ISLR | Chapter 10 Exercises

Marshall McQuillen

2/8/2019

Conceptual

1

NEED TO COME BACK TOO

- **A**. 10.12, illustrated below, is showing the the *within-cluster variation* is equal to twice the squared distance between each data point in cluster k (C_k) and that cluster's centroid, summed across all data points.

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{P}(x_{i,j}-x_{i',j})^2 = 2\sum_{i\in C_k}\sum_{j=1}^{P}(x_{i,j}-\bar{x}_{k,j})^2 \tag{1}$$

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{P}(x_{i,j}-x_{i',j})(x_{i,j}-x_{i',j}) = 2\sum_{i\in C_k}\sum_{j=1}^{P}(x_{i,j}-\bar{x}_{k,j})(x_{i,j}-\bar{x}_{k,j}) \tag{2}$$

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{P}(x_{i,j}^2-2x_{i,j}x_{i',j}+x_{i',j}^2) = 2\sum_{i\in C_k}\sum_{j=1}^{P}(x_{i,j}^2-2\bar{x}_{k,j}x_{i,j}+\bar{x}_{k,j}^2) \tag{3}$$

$$\frac{|C_k|}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{P}(x_{i,j}^2-2x_{i,j}x_{i',j}+x_{i',j}^2) = 2|C_k|\sum_{i\in C_k}\sum_{j=1}^{P}(x_{i,j}^2-2\bar{x}_{k,j}x_{i,j}+\bar{x}_{k,j}^2) \tag{4}$$

$$\sum_{i,i'\in C_k}\sum_{j=1}^{P}(x_{i,j}^2-2x_{i,j}x_{i',j}+x_{i',j}^2) = 2|C_k|\sum_{i\in C_k}\sum_{j=1}^{P}(x_{i,j}^2-2\bar{x}_{k,j}x_{i,j}+\bar{x}_{k,j}^2) \tag{5}$$

$$\sum_{i\in C_k}\sum_{j=1}^{P}x_{i,j}^2 - 2\sum_{i,i'\in C_k}\sum_{j=1}^{P}x_{i,j}x_{i',j} + \sum_{i'\in C_k}\sum_{j=1}^{P}x_{i',j}^2 = 2|C_k|\sum_{i\in C_k}\sum_{j=1}^{P}x_{i,j}^2 - 2|C_k|\sum_{i\in C_k}\sum_{j=1}^{P}2\bar{x}_{k,j}x_{i,j} + 2|C_k|\sum_{i\in C_k}\sum_{j=1}^{P}\bar{x}_{k,j}^2 \tag{6}$$

$$\sum_{i\in C_k}\sum_{j=1}^{P}x_{i,j}^2 - 2\sum_{i,i'\in C_k}\sum_{j=1}^{P}x_{i,j}x_{i',j} + \sum_{i'\in C_k}\sum_{j=1}^{P}x_{i',j}^2 = 2|C_k|\sum_{i\in C_k}\sum_{j=1}^{P}x_{i,j}^2 - 4|C_k|\sum_{i\in C_k}\sum_{j=1}^{P}\bar{x}_{k,j}x_{i,j} + 2|C_k|\sum_{i\in C_k}\sum_{j=1}^{P}\bar{x}_{k,j}^2 \tag{7}$$

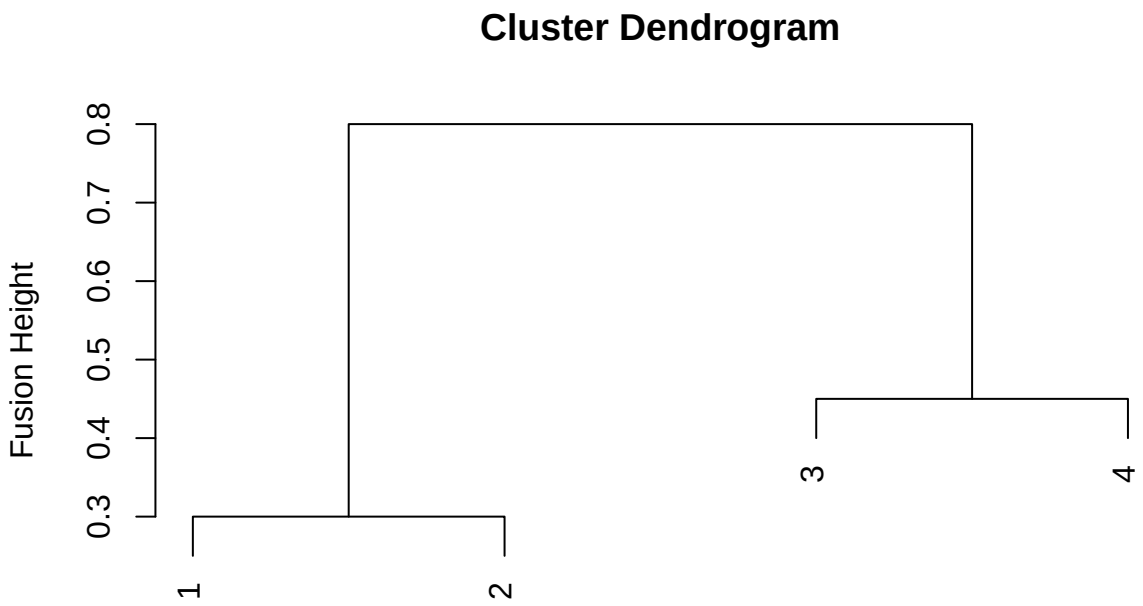
(8)

2

- **A**.

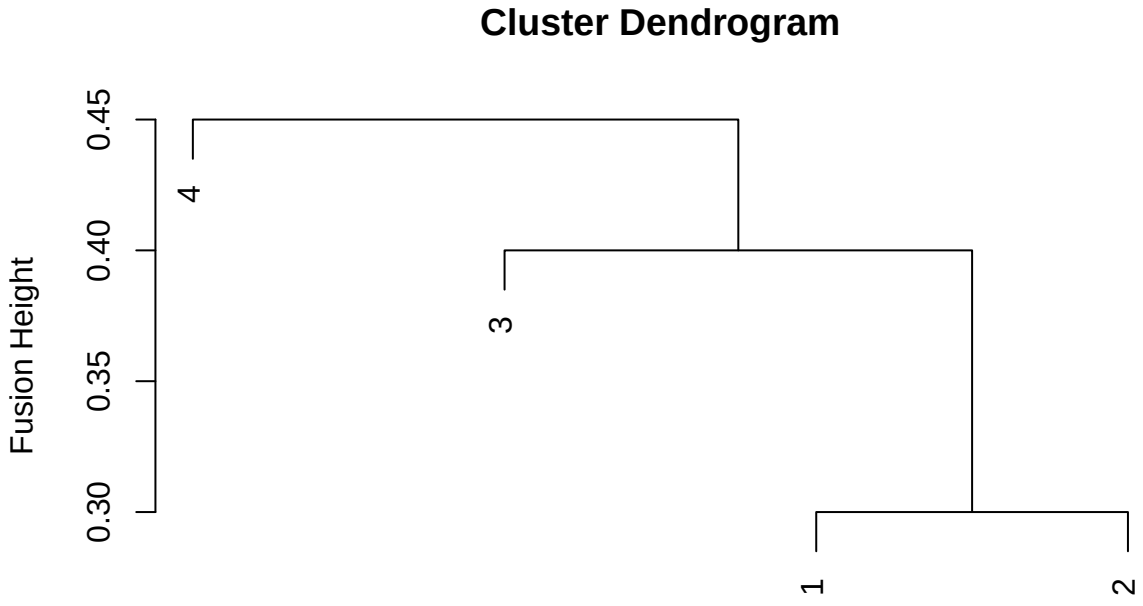
```
x <- matrix(c(0, 0.3, 0.4, 0.7,
              0.3, 0, 0.5, 0.8,
              0.4, 0.5, 0, 0.45,
              0.7, 0.8, 0.45, 0), ncol = 4, nrow = 4)
```

```
plot(hclust(as.dist(x), method = 'complete'),
     xlab = "",
     sub = "",
     ylab = "Fusion Height")
```



- B.

```
plot(hclust(as.dist(x), method = 'single'),
     xlab = "",
     sub = "",
     ylab = "Fusion Height")
```

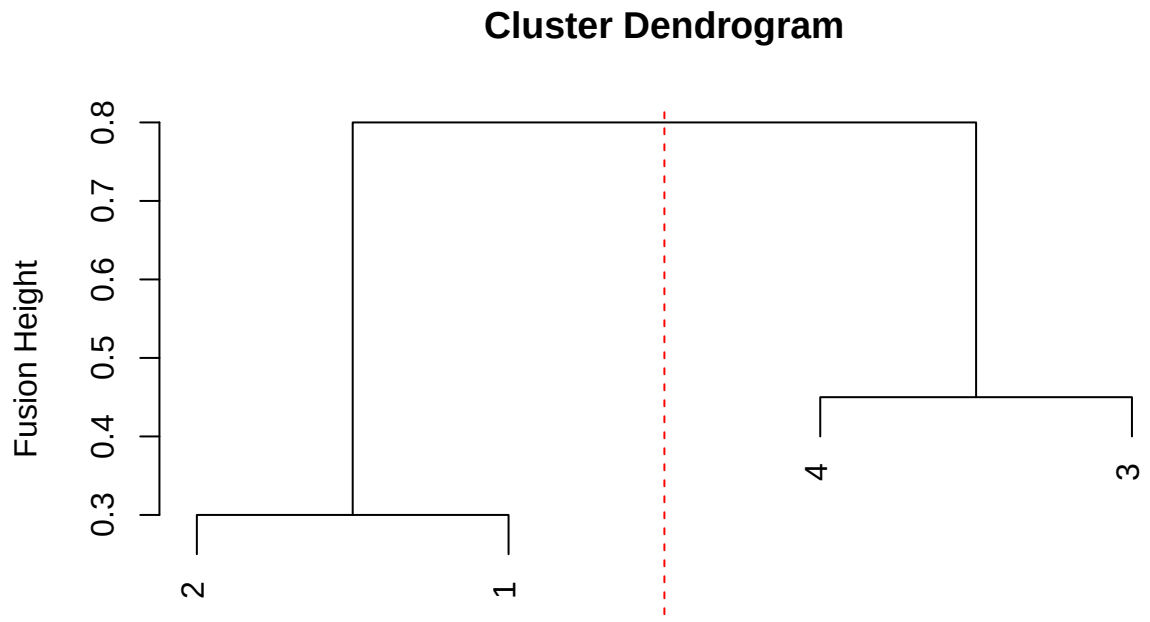


- **C**. Observations 1 & 2 will be in cluster A and observations 3 & 4 will be in cluster B (assuming one cuts the dendrogram at a height greater than 0.45).

- **D**. Although the answer to this question depends on where one cuts the dendrogram, the most likely clusters would contain observations 1 & 2 in cluster A and observations 3 & 4 in cluster B. This would results from a cut at a height greater than 0.3 and less than 0.4, which is the largest vertical distance on the dendrogram. If one were to make a cut between 0.4 and 0.45, then cluster A would contain observations 1, 2 & 3, while cluster B would consist of only observation 4. However, with the distance being greater between cluster's for the first grouping, that would be the more probable grouping.

- **E**. As shown below, one can simply switch the labels of the observations within each cluster to change the dendrogram without changing the meaning of the dendrogram. In addition, one could take the mirror image of the plot displayed along the dotted red line, producing a "new" dendrogram that has the same meaning.

```
plot(hclust(as.dist(x), method = 'complete'),
     xlab = "",
     sub = "",
     ylab = "Fusion Height", labels = c(2,1,4,3))
abline(v = 2.5, col = 'red', lty = 2)
```

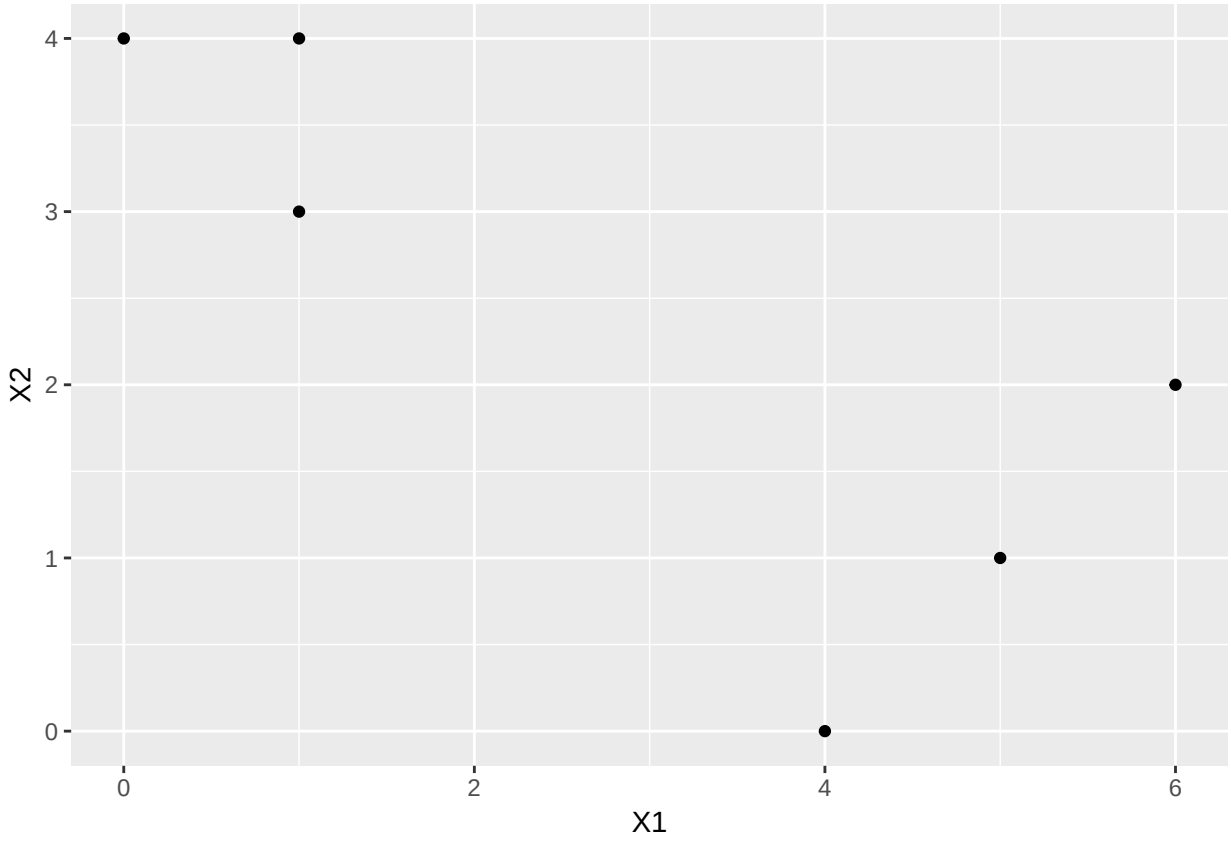


3

- **A**.

```
suppressPackageStartupMessages(library(ggplot2))
df <- data.frame(x1 = c(1,1,0,5,6,4),
```

```
                  x2 = c(4,3,4,1,2,0))
qplot(df$x1, df$x2, xlab = 'X1', ylab = 'X2')
```



- **B**.

```
suppressPackageStartupMessages(library(knitr))
df$group <- sample(c('A','B'), 6, replace = TRUE)
knitr::kable(df, caption = 'Sample Data with Group Assignments')
```

Table 1: Sample Data with Group Assignments

x1	x2	group
1	4	A
1	3	A
0	4	B
5	1	B
6	2	B
4	0	B

- **C**.

```r
group.a <- subset(df, group == 'A')
group.b <- subset(df, group == 'B')

# centroid calculation
calculate.centroid <- function(df, variables=c('x1','x2')) {

    coordinates <- NULL

    for (variable in variables) {
        column.mean <- mean(df[, variable])
        coordinates <- c(column.mean, coordinates)
    }

    return(coordinates)
}
group.a.centroid <- calculate.centroid(group.a)
group.b.centroid <- calculate.centroid(group.b)
print(paste("Group A centroid at coordinates",
            round(group.a.centroid[1], 2), 'and', round(group.a.centroid[2], 2)))
```

```
## [1] "Group A centroid at coordinates 3.5 and 1"
```

```r
print(paste("Group B centroid at coordinates",
            round(group.b.centroid[1], 2), 'and', round(group.b.centroid[2], 2)))
```

```
## [1] "Group B centroid at coordinates 1.75 and 3.75"
```

- **D**.

```r
centroid.matrix <- matrix(c(group.a.centroid, group.b.centroid),
                          nrow = 2,
                          ncol = 2)

reassign.cluster <- function(df,
                             col.idx=c(1,2),
                             cluster.column='group',
                             cluster.labels=c('A','B'),
                             centroids=centroid.matrix) {

    df.matrix <- as.matrix(df[, col.idx])

    updated.labels <- NULL
    for (i in 1:dim(df.matrix)[1]) {

        sqr.manhattan.dist <- (df.matrix[i, ] - centroids)^2
        euclidean.dist <- sqrt(colSums(sqr.manhattan.dist))
        closest.centroid <- which.min(euclidean.dist)
        updated.labels <- c(updated.labels, cluster.labels[closest.centroid])

    }

    return(data.frame(df.matrix, group = updated.labels))
```

```
}
updated.df <- reassign.cluster(df)
knitr::kable(updated.df, caption = "Sample Data After One Reassignment Iteration")
```

Table 2: Sample Data After One Reassignment Iteration

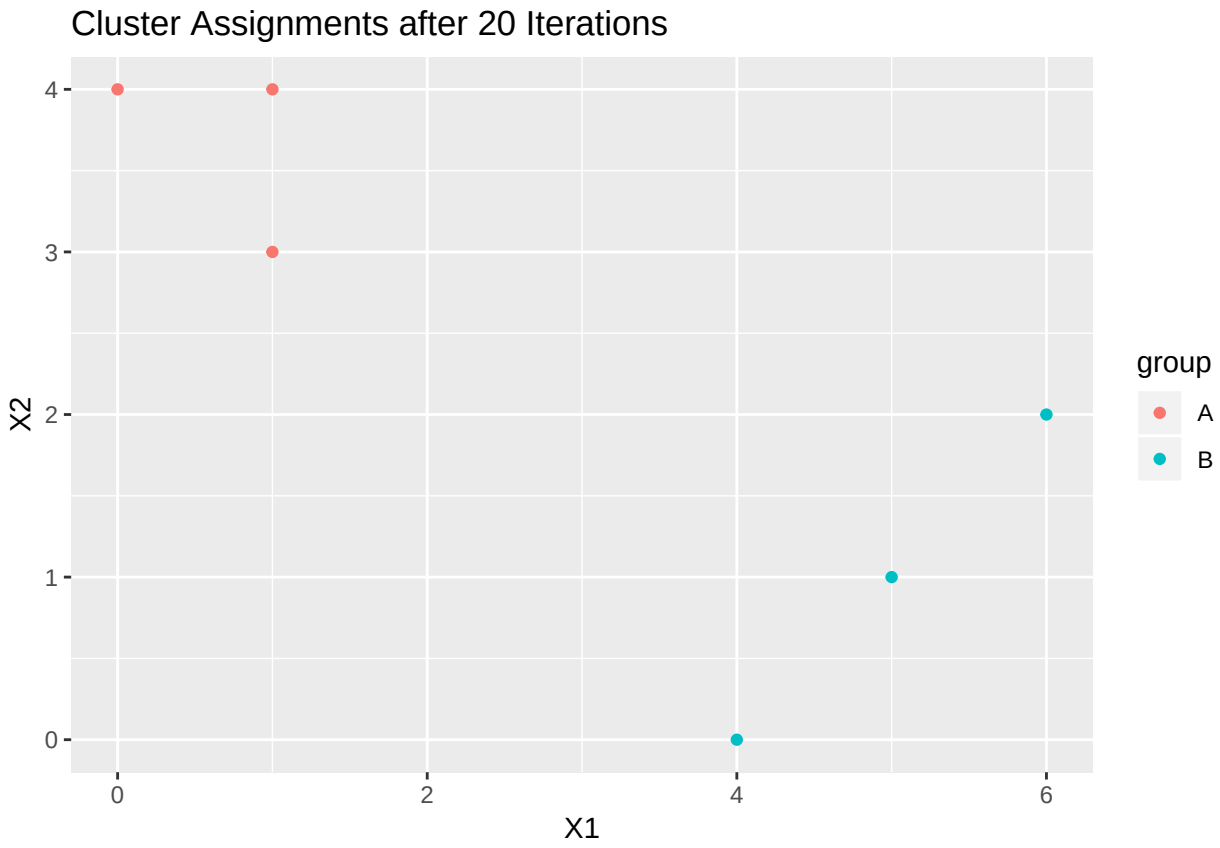
x1	x2	group
1	4	B
1	3	B
0	4	B
5	1	A
6	2	A
4	0	A

- **E**.

```
for (i in 1:20) {
    # split data
    group.a <- subset(df, group == 'A')
    group.b <- subset(df, group == 'B')

    # recalculate centroid
    group.a.centroid <- calculate.centroid(group.a)
    group.b.centroid <- calculate.centroid(group.b)

    # create centroid matrix
    centroid.matrix <- matrix(c(group.a.centroid, group.b.centroid), nrow = 2, ncol = 2)

    # re-assign group labels
    df <- reassign.cluster(df)
}
```

- **F**.

```
ggplot(df, aes(x = x1, y = x2, color = group)) +
    geom_point() +
    xlab('X1') +
    ylab('X2') +
    ggtitle('Cluster Assignments after 20 Iterations')
```



4

- **A**. *Assuming the maximal intercluster dissimilarity is* **not equal** *to the minimal intercluster dissimilarity*, the clusters {1, 2, 3} and {4, 5} will be fused higher on the dendrogram when complete linkage is used. While complete linkage uses the *maximal* intercluster dissimilarity, single linkage uses the *minimal* intercluster dissimilarity. So, if the first assumption is held, using complete linkage will lead to the fusion occurring higher on the dendrogram. However, if the maximal intercluster dissimilarity is equal to the minimal intercluster dissimilarity, then they would fuse at the same height, regardless of the linkage method used.
- **B**. Assuming that in both scenarios the clusters {5} and {6} are by themselves (i.e. not joined with any other clusters/observations), then the clusters would be joined at the same height regardless of whether single or complete linkage were to be used. As stated in part **A**, complete linkage uses the maximal intercluster dissimilarity while single linkage uses the minimal intercluster dissimilarity. In this scenario, both "clusters" only have one observation in them so the maximal and minimal intercluster dissimilarity will be the exact same.

5

The three different scaling options, and the results that one would expect by running the K-means algorithm with K = 2 are:

- *Variables unscaled* - Since the number of socks purchased will be greater than the number of computers purchased and will drive the dissimilarity between the clusters, one would expect the two clusters to be determined by the number of pairs of socks an indiviual shopper purchases. Perhaps those shoppers that purchased seven or fewer pairs of socks would be in one cluster and those who purchased more would be in the other cluser, or something similar.
- *Variables scaled to have standard deviation one* - When the number of items purchased is scaled by that item's standard deviation across the data set, then each variable is given (roughly) equal importance in the clustering algorithm, therefore one would expect clusters with those that bought computers and those that did not buy computers (the *quantity* of socks purchased would have very little affect on the clusters, since the computer variable could conceptually be grouped into the "haves" and the "have not's").
- *Variables scaled by dollars spent* - If the variables were measured in dollars spent (as opposed to quantity purchased), then one would expect the clusters to be dominated by whether a particular shopper purchased a computer or not.

6

- **A**. As shown in the equation below, the total variance in a data set is defined as the summation across all columns of the summation of the squared elements in attribute j divided by N (assuming each column has been centered to have mean 0).

$$\sum_{j=1}^{P} Var(\mathbf{X}) = \sum_{j=1}^{P} \frac{1}{N} \sum_{i=1}^{N} x_{i,j}^2 = Total\ Variance\ in\ \mathbf{X}$$

  After the data set has been "mapped" onto the principal components, saying the m^{th} principal component explaines 10% of the variance in the data set is saying that the summation of the squared elements *of the principal component* divided by N is equal to 10% of the original variation (given in the equation above) in the data set.

$$\frac{\sum_{i=1}^{N} z_{i,m}^2}{\sum_{j=1}^{P} \frac{1}{N} \sum_{i=1}^{N} x_{i,j}^2} = 0.1 \quad where \quad z_{i,m}^2 = \left(\sum_{j=1}^{P} \phi_{j,m} x_{i,j} \right)^2$$

  where $\phi_{j,m}$ = the loading associated with the j^{th} column for the m^{th} principal component.

- **B**. First and foremost, performing a two-sample t-test on *each* gene in the data set, where the two samples are the control group and the treatment group, would be 1000 statistical tests. This would require some statistical method for controlling the family wise error rate, either the Bonferroni Correction or the Benjamini-Hochberg Procedure[1], and even then, these procedures are not built to handle a test quantity of that size.

  In addition, each tissue sample (represented as columns in $\mathbf{X}$) was processed on a different day, and the columns are ordered in a manner such that, "the samples that were processed earliest are on the left, and the samples that were processed later are on the right." In addition, two different machines were used to measure gene expression, and machine A was used more often in the earlier tests, while machine B was used more often later.

  There are a couple points to make about information in the above paragraph:

1. Only one machine should have been used during the experiment. Even if the machines are of the exact same type, one can't rule out that small variations in the gene expression measurements could be attributed to the different machines, as opposed to actual differences in the expression of the gene. I would imagine this would be especially pertinent with gene expression data, however a geneticist would be required to speak more to this point.
2. Since the tissue measurements were taken across a period of time, the "linear relationship from left to right" that the researched observed in the first principal component is likely capturing the two different machines being used, as opposed to true differences in gene expression (in addition, the linear relationship could be indicating that gene expression is changing over time, although a geneticist would be needed to speak to the validity of that).

   Although this would appear obvious and not worth mentioning, there is a little ambiguity in the phrasing of the textbook question, therefore I will make one more point. Performing t-tests on the *transformed* data set would not serve the overall purpose of the analysis, that being determining if there is a statistically significant difference of gene expression between the control group and the treatment group *for each gene*. The transformed data set would be a 100 by M principal components data set, therefore running statistical tests would only tell the researcher whether there are statistically significant differences in the *principal component* scores across tissue samples. If the researcher performed PCA as a pre-analysis *and then reverted back to the original* $\mathbf{X}$, than this point is moot.

- **C**. The majority of what I would change about the experiment would fall into the experimental design category - i.e. only using one machine for gene expression measurement, implementing the Benjamini-Hochberg procedure (or, ideally, choose a few genes to test from this sample and only test those, as opposed to 1000 statistical tests), keeping the randomization of the tissue samples for the control and treatment group being processed on random days, etc. In addition, one would want to determine if the Normality assumption of the t-test holds for genetic variation, whether that be visually looking at QQ plots or performing the Shapiro-Wilk test.

As shown below, even with a False Discovery Rate of 0.25 (rather high), no tests are deemed significant using the Benjamini-Hochberg Procedure.

```
set.seed(10000)
control <- matrix(rnorm(1000*50), nrow = 50, ncol = 1000)
treatment <- matrix(rnorm(1000*50), nrow = 50, ncol = 1000)

df <- data.frame(rbind(control, treatment))
df$group <- sample(c('Control','Treatment'), 100, replace = TRUE)

genes <- colnames(df)
genes <- genes[1:length(genes) - 1]
p.values <- NULL
for (gene in genes) {
    result <- t.test(df[df$group == 'Control', gene], df[df$group != 'Control', gene])
    p.values <- c(p.values, result$p.value)
}

results.df <- data.frame(Gene = genes,
                         PValue = p.values)
results.df <- results.df[order(results.df$PValue),]
results.df$Rank <- seq(1, 1000, 1)
rownames(results.df) <- results.df$Rank

total.tests <- length(genes)
fdr <-  0.25

results.df$BHValue <- results.df$Rank / total.tests * fdr
results.df$Significant <- ifelse(results.df$PValue < results.df$BHValue, 1, 0)
knitr::kable(head(results.df),
             caption = 'Top Statistical Tests Results: Control vs. Treatment')
```

Table 3: Top Statistical Tests Results: Control vs. Treatment

Gene	PValue	Rank	BHValue	Significant
X926	0.0010228	1	0.00025	0
X379	0.0022865	2	0.00050	0
X962	0.0034444	3	0.00075	0
X896	0.0040057	4	0.00100	0
X323	0.0041138	5	0.00125	0
X885	0.0045142	6	0.00150	0

Applied

7

```
df <- USArrests
scaled.df <- t(scale(t(df)))
euc.df.scale <- as.matrix(dist(scaled.df))
# using the transpose to perform a row-wise correlation as opposed to column-wise correlation
cor.df <- cor(t(scaled.df))

constant <- (euc.df.scale^2) / (1 - cor.df)
knitr::kable(constant[1:5,1:5],
             caption = "$(1 - Cor(x_i, y_i)) = Dist(x_i, y_i)^2 * Constant$")
```

Table 4: $(1 - Cor(x_i, y_i)) = Dist(x_i, y_i)^2 * Constant$

	Alabama	Alaska	Arizona	Arkansas	California
Alabama	NaN	6	6	6	6
Alaska	6	NaN	6	6	6
Arizona	6	6	NaN	6	6
Arkansas	6	6	6	NaN	6
California	6	6	6	6	NaN

8

- **A**. Noting that I chose to standardize the data so each attribute is weighted equally.

```
scaled.df <- scale(USArrests)
pca <- prcomp(scaled.df)
pve <-  pca$sdev^2 / sum(pca$sdev^2)
```

- **B**.

```
# total variance in X
total.var <- 0
len <- dim(scaled.df)[1]
for (i in 1:dim(scaled.df)[2]) {
    col.var <- sum(scaled.df[,i]^2)
    total.var <- total.var + col.var
}

transformed.df <- scaled.df %*% pca$rotation
manual.pve <- NULL
for (i in 1:dim(transformed.df)[2]){
    col.var <- sum(transformed.df[,i]^2)
```

```
    manual.pve <- c(manual.pve, col.var/total.var)
}
knitr::kable(data.frame(PrincipalComponent = 1:4,
                        AutomaticPVE = pve,
                        ManualPVE = manual.pve),
             caption = 'Proportion of Variance Explained')
```

Table 5: Proportion of Variance Explained

PrincipalComponent	AutomaticPVE	ManualPVE
1	0.6200604	0.6200604
2	0.2474413	0.2474413
3	0.0891408	0.0891408
4	0.0433575	0.0433575

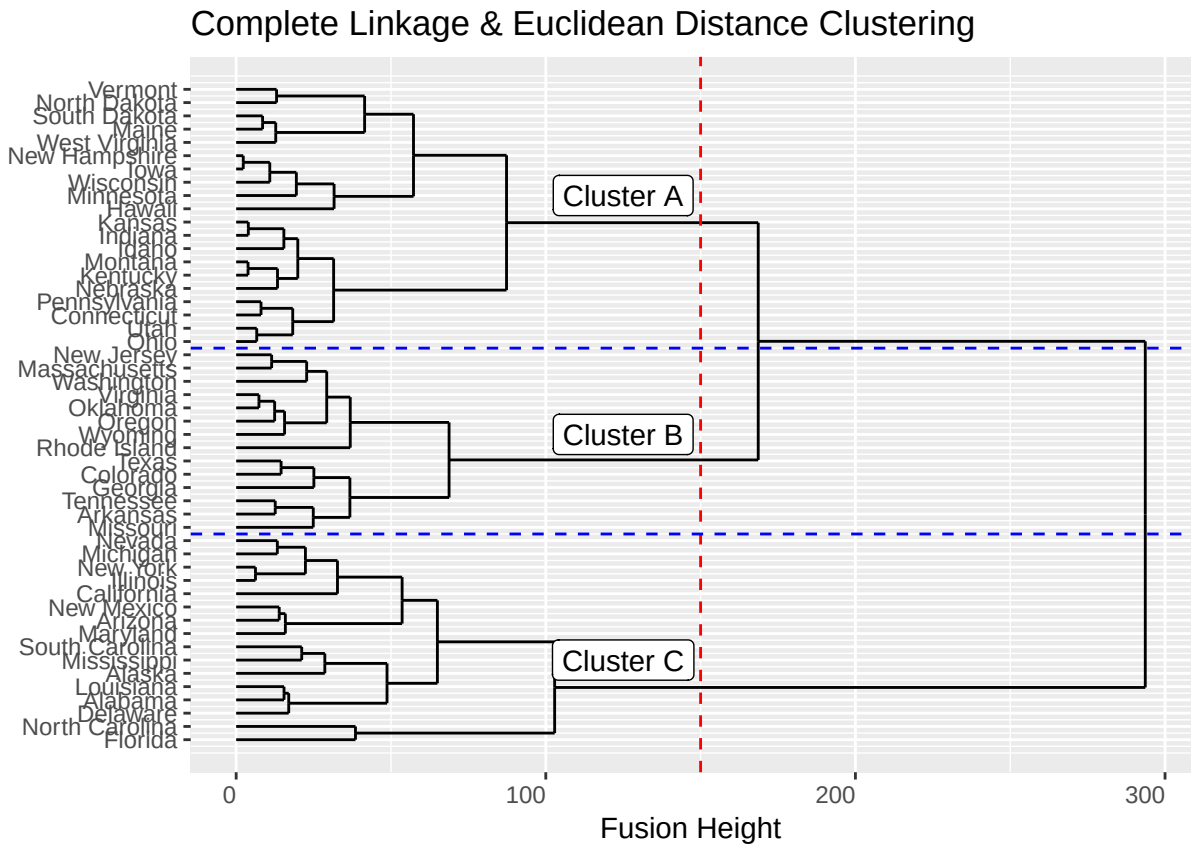
9

- **A**.

```
df <- USArrests
complete <- hclust(dist(df), method = 'complete')
```

- **B**.

```
suppressPackageStartupMessages(library(ggdendro))

label.df <- data.frame(labels = c('Cluster C','Cluster B','Cluster A'),
                       x = c(7, 24, 42),
                       y = (rep(125, 3)))

ggdendrogram(complete, rotate = TRUE, theme_dendro = FALSE) +
    geom_hline(yintercept = 150, col = 'red', lty = 2) +
    geom_vline(xintercept = 16.5, col = 'blue', lty = 2) +
    geom_vline(xintercept = 30.5, col = 'blue', lty = 2) +
    geom_label(data = label.df, aes(x = x, y = y, label = labels)) +
    ggtitle("Complete Linkage & Euclidean Distance Clustering") +
    xlab('') +
    ylab('Fusion Height')
```




```
clusters <- cutree(complete, 3)
clusters
```

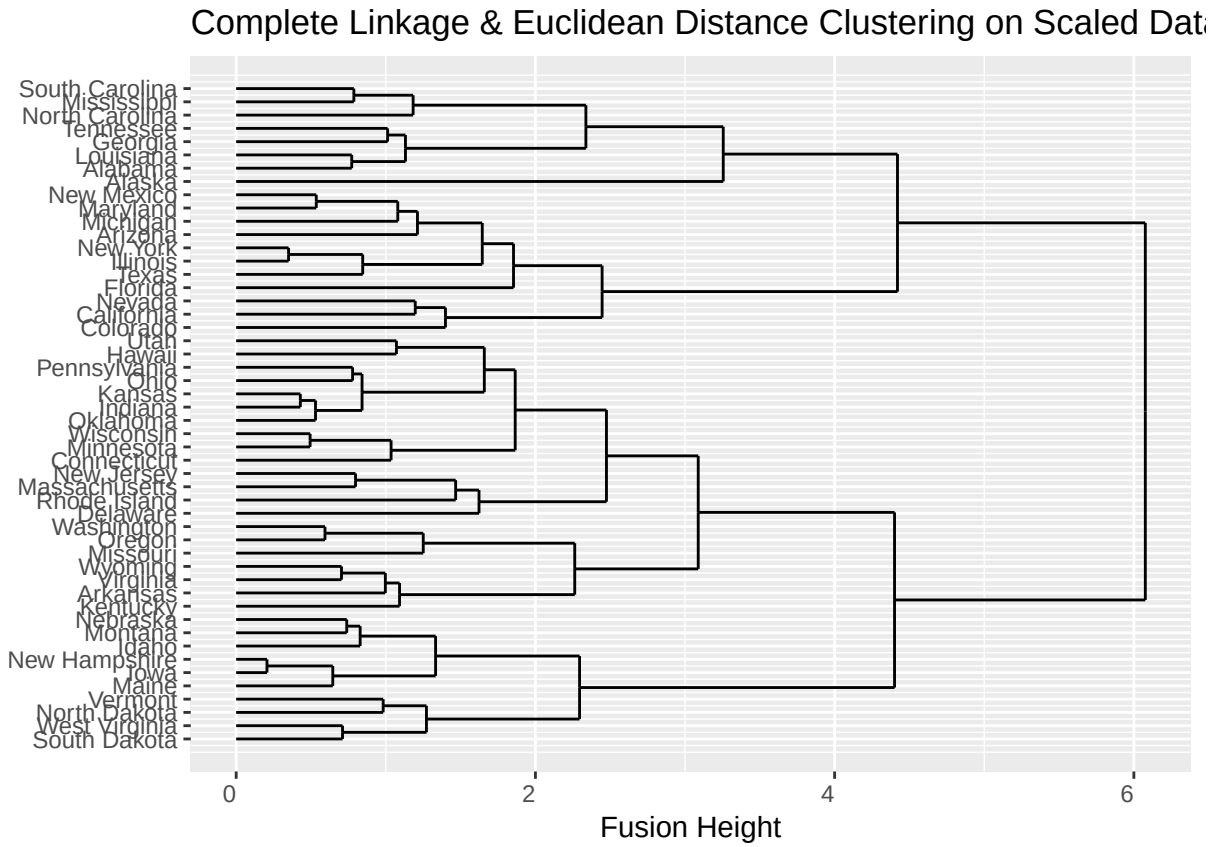
```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              2              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              1              1              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**.

```
complete.scaled <- hclust(dist(scale(df)), method = 'complete')
```

- **D**. Clustering on scaled data yields what could be described as a more 'uniform' dendrogram - the branches seem to fuse at a more continuous pace, with no blatantly obvious height at which the tree should be cut, creating K clusters. That said, it is my opinion that the attributes should be scaled in order standardize the units of each variable and not overweight any variable that might have been measured on a completely different scale than the others.

```
ggdendrogram(complete.scaled, rotate = TRUE, theme_dendro = FALSE) +
    ggtitle("Complete Linkage & Euclidean Distance Clustering on Scaled Data") +
    xlab('') +
    ylab('Fusion Height')
```



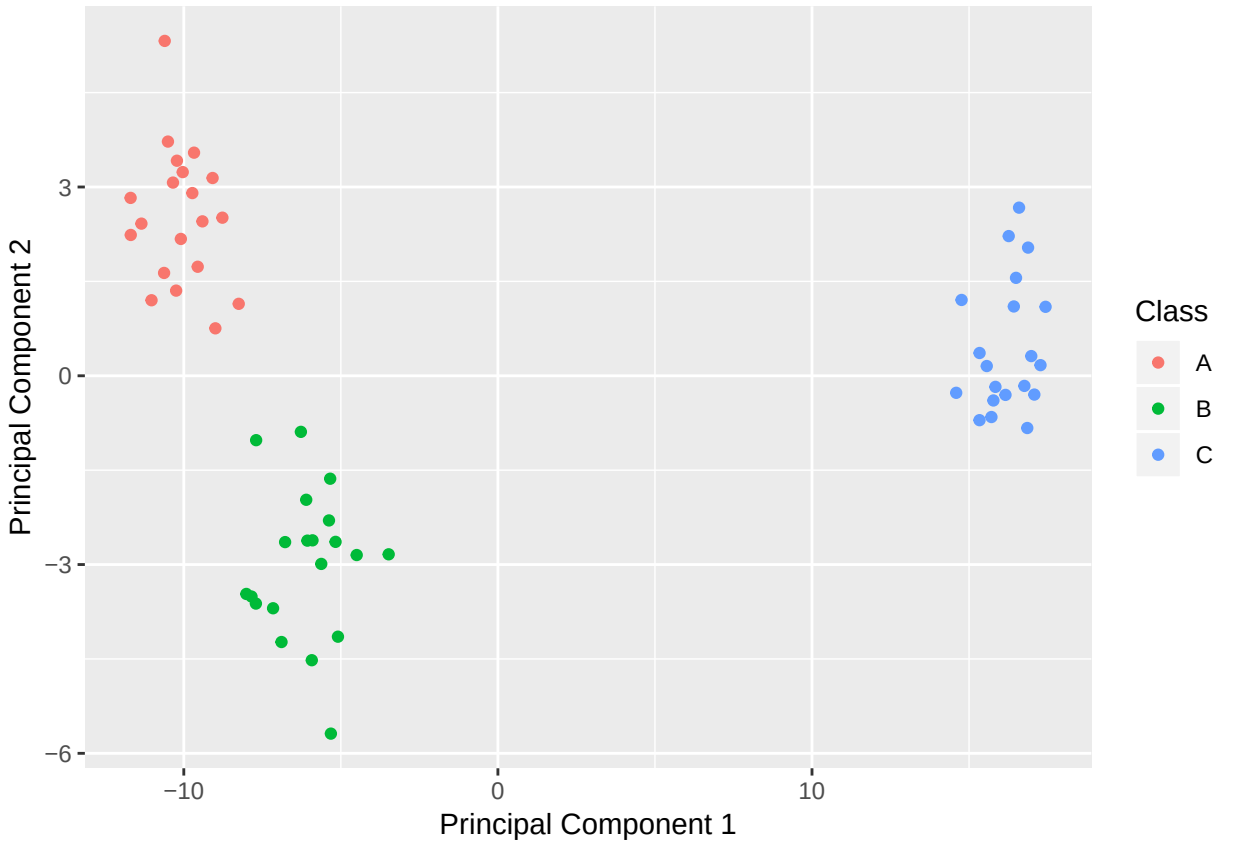
10

- **A**. Creating separation in three dimension between the three classes.

```
set.seed(10000)
df <- data.frame(
    matrix(rnorm(60*50),
           ncol = 50,
           nrow = 60)
)
df$class <- sample(c('A','B','C'), replace = TRUE)

dimensions <- sample(1:50, 3)
for (i in dimensions) {
    df[df$class == 'A',i] <- df[df$class == 'A', i] + sample(5:10, 1)
    df[df$class == 'B',i] <- df[df$class == 'B', i] - sample(-15:-5, 1)
    df[df$class == 'C',i] <- df[df$class == 'C', i] + sample(20:30, 1)
}
```

- **B**.

```
# excluding the class labels column
pca <- prcomp(df[,1:(dim(df)[2] - 1)])
transformed.df <- data.frame(pca$x)
transformed.df$class <- df$class
ggplot(df, aes(x = transformed.df$PC1, y = transformed.df$PC2, color = transformed.df$class)) +
    geom_point() +
    xlab("Principal Component 1") +
    ylab("Principal Component 2") +
    labs(color = 'Class')
```



- **C**. K-means perfectly classifies all observations.

```
# excluding the class labels column
km.out <- kmeans(df[,1:(dim(df)[2] - 1)], centers = 3, nstart = 20)
table(km.out$cluster, df$class)
```

```
##
##      A  B  C
##   1  0  0 20
##   2  0 20  0
##   3 20  0  0
```

,

- **D**. As shown in the table below, performing K-means with 2 clusters results in the algorithm grouping classes A and B together. Looking at the plot of the three distinct clusters above, this makes sense as there is a greater division between class C and the other two if grouped together than any other grouping. Another way to look at this is, if one were to create a "self vs. others" SVM for each of the three classes, the margin would be far and away the widest for the "class C vs. others" classfier.

```
km.out <- kmeans(df[,1:(dim(df)[2] - 1)], centers = 2, nstart = 20)
table(km.out$cluster, df$class)
```

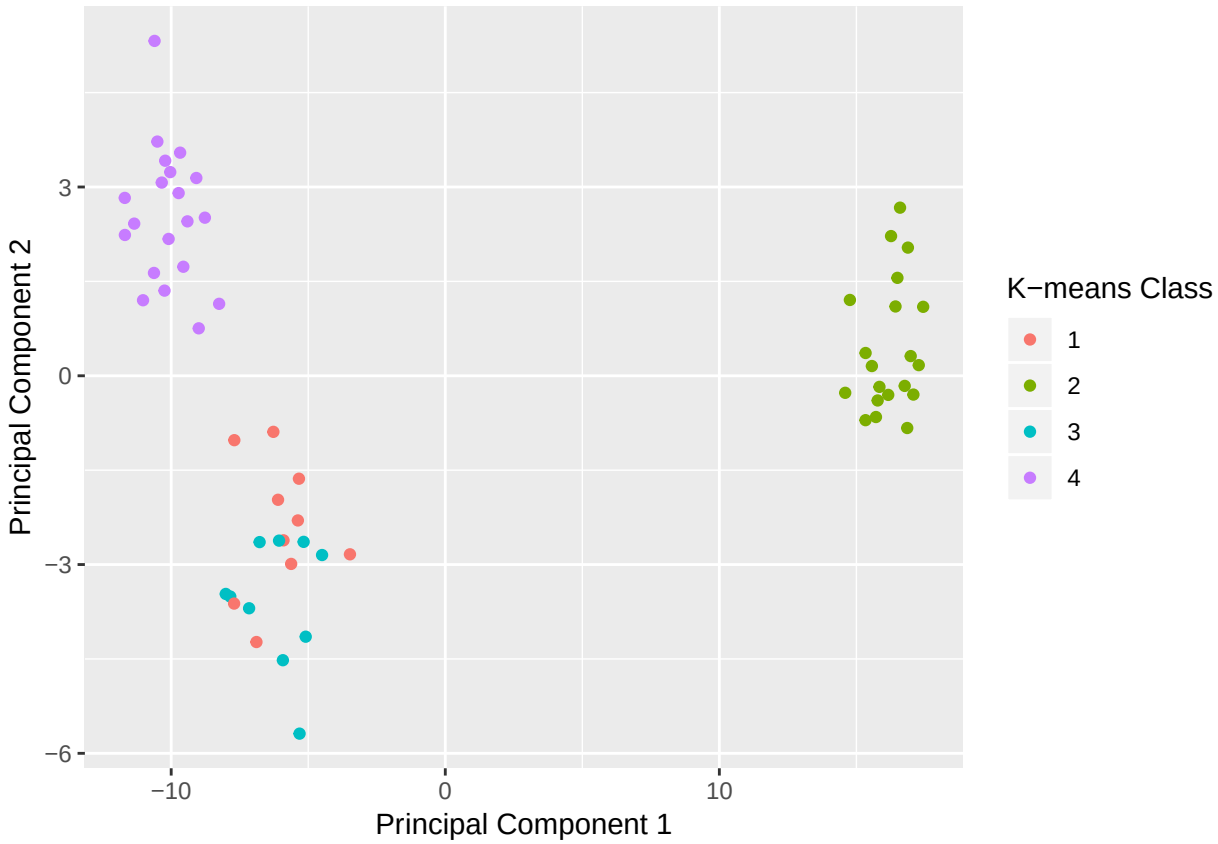
```
##
##      A  B  C
##   1  0  0 20
##   2 20 20  0
```

- **E**. Using 4 clusters, class B gets split into two separate clusters. Replicating the above plot below, this time coloring the observations based on the assignments from K-means, this becomes apparent. Although there isn't a clear delineation between classes 1 and 3 in the below plot, that is an artifact of only plotting in two dimensions.

```
km.out <- kmeans(df[,1:(dim(df)[2] - 1)], centers = 4, nstart = 20)
table(km.out$cluster, df$class)
```

```
##
##      A  B  C
##   1  0 10  0
##   2  0  0 20
##   3  0 10  0
##   4 20  0  0
```

```
ggplot(df, aes(x = transformed.df$PC1, y = transformed.df$PC2, color = factor(km.out$cluster))) +
    geom_point() +
    xlab("Principal Component 1") +
    ylab("Principal Component 2") +
    labs(color = 'K-means Class')
```



- **F**. Performing K-means on the first two principal components perfectly classifies the data

```
km.pca <- kmeans(pca$x[,1:2], centers = 3, nstart = 20)
table(km.pca$cluster, df$class)
```

```
##
##      A  B  C
##   1 20  0  0
##   2  0  0 20
##   3  0 20  0
```

- **G**. As shown in the table below, performing K-means on scaled data does *not* always lead to superior results.

```
km.out <- kmeans(scale(df[,1:(dim(df)[2] - 1)]), centers = 3, nstart = 20)
table(km.out$cluster, df$class)
```

```
##
##      A  B  C
##   1 14 13  0
##   2  6  7  0
##   3  0  0 20
```

Resources & References

Dendrograms in R

[1] Multiple Comparisons Overview