ISL Chapter 10 Exercises

Jonathan Bryan

September 4, 2018

1. This problem involves the K-means clustering algorithm.

(a) Prove (10.12).

$$\frac{1}{|C_k|}\sum_{j=1}^{p}\sum_{i\in C_K}\sum_{i'\in C_k}(x_{ij}-x_{i'j})^2 = \frac{1}{|C_k|}\sum_{j=1}^{p}\sum_{i\in C_K}\sum_{i'\in C_k} x_{ij}^2 - 2x_{ij}x_{i'j} + x_{i'j}^2$$

$$\frac{1}{|C_k|}\sum_{j=1}^{p}\sum_{i\in C_K}\sum_{i'\in C_k}(x_{ij}-x_{i'j})^2 = \frac{1}{|C_k|}\sum_{j=1}^{p}\sum_{i\in C_K} |C_k|x_{ij}^2 - 2x_{ij}\sum_{i'\in C_k} x_{i'j} + \sum_{i'\in C_k} x_{i'j}^2$$

$$\frac{1}{|C_k|}\sum_{j=1}^{p}\sum_{i\in C_K}\sum_{i'\in C_k}(x_{ij}-x_{i'j})^2 = \frac{1}{|C_k|}\sum_{j=1}^{p}\sum_{i\in C_K} |C_k|x_{ij}^2 - 2x_{ij}|C_k|\bar{x_{jk}} + \sum_{i'\in C_k} x_{i'j}^2$$

$$\frac{1}{|C_k|}\sum_{j=1}^{p}\sum_{i\in C_K}\sum_{i'\in C_k}(x_{ij}-x_{i'j})^2 = \frac{1}{|C_k|}\sum_{j=1}^{p}|C_k|\sum_{i\in C_K} x_{ij}^2 - 2\sum_{i\in C_K} x_{ij}|C_k|\bar{x_{jk}} + |C_k|\sum_{i'\in C_k} x_{i'j}^2$$

$$\frac{1}{|C_k|}\sum_{j=1}^{p}\sum_{i\in C_K}\sum_{i'\in C_k}(x_{ij}-x_{i'j})^2 = \frac{1}{|C_k|}\sum_{j=1}^{p}|C_k|\sum_{i\in C_K} x_{ij}^2 - 2|C_k|\bar{x_{ij}}|C_k|\bar{x_{jk}} + |C_k|\sum_{i'\in C_k} x_{i'j}^2$$

$$\frac{1}{|C_k|}\sum_{j=1}^{p}\sum_{i\in C_K}\sum_{i'\in C_k}(x_{ij}-x_{i'j})^2 = \frac{1}{|C_k|}\sum_{j=1}^{p}2|C_k|\sum_{i\in C_K} x_{ij}^2 - 2|C_k|\bar{x_{ij}}|C_k|\bar{x_{jk}},\ (\sum_{i\in C_k} x_{ij}^2 = \sum_{i'\in C_k} x_{i'j}^2)$$

$$\frac{1}{|C_k|}\sum_{j=1}^{p}\sum_{i\in C_K}\sum_{i'\in C_k}(x_{ij}-x_{i'j})^2 = 2\sum_{j=1}^{p}(\sum_{i\in C_K} x_{ij}^2 - |C_k|\bar{x_{ij}}^2)$$

$$2\sum_{i\in C_K}\sum_{j=1}^{p}(x_{ij}-\bar{x_{ij}})^2 = 2\sum_{i\in C_K}\sum_{j=1}^{p} x_{ij}^2 - 2x_{ij}\bar{x_{ij}} + \bar{x_{ij}}^2$$

$$2\sum_{i\in C_K}\sum_{j=1}^{p}(x_{ij}-\bar{x_{ij}})^2 = 2\sum_{j=1}^{p}(\sum_{i\in C_K} x_{ij}^2 - 2\bar{x_{ij}}\sum_{i\in C_K} x_{ij} + \sum_{i\in C_K}\bar{x_{ij}}^2)$$

$$2\sum_{i\in C_K}\sum_{j=1}^{p}(x_{ij}-\bar{x_{ij}})^2 = 2\sum_{j=1}^{p}(\sum_{i\in C_K} x_{ij}^2 - 2\bar{x_{ij}}|C_k|\bar{x_{ij}} + |C_k|\bar{x_{ij}}^2)$$

$$2\sum_{i\in C_K}\sum_{j=1}^{p}(x_{ij}-\bar{x_{ij}})^2 = 2\sum_{j=1}^{p}(\sum_{i\in C_K} x_{ij}^2 - 2|C_k|\bar{x_{ij}}^2 + |C_k|\bar{x_{ij}}^2)$$

$$2\sum_{i\in C_K}\sum_{j=1}^{p}(x_{ij}-\bar{x_{ij}})^2 = 2\sum_{j=1}^{p}(\sum_{i\in C_K} x_{ij}^2 - |C_k|\bar{x_{ij}}^2)$$

$$\frac{1}{|C_k|}\sum_{j=1}^{p}\sum_{i\in C_K}\sum_{i'\in C_k}(x_{ij}-x_{i'j})^2 = 2\sum_{j=1}^{p}(\sum_{i\in C_K} x_{ij}^2 - |C_k|\bar{x_{ij}}^2) = 2\sum_{i\in C_K}\sum_{j=1}^{p}(x_{ij}-\bar{x_{ij}})^2$$

(b) On the basis of this identity, argue that the K-means clustering algorithm (Algorithm 10.1) decreases the objective (10.11) at each iteration.

This identity proves that minimizing the squared Euclidean distance between assigned observations in each cluster is equivalent to minimizing the variance of the clusters. Therefore, at each iteration we seek to assign observations to the closest p mean vector (centroid) to reduce the variance of each cluster.

2. Suppose that we have four observations, for which we compute a dissimilarity matrix, given by

$$\begin{bmatrix} 0 & 0.3 & 0.4 & 0.7 \\ 0.3 & 0 & 0.5 & 0.8 \\ 0.4 & 0.5 & 0 & 0.45 \\ 0.7 & 0.8 & 0.45 & 0 \end{bmatrix}$$

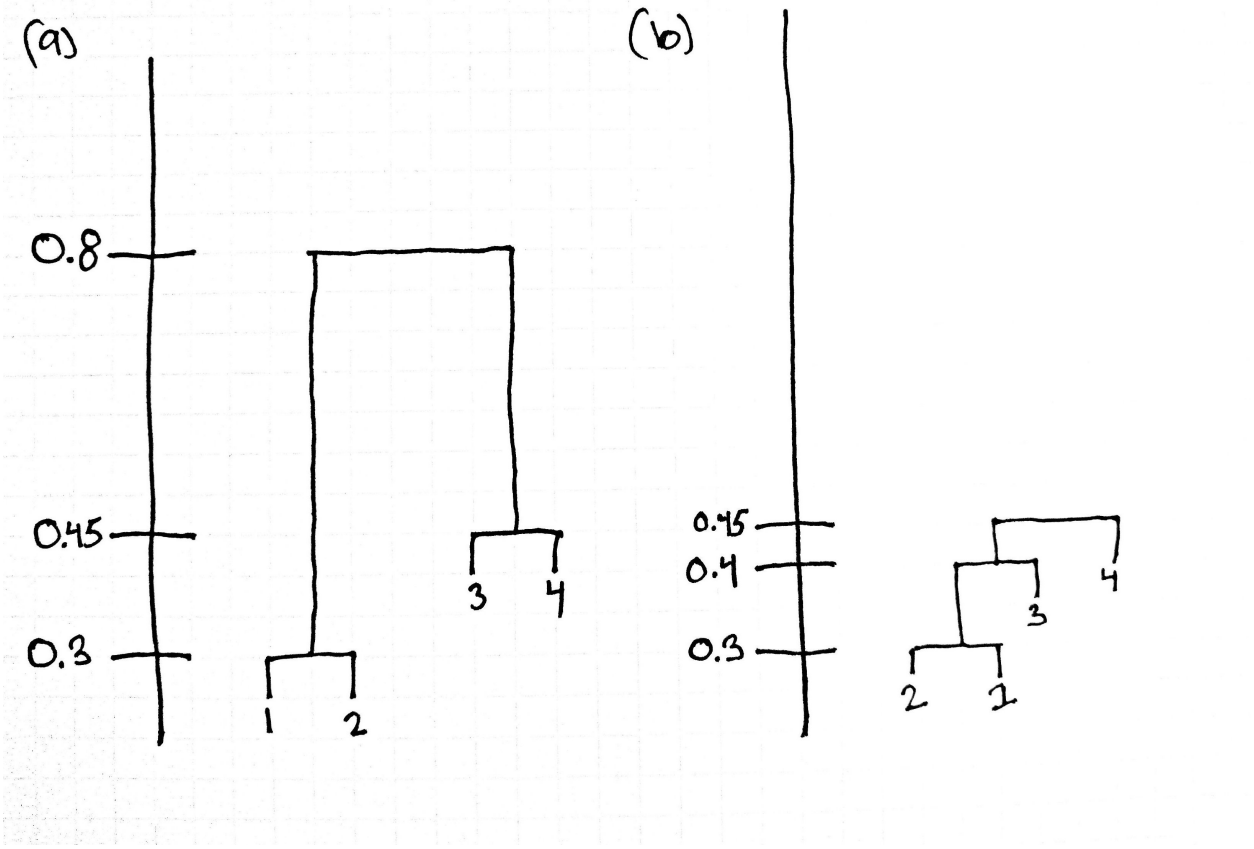
For instance, the dissimilarity between the first and second observations is 0.3, and the dissimilarity between the second and fourth observations is 0.8.

(a) On the basis of this dissimilarity matrix, sketch the dendrogram that results from hierarchically clustering these four observations using complete linkage. Be sure to indicate on the plot the height at which each fusion occurs, as well as the observations corresponding to each leaf in the dendrogram.

See below.

(b) Repeat (a), this time using single linkage clustering.

See below.



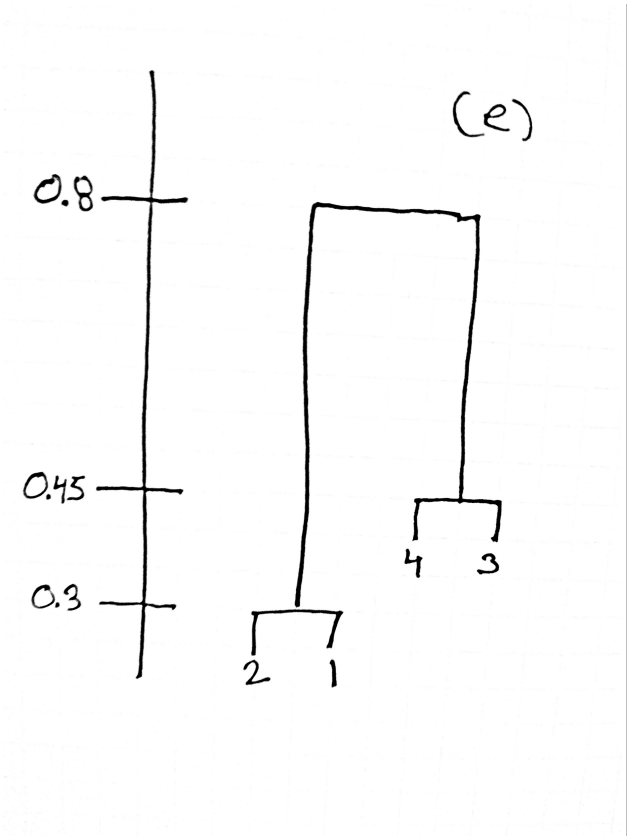
(c) Suppose that we cut the dendogram obtained in (a) such that two clusters result. Which observations are in each cluster?

The first cluster contains 1 and 2 and the second cluster contains 3 and 4.

(d) Suppose that we cut the dendogram obtained in (b) such that two clusters result. Which observations are in each cluster?

The first cluster contains 1, 2 and 3. The second cluster contains 4.

(e) It is mentioned in the chapter that at each fusion in the dendrogram, the position of the two clusters being fused can be swapped without changing the meaning of the dendrogram. Draw a dendrogram that is equivalent to the dendrogram in (a), for which two or more of the leaves are repositioned, but for which the meaning of the dendrogram is the same.

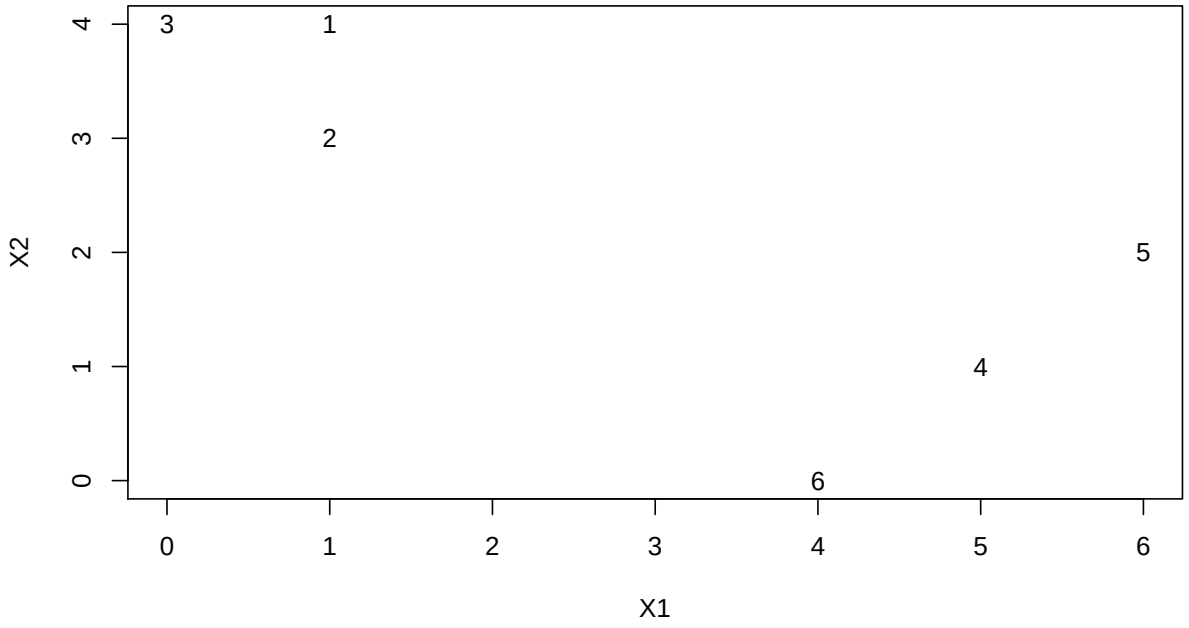


3. In this problem, you will perform K-means clustering manually, with $K = 2$, on a small example with $n = 6$ observations and $p = 2$ features. The observations are as follows.

Obs.	X1	X2
1	1	4
2	1	3
3	0	4
4	5	1
5	6	2
6	4	0

(a) Plot the observations.

```
plot(table[,2],table[,3], type = "n",
     xlab = "X1",
     ylab = "X2")
text(table[,2],table[,3],labels=row.names(table))
```

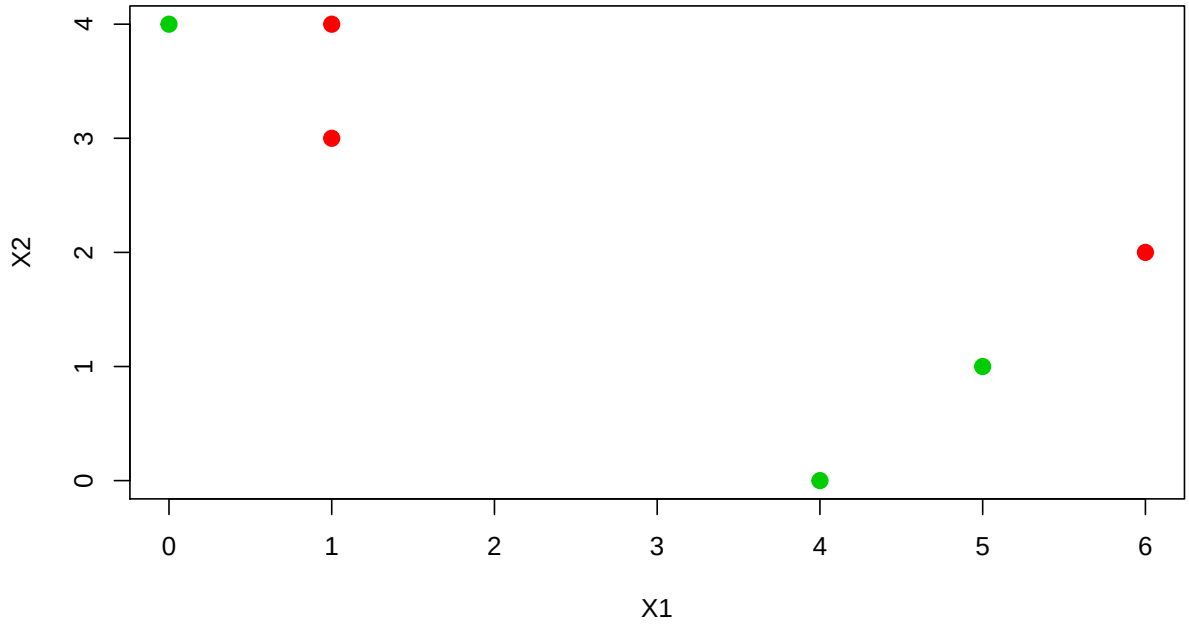


(b) Randomly assign a cluster label to each observation. You can use the sample() command in R to do this. Report the cluster labels for each observation.

```
set.seed(1)
labels = sample(1:2, size = 6, replace = TRUE)
table = cbind(table,labels)
knitr::kable(table)
```

Obs.	X1	X2	labels
1	1	4	1
2	1	3	1
3	0	4	2
4	5	1	2
5	6	2	1
6	4	0	2

```
plot(table[,c(2,3)], col = table[,4]+1, pch = 20, cex =2)
```



(c) Compute the centroid for each cluster.

```
library(dplyr)
centroids = function(table){
  as.data.frame(table) %>%
    group_by(labels) %>%
    summarise(X1_mean = mean(X1),
              X2_mean = mean(X2))
}
cents = centroids(table)
cents
```

```
## # A tibble: 2 x 3
##   labels X1_mean X2_mean
##    <dbl>   <dbl>   <dbl>
## 1      1    2.67    3
## 2      2    3       1.67
```

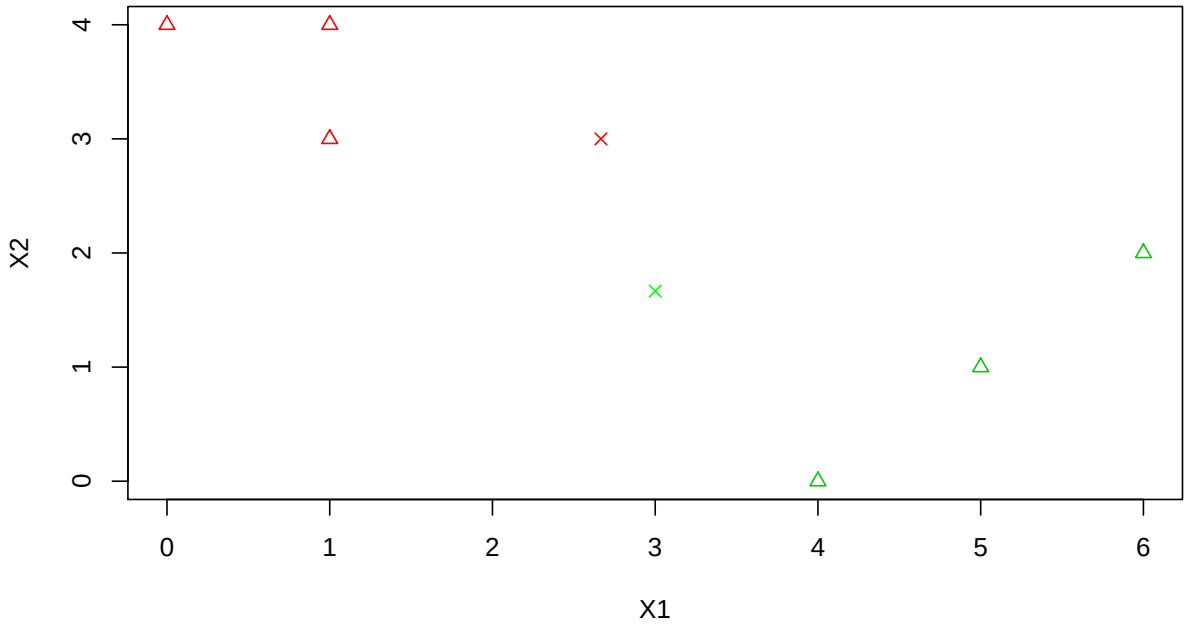
(d) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

```
get_cluster = function(matrix, centroids){
              new_labels = rep(NA, nrow(matrix))
              for (r in 1:nrow(matrix)){
                new_labels[r] = which.min(sqrt(apply((t(table[r,c(2,3)] - centroids[,c(2,3)]))^2,1,sum)))
              }
              new_labels
}
```

```r
new_labels = get_cluster(table,cents)
table[,4] = new_labels
table
```

```
##   Obs. X1 X2 labels
## 1    1  1  4      1
## 2    2  1  3      1
## 3    3  0  4      1
## 4    4  5  1      2
## 5    5  6  2      2
## 6    6  4  0      2
```

```r
plot(table[,c(2,3)], col = table[,4]+1, pch = 2)
points(x = cents$X1_mean, y = cents$X2_mean, pch = 4, col = c("red", "green"))
```

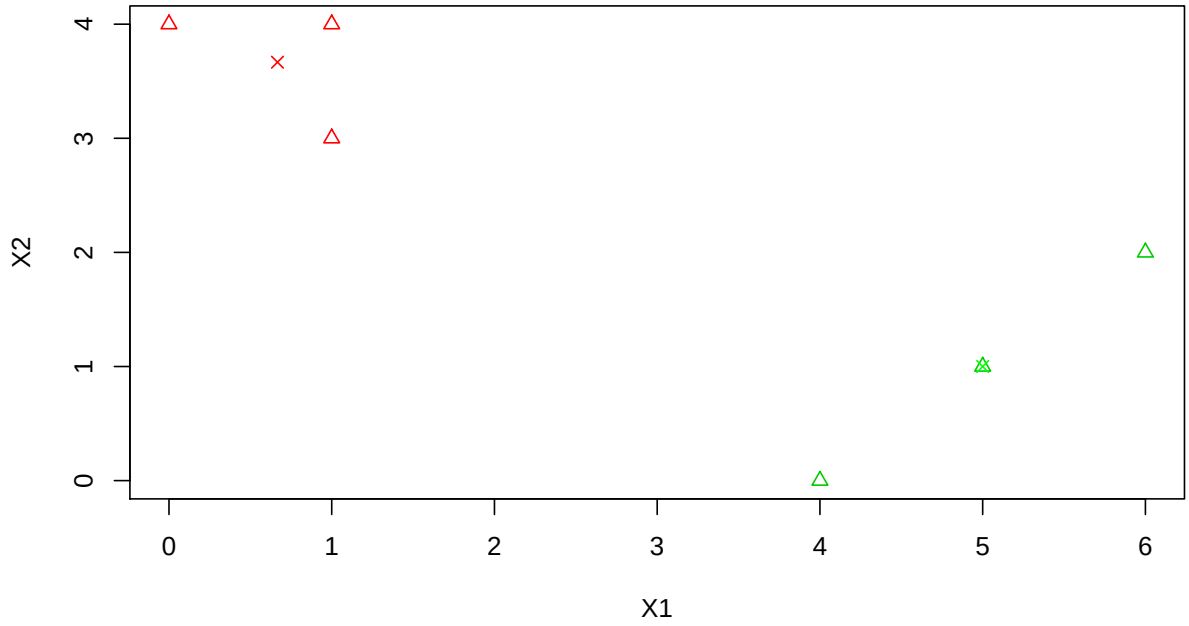


(e) Repeat (c) and (d) until the answers obtained stop changing.

```r
#run 10 times
for (i in 1:10){
  cents = centroids(table)
  table[,4] = get_cluster(table, cents)
}
```

(f) In your plot from (a), color the observations according to the cluster labels obtained.

```r
plot(table[,c(2,3)], col = table[,4]+1, pch=2)
points(x = cents$X1_mean, y = cents$X2_mean, pch = 4, col = c("red", "green"))
```



4. Suppose that for a particular data set, we perform hierarchical clustering using single linkage and using complete linkage. We obtain two dendrograms.

(a) At a certain point on the single linkage dendrogram, the clusters $\{1, 2, 3\}$ and $\{4, 5\}$ fuse. On the complete linkage dendrogram, the clusters $\{1, 2, 3\}$ and $\{4, 5\}$ also fuse at a certain point. Which fusion will occur higher on the tree, or will they fuse at the same height, or is there not enough information to tell?

There is not enough information to tell. It is possible that the largest and smallest dissimilarity between the unique leafs of each cluster are equal, in which case the height of the fuse would be the same for both clusters (i.e, $d(4, i) = d(5, j)\ \forall\ i, j\ \in\ \{1, 2, 3\}$). If $\exists\ i, j\ s.t.\ d(4, i) \neq d(5, j)$ then the complete linkage dendrogram will have the higher height.

(b) At a certain point on the single linkage dendrogram, the clusters $\{5\}$ and $\{6\}$ fuse. On the complete linkage dendrogram, the clusters $\{5\}$ and $\{6\}$ also fuse at a certain point. Which fusion will occur higher on the tree, or will they fuse at the same height, or is there not enough information to tell?

If $\{5\}$ and $\{6\}$ have the smallest dissimilarity out of all of the observations and clusters at a specific iteration, then both the complete and single linkage dendrogram will have the fuse at the same height.

5. In words, describe the results that you would expect if you performed K-means clustering of the eight shoppers in Figure 10.14, on the basis of their sock and computer purchases, with $K = 2$. Give three answers, one for each of the variable scalings displayed. Explain.

For the chart on the left, the two clusters would most likely encompass the 1) black, gold, and pink; 2) light blue, green, yellow, blue, and orange. This is because the difference in sock purchases dominates the

dissimilarity with some contribution from computer purchases.

For the chart in the middle, the two clusters would most likely encompass the 1) black, gold, light blue, and green; 2) yellow, blue, orange, and pink. This is because the difference in computer purchases dominates the dissimilarity.

For the chart on the right, the two clusters would most likely encompass the 1) black, gold, light blue, and green; 2) yellow, blue, orange, and pink. This is because the difference in computer purchases dominates the dissimilarity.

6. A researcher collects expression measurements for 1,000 genes in 100 tissue samples. The data can be written as a 1,000 X 100 matrix, which we call X, in which each row represents a gene and each column a tissue sample. Each tissue sample was processed on a different day, and the columns of X are ordered so that the samples that were processed earliest are on the left, and the samples that were processed later are on the right. The tissue samples belong to two groups: control(C) and treatment (T). The C and T samples were processed in a random order across the days. The researcher wishes to determine whether each gene's expression measurements differ between the treatment and control groups.

As a pre-analysis (before comparing T versus C), the researcher performs a principal component analysis of the data, and finds that the first principal component (a vector of length 100) has a strong linear trend from left to right, and explains 10% of the variation. The researcher now remembers that each patient sample was run on one of two machines, A and B, and machine A was used more often in the earlier times while B was used more often later. The researcher has a record of which sample was run on which machine.

(a) Explain what it means that the first principal component "explains 10% of the variation".

The mean-centered variance of the data points projected along the first principal component is 10 percent of the overall variance found in the mean-centered data.

(b) The researcher decides to replace the (j, i)th element of X with

$$x_{ji} - \phi_{j1} z_{i1}$$

where z_{i1} is the ith score, and ϕ_{j1} is the jth loading, for the first principal component. He will then perform a two-sample t-test on each gene in this new data set in order to determine whether its expression differs between the two conditions. Critique this idea, and suggest a better approach. (The principal component analysis is performed on X^T).

The approach in part(b) removes the variation present in the first principle component across the genes and weighted by the scores of the tissue. A better approach would be to remove the variation across the tissues (because that is where the variation in machine batch can be found) and weighted by the gene scores. Represented below as:

$$x_{ji} - \phi_{i1} z_{j1}$$

(c) Design and run a small simulation experiment to demonstrate the superiority of your idea.

```
#generate simulated gene expression data with treament effects and systematic machine error
set.seed(1)
gexp_t = as.data.frame(matrix(rnorm(1000*100),
```

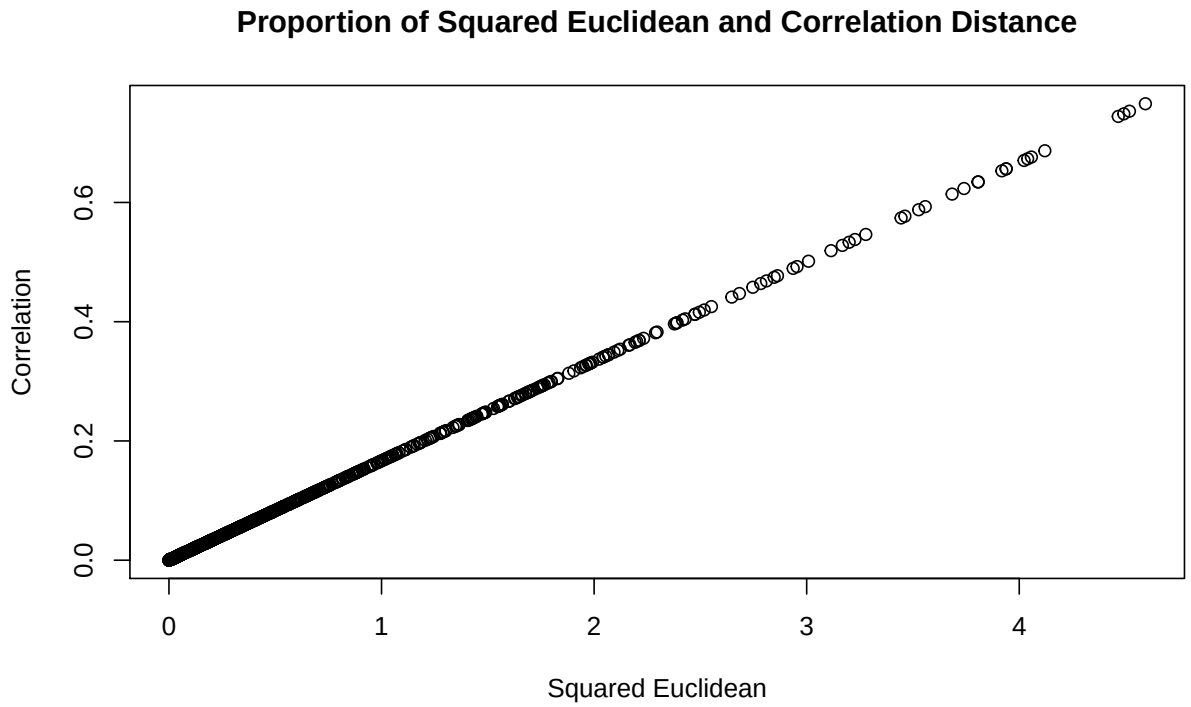
```
                    nrow = 100,
                    ncol = 1000
                    )
      )
rownames(gexp_t) = paste0("tissue_", 1:100)
colnames(gexp_t) = seq(1:1000)
gexp_t$treat = sample(c("C", "T"), size = 100, replace = TRUE)
gexp_t$machine = c(rep("A",50), rep("B",50))

#add treatment effect
gexp_t[gexp_t$treat == "T",1:1000] = gexp_t[gexp_t$treat == "T",1:1000] + rnorm(1000,1,1)

#add machine effect
gexp_t[gexp_t$machine == "A",1:1000] = gexp_t[gexp_t$machine == "A",1:1000] - rnorm(1000, 1,2)

#convert to non-transpose form in part (b)
gexp = matrix(as.numeric(t(gexp_t)[1:1000,]),
              nrow = 1000,
              ncol = 100)

#function to t.test on each gene between T and C
multi_t.test = function(df){
            p.values = rep(NA,1000)
            for (c in 1:1000){
            p.values[c] = t.test(df[c, gexp_t$treat == "T"],
                     df[c,gexp_t$treat == "C"])$p.value
            }
            p.values
}

#part (b) method
gexp_t_pc = prcomp(gexp_t[,1:1000], scale = TRUE)
gexp_t_pc1_scores = gexp_t_pc$x[,1]
gexp_t_pc1_loadings = gexp_t_pc$rotation[,1]
gexp_book = gexp[1:1000,] - gexp_t_pc1_loadings%*%t(gexp_t_pc1_scores)

#improved method
gexp_pc = prcomp(gexp[1:1000,], scale = TRUE)
gexp_pc1_scores = gexp_pc$x[,1]
gexp_pc1_loadings = gexp_pc$rotation[,1]
gexp_imp = gexp[1:1000,] - gexp_pc1_scores%*%t(gexp_pc1_loadings)

# t-testing with Bonferroni correction for multiple testing
base = multi_t.test(gexp)
book = multi_t.test(gexp_book)
imp =  multi_t.test(gexp_imp)

data.frame("No Adustment" = sum(base > 0.00005),
           "Part (b) method" = sum(book > 0.00005),
           "Improved method" = sum(imp > 0.00005)
          )
```

```
##   No.Adustment Part..b..method Improved.method
## 1          952             999              949
```

We simulate a data set where each of the 1000 genes has a +1 mean offset for the treatment group and -1 mean offset for the A machine. Using a conservative Bonferroni correction to the type I error rate ($\alpha = 0.00005$), we see that the method in part(b) actually increases the false negative rate. The improved method is marginally better than the non-adjusted method.

7. In the chapter, we mentioned the use of correlation-based distance and Euclidean distance as dissimilarity measures for hierarchical clustering. It turns out that these two measures are almost equivalent: if each observation has been centered to have mean zero and standard deviation one, and if we let rij denote the correlation between the ith and jth observations, then the quantity $1-r_{ij}$ is proportional to the squared Euclidean distance between the ith and jth observations. On the USArrests data, show that this proportionality holds.

```
#squared Euclidean distance of scaled observations
USArrests_scaled = t(scale(t(USArrests)))
sqr_Euc = dist(USArrests_scaled, method = "euclidean")^2

#correlation distance
corr = 1 - cor(t(USArrests_scaled))
corr = corr[lower.tri(corr)]

#plot distance metrics
plot(sqr_Euc,corr,
     main = "Proportion of Squared Euclidean and Correlation Distance",
     xlab = "Squared Euclidean",
     ylab = "Correlation")
```



We observe that the squared Euclidean distance is proportionally six times that of the correlation distance metric for this data.

8. In Section 10.2.3, a formula for calculating PVE was given in Equation 10.8. We also saw that the PVE can be obtained using the sdev output of the prcomp() function. On the USArrests data, calculate PVE in two ways:

(a) Using the sdev output of the prcomp() function, as was done in Section 10.2.3.

```
arrests_prc = prcomp(USArrests, scale = TRUE)
arrests_pve = arrests_prc$sdev^2/sum(arrests_prc$sdev^2)
round(arrests_pve,2)
```

```
## [1] 0.62 0.25 0.09 0.04
```

(b) By applying Equation 10.8 directly. That is, use the prcomp() function to compute the principal component loadings. Then, use those loadings in Equation 10.8 to obtain the PVE.

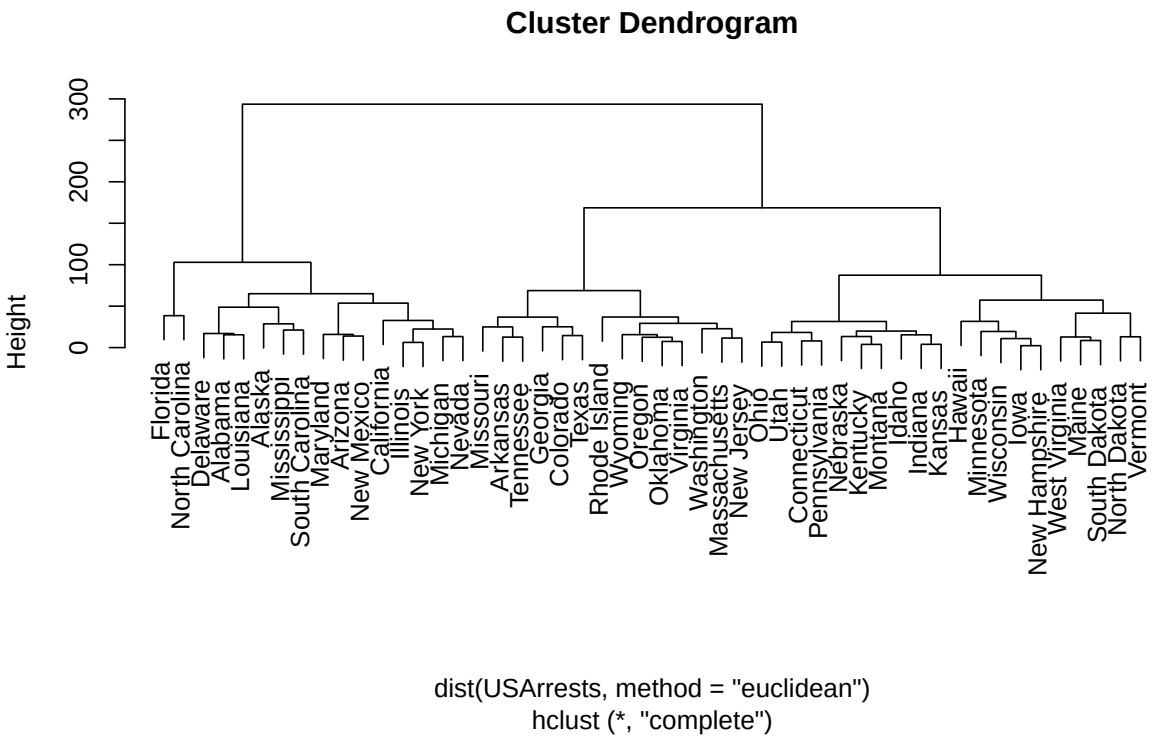
```
arrests_pve2 = colSums((as.matrix(scale(USArrests))%*%arrests_prc$rotation)^2)/sum(colSums(scale(USArres
round(arrests_pve2,2)
```

```
##  PC1  PC2  PC3  PC4
## 0.62 0.25 0.09 0.04
```

9. Consider the USArrests data. We will now perform hierarchical clustering on the states.

(a) Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```
arrests_clust = hclust(dist(USArrests,
                            method = "euclidean"),
                       method = "complete"
                      )
plot(arrests_clust)
```



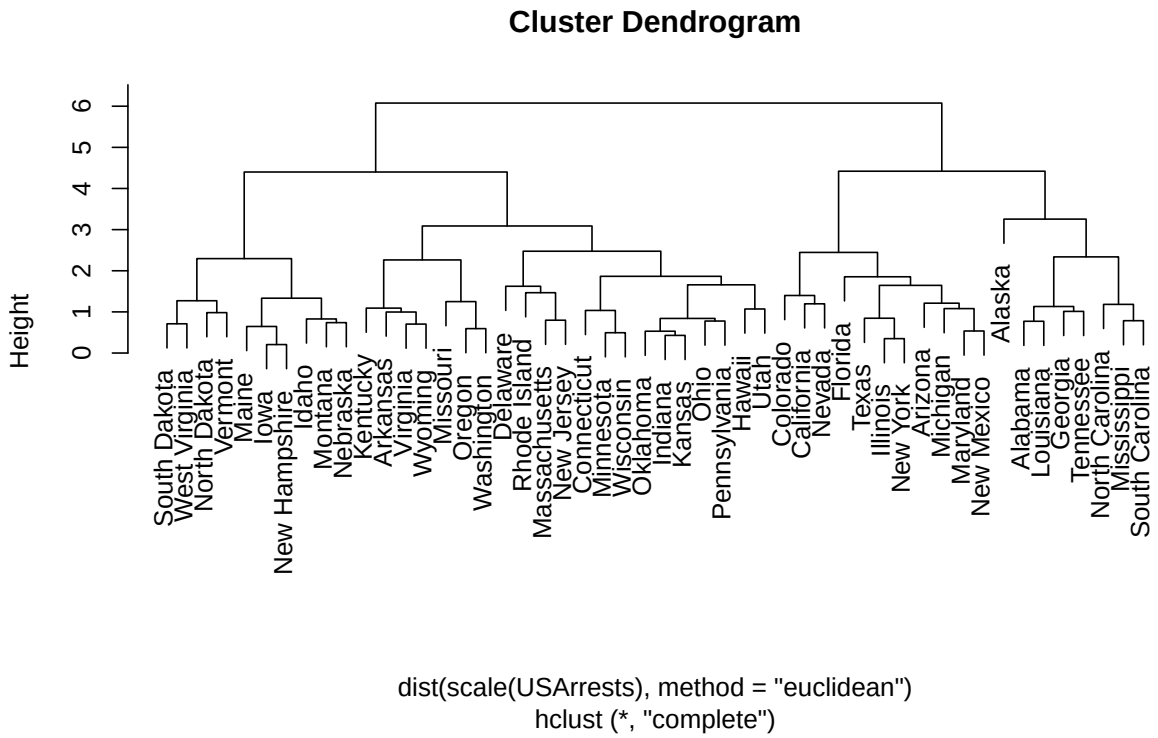
(b) Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters?

```
cutree(arrests_clust, k=3)
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              2              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              1              1              2
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              3              1              3              3
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              3              1              3              1
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              2              1              3              1              2
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              3              3              1              3              2
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              1              1              1              3              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              2              2              3              2              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              3              2              2              3              3
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              2              2              3              3              2
```

(c) Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```
arrests_clust = hclust(dist(scale(USArrests),
                            method = "euclidean"),
                       method = "complete"
                      )
plot(arrests_clust)
```



(d) What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer.

Scaling the variables prior to clustering reduces the absolute difference in dissimilarity measures between major clusters but increases the visible dissimilarity differences between individual observations. This allows for greater subjective discrimination among clusters and observations. These findings support scaling of observations before inter-observation dissimilarities are computed.

10. In this problem, you will generate simulated data, and then perform PCA and K-means clustering on the data.

(a) Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables. Hint: There are a number of functions in R that you can use to generate data. One example is the rnorm() function; runif() is another option. Be sure to add a mean shift to the observations in each class so that there are three distinct classes.

```
#generate simulated data
set.seed(1)
data_sim = matrix(rnorm(60*50),
                  nrow = 60,
                  ncol = 50)
```

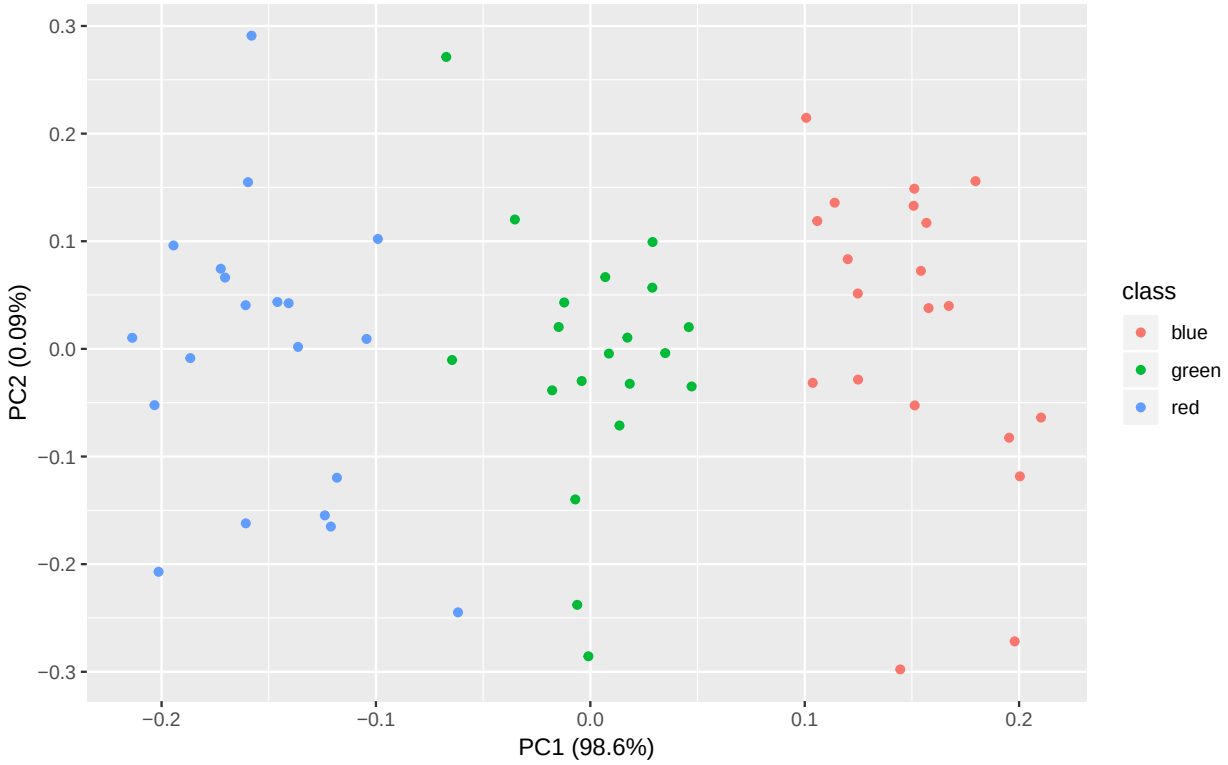
```
data_sim[1:20,] = data_sim[1:20,] + rnorm(20,10,2)
data_sim[21:40,] = data_sim[21:40,] - rnorm(20,10,2)
data_sim[41:60,] = data_sim[41:60,] + rnorm(20,0,2)
data_sim = as.data.frame(data_sim)
data_sim$class = c(rep("red", 20), rep("blue", 20), rep("green",20))
```

(b) Perform PCA on the 60 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, then continue on to part (c). If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes. Do not continue to part (c) until the three classes show at least some separation in the first two principal component score vectors.

```
library(ggfortify)
```

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

```
pca = prcomp(data_sim[,-51], scale = TRUE)
autoplot(pca, data = data_sim, colour = 'class')
```



(c) Perform K-means clustering of the observations with K = 3. How well do the clusters that you obtained in K-means clustering compare to the true class labels? Hint: You can use the table() function in R to compare the true class labels to the class labels obtained by clustering. Be careful how you interpret the results: K-means clustering will arbitrarily number the clusters, so you cannot simply check whether the true class labels and clustering labels are the same.

```
set.seed(1)
kcluster = kmeans(x = data_sim[,-51], center = 3, nstart = 20)

#peak at cluster numbers to match with class colors
kcluster$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```
kcluster$cluster[kcluster$cluster == 1] = "red"
kcluster$cluster[kcluster$cluster == 2] = "blue"
kcluster$cluster[kcluster$cluster == 3] = "green"

#cluster accuracy
table(kcluster$cluster, data_sim$class)
```

```
##
##         blue green red
##   blue    20     0   0
##   green    0    20   1
##   red      0     0  19
```

We find 100 percent accurate clustering for the blue and green classes. One red observation is misclassified as green.

(d) Perform K-means clustering with K = 2. Describe your results.

```
set.seed(1)
kcluster = kmeans(x = data_sim[,-51], center = 2, nstart = 20)

#peak at cluster numbers to match with class colors
kcluster$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2
```

```
kcluster$cluster[kcluster$cluster == 1] = "red"
kcluster$cluster[kcluster$cluster == 2] = "blue"

#cluster accuracy
table(kcluster$cluster, data_sim$class)
```

```
##
##        blue green red
##   blue   20    17   0
##   red     0     3  20
```

We find 100% accurate clustering for the blue and red observations. Most of the green observations are classified as blue.

(e) Now perform K-means clustering with K = 4, and describe your results.

```
set.seed(1)
kcluster = kmeans(x = data_sim[,-51], center = 4, nstart = 20)

#peak at cluster numbers to match with class colors
kcluster$cluster
```

```
##  [1] 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1 1 2 2 1 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [36] 4 4 4 4 4 3 3 3 3 3 3 3 3 2 3 3 2 3 3 3 3 3 3 3 3
```

```
kcluster$cluster[kcluster$cluster == 1] = "red"
kcluster$cluster[kcluster$cluster == 2] = "mystery"
kcluster$cluster[kcluster$cluster == 3] = "green"
kcluster$cluster[kcluster$cluster == 4] = "blue"

#cluster accuracy
table(kcluster$cluster, data_sim$class)
```

```
##
##            blue green red
##   blue       20     0   0
##   green       0    18   0
##   mystery     0     2   6
##   red         0     0  14
```

We find 100% accurate clustering for the blue observations. Two of the green observations and 6 of the red observations are misclassified into the "mystery" fourth cluster.

(f) Now perform K-means clustering with K = 3 on the first two principal component score vectors, rather than on the raw data. That is, perform K-means clustering on the 60 x 2 matrix of which the first column is the first principal component score vector, and the second column is the second principal component score vector. Comment on the results.

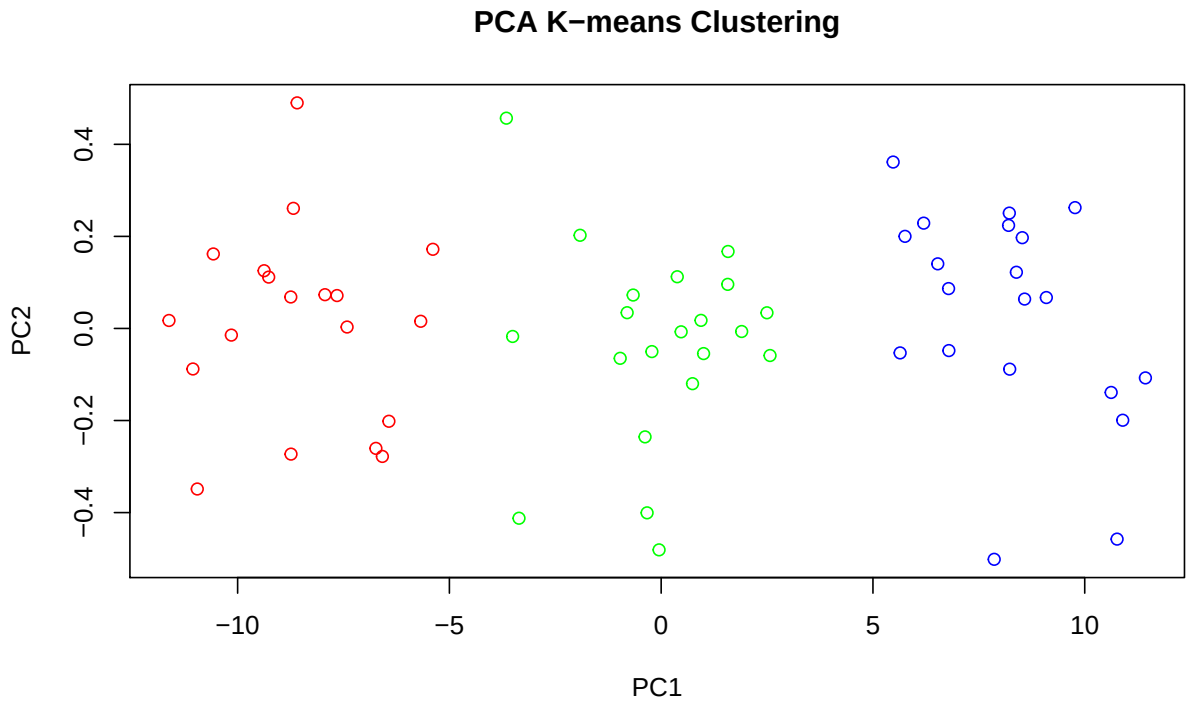
```
set.seed(1)
kcluster = kmeans(x = pca$x[,c(1,2)], center = 3, nstart = 20)

#peak at cluster numbers to match with class colors
kcluster$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```
kcluster$cluster[kcluster$cluster == 1] = "red"
kcluster$cluster[kcluster$cluster == 2] = "blue"
kcluster$cluster[kcluster$cluster == 3] = "green"

#cluster accuracy
table(kcluster$cluster, data_sim$class)
```

```
##
##          blue green red
##   blue     20     0   0
##   green     0    20   1
##   red       0     0  19
```

```r
#plot
plot(pca$x[,c(1,2)], col = kcluster$cluster,
     main = "PCA K-means Clustering",
     ylab = "PC2",
     xlab = "PC1"
  )
```



This procedure gives us the same level of accuracy as k-means clustering on the full column set, however, we can now easily visualize the results.

(g) Using the scale() function, perform K-means clustering with K = 3 on the data after scaling each variable to have standard deviation one. How do these results compare to those obtained in (b)? Explain.

```r
set.seed(1)
kcluster = kmeans(x = scale(data_sim[,-51]), center = 3, nstart = 20)

#peak at cluster numbers to match with class colors
kcluster$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```r
kcluster$cluster[kcluster$cluster == 1] = "red"
kcluster$cluster[kcluster$cluster == 2] = "blue"
kcluster$cluster[kcluster$cluster == 3] = "green"

#cluster accuracy
table(kcluster$cluster, data_sim$class)
```

```
##
##          blue green red
##   blue    20     0   0
##   green    0    20   1
##   red      0     0  19
```

We find nearly perfectly clustering with the exception of one red observation misclassfied as green.

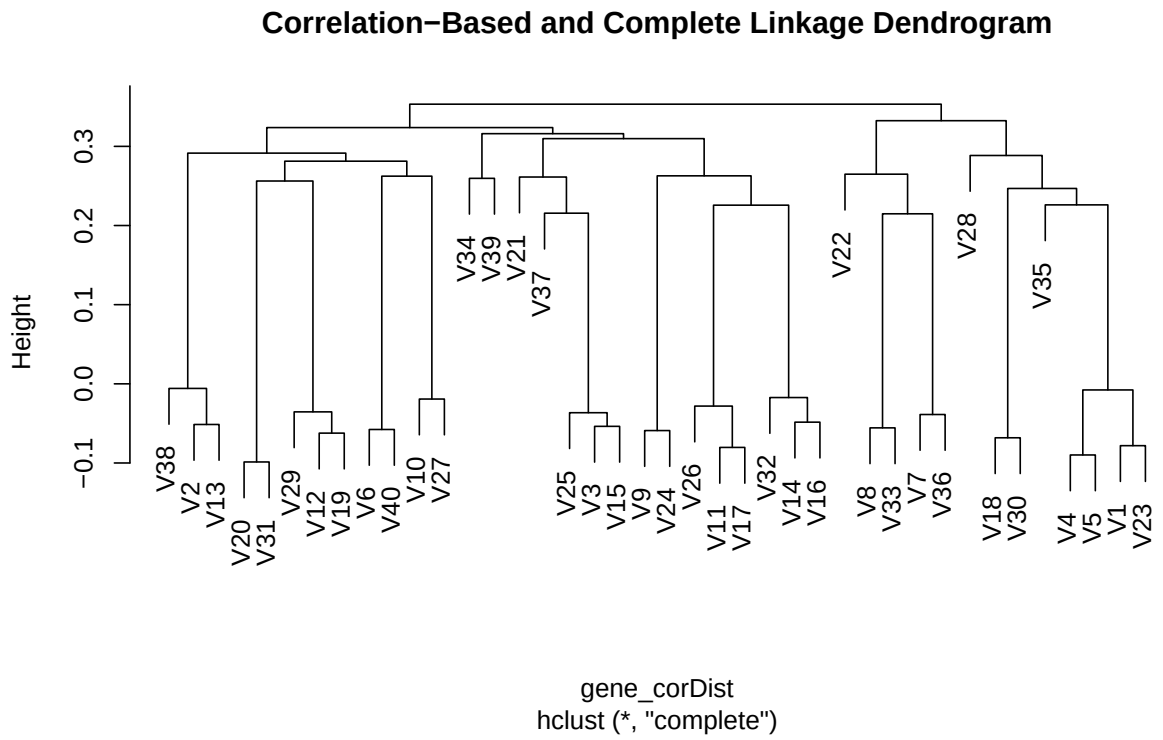
11. On the book website, www.StatLearning.com, there is a gene expression data set (Ch10Ex11.csv) that consists of 40 tissue samples with measurements on 1,000 genes. The first 20 samples are from healthy patients, while the second 20 are from a diseased group.

(a) Load in the data using read.csv(). You will need to select header=F.

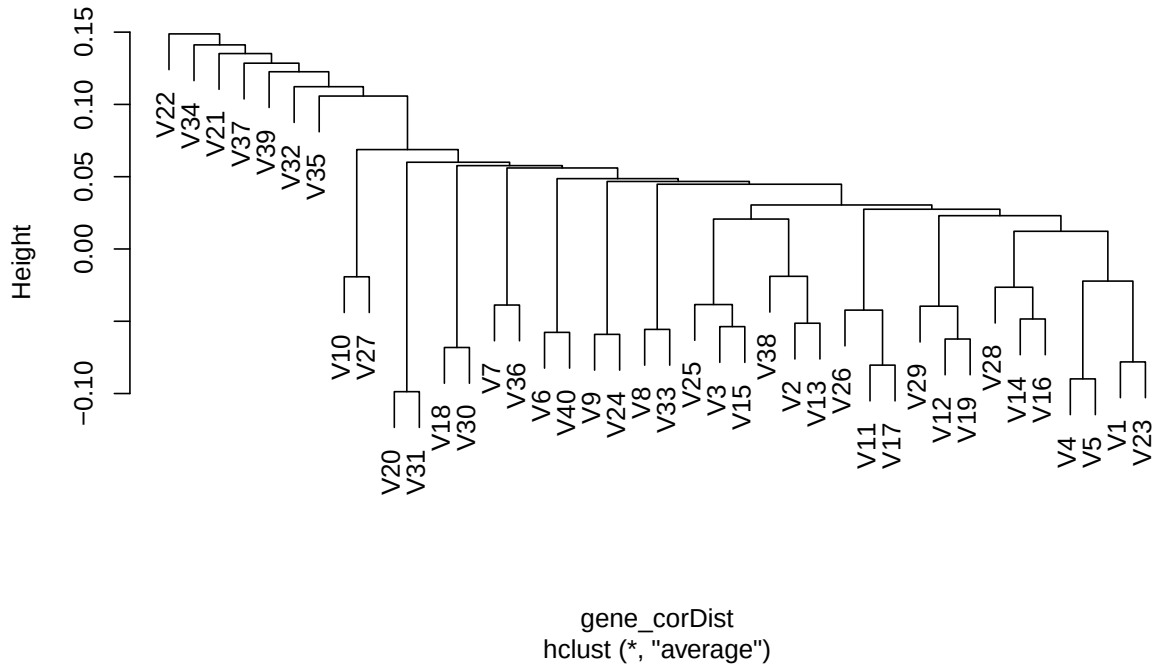
```
gene = read.csv("http://www-bcf.usc.edu/~gareth/ISL/Ch10Ex11.csv", header = FALSE)
```

(b) Apply hierarchical clustering to the samples using correlation-based distance, and plot the dendrogram. Do the genes separate the samples into the two groups? Do your results depend on the type of linkage used?

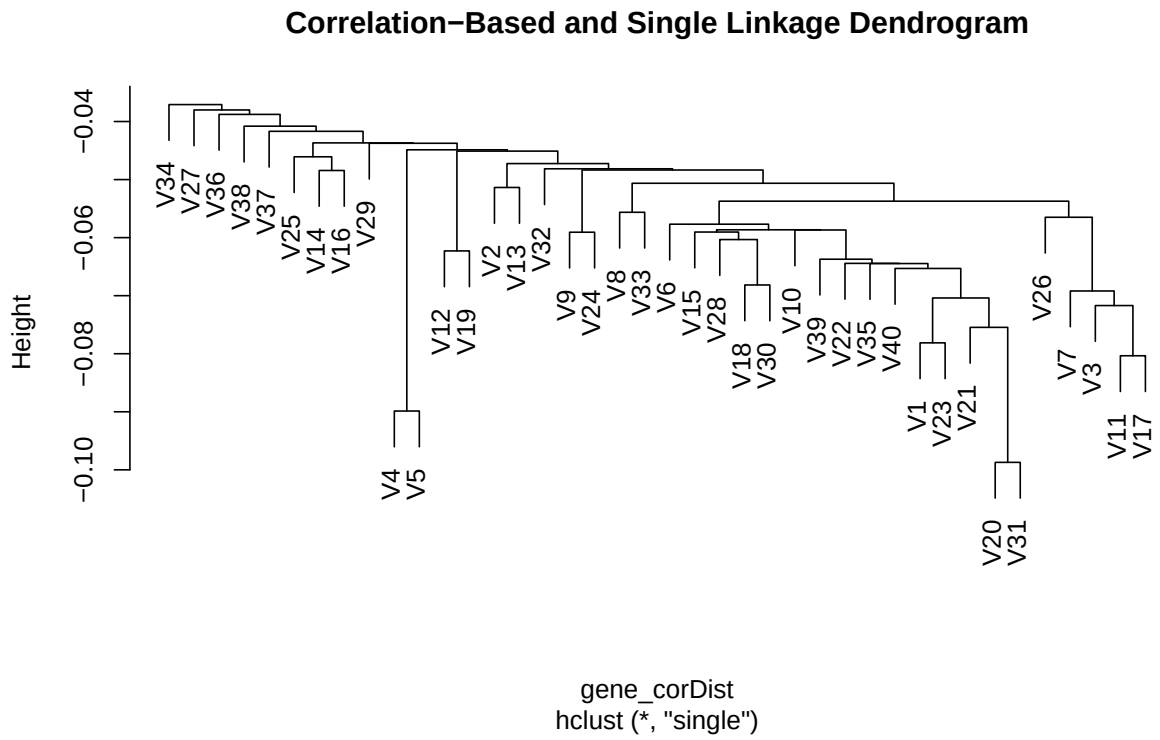
```
#correlation-based distance and complete linkage
gene_cor = cor(gene)
gene_corDist = as.dist(gene_cor)
gene_corClust = hclust(gene_corDist)

#average linkage
gene_complete = hclust(gene_corDist, method = "average")

#single linkage
gene_avg = hclust(gene_corDist, method = "single")

#plots
plot(gene_corClust,
     main = "Correlation-Based and Complete Linkage Dendrogram")
```




```
plot(gene_complete,
     main = "Correlation-Based and Average Linkage Dendrogram")
```



```
plot(gene_avg,
     main = "Correlation-Based and Single Linkage Dendrogram")
```

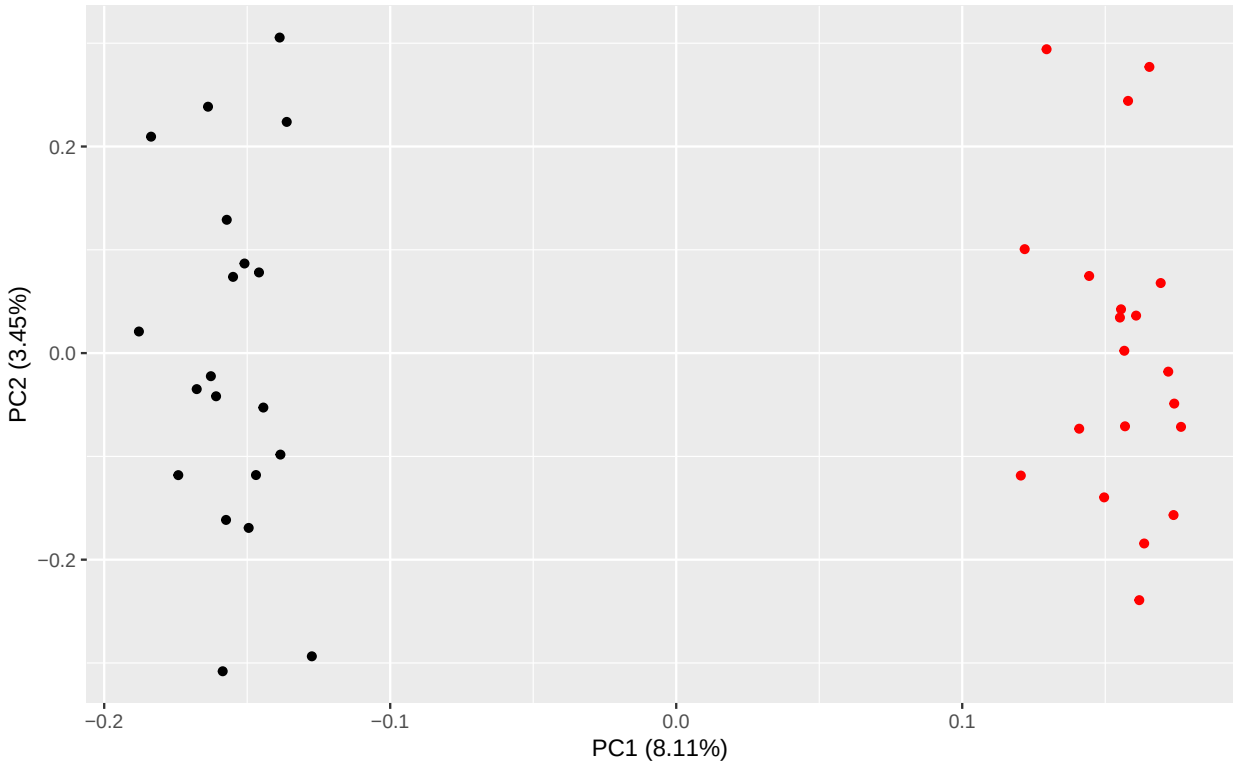


Using a correlation-based distance measure and complete linkage there is good separation between two groups of patients. Average and single linkage do a poor job of separating the genes into two well balanced groups.

(c) Your collaborator wants to know which genes differ the most across the two groups. Suggest a way to answer this question, and apply it here.

One way to calculate the largest gene "difference" across the groups would be to derive the first and second principal components and plot the patient classes on this graph. We can then assess for good separation, and if found, can use the ranked absolute value of the PC1 and/or PC2 loadings for each gene. We see below that the first principal component generates good separation along its axis. The top ten genes with the largest absolute value PC1 loadings are shown below and would indicate the genes with the largest differences across the groups.

```
#PCA on genes
gene_t = as.data.frame(t(gene))
gene_pca = prcomp(gene_t, scale = TRUE)
autoplot(gene_pca, colour = c(rep("black", 20), rep("red", 20)))
```




```
#top 10 absolute value ranked PC1 loadings
sort(abs(gene_pca$rotation[,1]), decreasing = TRUE)[1:10]
```

```
##       V502       V589       V565       V590       V600       V551
## 0.09485044 0.09449766 0.09183823 0.09173169 0.09167322 0.08768360
##       V593       V538       V584       V509
## 0.08758616 0.08745400 0.08690858 0.08661015
```