hw2

Khoa Vu 705600710

In this homework, we will use the NCI60 cancer cell line microarray data, which consist of 6,830 gene expression measurements on 64 cancer cell lines. You need to install the ISLR package in R by

```
## Installing package into '/home/kvu1702/R/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library("ISLR")
```

Then you have the object `NCI60` in your R workspace. The object is a list of two elements `NCI60$data` and `NCI60$labs`, where `NCI60$data` is a numeric matrix of 64 rows (i.e., cancer cell lines) and 6, 830 columns (i.e., genes) and `NCI60$labs` is a 64-element character vector containing the cancer types of the cell lines. For example, you can explore the data using the following commands.

```
## [1]   64 6830
```

```
## [1] "CNS"    "CNS"    "CNS"    "RENAL"  "BREAST" "CNS"
```

```
##
##      BREAST         CNS       COLON K562A-repro K562B-repro    LEUKEMIA
##           7           5           7           1           1           6
## MCF7A-repro MCF7D-repro    MELANOMA       NSCLC     OVARIAN    PROSTATE
##           1           1           8           9           6           2
##       RENAL     UNKNOWN
##           9           1
```

Problem 1: Hierarchical Clustering (20 pts) 1. Perform hierarchical clustering on the 64 cancer cell lines using all the 6,830 genes. Use the

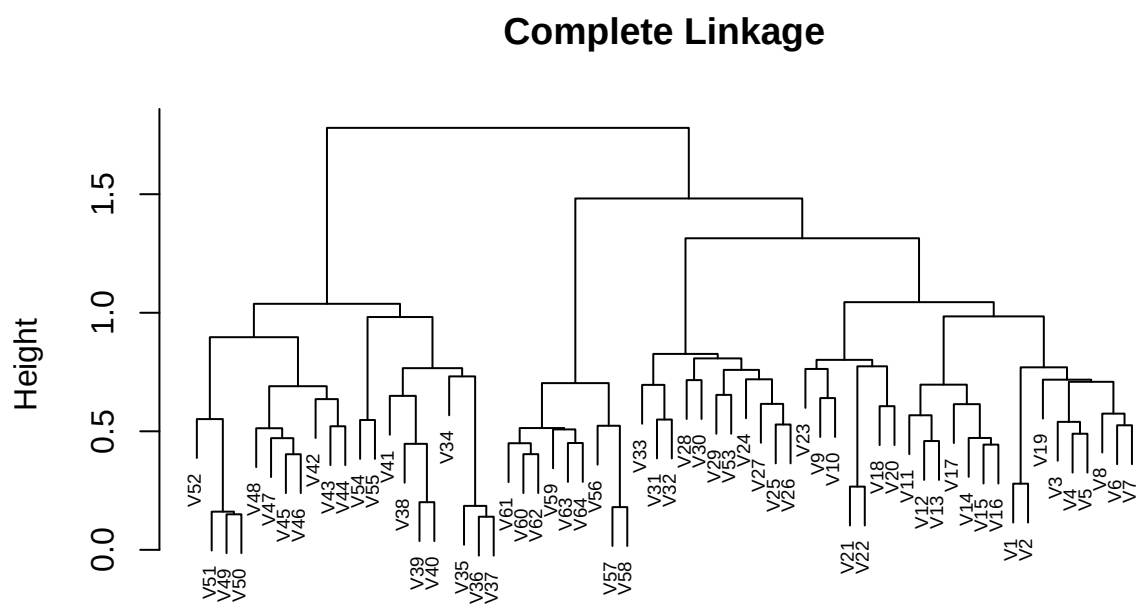
$$d_{ij} = \frac{1 - \text{Pearson correlation between cell lines i and j}}{2}$$

as the distance measure. Plot three dendrograms using complete, single and average linkage. You can use R functions hclust() and dist() to do the hierarchical clustering.

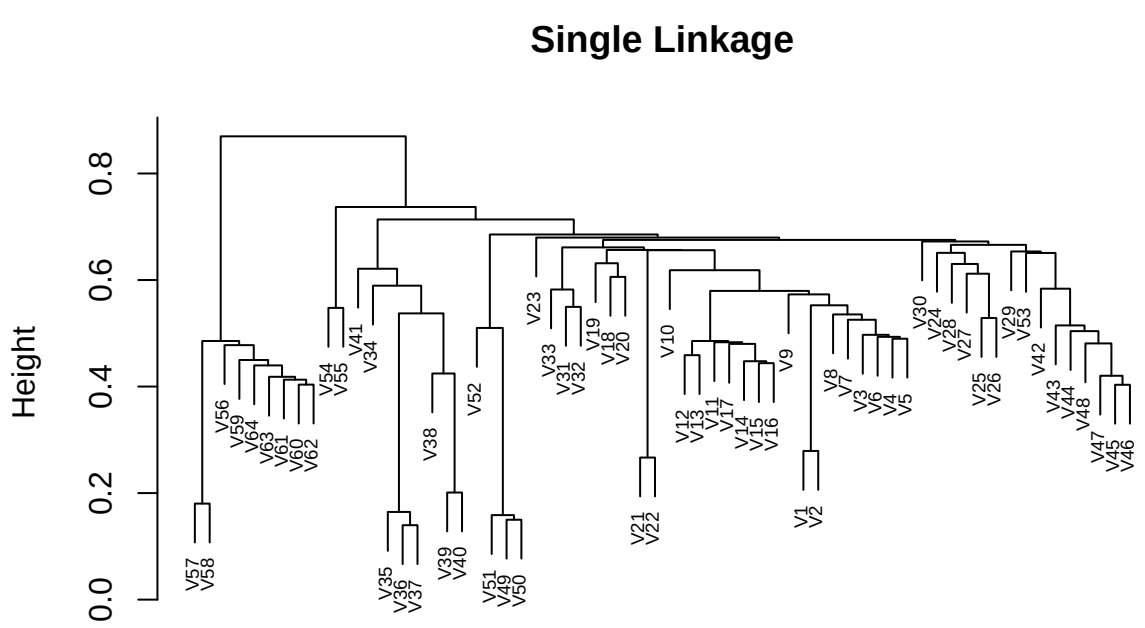
Solution:

```
#We first calculate our distance matrix
#In order to calculate our distance matrix, we also need to calculate the
#correlation matrix between all 64 cell lines using all 6830 genes
corr <- cor(t(NCI60$data))
distance <- dist((1 - corr)/2)

#Compute all three kinds of linkages: complete, single, and average
hclust_complete <- hclust(distance, method = "complete")
hclust_single <- hclust(distance, method = "single")
hclust_average <- hclust(distance, method = "average")
```

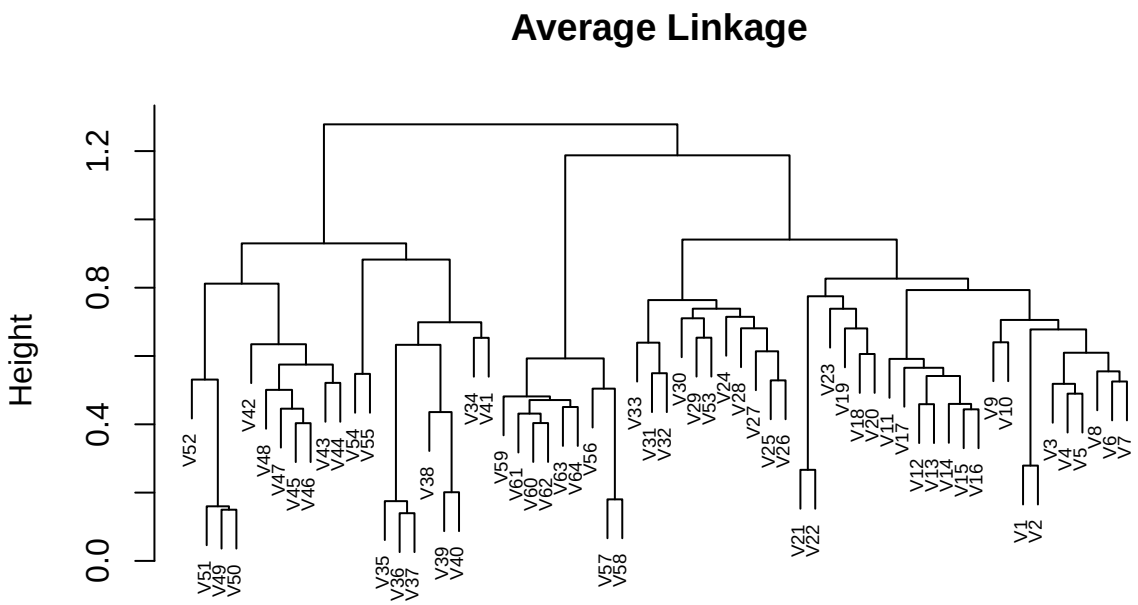
```r
#Plotting the clustering results
plot(hclust_complete, main = "Complete Linkage", xlab = "", sub = "", cex = 0.6)
```




```r
plot(hclust_single, main = "Single Linkage", xlab = "", sub = "", cex = 0.6)
```



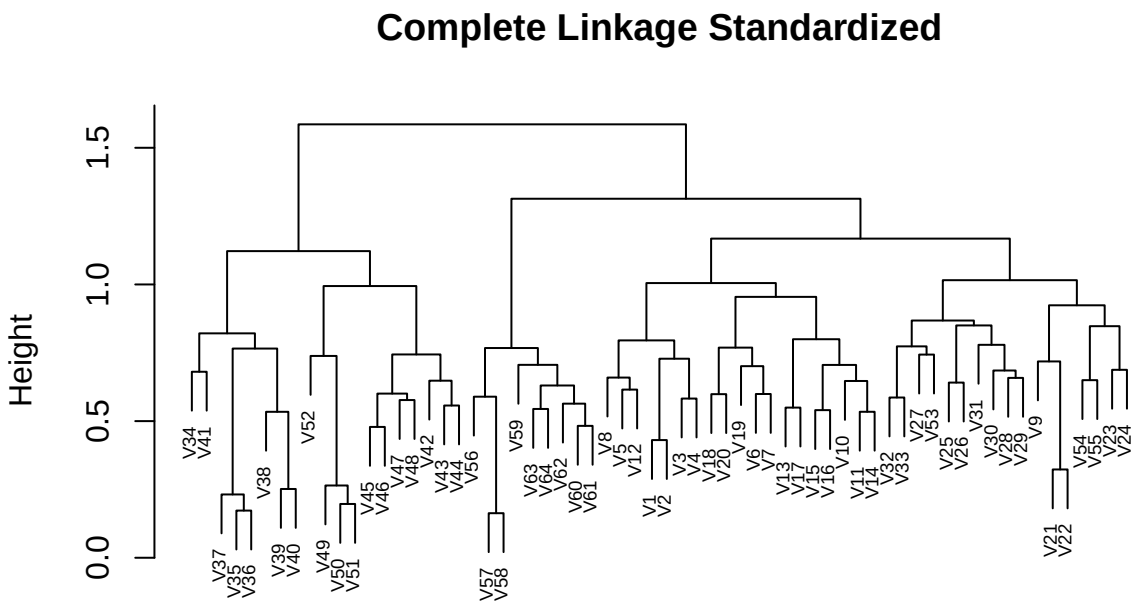

```r
plot(hclust_average, main = "Average Linkage", xlab = "", sub = "", cex = 0.6)
```



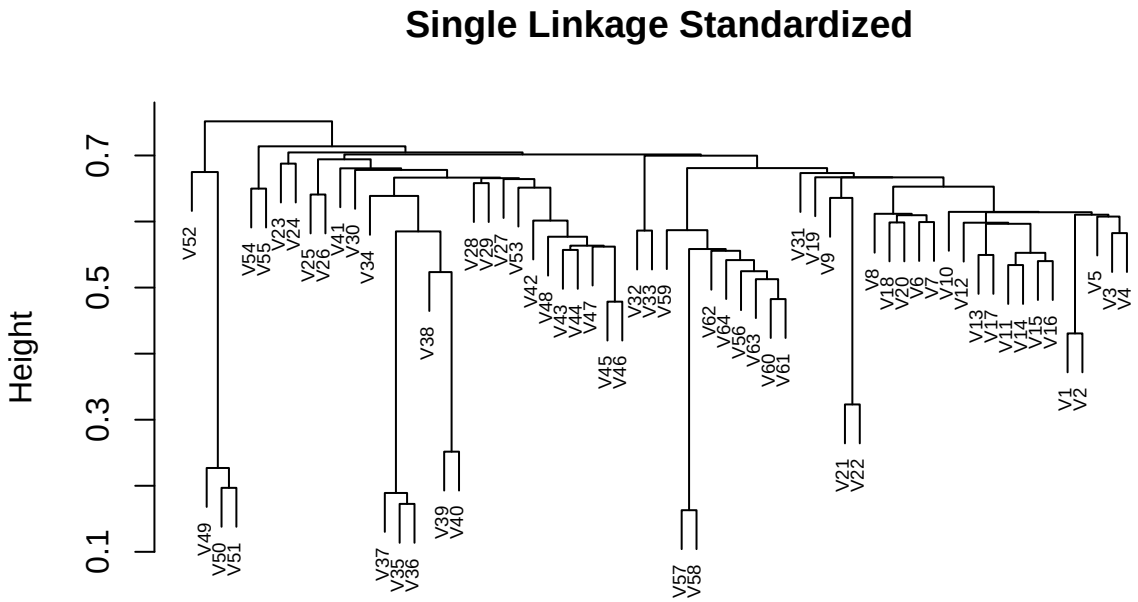
2. Standardize the data such that every gene will have mean zero and standard deviation one across the 64 samples. This standardization will make every gene on the same scale. Then perform the hierarchical clustering as in Problem 1.1. Plot the three resulting dendrograms.

Solution:

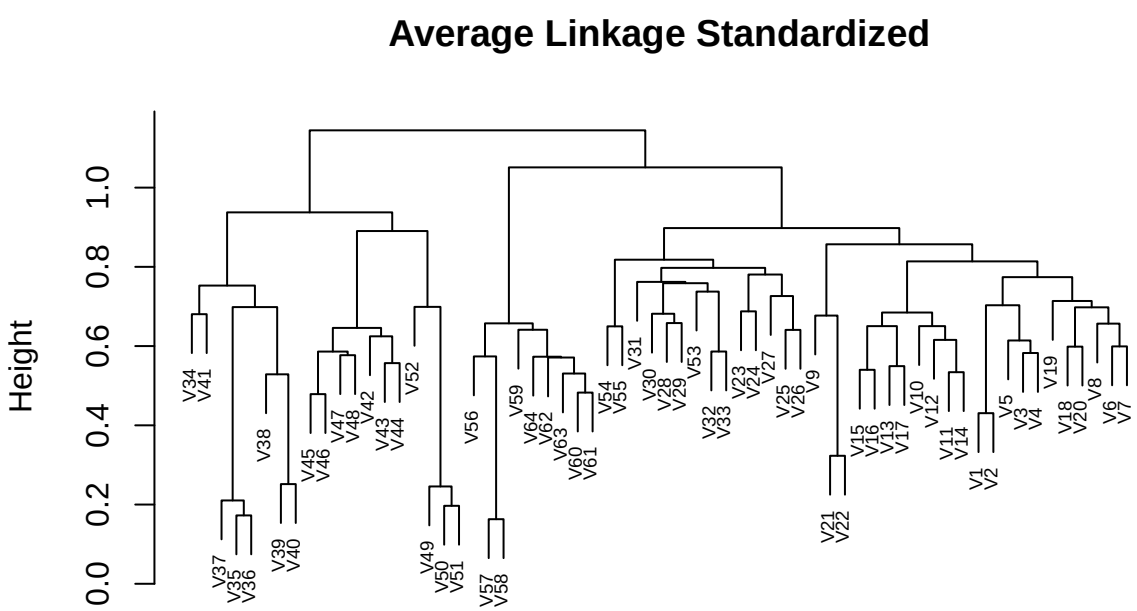
```
#We first standardize the data using scale()
NCI60$data_standardized <- scale(NCI60$data, center=TRUE, scale=TRUE)

#We repeat our code from problem 1.1
#We first calculate our distance matrix
#In order to calculate our distance matrix, we also need to calculate the
#correlation matrix between all 64 cell lines using all 6830 genes
corr_standardized <- cor(t(NCI60$data_standardized))
dist_standardized <- dist((1 - corr_standardized)/2)

#Compute all three kinds of linkages: complete, single, and average
hclust_complete_standardized <- hclust(dist_standardized, method = "complete")
hclust_single_standardized <- hclust(dist_standardized, method = "single")
hclust_average_standardized <- hclust(dist_standardized, method = "average")

#Plotting the clustering results
plot(hclust_complete_standardized, main = "Complete Linkage Standardized",
     xlab = "", sub = "", cex = 0.6)
```




```r
plot(hclust_single_standardized, main = "Single Linkage Standardized",
     xlab = "", sub = "", cex = 0.6)
```




```r
plot(hclust_average_standardized, main = "Average Linkage Standardized",
     xlab = "", sub = "", cex = 0.6)
```



3. Perform quantile normalization on the original data, so that all the 6,830 genes have the same empirical distribution in every cell line. Then perform the hierarchical clustering as in Problem 1.1. Plot the three resulting dendrograms.

Solution:

```
#eval|echo=FALSE
#Installing the required packages
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("preprocessCore")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: https://cloud.r-project.org
```

```
## Bioconductor version 3.20 (BiocManager 1.30.25), R 4.4.2 (2024-10-31)
```

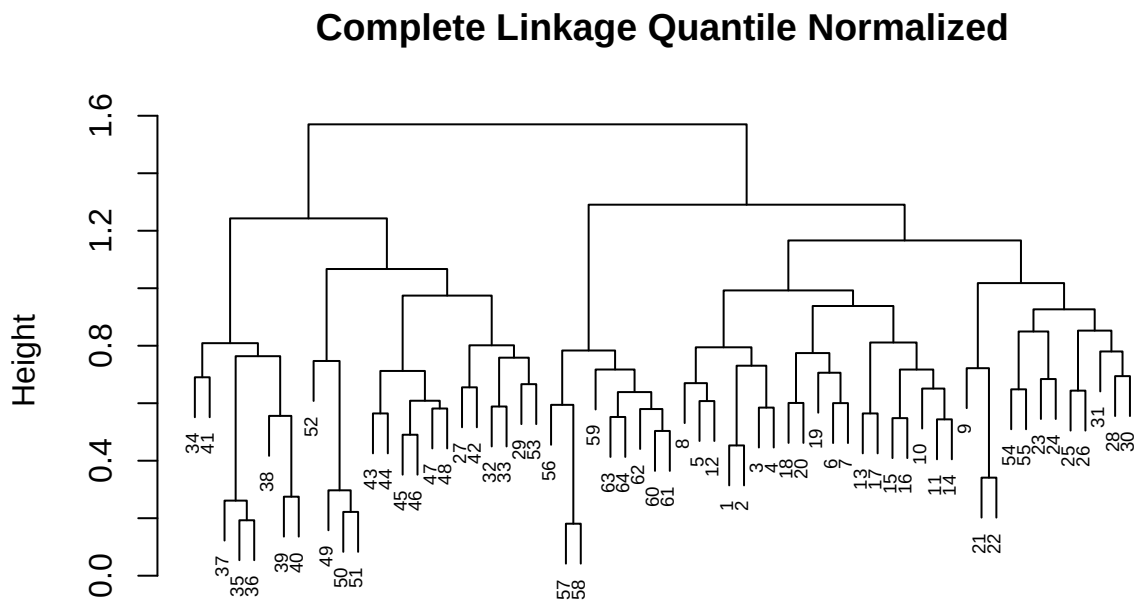
```
## Warning: package(s) not installed when version(s) same as or greater than current; use
##   `force = TRUE` to re-install: 'preprocessCore'
```

```
## Installation paths not writeable, unable to update packages
##   path: /usr/lib/R/library
##   packages:
##     boot, class, cluster, codetools, foreign, KernSmooth, lattice, MASS,
##     Matrix, nlme, nnet, rpart, spatial, survival
```

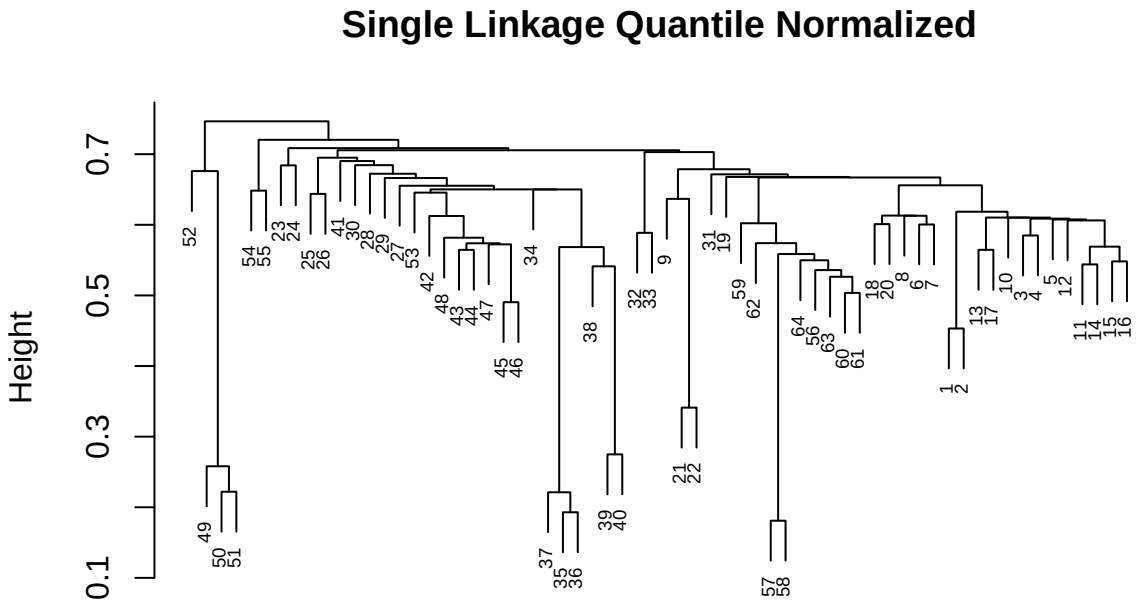
```
## Old packages: 'xml2'
```

```
library(preprocessCore)
```

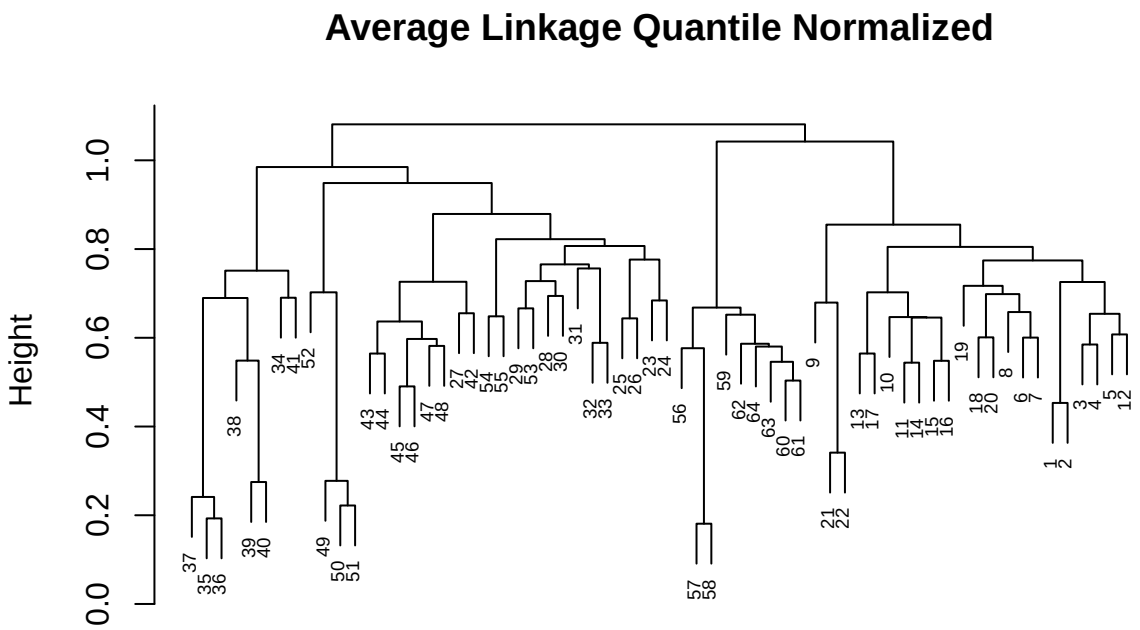
```
#We utilize the built in quantile normalization function from preprocessCore,
#quantile_normalize()
library(preprocessCore)

#We first perform quantile normalization on the data
NCI60$data_quantile_normalized <- normalize.quantiles(NCI60$data)

#We repeat our code from problem 1.1
#We first calculate our distance matrix
#In order to calculate our distance matrix, we also need to calculate the
#correlation matrix between all 64 cell lines using all 6830 genes
corr_quantile_normalized <- cor(t(NCI60$data_quantile_normalized))
dist_quantile_normalized <- dist((1 - corr_quantile_normalized)/2)

#Compute all three kinds of linkages: complete, single, and average
hclust_complete_quantile_normalized <- hclust(dist_quantile_normalized,
                                              method = "complete")
hclust_single_quantile_normalized <- hclust(dist_quantile_normalized,
                                            method = "single")
hclust_average_quantile_normalized <- hclust(dist_quantile_normalized,
                                             method = "average")

#Plotting the clustering results
plot(hclust_complete_quantile_normalized,
     main = "Complete Linkage Quantile Normalized",
     xlab = "", sub = "", cex = 0.6)
```




```
plot(hclust_single_quantile_normalized,
     main = "Single Linkage Quantile Normalized",
     xlab = "", sub = "", cex = 0.6)
```




```
plot(hclust_average_quantile_normalized,
     main = "Average Linkage Quantile Normalized",
     xlab = "", sub = "", cex = 0.6)
```



4. Cut each of the six dendrograms generated in Problems 1.1, 1.2 and 1.3 into four clusters using R function cutree(), which takes the output of hclust() as input. Compare the nine sets of clustering results, which one looks most reasonable to you?

Solution:

```
#eval|echo=FALSE
#Installing the required packages
install.packages("cluster")
```

```
## Installing package into '/home/kvu1702/R/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
#Importing the package
library(cluster)

#Problem 1.1
cut_hcluster_complete <- cutree(hclust_complete, k = 4)
cut_hcluster_single <- cutree(hclust_single, k = 4)
cut_hcluster_average <- cutree(hclust_average, k = 4)

#We use the average of the silhouette score from the package cluster to assess the quality of our clust
cut_hcluster_complete_silhoutte_average <- mean(
  silhouette(cut_hcluster_complete, distance)[,3])
cut_hcluster_single_silhoutte_average <- mean(
  silhouette(cut_hcluster_single, distance)[,3])
cut_hcluster_average_silhoutte_average <- mean(
  silhouette(cut_hcluster_average, distance)[,3])
```

```r
#Problem 1.2
cut_hcluster_complete_standardized <- cutree(
  hclust_complete_standardized, k = 4)
cut_hcluster_single_standardized <- cutree(
  hclust_single_standardized, k = 4)
cut_hcluster_average_standardized <- cutree(
  hclust_average_standardized, k = 4)

cut_hcluster_complete_standardized_silhoutte_average <- mean(
  silhouette(cut_hcluster_complete_standardized, distance)[,3])
cut_hcluster_single_standardized_silhoutte_average <- mean(
  silhouette(cut_hcluster_single_standardized, distance)[,3])
cut_hcluster_average_standardized_silhoutte_average <- mean(
  silhouette(cut_hcluster_average_standardized, distance)[,3])

#Problem 1.3
cut_hcluster_complete_quantile_normalized <- cutree(
  hclust_complete_quantile_normalized, k = 4)
cut_hcluster_single_quantile_normalized <- cutree(
  hclust_single_quantile_normalized, k = 4)
cut_hcluster_average_quantile_normalized <- cutree(
  hclust_average_quantile_normalized, k = 4)

cut_hcluster_complete_quantile_normalized_silhoutte_average <- mean(
  silhouette(cut_hcluster_complete_quantile_normalized, distance)[,3])
cut_hcluster_single_quantile_normalized_silhoutte_average <- mean(
  silhouette(cut_hcluster_single_quantile_normalized, distance)[,3])
cut_hcluster_average_quantile_normalized_silhoutte_average <- mean(
  silhouette(cut_hcluster_average_quantile_normalized, distance)[,3])

average_silhouette_scores <- c(cut_hcluster_complete_silhoutte_average, #1
  cut_hcluster_single_silhoutte_average, #2
  cut_hcluster_average_silhoutte_average, #3
  cut_hcluster_complete_standardized_silhoutte_average, #4
  cut_hcluster_single_standardized_silhoutte_average, #5
  cut_hcluster_average_standardized_silhoutte_average, #6
  cut_hcluster_complete_quantile_normalized_silhoutte_average, #7
  cut_hcluster_single_quantile_normalized_silhoutte_average, #8
  cut_hcluster_average_quantile_normalized_silhoutte_average) #9

#We compare the average silhouette scores of the three clustering methods
#for each of the three data preprocessing methods, taking the max silhouette score
index_max_silhouette_score <- which.max(average_silhouette_scores)
index_max_silhouette_score
```

```
## [1] 6
```

```r
#The preprocessing and linkage method with the highest silhouette score is
#standardized average linkage (the sixth index of our list
#average_silhouette_scores)
cut_hcluster_average_standardized_silhoutte_average
```

```
## [1] 0.2926966
```

Using the silhouette score as the comparison and reasonability metric, the standardized, average linkage

method has the highest silhouette score of 0.29 and is the most reasonable clustering method.

5. Based on the results in Problem 1.4, please explain whether you think standardization should be performed, whether you think quantile normalization is reasonable, and which linkage is the most suitable in this case.

Solution:

Based on the results in Problem 1.4, I think standardization should be performed, and average linkage should be used as it results in the highest silhouette score of 0.29. Quantile normalization is also reasonable, resulting in a silhouette score of 0.267 for complete and average linkages, higher than 0.25 for complete and average linkages on the original data.

Problem 2: K-means Clustering (20 pts)

Perform K-means clustering on the the 64 cancer cell lines using all the 6,830 genes. Use the Euclidean distance as the distance measure. Set K = 4. Use 20 random initial starts in the R function kmeans() by setting the nstart argument. Set the random seed as 1 every time before you run the algorithm using

```
set.seed(1)
```

so your results can be reproducible.

1. Perform K-means clustering on the original data

Solution:

```
set.seed(1)

#Importing factoextra package to plot gap_statistic
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
kmeans_original <- kmeans(NCI60$data, centers = 4, nstart = 20)
kmeans_original_silhoutte_average <- mean(silhouette(kmeans_original$cluster,
                                                     dist(NCI60$data))[,3])

#Visualizing the cluster results
fviz_cluster(kmeans_original,
             NCI60$data,
             geom = "point",
             ellipse.type = "convex",
             main = "K-Means Clustering with K = 4",
             ggtheme = theme_bw()
             )
```


```
kmeans_original_silhoutte_average
```

```
## [1] 0.1161059
```

2. Perform K-means clustering on the standardized data from Problem 1.2.

Solution:

```
set.seed(1)
kmeans_standardized <- kmeans(NCI60$data_standardized, centers = 4, nstart = 20)
kmeans_standardized_silhoutte_average <- mean(
  silhouette(kmeans_standardized$cluster, dist(NCI60$data_standardized))[,3])

#Visualizing the cluster results
fviz_cluster(kmeans_standardized,
             NCI60$data_standardized,
             geom = "point",
             ellipse.type = "convex",
             main = "K-Means Clustering with K = 4 and Data Standardization",
             ggtheme = theme_bw()
             )
```

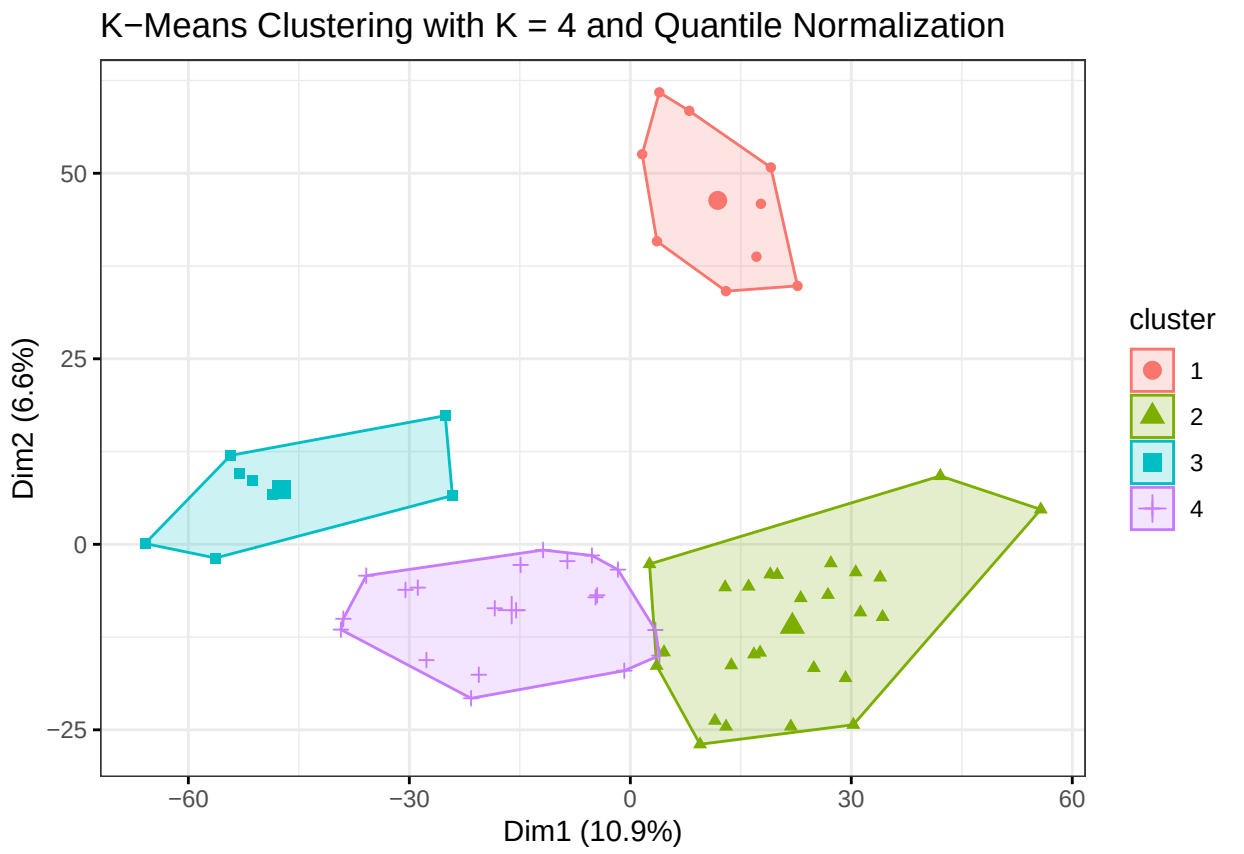

```
kmeans_standardized_silhoutte_average
```

```
## [1] 0.08040558
```

3. Perform K-means clustering on the quantile normalized data from Problem 1.3

Solution:

```
set.seed(1)
kmeans_quantile_normalized <- kmeans(NCI60$data_quantile_normalized,
                                     centers = 4, nstart = 20)
kmeans_quantile_normalized_silhoutte_average <- mean(
  silhouette(kmeans_quantile_normalized$cluster,
             dist(NCI60$data_quantile_normalized))[,3])

#Visualizing the cluster results
fviz_cluster(kmeans_quantile_normalized,
             NCI60$data_quantile_normalized,
             geom = "point",
             ellipse.type = "convex",
             main = "K-Means Clustering with K = 4 and Quantile Normalization",
             ggtheme = theme_bw()
             )
```




```
kmeans_quantile_normalized_silhoutte_average
```

```
## [1] 0.07662218
```

4. Compare the three sets of clustering results from Problems 2.1, 2.2 and 2.3. Also compare them to the most reasonable clustering result you obtained in Problem 1.4.

Using the silhouette score as the comparison and reasonability metric, the K means algorithm applied to the original data is the most reasonable clustering method. This is opposite to the hierarchical clustering results, where the standardized data had the highest silhouette score. However, the silhouette score for the K-means algorithm overall is lower compared to hierarchical clustering regardless of data preprocessing steps, meaning hierarchical clustering is a better method for this dataset.

Solution:

5. Based on the results in Problem 2.4, please explain whether you think standardization should be performed for K-means, whether you think quantile normalization should be used before K-means, and which clustering method, K-means with Euclidean distance or hierarchical clustering with correlation distance, performs better in this application.

Solution:

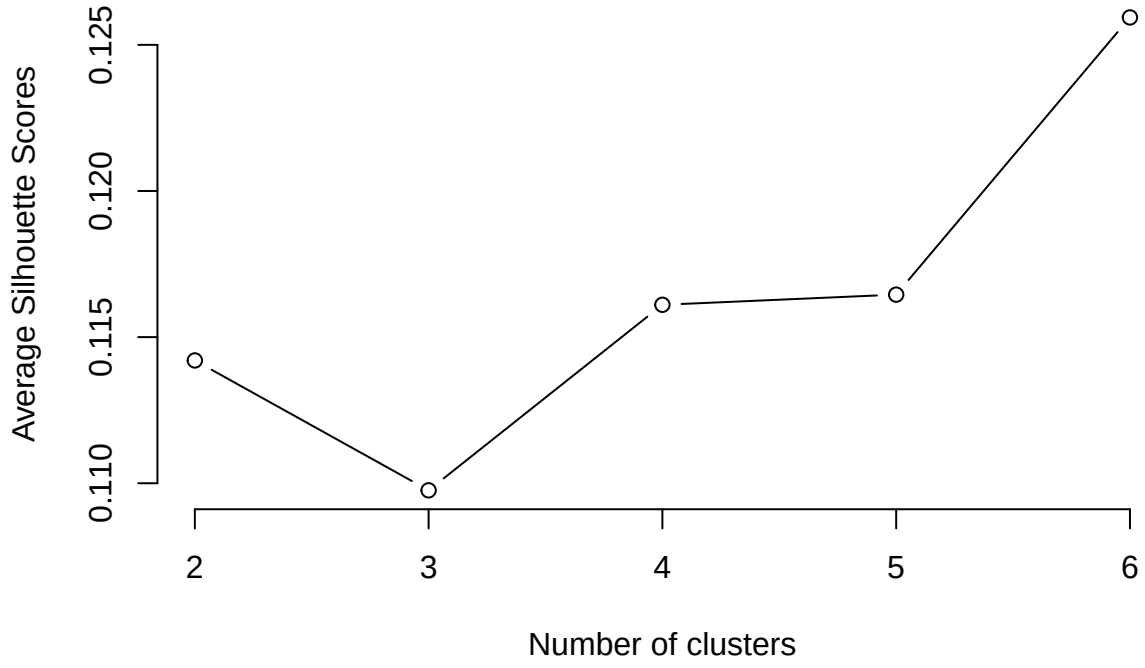
As stated in problem 2.4, because the silhouette score for the K-means algorithm without standardization or quantile normalization is the highest, it would be best not to perform standardization or quantile normalization for K-means. The silhouette scores for hierarchical clustering are higher overall than K-means clustering, making it more suitable for this dataset, at least for k = 4.

Problem 3: Choose K in K-means Clustering (10 pts) Choose the number of clusters K for the K-means clustering on the 64 samples using the original data and the same settings (20 random starts and random seed 1) as in Problem 2.1. Use the following two approaches.

1. Use the R function silhouette() in the cluster package. Show the silhouette plot fors K = 2, 3, 4, 5 and 6. Which value will you choose for K based on the plots?

Solution:

```r
set.seed(1)

#Define k <- 2, 3, 4, 5, 6
k <- 2:6

#We define a function that calculates the silhouette score for
#each chosen cluster number

sihoutte_score <- function(k){
  kmeans_original <- kmeans(NCI60$data, centers = k, nstart = 20)
  sihoutte_scores <- silhouette(kmeans_original$cluster, dist(NCI60$data))
  mean(sihoutte_scores[,3])
}

#We calculate the silhouette score for each k
silhouette_scores <- sapply(k, sihoutte_score)
plot(k, type='b', silhouette_scores,
     xlab='Number of clusters',
     ylab='Average Silhouette Scores',
     frame=FALSE)
```



Based on the silhouette scores for varying k clusters, the best value for k is 6, with the highest silhouette score of ~0.13. This, however, is still far from the silhouette scores achieved by hierarchical clustering.

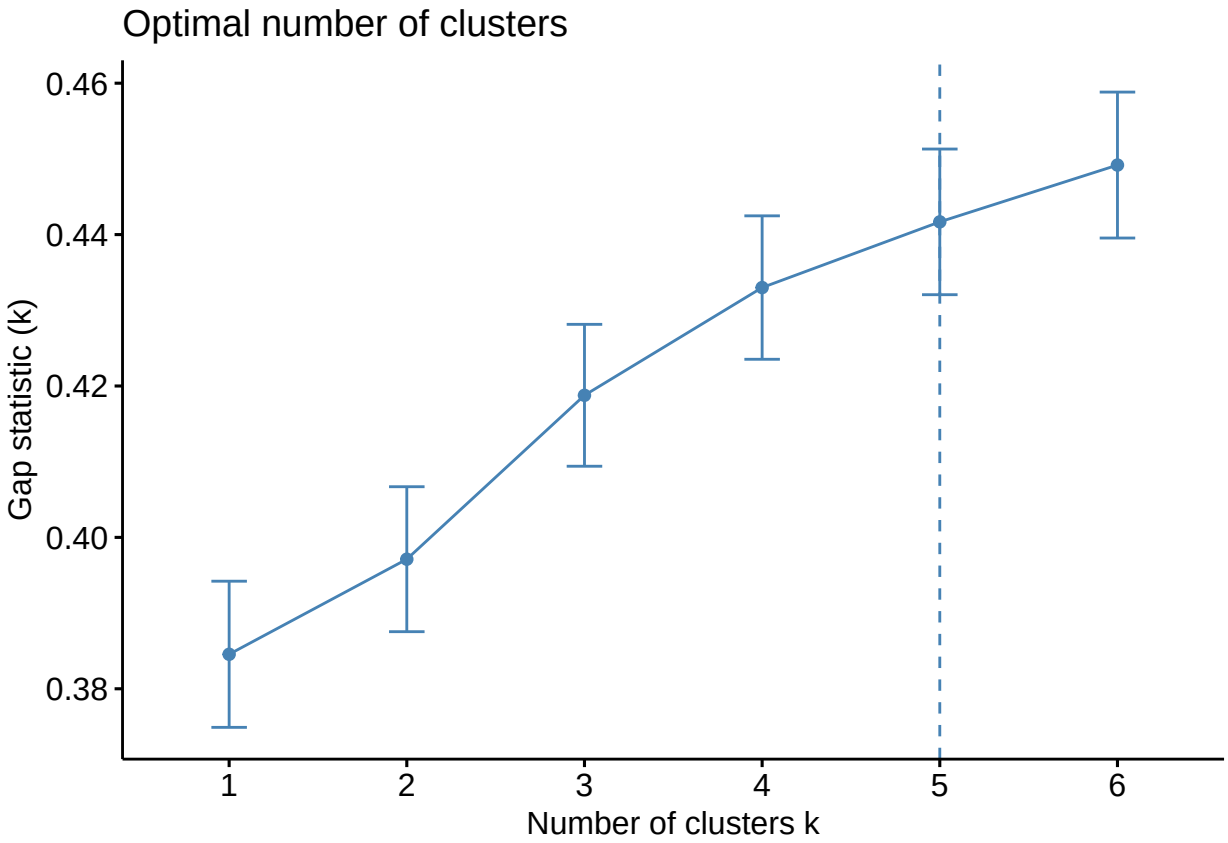
2. Use the R function clusGap() for computing the gap statistic in the cluster package. Show the gap statistic plot fors K = 2, 3, 4, 5 and 6. Which value will you choose for K based on the plot? *Hint: Follow the examples in the help files of these R functions.*

Solution:

```
#eval|echo=FALSE
#Installing the required packages
install.packages("factoextra")
```

```
## Installing package into '/home/kvu1702/R/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
set.seed(1)

#Importing factoextra package to plot gap_statistic
library(factoextra)

hcluster_gap <- clusGap(NCI60$data,
                        FUN = kmeans,
                        nstart = 20,
                        K.max = 6,
                        B = 100)
gap_statistic <- hcluster_gap$Tab

fviz_gap_stat(hcluster_gap)
```



Based on the gap statistic for the varying k clusters, the best value for k is 5, one less k then the value

derived in question 3.1.

Problem 4: Tight Clustering Algorithm (10 pts)

Apply the Tight Clustering algorithm in the R package tightClust to find 4 tight clusters among the 64 samples. Interpret the results. Are the results reasonable and more meaningful than the K-means clustering results?

Solution:

```
#eval|echo=FALSE
#Installing the required packages
install.packages("tightClust")
```

```
## Installing package into '/home/kvu1702/R/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
set.seed(1)

#Importing the package
library(tightClust)
library(factoextra)

#Calculating 4 tight clusters
k <- 4
#It is recommended k.min = target + 5
tight_clusters <- tight.clust(NCI60$data, target = k, k.min = k + 5)
```

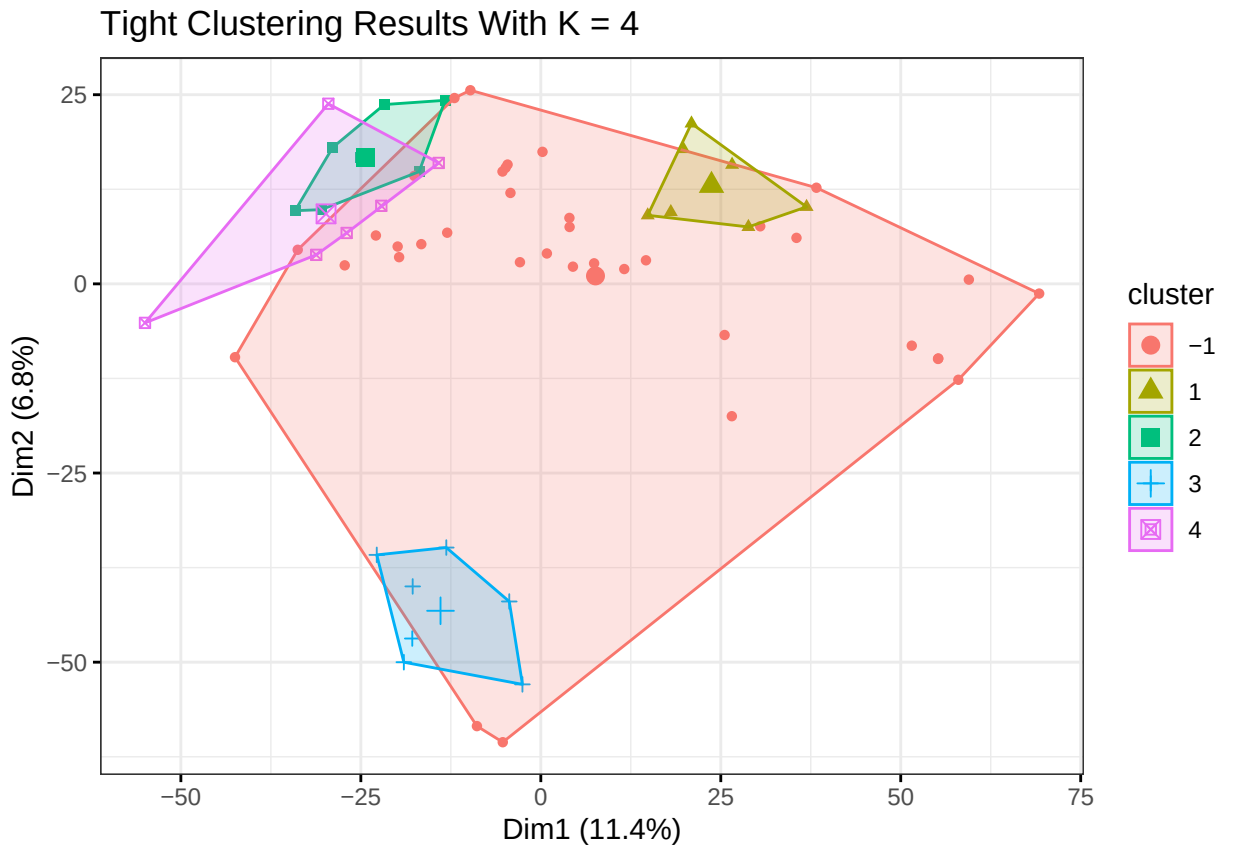
```
## Number of points: 64     Dimension: 6830
##
## Looking for tight cluster 1 ...
## k = 9
## k = 10
## 1 tight cluster(s) found!
## Cluster size: 7  Remaining number of points: 57
##
## Looking for tight cluster 2 ...
## k = 8
## k = 9
## 2 tight cluster(s) found!
## Cluster size: 7  Remaining number of points: 50
##
## Looking for tight cluster 3 ...
## k = 7
## k = 8
## 3 tight cluster(s) found!
## Cluster size: 7  Remaining number of points: 43
##
## Looking for tight cluster 4 ...
## k = 6
## k = 7
## 4 tight cluster(s) found!
## Cluster size: 6  Remaining number of points: 37
```

```
#Calculating the silhouette score
average_silhouette_score_tight_clusters <- mean(
  silhouette(tight_clusters$cluster, dist(NCI60$data))[, 3])
```

```
average_silhouette_score_tight_clusters
```

```
## [1] 0.01841897
```

```
#Visualizing the cluster results
fviz_cluster(tight_clusters,
             NCI60$data,
             geom = "point",
             ellipse.type = "convex",
             main = "Tight Clustering Results With K = 4",
             ggtheme = theme_bw()
             )
```



Based on the graph of the results, the tight clustering results do not seem very reasonable, with outliers (cluster = -1) all over the place despite being very close to established clusters. However, the 2D distance could not be meaningful due to the high dimensionality of the original data. Similarly, there appear to be overlapping clusters, but this could also be due to the curse of dimensionality. The silhouette score is also very low, much lower than previous methods, resulting in a less meaningful result than the K-means clustering results.

Problem 5: Justification of the K-means Algorithm (10 pts)

Solution: See attached PDF.

Problem 6: Practice of the Hierarchical Clustering (10 pts)

Suppose that we have four observations, for which we have a dissimilarity matrix:

[0 0.3 0.4 0.7]

[0.3 0 0.5 0.8]

[0.4 0.5 0 0.9]

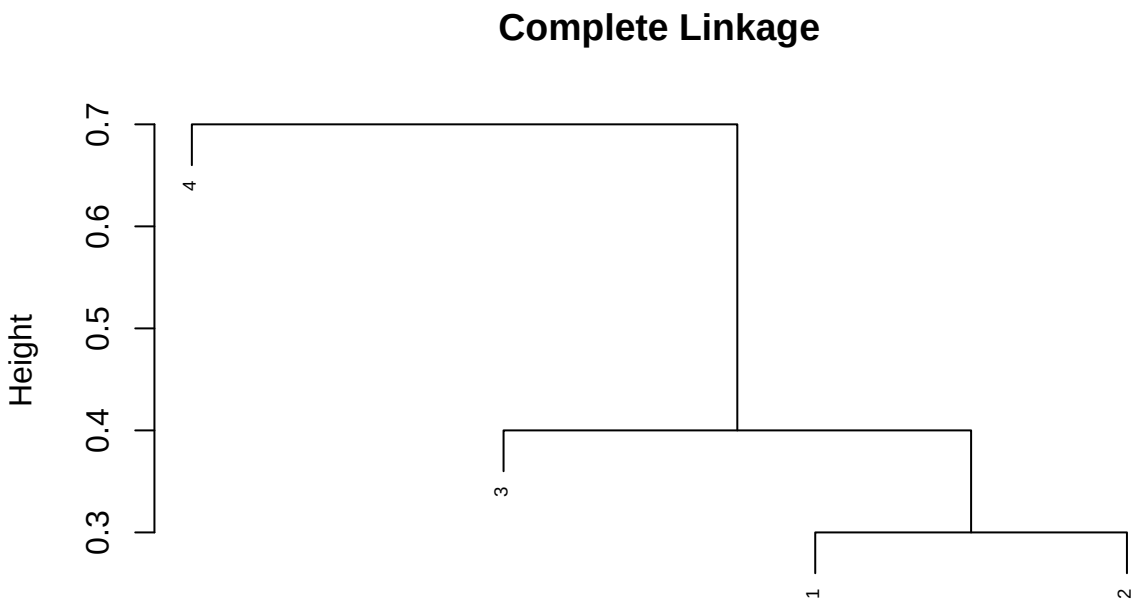
[0.7 0.8 0.9 0]

For instance, the dissimilarity between the first and second observations is 0.3, and the dissimilarity between the second and fourth observations is 0.8.

1. On the basis of this dissimilarity matrix, sketch the dendrogram that results from hierarchical clustering on these four observations using complete linkage. Be sure to indicate on the plot the height at which each merging occurs, as well as the observations corresponding to each leaf in the dendrogram. Cut the dendrogram to obtain two clusters. Which observations are in each cluster?

Solution: Also see the attached pdf for the hand-drawn dendrograms.

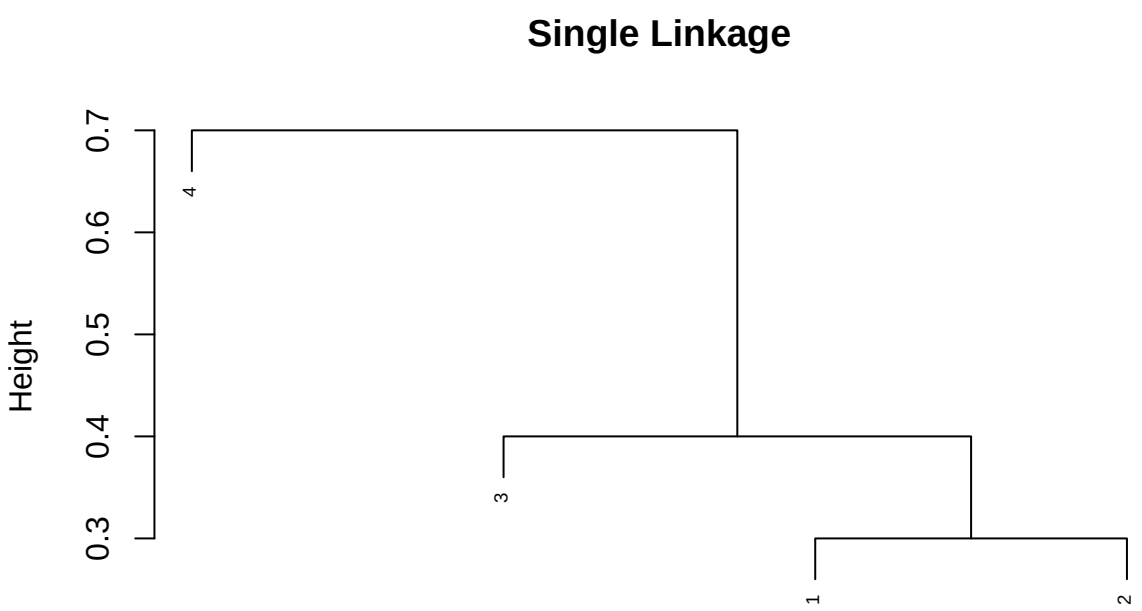
```r
set.seed(1)
#Creating the dissimilarity matrix
#    1   2   3   4
# 1 [0   0.3 0.4 0.7]
# 2 [0.3 0   0.5 0.8]
# 2 [0.4 0.5 0   0.9]
# 3 [0.7 0.8 0.9 0  ]
dissimilarity_matrix <- as.dist(
  matrix(c(0, 0.3, 0.4, 0.7,
           0.3, 0, 0.5, 0.8,
           0.4, 0.5, 0, 0.9,
           0.7, 0.8, 0.9, 0), nrow = 4))

#Computing complete linkage
hclust_dissimilarity_complete <- hclust(dissimilarity_matrix, method = "single")

plot(hclust_dissimilarity_complete,
     main = "Complete Linkage",
     xlab = "",
     sub = "",
     cex = 0.6)
```



2. Repeat 1, this time using single linkage.

Solution: Also see the attached pdf for the hand-drawn dendrograms.

```
set.seed(1)
#Computing single linkage
hclust_dissimilarity_single <- hclust(dissimilarity_matrix, method = "single")
plot(hclust_dissimilarity_single,
     main = "Single Linkage",
     xlab = "",
     sub = "",
     cex = 0.6)
```



Problem 7: EM Algorithm for the Gaussian Mixture Model (20 pts)

1-3. **Solution** See the attached PDF.

4. Using $\mu_0 = 0$, $\mu_1 = 2$, $^2 = 1$, and $= 0.5$, simulate $(x_1, z_1), \ldots, (x_n, z_n)$ for $n = 500$. Use the full data $(x_1, z_1), \ldots, (x_n, z_n)$ to calculate the maximum likelihood estimates of $(\mu_0, \mu_1, {}^2_1, {}^2_2,)$.

Solution:

```
set.seed(1)

#Defining our data
mu_0 <- 0
mu_1 <- 2
sigma_1_squared <- 1
sigma_2_squared <- 1
gamma_bernoulli <- 0.5
n <- 500

#Simulate the latent variable from a binomial distribution
z <- replicate(n, rbinom(n = 1, prob = gamma_bernoulli, size=1))
#Simulating our observed data X based on the latent variable value
#If Z = 0, then N(mu_0, sigma_1^2)
#If Z = 1, then N(mu_1, sigma_2^2)
#Creating our vector of observed data

x <- numeric(n)
for (i in 1:n){
```

```
  if (z[i] == 0){
    x[i] <- rnorm(n = 1, mean = mu_0, sd = sqrt(sigma_1_squared))
  } else {
    x[i] <- rnorm(n = 1, mean = mu_1, sd = sqrt(sigma_2_squared))
  }
}

#We calculate the MLE estimates using the formulas we learned in class
MLE_mu_0 = mean(x[z == 0])
MLE_mu_1 = mean(x[z == 1])
MLE_sigma_1_squared = var(x[z == 0])
MLE_sigma_2_squared = var(x[z == 1])
MLE_gamma = mean(z)

cat("Our direction MLE calculations are: \n")
```

```
## Our direction MLE calculations are:
```

```
cat("MLE_mu_0:", MLE_mu_0, "\n")
```

```
## MLE_mu_0: 0.01074411
```

```
cat("MLE_mu_1:", MLE_mu_1, "\n")
```

```
## MLE_mu_1: 1.913897
```

```
cat("MLE_sigma_1^2:", MLE_sigma_1_squared, "\n")
```

```
## MLE_sigma_1^2: 1.053091
```

```
cat("MLE_sigma_2^2:", MLE_sigma_2_squared, "\n")
```

```
## MLE_sigma_2^2: 1.189878
```

```
cat("MLE_gamma:", MLE_gamma, "\n")
```

```
## MLE_gamma: 0.46
```

5. Implement the EM algorithm you derived in 2 and 3 on $(x_1, \ldots, x_n)$ to estimate $(\mu_0, \mu_1, {}^2{}_1, {}^2{}_2,)$. Use initial parameter estimates $\mu\_hat^{(0)}{}_0 = 0.5$, $\mu\_hat^{(0)}{}_1 = 1.5$, $(\_hat^{(0)}{}_1)^2 = 0.5$, $(\_hat^{(0)}{}_2)^2 = 0.5$ and $\_hat^{(0)} = 0.3$. Display your estimates in the first 10 iterations. After how many iterations, are your EM estimates within $\pm 10{-}4$ of the maximum likelihood estimates you calculated in 4 for every parameter?

Solution:

```
set.seed(1)

# Initial parameter estimates
mu_0 <- 0.5
mu_1 <- 1.5
sigma_1_squared <- 0.5
sigma_2_squared <- 0.5
gamma_bernoulli <- 0.3

#We implement the EM Algorithm in a while loop that breaks once EM estimates
#are within ±10-4 of the maximum likelihood estimates

#Initializing our iterations counter
```

```r
iterations <- 1
max_iterations <- 10000

while(iterations <= max_iterations) {
  #Implementing the E step
  #We first calculate the conditional probabilities needed to derive tau using
  #the equations from problem 5.2
  #P(X_i | Z_i = 1) and P(X_i | Z_i = 0)
  p_xi_zi_0 <- dnorm(x, mean = mu_0, sd = sqrt(sigma_1_squared))
  p_xi_zi_1 <- dnorm(x, mean = mu_1, sd = sqrt(sigma_2_squared))
  tau <- (gamma_bernoulli * p_xi_zi_1) /
    (gamma_bernoulli * p_xi_zi_1 + (1 - gamma_bernoulli) * p_xi_zi_0)

  #Implementing the M step
  #We calculate our parameters using the equations from problem 5.3
  gamma_bernoulli_new <- sum(tau) / n
  mu_0_new <- sum((1 - tau) * x) / sum(1 - tau)
  mu_1_new <- sum(tau * x) / sum(tau)
  sigma_1_squared_new <- sum((1 - tau) * (x - mu_0)^2) / sum(1 - tau)
  sigma_2_squared_new <- sum(tau * (x - mu_1)^2) / sum(tau)

  #Checking to see if new parameters are within ±10-4 of the previous iteration
  if ((abs(mu_0 - mu_0_new) < 1e-4) &&
      (abs(mu_1 - mu_1_new) < 1e-4) &&
      (abs(sigma_1_squared - sigma_1_squared_new) < 1e-4) &&
      (abs(sigma_2_squared - sigma_2_squared_new) < 1e-4) &&
      (abs(gamma_bernoulli - gamma_bernoulli_new) < 1e-4)) {
    cat("The EM Algorithm converged after ", iterations, " iterations. \n")
    cat("mu_0:", mu_0, "\n")
    cat("mu_1:", mu_1, "\n")
    cat("sigma_1^2:", sigma_1_squared, "\n")
    cat("sigma_2^2:", sigma_2_squared, "\n")
    cat("gamma:", gamma_bernoulli, "\n")
    cat("The difference between the EM  estimates and the direct MLE are: \n")
    cat("mu_0 - MLE_mu_0:", mu_0 - MLE_mu_0, "\n")
    cat("mu_1 - MLE_mu_1:", mu_1 - MLE_mu_1, "\n")
    cat("sigma_1^2 - MLE_sigma_1^2:", sigma_1_squared - MLE_sigma_1_squared, "\n")
    cat("sigma_2^2 - MLE_sigma_2^2:", sigma_2_squared - MLE_sigma_2_squared, "\n")
    cat("gamma - MLE_gamma:", gamma_bernoulli - MLE_gamma, "\n")
    break
  }

  #Update our current parameters as the new one
  mu_0 <- mu_0_new
  mu_1 <- mu_1_new
  sigma_1_squared <- sigma_1_squared_new
  sigma_2_squared <- sigma_2_squared_new
  gamma_bernoulli <- gamma_bernoulli_new

  #Displaying the results for the first 10 iterations
  if (iterations <= 10) {
    cat("Iteration", iterations, ":\n")
    cat("mu_0:", mu_0, "\n")
```

```
    cat("mu_1:", mu_1, "\n")
    cat("sigma_1^2:", sigma_1_squared, "\n")
    cat("sigma_2^2:", sigma_2_squared, "\n")
    cat("gamma:", gamma_bernoulli, "\n")
  }

  iterations <- iterations + 1
}
```

```
## Iteration 1 :
## mu_0: 0.1357268
## mu_1: 2.152165
## sigma_1^2: 1.227854
## sigma_2^2: 1.429125
## gamma: 0.3721749
## Iteration 2 :
## mu_0: 0.1941128
## mu_1: 2.049709
## sigma_1^2: 1.206825
## sigma_2^2: 1.220794
## gamma: 0.37297
## Iteration 3 :
## mu_0: 0.1915928
## mu_1: 2.046547
## sigma_1^2: 1.217984
## sigma_2^2: 1.18399
## gamma: 0.3744576
## Iteration 4 :
## mu_0: 0.1882869
## mu_1: 2.048619
## sigma_1^2: 1.217879
## sigma_2^2: 1.17015
## gamma: 0.3751523
## Iteration 5 :
## mu_0: 0.1854689
## mu_1: 2.051129
## sigma_1^2: 1.214543
## sigma_2^2: 1.162343
## gamma: 0.3755912
## Iteration 6 :
## mu_0: 0.1830056
## mu_1: 2.053437
## sigma_1^2: 1.210763
## sigma_2^2: 1.156714
## gamma: 0.3759501
## Iteration 7 :
## mu_0: 0.1808344
## mu_1: 2.055409
## sigma_1^2: 1.207234
## sigma_2^2: 1.152177
## gamma: 0.3762775
## Iteration 8 :
## mu_0: 0.1789143
## mu_1: 2.057044
```

```
## sigma_1^2: 1.204094
## sigma_2^2: 1.148384
## gamma: 0.3765875
## Iteration 9 :
## mu_0: 0.1772123
## mu_1: 2.058376
## sigma_1^2: 1.20134
## sigma_2^2: 1.145187
## gamma: 0.376885
## Iteration 10 :
## mu_0: 0.1756997
## mu_1: 2.059441
## sigma_1^2: 1.198933
## sigma_2^2: 1.142495
## gamma: 0.3771723
## The EM Algorithm converged after  2070  iterations.
## mu_0: -0.3163878
## mu_1: 1.480536
## sigma_1^2: 0.8611628
## sigma_2^2: 1.511631
## gamma: 0.6692452
## The difference between the EM  estimates and the direct MLE are:
## mu_0 - MLE_mu_0: -0.3271319
## mu_1 - MLE_mu_1: -0.4333616
## sigma_1^2 - MLE_sigma_1^2: -0.1919281
## sigma_2^2 - MLE_sigma_2^2: 0.3217533
## gamma - MLE_gamma: 0.2092452
```