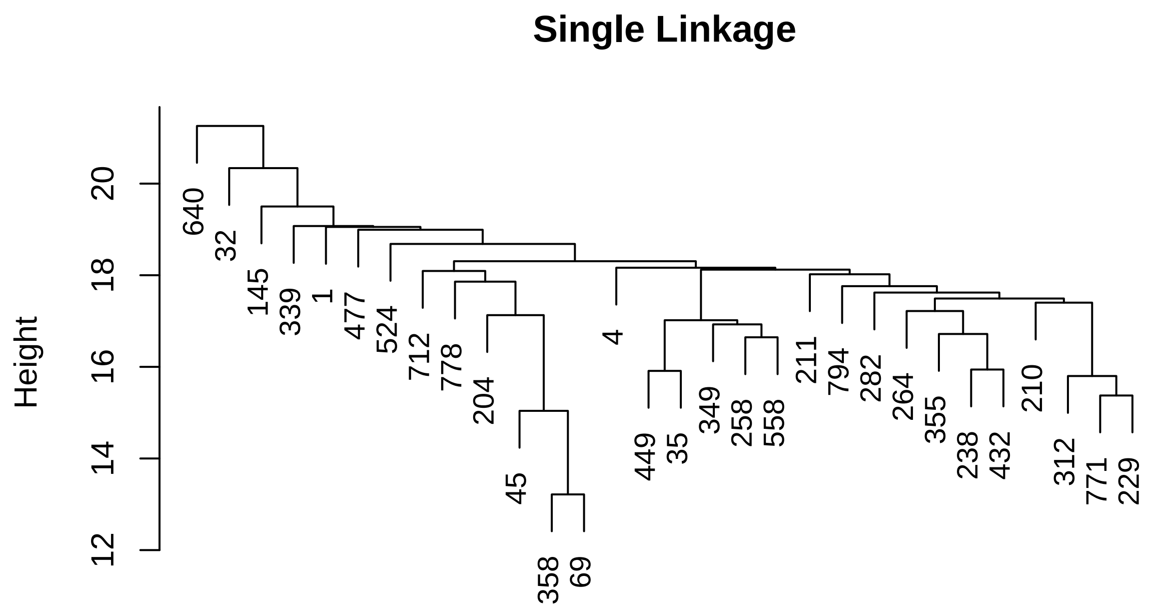
The clustering results using average linkage were not useful. Most observations were grouped into cluster 1 regardless of their true label. The only exception is that it did well differentiating KIRC, and placed all KIRC observations into a single cluster with no other cancer types.

```r
set.seed(123)
# Subset to 250 genes
feature_ind <- sample(seq(1, dim(stdgexpProj1)[2]), 250, replace = FALSE)
gene_subset <- stdgexpProj1[,feature_ind]
gene_subset$X <- stdgexpProj1$X

# Perform hierarchical clustering
hc_average <- hclust(dist(select(gene_subset, -X)), method = "average")
hc_single <- hclust(dist(select(gene_subset, -X)), method = "single")
hc_complete <- hclust(dist(select(gene_subset, -X)), method = "average")
```

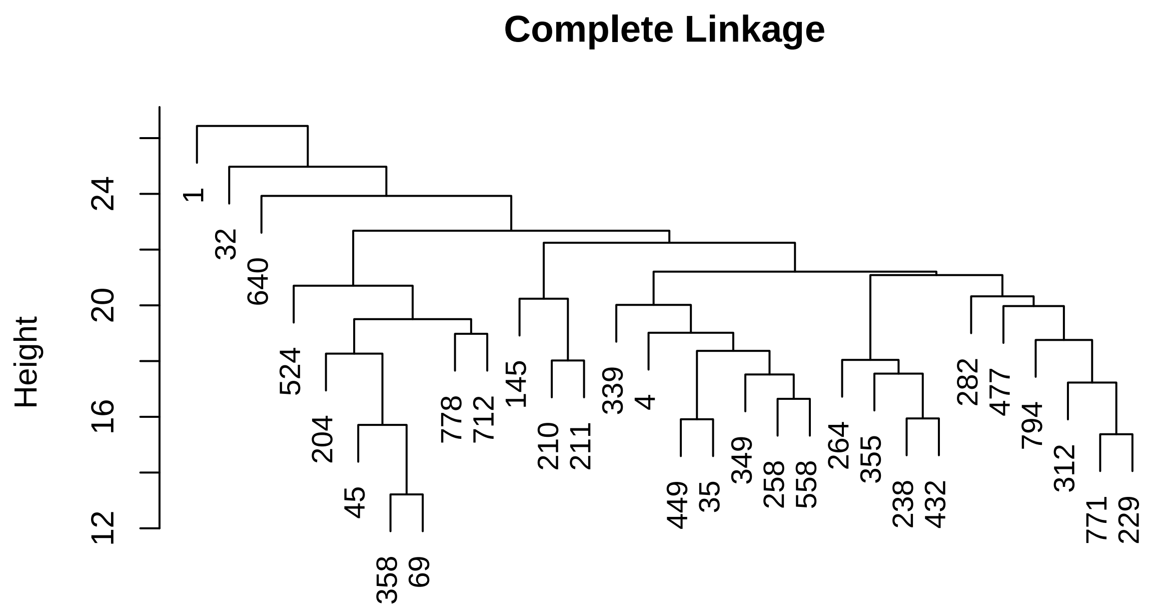
```r
# Plot dendrograms
plot(hc_average, main = "Average Linkage", xlab = "", sub = "", cex = .9)
```



```r
plot(hc_single, main = "Single Linkage", xlab = "", sub = "", cex = .9)
```



```r
plot(hc_complete, main = "Complete Linkage", xlab = "", sub = "", cex = .9)
```



```
tree_result <- cutree(hc_average, h = 23)
gene_subset$cluster <- tree_result
gene_subset$label <- labels1$Class

table(gene_subset$cluster, gene_subset$label)
```

```
##
##      BRCA COAD KIRC LUAD PRAD
##   1     6    4    7    3    7
##   2     1    0    0    0    0
##   3     1    0    0    0    0
##   4     0    0    0    0    1
```

Task B. Classification

For this task, we will use the same data set you would have downloaded. Please use `set.seed(123)` for random sampling via the command `sample` and any other process where artificial randomization is needed.

(**Task B1**) After you obtain "labels.csv" and "data.csv", do the following:

- Filter out genes (from "data.csv") whose expressions are zero for at least 300 subjects, and save the filtered data as R object "gexp2".

- Use the command `sample` to randomly select 1000 genes and their expressions from “gexp2”, and save the resulting data as R object “gexp3”.
- Pick the samples from “labels.csv” that are for cancer type “LUAD” or “BRCA”, and save them as object “labels2”. For these samples, pick the corresponding gene expressions from “gexp3” and save them as object “stdgexp2”

```r
set.seed(123)

# Remove columns with at least 300 rows where the expression is 0
gexp2 <- df[,which(colSums(df[,-1] == 0) >= 300)]

# Randomly select 1000 genes
feature_ind <- sample(seq(2, dim(gexp2)[2]), 1000, replace = FALSE)
gexp3 <- gexp2 %>% select(1, feature_ind)

# Select observations with type LUAD or BRCA

labels2 <- labels %>% filter(Class %in% c("LUAD", "BRCA"))
stdgexp2 <- gexp3 %>% filter(X %in% labels2$X)

# Remove columns with NaNs
stdgexpProj1 <- data.frame(X = gexpProj1$X, scale(gexpProj1[,-1])) %>%
  select_if(~ !any(is.na(.)))
```

(**Taks B2**) The assumptions of linear or quadratic discriminant analysis requires that each observation follows a Gaussian distribution given the class or group membership of the observation, and that each observation follows a Gaussian mixture model. In our settings here, each observation (as a row) within a group would follow a Gaussian with dimensionality equal to the number of genes (i.e., number of entries of the row). So, the more genes whose expressions we use for classification, the higher the dimension of these Gaussian distributions. Nonetheless, you need to check if the Gaussian mixture assumption is satisfied. Note that we only consider two classes “LUAD” and “BRCA”, for which the corresponding Gaussian mixture has 2 components and hence has 2 bumps when its density is plotted.

Do the following and report your findings on classification:

- Randomly pick 3 genes and their expressions from “stdgexp2”, and save them as object “stdgexp2a”.

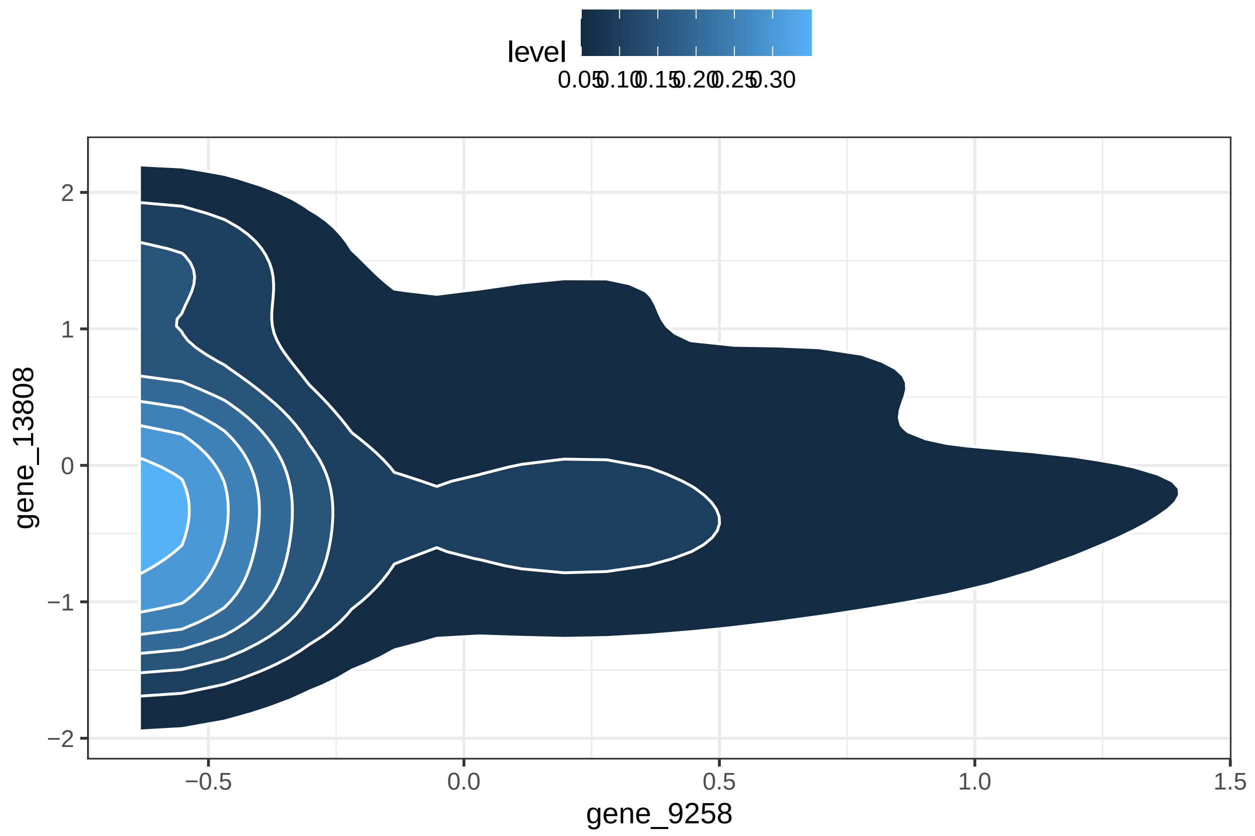
After randomly selecting three genes, every observation but one had an expression of 0. Randomly sampling a new gene to replace it yielded the same result. To rectify this and complete the assignment I manually assigned gene_11648 as the third gene.

```r
set.seed(123)

# Randomly select three genes
feature_ind <- sample(seq(1,dim(stdgexp2)[2]), 3)
stdgexp2a <- stdgexp2[,feature_ind]
stdgexp2a[,3] <- select(stdgexp2, gene_11648)
colnames(stdgexp2a)[3] <- "gene_11648"
```

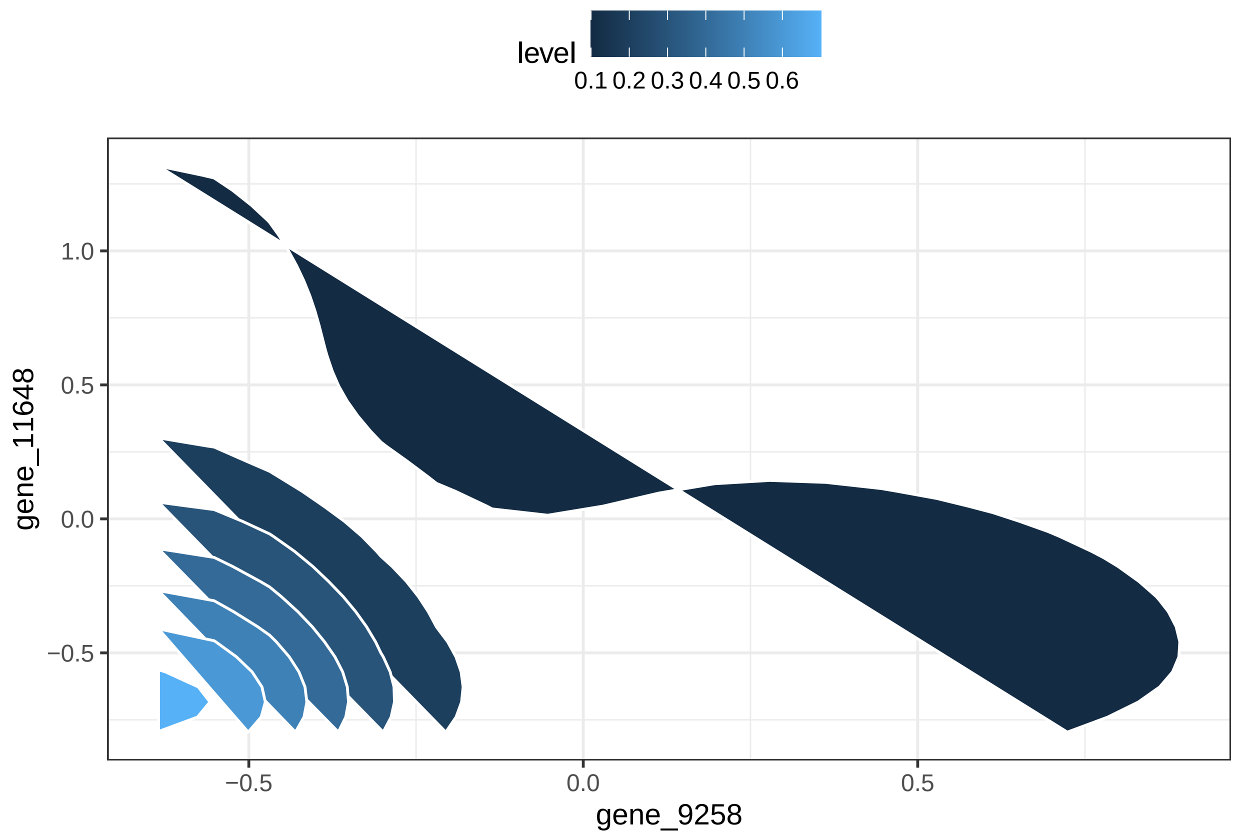
```r
stdgexp2a$X <- stdgexp2$X

# Scale dataset
stdgexp2a <- data.frame(X = stdgexp2a$X, scale(select(stdgexp2a, -X)))
```

The plots below suggest that gene_13808 does have a Gaussian distribution, but gene_9258 and gene_13808 are not symmetric and therefore not Guassian.

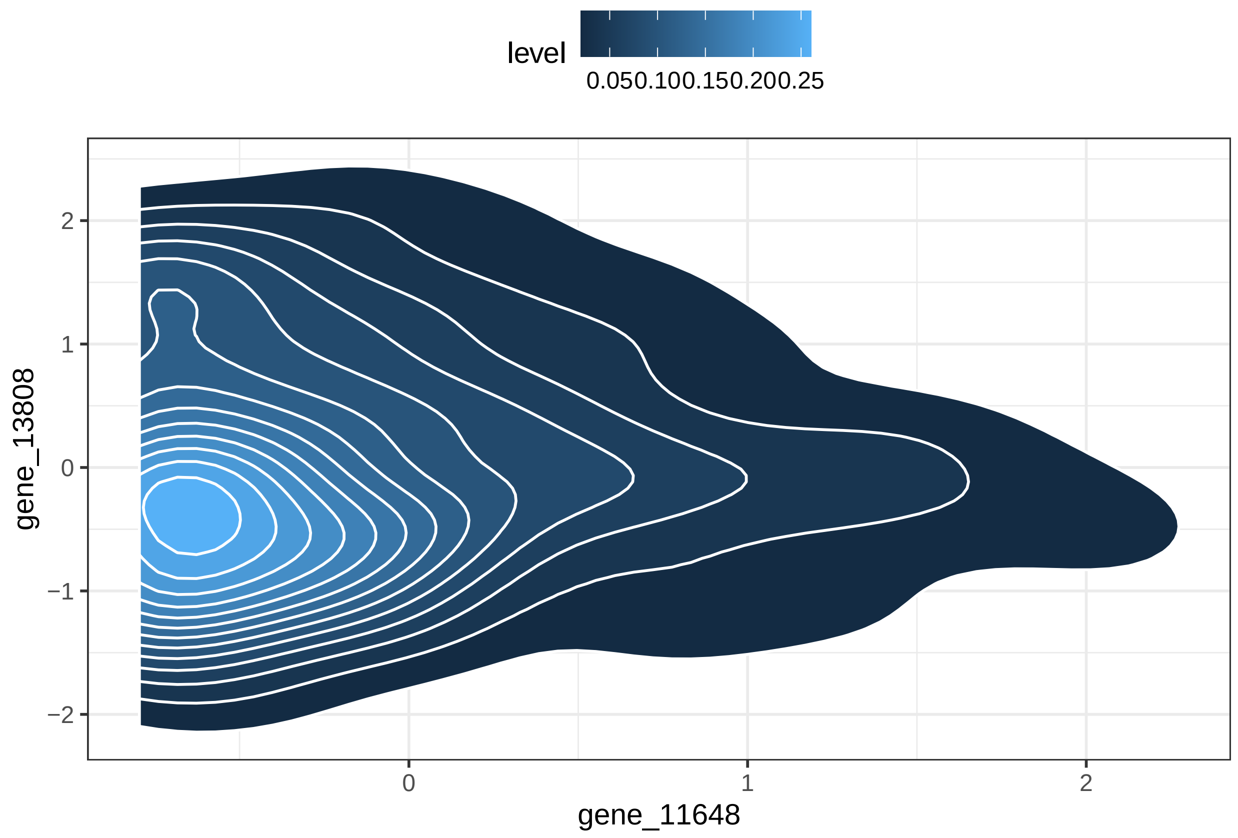
```r
ggplot(stdgexp2a, aes(gene_9258, gene_13808)) +
  theme_bw() +
  stat_density_2d(aes(fill = after_stat(level)),
                  geom = "polygon", colour = "white") +
  theme(legend.position = "top", legend.direction = "horizontal")
```



```r
ggplot(stdgexp2a, aes(gene_9258, gene_11648)) +
  theme_bw() +
  stat_density_2d(aes(fill = after_stat(level)),
                  geom = "polygon", colour = "white") +
  theme(legend.position = "top", legend.direction = "horizontal")
```



```
ggplot(stdgexp2a, aes(gene_11648, gene_13808)) +
  theme_bw() +
  stat_density_2d(aes(fill = after_stat(level)),
                  geom = "polygon", colour = "white") +
  theme(legend.position = "top", legend.direction = "horizontal")
```



- Randomly pick 60% of samples from "stdgexp2a", use them as the training set, and use the rest as the test set. You can round down the number of samples in the training set by the command `floor` if it is not an integer.

```
n_row <- floor(.6 * dim(stdgexp2a)[1])
train_ind <- sample(seq(1,dim(stdgexp2a)[1]), n_row)
train_df <- stdgexp2a[train_ind, ]
test_df <- stdgexp2a[-train_ind, ]

train_label <- labels2[train_ind, ]
train_df$label <- train_label$Class
test_label <- labels2[-train_ind,]$Class
```

Build a quadratic discriminant analysis model using the training set, and apply the obtained model to the test set to classify each of its observations. You should code "BRCA" as 0 and "LUAD" as 1. If for an observation the posterior probability of being "BRCA" is predicted by the model to be greater than 0.5, the observation is classified as "BRCA". Report via a 2-by-2 table on the classification errors. Note that the predicted posterior probability given by `qda` is for an observation to belong to class "BRCA".

Before building a quadratic discriminant analysis model, you need to check for highly correlated gene expressions, i.e., you need to check the sample correlations between each pair of columns of the training set. If there are highly correlated gene expressions, the estimated covariance matrix can be close to to being singular, leading to unstable inference. You can remove a column from two columns when their contained expressions have sample correlation greater than 0.9 in absolute value.

No pair of genes under consideration are highly correlated, and we can proceed with building the QDA model.

The QDA had an overall average accuracy of 66.6%, which is a strong performance. It did a very good job classifying BRCA observations, with an almost 90% accuracy rate for the group. Most of the error was in separating out LUAD observations, for which the classifier only had a 44% accuracy rate.

```
# Check correlations between gene expressions
cor(train_df$gene_9258, train_df$gene_13808)
```

```
## [1] 0.06754369
```

```
cor(train_df$gene_9258, train_df$gene_11648)
```

```
## [1] 0.2395415
```

```
cor(train_df$gene_11648, train_df$gene_13808)
```

```
## [1] 0.01740552
```

```
# Build QDA model
qda_fit <- qda(label ~ ., data = select(train_df, -X))

# Check the results
qda_pred <- predict(qda_fit, test_df)$class

table(qda_pred, test_label)
```

```
##         test_label
## qda_pred BRCA LUAD
##     BRCA   79    9
##     LUAD   50   39
```

```
mean(qda_pred == test_label)
```

```
## [1] 0.6666667
```

(**Taks B3**) Do the following:

- Randomly pick 100 genes and their expressions from “stdgexp2”, and save them as object “stdgexp2b”.

Random selection of 100 genes yielded a dataset with some features where every observation had an expression of 0 for the selected rows. These were removed, yielding a dataset with 87 genes.

```
set.seed(123)

# Randomly select 100 genes
feature_ind <- sample(seq(1, dim(stdgexp2)[2]), 100)
stdgexp2b <- stdgexp2[,feature_ind]
```

```r
stdgexp2b$X <- stdgexp2$X
```

- Randomly pick 75% of samples from "stdgexp2b", use them as the training set, and use the rest as the test set. You can round down the number of samples in the training set by the command `floor` if it is not an integer.

Some of the selected features contained all 0s for the selected rows. These features were coerced to NA by scaling then subsequently removed.

```r
set.seed(123)
n_row <- floor(.75 * dim(stdgexp2b)[1])
train_ind <- sample(seq(1,dim(stdgexp2b)[1]), n_row)
train_df <- stdgexp2b[train_ind, ]
test_df <- stdgexp2b[-train_ind, ]

train_label <- labels2[train_ind, ]
test_label <- labels2[-train_ind,]

# Standardize the data
train_df <- train_df %>% select(-X) %>% scale()
test_df <- test_df %>% select(-X) %>% scale()

# Remove columns with NaNs
na_col <- c()
for(i in seq(1,dim(train_df)[2])){
  if(any(is.na(train_df[,i])) | any(is.na(test_df[,i]))){
    na_col <- append(na_col, i)
  }
}

# Create dataframes with labels
train_df <- data.frame(label = factor(train_label$Class), train_df[, -na_col])
test_df <- data.frame(label = factor(test_label$Class), test_df[, -na_col])
```

Then apply quadratic discriminant analysis by following the requirements given in **Taks B2**. Compare classification results you find here with those found in **Taks B2**, and explain on any difference you find between the classification results.

Due to an error when trying to fit the model I was unable to perform QDA for the gene set with more features. Based on the performance of the clustering algorithm on the expanded feature space, I expect that the accuracy of QDA would improve with additional features as the data would be more separable.

```r
set.seed(123)
# Check for correlations
gene_cor <- round(cor(select(train_df, -label)), 3)
melted_gene_cor <- melt(gene_cor)
high_cor <- melted_gene_cor %>% filter(value != 1, abs(value) > .8)
dim(high_cor)[1]
```

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

```
#
# # Build QDA model
# qda_fit <- qda(label~., data = train_df)
#
# # Check the results
# qda_pred <- predict(qda_fit, test_df)$class
#
# table(qda_pred, test_label)
# mean(qda_pred == test_label)
```

(**Taks B4**) Do the following:

- Randomly pick 100 genes and their expressions from “stdgexp2”, and save them as object “stdgexp2b”.

```
set.seed(123)
feature_ind <- sample(seq(1, dim(stdgexp2)[2]), 100)
stdgexp2b <- stdgexp2[,feature_ind]
```

- Randomly pick 75% of samples from “stdgexp2b”, use them as the training set, and use the rest as the test set. You can round down the number of samples in the training set by the command `floor` if it is not an integer.

```
set.seed(123)
n_row <- floor(.75 * dim(stdgexp2b)[1])
train_ind <- sample(seq(1,dim(stdgexp2b)[1]), n_row)
train_df <- stdgexp2b[train_ind, ]
test_df <- stdgexp2b[-train_ind, ]

train_label <- labels2[train_ind, ]
test_label <- labels2[-train_ind,]

# Standardize the data
train_df <- train_df  %>% scale()
test_df <- test_df %>% scale()

# Remove columns with NaNs
na_col <- c()
for(i in seq(1,dim(train_df)[2])){
  if(any(is.na(train_df[,i])) | any(is.na(test_df[,i]))){
    na_col <- append(na_col, i)
  }
}

# Create dataframes
train_df <- data.frame(train_df[, -na_col])
test_df <- data.frame(test_df[, -na_col])
```

Then apply k-nearest-neighbor (k-NN) method with neighborhood size k=3 to the test data to classify each observation in the test set into one of the cancer types. Here, for an observation, if the average of being cancer type "BRCA" is predicted by k-NN to be greater than 0.5, then the observation is classified as being "BRCA". Report via a 2-by-2 table on the classification errors. Compare and comment on the classification results obtained here to those obtain in **Taks B3**. If there is any difference between the classification results, explain why.

KNN with three clusters performed nearly flawlessly with an accuracy rate of 97%. This is significantly better than QDA with three features, but I expect that it would be similar to the performance of QDA with 100 features. The main consideration in this expectation is that in the case of this dataset, expanding the feature space leads to data that with more distinct neighborhoods and therefore more separable neighborhoods. Both of these classification methods, as well as kmeans clustering is attempting to classify observations based on the neighborhood of observations around them. When the neighborhood is more "pure" in the sense it mostly contains observations of a single type, then these classifiers will perform better. By adding dimensions, neighborhoods that may be mixed in three dimensions are in fact distinct - this makes them separable.

```r
knn_fit <- knn(train_df, test_df, cl=train_label$Class, k =3)

table(knn_fit, test_label$Class)
```

```
##
## knn_fit BRCA LUAD
##    BRCA   80    1
##    LUAD    2   28
```