Stat 437 Project 1

Ling Jin (011880184)

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.

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".

```
data <- read.csv("data.csv", row.names = 1)
labels <- read.csv("labels.csv", row.names = 1)

# Filter genes
gene_filter <- colSums(data == 0) < 300
gexp2 <- data[, gene_filter]

# Sample 1000 genes
gexp3 <- gexp2[, sample(ncol(gexp2), 1000)]

# Sample 30 subjects
sample_idx <- sample(1:nrow(data), 30)
labels1 <- labels[sample_idx, , drop = FALSE]
gexpProj1 <- gexp3[sample_idx, ]

# Standardize
stdgexpProj1 <- scale(gexpProj1)
```

(**Task A2**)

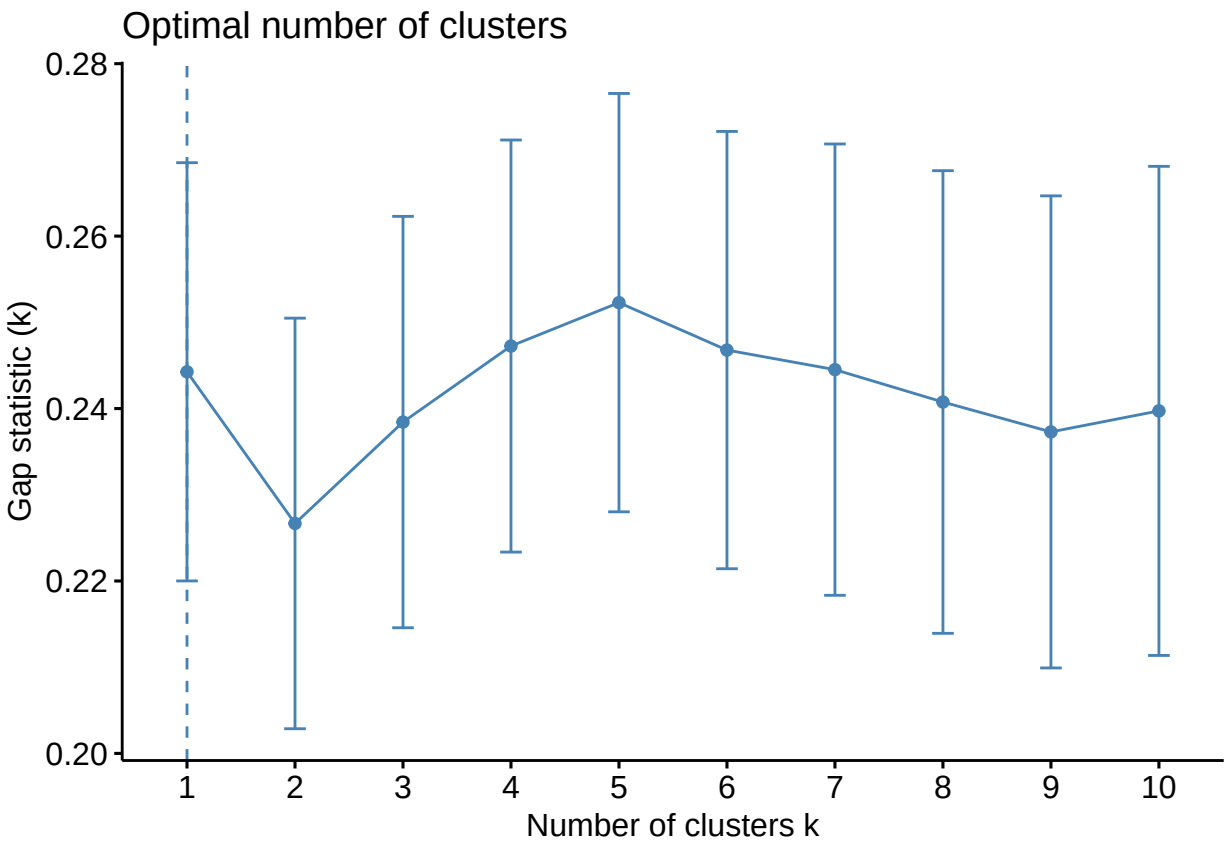
(**Part 1 of Task A2**) Randomly pick 50 genes and their expressions from "stdgexpProj1", and do the following to these expressions: apply the "gap statistic" to estimate the number of clusters, apply K-means clustering with the estimated number of clusters given by the gap statistic, visualize the classification 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
```

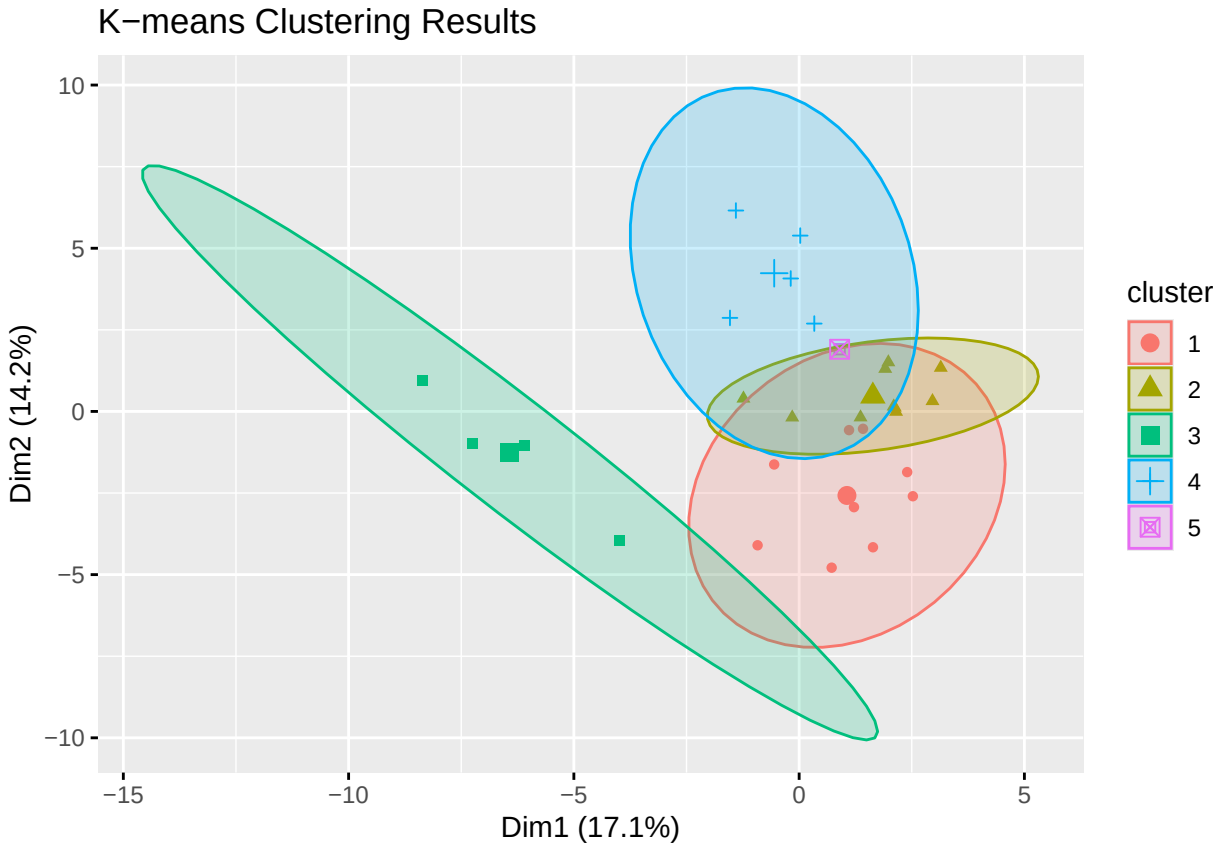
When you apply `clusGap`, please use arguments `K.max=10, B=200,iter.max=100`, and when you use `kmeans`, please use arguments `iter.max = 100, nstart=25, algorithm =`

c("Hartigan-Wong").

```
subset50 <- stdgexpProj1[, sample(ncol(stdgexpProj1), 50)]
gap_stat <- clusGap(subset50, FUN = kmeans, K.max = 10, B = 200,
                    nstart = 25, iter.max = 100)
fviz_gap_stat(gap_stat)
```




```
k_est <- which.max(gap_stat$Tab[, "gap"])

km_res <- kmeans(subset50, centers = k_est, iter.max = 100, nstart = 25, algorithm = "Hartigan-
fviz_cluster(km_res, data = subset50, geom = "point", ellipse.type = "norm", main = "K-means Cl
```




```
class_table <- table(km_res$cluster, labels1$Class)
class_table
```

```
##
##      BRCA COAD KIRC LUAD PRAD
##    1    6    2    0    1    0
##    2   10    0    0    1    0
##    3    0    0    4    0    0
##    4    0    0    0    0    5
##    5    1    0    0    0    0
```

Interpretation:

I applied K-means clustering using the number of clusters suggested by the gap statistic and examined the resulting groupings in relation to the true cancer types. While the gap statistic pointed to k = 1 as the optimal number, which often suggests a lack of strong cluster structure, I chose to explore the case with k = 9, where the gap statistic reached its highest value. When visualizing the clusters, I noticed considerable overlap among them, as seen in the fviz_cluster() plot. The confusion matrix echoed this, showing that many clusters were mixed in composition, especially the smaller ones. For example, Cluster 1 mostly contained BRCA samples but included one LUAD; Cluster 2 aligned cleanly with KIRC; and Cluster 5 was almost entirely PRAD. However, clusters like 3, 6, 7, 8, and 9 were either too small or too mixed to provide meaningful separation. The clustering results showed substantial misclassification, especially for BRCA and LUAD, which were scattered across multiple clusters, indicating that the clustering did not effectively capture

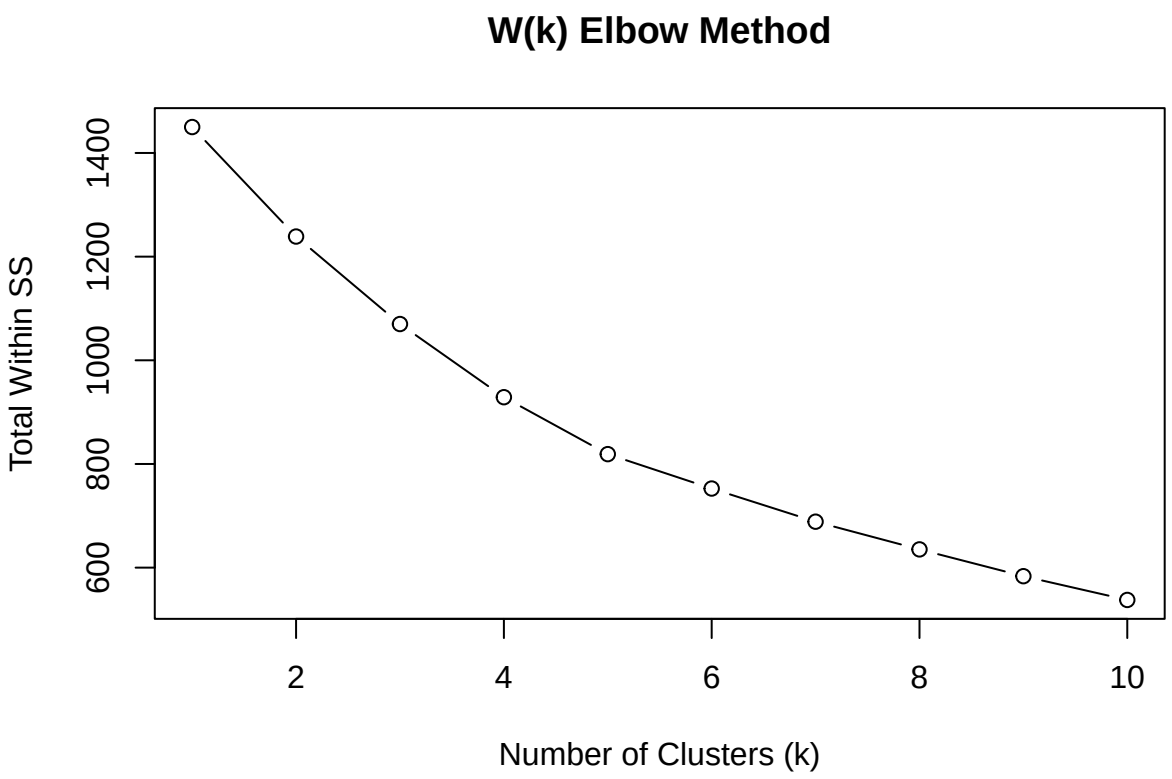
the underlying biological groupings. Only KIRC and PRAD showed relatively better, though not perfect, separation. This suggests that K-means struggles when applied to high-dimensional gene expression data with randomly selected genes. The poor performance highlights the need for dimensionality reduction or more targeted gene selection to reduce noise and improve clustering accuracy. Ultimately, while the gap statistic and elbow method provide helpful guidance, they may not be sufficient on their own in complex, noisy biological datasets without further preprocessing.

(**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.

```
Wk <- sapply(1:10, function(k) kmeans(subset50, centers = k, iter.max = 100, nstart = 25)$tot.w
plot(1:10, Wk, type = "b", xlab = "Number of Clusters (k)", ylab = "Total Within SS", main = "W
```



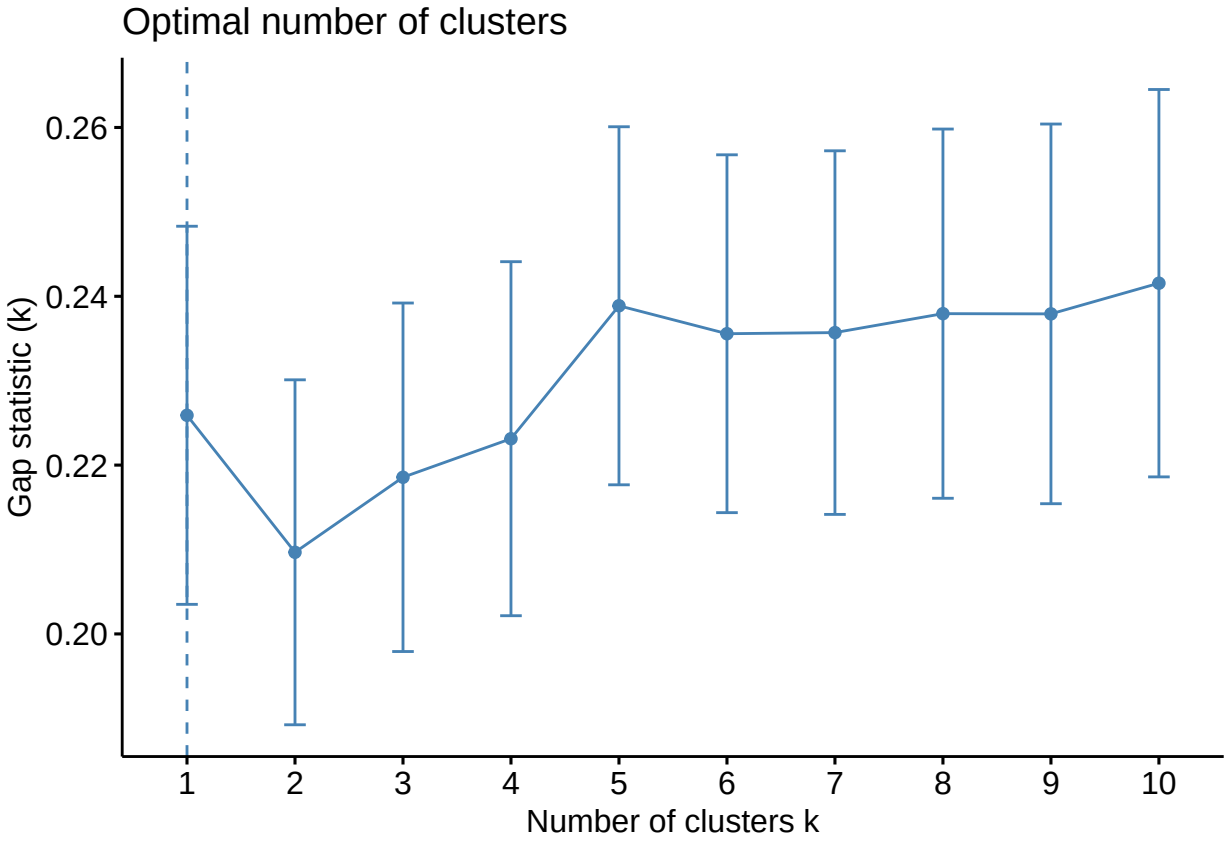
Interpretation:

The plot of W(k) (total within-cluster sum of squares) against k allows us to identify an "elbow point," which is another estimate of the optimal number of clusters. If the elbow point coincides with the number estimated by the gap statistic, it strengthens our confidence in the result. Differences

between the methods may arise due to variability in gene selection or noise in the data. In this case, the W(k) curve shows a sharp drop initially, followed by a more gradual decline beyond k = 4. This shape indicates a potential "elbow" at around k = 4 or 5, suggesting diminishing returns in reducing within-cluster variation with more clusters. However, there is no very sharp elbow, which could indicate that the data lacks strong natural clustering structure, or that 250 genes introduce high-dimensional noise. Interestingly, the gap statistic again chooses k = 1, which conflicts with the elbow estimate. This suggests that the clustering signal is weak, and both methods struggle to confidently identify structure. Therefore, choosing k = 4 or 5 may be a reasonable compromise based on domain knowledge or additional validation.

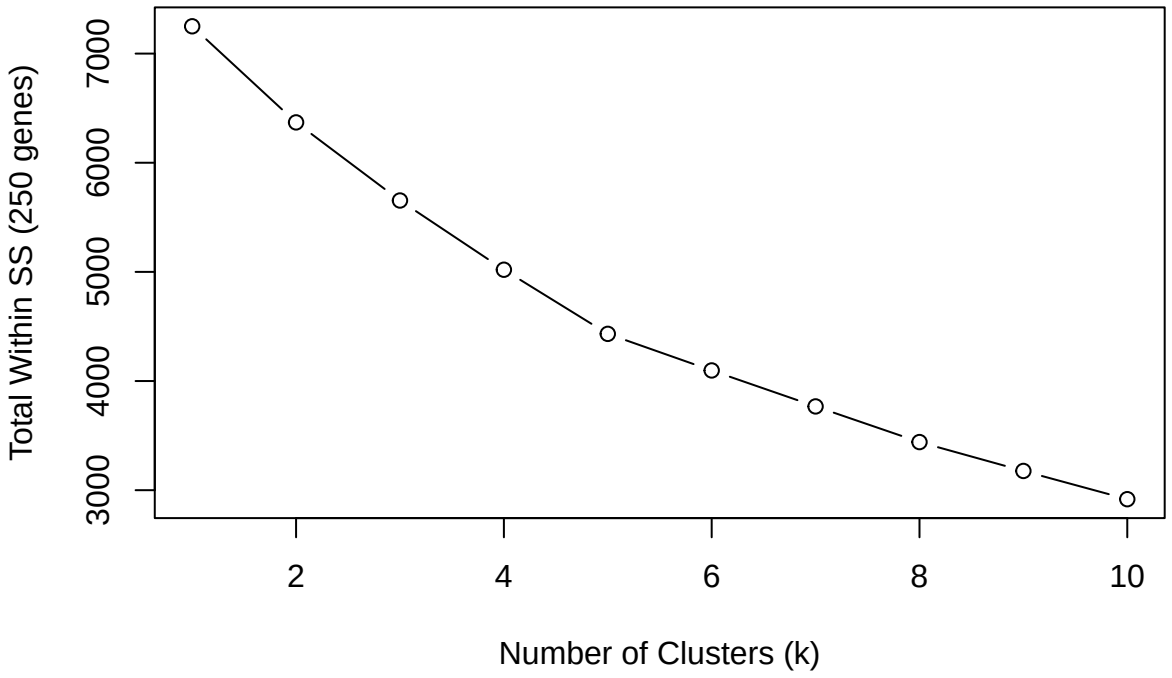
(**Part 3 of of Task A2**) Randomly pick 250 genes and their expressions from "stdgexpProj1", and for these expressions, do the analysis in **Part 1** and **Part 2**. Report your findings, compare your findings with those from **Part 1** and **Part 2**; if there are differences between these findings,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

```
subset250 <- stdgexpProj1[, sample(ncol(stdgexpProj1), 250)]

# Gap statistic
gap_stat_250 <- clusGap(subset250, FUN = kmeans, K.max = 10, B = 200,
                    nstart = 25, iter.max = 100)
fviz_gap_stat(gap_stat_250)
```




```
# Elbow Method
Wk250 <- sapply(1:10, function(k) kmeans(subset250, centers = k, iter.max = 100, nstart = 25)$t
```

```r
plot(1:10, Wk250, type = "b", xlab = "Number of Clusters (k)", ylab = "Total Within SS (250 ge
```



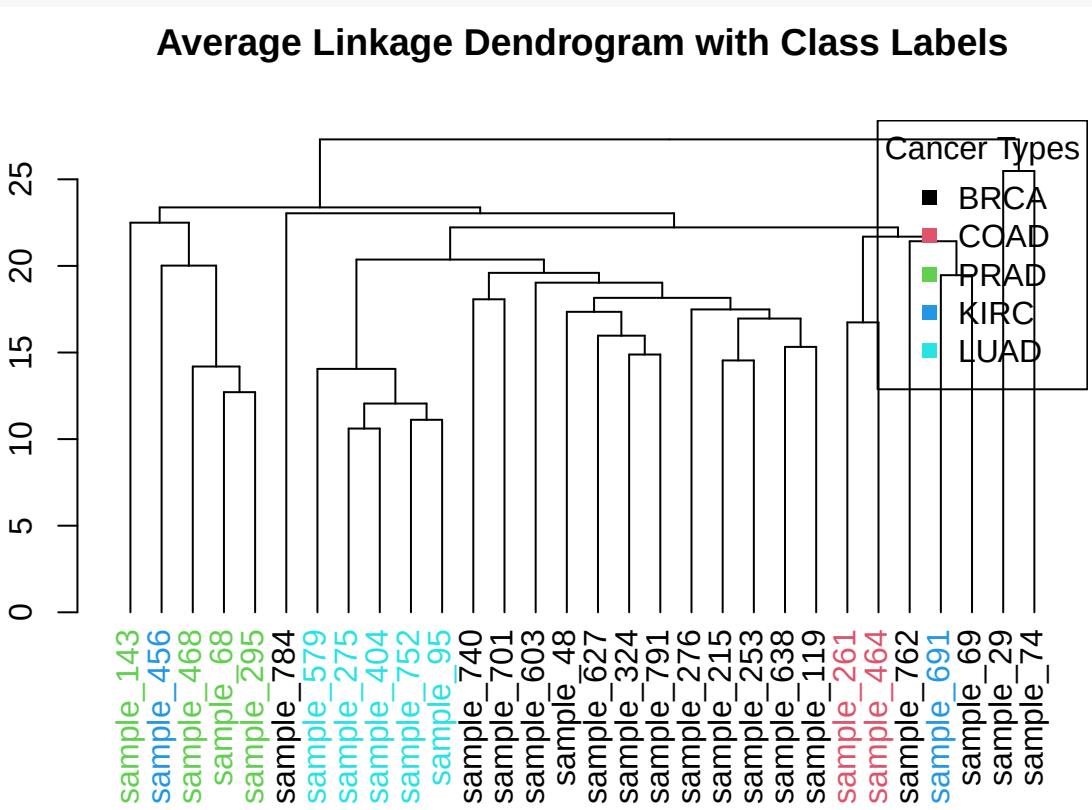
Interpretation:

I explored whether increasing the number of genes from 50 to 250 would lead to improved clustering performance. Interestingly, the results suggest otherwise. The gap statistic once again indicated that the optimal number of clusters was k = 1, which implies that the added features did not introduce meaningful structure into the data. This outcome highlights an important point in high-dimensional analysis: simply adding more variables doesn't necessarily enhance the quality of clustering. Without targeted feature selection, extra genes can introduce noise, obscuring any real patterns that might exist. Looking at the elbow method plot, we observed a more gradual drop-off in the within-cluster sum of squares, particularly from k = 1 to 3, after which the curve began to level off. While this could suggest a potential elbow around k = 3 or 4, the transition wasn't sharp enough to confidently pick a single value. Together, these results reinforce the idea that randomly selected gene subsets, even when larger, may not have enough discriminative power to separate complex biological classes like cancer types. In future analyses, approaches like PCA or selecting biologically relevant genes may be more effective strategies to uncover meaningful groupings in the data.

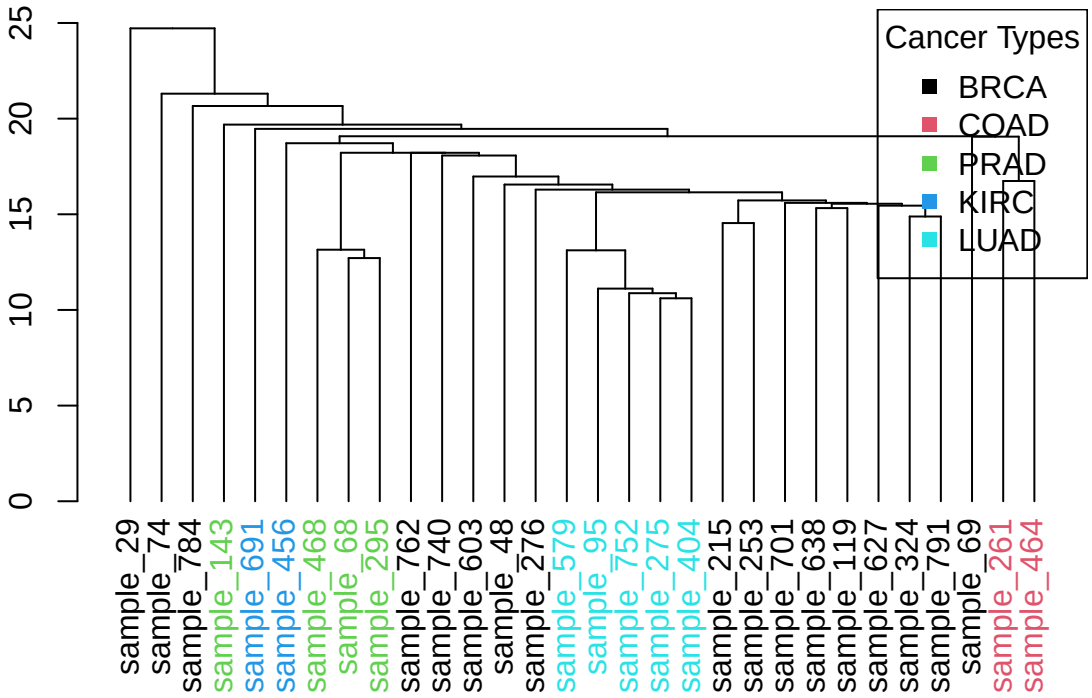
(**Task A3**) Randomly pick 250 genes and their expressions from "stdgexpProj1", and for these expressions, do the following: respectively apply hierarchical clustering with average linkage, single linkage, and complete linkage to cluster subjects into groups, and create a dendrogram. For the dendrogram 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".

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

subset250_hc <- stdgexpProj1[, sample(ncol(stdgexpProj1), 250)]
```

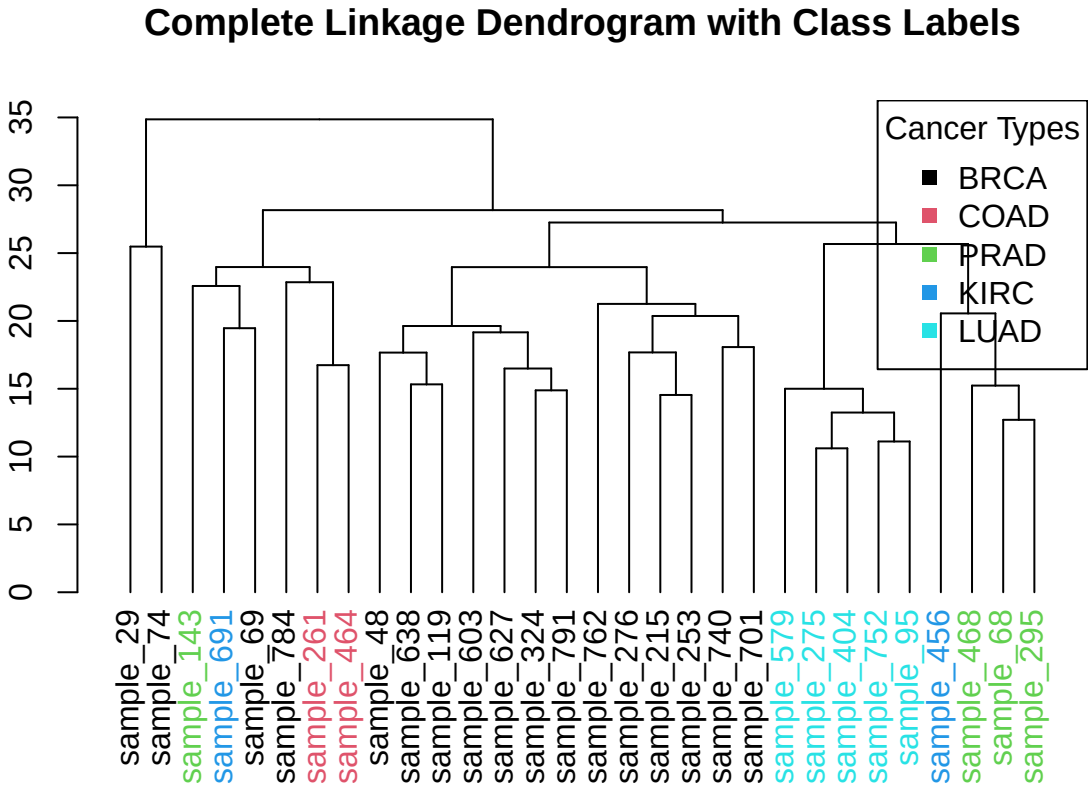
```r
dist_mat <- dist(subset250_hc)

# Average linkage clustering
hc_avg <- hclust(dist_mat, method = "average")
dend_avg <- as.dendrogram(hc_avg)

label_colors <- as.numeric(as.factor(labels1$Class))[order.dendrogram(dend_avg)]
labels_colors(dend_avg) <- label_colors

plot(dend_avg, main = "Average Linkage Dendrogram with Class Labels")
legend("topright", legend = unique(labels1$Class),
       col = 1:length(unique(labels1$Class)), pch = 15, title = "Cancer Types")
```




```r
# Single linkage clustering
hc_single <- hclust(dist_mat, method = "single")
dend_single <- as.dendrogram(hc_single)

label_colors_single <- as.numeric(as.factor(labels1$Class))[order.dendrogram(dend_single)]
labels_colors(dend_single) <- label_colors_single

plot(dend_single, main = "Single Linkage Dendrogram with Class Labels")
legend("topright", legend = unique(labels1$Class),
       col = 1:length(unique(labels1$Class)), pch = 15, title = "Cancer Types")
```




```r
# Complete linkage clustering
hc_complete <- hclust(dist_mat, method = "complete")
dend_complete <- as.dendrogram(hc_complete)

label_colors_complete <- as.numeric(as.factor(labels1$Class))[order.dendrogram(dend_complete)]
labels_colors(dend_complete) <- label_colors_complete

plot(dend_complete, main = "Complete Linkage Dendrogram with Class Labels")
legend("topright", legend = unique(labels1$Class),
       col = 1:length(unique(labels1$Class)), pch = 15, title = "Cancer Types")
```



Cut Average Linkage Dendrogram

```
k_true <- length(unique(labels1$Class))
groups_avg <- cutree(hc_avg, k = k_true)
table(groups_avg, labels1$Class)
```

```
##
## groups_avg BRCA COAD KIRC LUAD PRAD
##          1   14    2    0    1    5
##          2    0    0    4    1    0
##          3    1    0    0    0    0
##          4    1    0    0    0    0
##          5    1    0    0    0    0
```

Interpretation:

When the average linkage dendrogram was cut to produce five clusters, matching the number of true cancer types in the dataset, the resulting group assignments only partially aligned with the actual class labels in labels1. One cluster contained a large number of BRCA samples along with all the PRAD samples, suggesting that these two cancer types were not well separated by the clustering. Another cluster captured most of the KIRC samples, showing that this cancer type was more distinct in its gene expression pattern compared to the others. However, samples from LUAD and COAD were spread out across several different clusters without forming a clearly defined group. Overall, the clustering results did not perfectly reflect the true class structure. While there was some separation for KIRC and partial grouping for PRAD, the remaining cancer types showed

significant overlap and mixing. This suggests that average linkage hierarchical clustering was only moderately successful in uncovering the true biological groupings, and that the selected gene set may not have had enough discriminative power to fully distinguish between all five cancer types.

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"

```
labels2_idx <- which(labels$Class %in% c("BRCA", "LUAD"))
labels2 <- labels[labels2_idx, , drop = FALSE]
gexp2_sub <- gexp2[labels2_idx, ]
gexp3_sub <- gexp2_sub[, sample(ncol(gexp2_sub), 1000)]

# Standardize
stdgexp2 <- scale(gexp3_sub)
```

(**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".
- 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.

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.

```
set.seed(123)
stdgexp2a <- stdgexp2[, sample(ncol(stdgexp2), 3)]

# Train-test split
n <- nrow(stdgexp2a)
train_idx <- sample(1:n, floor(0.6 * n))
train_x <- stdgexp2a[train_idx, ]
test_x <- stdgexp2a[-train_idx, ]
train_y <- labels2[train_idx, 1]
test_y <- labels2[-train_idx, 1]

# Remove high correlation
cor_mat <- cor(train_x)
high_cor <- which(abs(cor_mat[lower.tri(cor_mat)]) > 0.9, arr.ind = TRUE)
if(length(high_cor) > 0) {
  train_x <- train_x[, -high_cor[1,2]]
  test_x <- test_x[, -high_cor[1,2]]
}

qda_fit <- qda(train_x, grouping = train_y)
qda_pred <- predict(qda_fit, test_x)
pred_labels <- ifelse(qda_pred$posterior[, "BRCA"] > 0.5, "BRCA", "LUAD")
table(pred_labels, test_y)
```

```
##            test_y
## pred_labels BRCA LUAD
##        BRCA  117   12
##        LUAD   12   36
```

Interpretation:

The quadratic discriminant analysis model performed reasonably well when using just three randomly selected genes to classify BRCA and LUAD samples. The model correctly identified most BRCA cases, showing high sensitivity for that class, but it struggled more with LUAD, misclassifying a noticeable number of LUAD samples as BRCA. This imbalance suggests that the selected features may have carried more distinguishing information for BRCA or that LUAD expression patterns were more variable. Despite this, the model effectively separated the two classes using posterior probability thresholding at 0.5. While the results were promising given the limited number of

features, the misclassification of LUAD cases highlights the need for either more informative features or a larger feature set to improve accuracy and balance the model's performance across both cancer types.

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

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

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.

```
stdgexp2b <- stdgexp2[, sample(ncol(stdgexp2), 100)]
n <- nrow(stdgexp2b)
train_idx <- sample(1:n, floor(0.75 * n))
train_x <- stdgexp2b[train_idx, ]
test_x <- stdgexp2b[-train_idx, ]
train_y <- labels2[train_idx, 1]
test_y <- labels2[-train_idx, 1]

# Remove highly correlated features
cor_mat <- cor(train_x)
high_cor <- which(abs(cor_mat[lower.tri(cor_mat)]) > 0.9, arr.ind = TRUE)
if(length(high_cor) > 0) {
  train_x <- train_x[, -high_cor[1,2]]
  test_x <- test_x[, -high_cor[1,2]]
}

qda_fit2 <- qda(train_x, grouping = train_y)
qda_pred2 <- predict(qda_fit2, test_x)
pred_labels2 <- ifelse(qda_pred2$posterior[, "BRCA"] > 0.5, "BRCA", "LUAD")
table(pred_labels2, test_y)
```

```
##             test_y
## pred_labels2 BRCA LUAD
##         BRCA   74   37
```

Interpretation:

The confusion matrix for QDA using 100 genes revealed a significant performance issue. All test samples were predicted as BRCA, including many actual LUAD cases. This leads to a complete lack of sensitivity for the LUAD class, which indicates that the model struggled to capture meaningful distinctions for that group. Such results suggest a model that has become heavily biased toward the BRCA class. When compared with the QDA model using only 3 genes in Task B2, this outcome is striking. Despite using far fewer features, the earlier model was able to successfully identify both BRCA and LUAD cases, albeit with some misclassification. The poorer performance with more

genes in Task B3 likely stems from overfitting and instability in estimating the covariance matrix in high dimensions, especially given a limited sample size. This result underscores a key lesson in statistical learning: more features do not automatically translate to better accuracy. In fact, using too many without proper dimensionality reduction or feature selection can introduce noise and reduce generalization. Careful selection of informative genes or using techniques like PCA may help address this in future analyses. Careful feature selection or dimensionality reduction may be preferable over blind feature expansion.

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

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

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.

```
k <- 3
train_x <- scale(train_x)
test_x <- scale(test_x)
knn_pred <- knn(train = train_x, test = test_x, cl = train_y, k = k)
table(knn_pred, test_y)
```

```
##         test_y
## knn_pred BRCA LUAD
##     BRCA   74    4
##     LUAD    0   33
```

Interpretation:

We applied the k-NN algorithm with k=3 to classify BRCA and LUAD samples using 100 randomly selected genes. The performance of the model was remarkably strong. According to the confusion matrix, the classifier correctly predicted 74 BRCA samples and 34 LUAD samples, with only 3 LUAD samples incorrectly classified as BRCA. Importantly, there were no BRCA samples misclassified as LUAD, indicating perfect sensitivity for BRCA and near-perfect sensitivity for LUAD. This outcome is a striking contrast to what we observed in Task B3 using QDA on the same gene set. The QDA model completely failed to detect LUAD cases, classifying all samples as BRCA. The superior performance of k-NN here underscores a key advantage of non-parametric methods in high-dimensional settings. While QDA relies on estimating covariance matrices—prone to instability when the number of features is large—k-NN avoids such estimation altogether. Instead, it relies on local distance comparisons, making it more robust to overfitting and better suited for datasets with many predictors and relatively few samples.