STK2100: Solutions Week 17

Lars H. B. Olsen

06.05.2021

ISLR

Exercise 1

a)

Prove equation (10.12), which is the same as showing that the left hand side equals the right hand side. Let $\bar{x}_{kj} = \frac{1}{|C_k|}\sum_{i\in C_k} x_{ij}$, then we get

$$
\begin{aligned}
\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 &= \frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj}+\bar{x}_{kj}-x_{i'j})^2 \\
&= \frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}((x_{ij}-\bar{x}_{kj})-(x_{i'j}-\bar{x}_{kj}))^2 \\
&= \frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}\left((x_{ij}-\bar{x}_{kj})^2-2(x_{ij}-\bar{x}_{kj})(x_{i'j}-\bar{x}_{kj})+(x_{i'j}-\bar{x}_{kj})^2\right) \\
&= \frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2-\frac{2}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})(x_{i'j}-\bar{x}_{kj})+\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{i'j}-\bar{x}_{kj})^2 \\
&= \frac{|C_k|}{|C_k|}\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2-\frac{2}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})(x_{i'j}-\bar{x}_{kj})+\frac{|C_k|}{|C_k|}\sum_{i'\in C_k}\sum_{j=1}^{p}(x_{i'j}-\bar{x}_{kj})^2 \\
&= \sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2-\frac{2}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})(x_{i'j}-\bar{x}_{kj})+\sum_{i\in C_k}\sum_{j=1}^{p}(\bar{x}_{kj}-x_{ij})^2 \\
&= 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2-\frac{2}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})(x_{i'j}-\bar{x}_{kj}) \\
&= 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2-0 \\
&= 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2,
\end{aligned}
$$

which is what we were asked to show.

The last term is zero as

$$\begin{aligned}
\sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})(x_{i'j} - \bar{x}_{kj}) &= \sum_{i \in C_k} \sum_{i' \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})(x_{i'j} - \bar{x}_{kj}) \\
&= \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj}) \sum_{i' \in C_k} (x_{i'j} - \bar{x}_{kj}) \\
&= \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj}) \left(\sum_{i' \in C_k} x_{i'j} - |C_k| \bar{x}_{kj} \right) \\
&= \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj}) \left(\sum_{i' \in C_k} x_{i'j} - \sum_{i' \in C_k} x_{i'j} \right) \\
&= \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj}) \, (0) \\
&= 0.
\end{aligned}$$

Thus we have shown that equation (10.12) holds, namely,

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

b)

We are asked to argue, on the basis of this identity, that the K-means clustering algorithm (Algorithm 10.1) decreases the objective (10.11) at each iteration.

As K-means clustering algorithm assigns the observations to the clusters to which they are nearest, after each iteration, the value of RHS will decrease (as this quantity is the sum of squared distance of each observation from the cluster mean). Hence, the clustering algorithm decreases the objective at each iteration.

Exercise 3

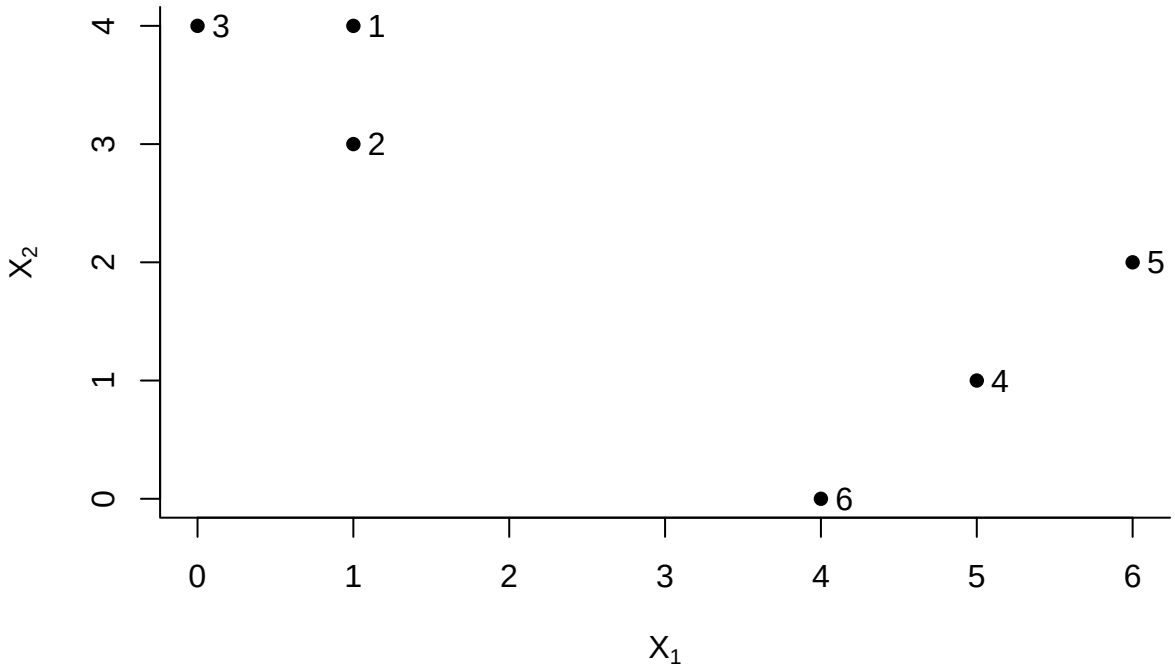
In this problem, we will perform K-means clustering manually, with $K = 2$, on a small example with $n = 6$ observations and $p = 2$ features. The observations are as follows.

Observation	X_1	X_2
1	1	4
2	1	3
3	0	4
4	5	1
5	6	2
6	4	0

a)

```
x1 = c(1, 1, 0, 5, 6, 4)
x2 = c(4, 3, 4, 1, 2, 0)
x = cbind(x1, x2)
plot(x[,1], x[,2],
     xlab = expression(X[1]),
     ylab = expression(X[2]),
     bty = "l",
```

```
     pch = 16)
text(x[,1]+0.15, x[,2], 1:6)
```



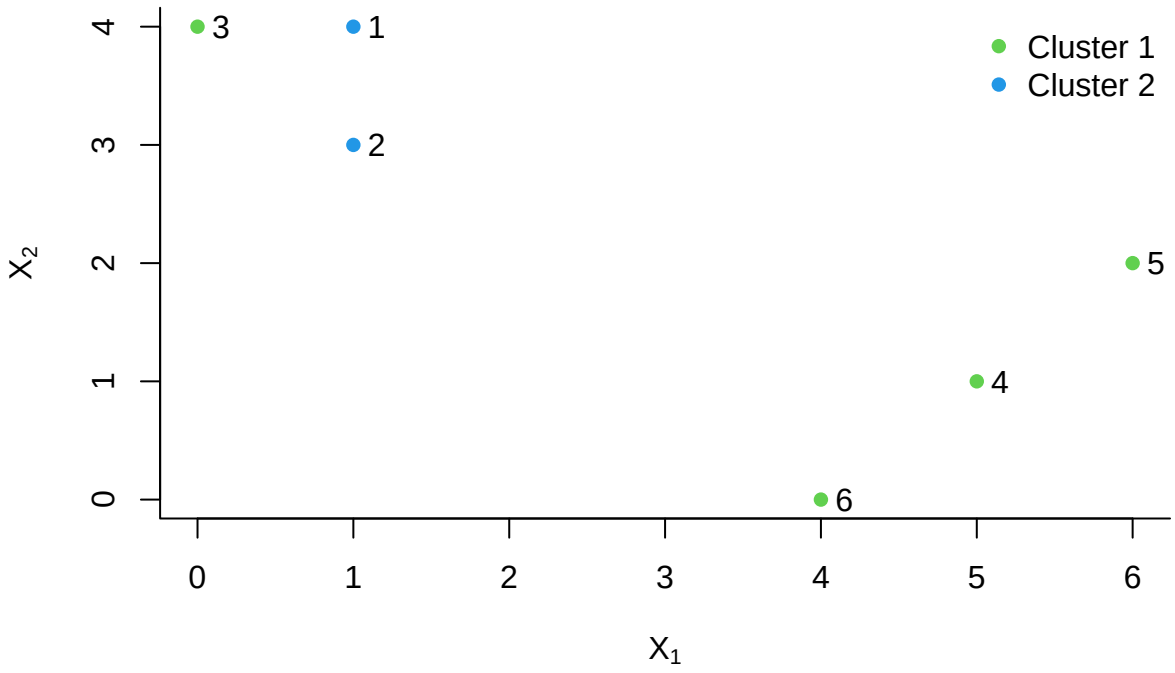
b)

Randomly assign a cluster label to each observation. Report the cluster labels for each observation.

```
set.seed(2100)
labels = sample(2, nrow(x), replace = T)
x = cbind(x, labels)
x
```

```
##      x1 x2 labels
## [1,]  1  4      2
## [2,]  1  3      2
## [3,]  0  4      1
## [4,]  5  1      1
## [5,]  6  2      1
## [6,]  4  0      1
```

```
plot(x[,1], x[,2],
     xlab = expression(X[1]),
     ylab = expression(X[2]),
     bty = "l",
     col = labels+2,
     pch = 16)
text(x[,1]+0.15, x[,2], 1:6)
legend("topright", legend = c("Cluster 1", "Cluster 2"),
       col = c(1,2)+2, pch = 16, bty = "n")
```



c)

Compute the centroid for each cluster.

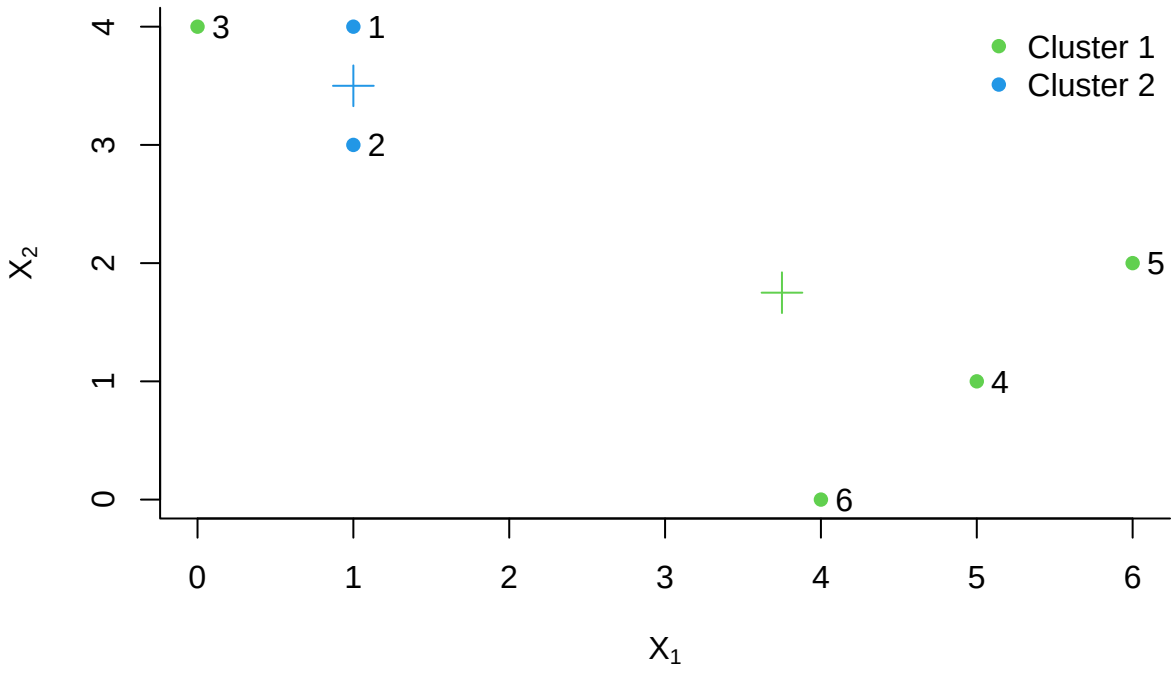
```
# Cluster one
centroid_one = colMeans(x[x[,3] == 1,, drop = FALSE])[1:2]
centroid_one
```

```
##   x1   x2
## 3.75 1.75
```

```
# Cluster two
centroid_two = colMeans(x[x[,3] == 2,, drop = FALSE])[1:2]
centroid_two
```

```
##  x1  x2
## 1.0 3.5
```

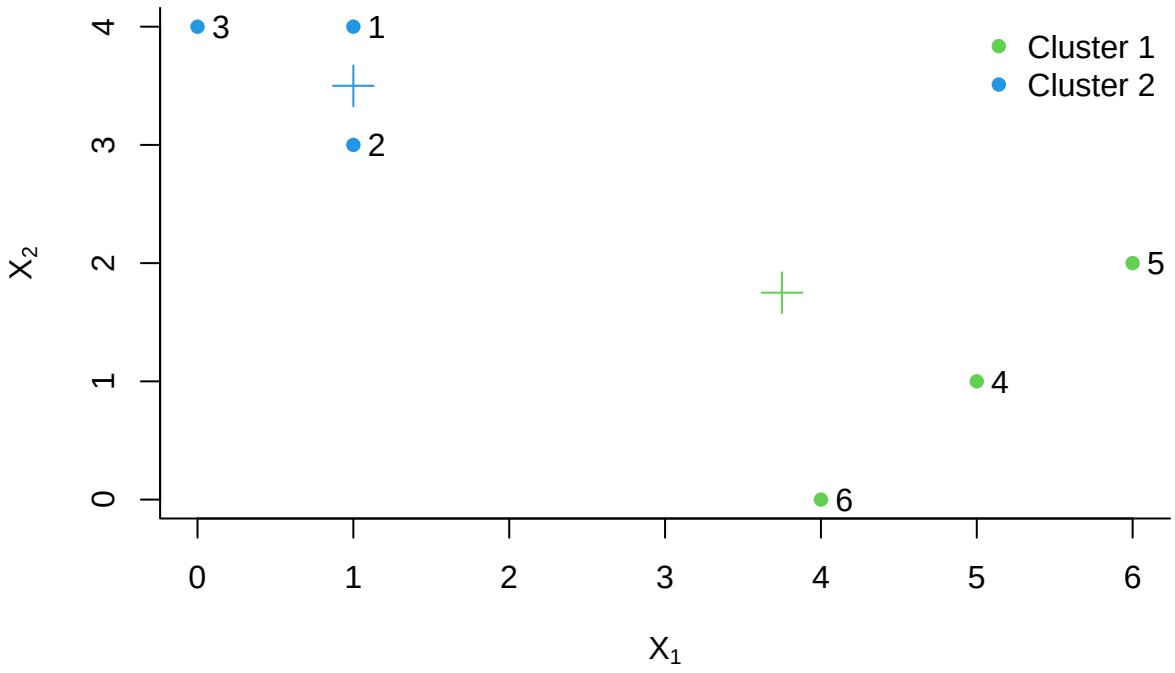
```
plot(x[,1], x[,2],
     xlab = expression(X[1]),
     ylab = expression(X[2]),
     col = labels+2,
     bty = "l",
     pch = 16)
text(x[,1]+0.15, x[,2], 1:6)
legend("topright", legend = c("Cluster 1", "Cluster 2"),
       col = c(1,2)+2, pch = 16, bty = "n")
points(centroid_one[1], centroid_one[2], pch = 3, col = 3, cex = 2)
points(centroid_two[1], centroid_two[2], pch = 3, col = 4, cex = 2)
```



d)

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

```
labels = c(2, 2, 2, 1, 1, 1)
x[,3] = labels
plot(x[, 1], x[, 2],
     xlab = expression(X[1]),
     ylab = expression(X[2]),
     bty = "l",
     col = labels + 2,
     pch = 16)
text(x[,1]+0.15, x[,2], 1:6)
legend("topright", legend = c("Cluster 1", "Cluster 2"),
       col = c(1,2)+2, pch = 16, bty = "n")
points(centroid_one[1], centroid_one[2], pch = 3, col = 3, cex = 2)
points(centroid_two[1], centroid_two[2], pch = 3, col = 4, cex = 2)
```



e)

Repeat (c) and (d) until the answers obtained stop changing. We only need one more iteation. Start by computing the two new centroids.

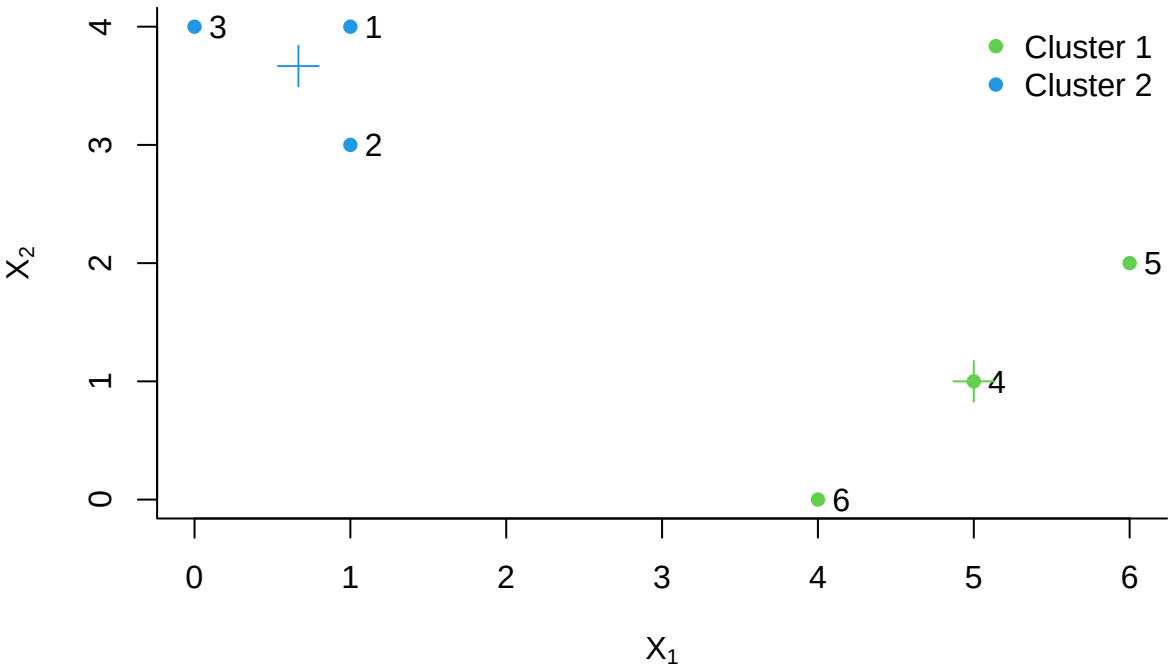
```
# Cluster one
centroid_one = colMeans(x[x[,3] == 1,, drop = FALSE])[1:2]
centroid_one
```

```
## x1 x2
##  5  1
```

```
# Cluster two
centroid_two = colMeans(x[x[,3] == 2,, drop = FALSE])[1:2]
centroid_two
```

```
##        x1        x2
## 0.6666667 3.6666667
```

```
plot(x[,1], x[,2],
     xlab = expression(X[1]),
     ylab = expression(X[2]),
     col = labels+2,
     bty = "l",
     pch = 16)
text(x[,1]+0.15, x[,2], 1:6)
legend("topright", legend = c("Cluster 1", "Cluster 2"),
       col = c(1,2)+2, pch = 16, bty = "n")
points(centroid_one[1], centroid_one[2], pch = 3, col = 3, cex = 2)
points(centroid_two[1], centroid_two[2], pch = 3, col = 4, cex = 2)
```



If we assign each observation to the centroid to which it is closest, nothing changes, so the algorithm is terminated at this step.

f)

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

Just look at the figure above.

Exercise 4

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

See section 10.3.2 in ISLR for information on Hierarchical clustering. In particular, *complete linkage* means the maximal intercluster dissimilarity and *single linkage* means the minimal intercluster dissimilarity. That is, compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* or *smallest* of these dissimilarities, respectively. Here, we denote the dissimilarity, between two observations x and y, as $d(x, y)$. Often one uses the Euclidean distance for numerical observations. Note that complete linkage is generally preferred over single linkage, as it yields more balanced dendrograms. And recall that the height on the y-axis of dendrogram represent the dissimilarity at the fusion step between two clusters.

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. For example, if $d(1, 4) = 2$, $d(1, 5) = 3$, $d(2, 4) = 1$, $d(2, 5) = 3$, $d(3, 4) = 4$ and $d(3, 5) = 1$, the single linkage dissimilarity between $\{1, 2, 3\}$ and $\{4, 5\}$ would be equal to 1 and the complete linkage dissimilarity between $\{1, 2, 3\}$ and $\{4, 5\}$ would be equal to 4. So, with single linkage, they would fuse at a height of 1, and with complete linkage, they would fuse at a height of 4. But, if all inter-observations distance are equal to 2, we would have that the single and complete linkage dissimilarities between $\{1, 2, 3\}$ and $\{4, 5\}$ are equal to 2.

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?

They would fuse at the same height. For example, if $d(5, 6) = 2$, the single and complete linkage dissimilarities between {5} and {6} would be equal to 2. So, they would fuse at a height of 2 for single and complete linkage.

Exercise 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.

```
data = USArrests

# Look a bit at the data
head(data)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

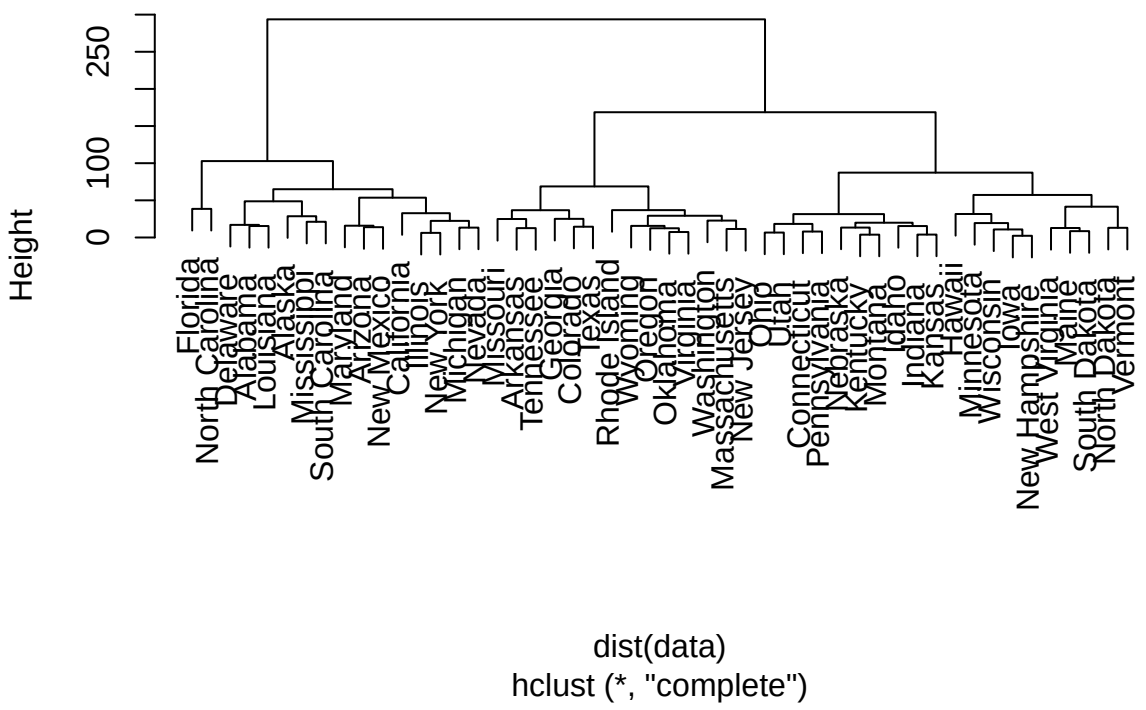
```
summary(data)
```

```
##      Murder          Assault         UrbanPop          Rape
##  Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
##  1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
##  Median : 7.250   Median :159.0   Median :66.00   Median :20.10
##  Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
##  3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
##  Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

```
# Set seed for reproducibility
set.seed(2100)

# Create the complete linkage cluster
# 'dist()' computes the Euclidean distance between the observations.
complete_cluster = hclust(dist(data), method="complete")

# Plot the complete linkage cluster
plot(complete_cluster)
```



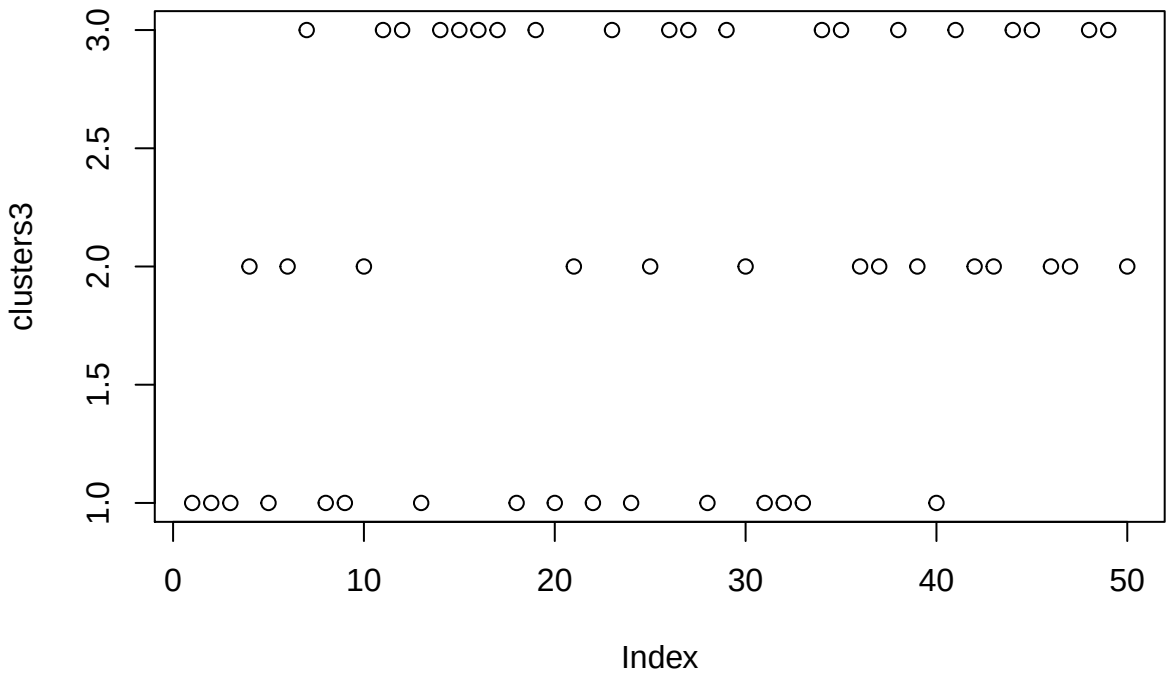
b)

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

```
# Call the cutree function and specify the number of clusters.
clusters3 = cutree(complete_cluster, 3)
clusters3
```

```
##        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
```

```r
plot(clusters3)
```




```r
table(clusters3)
```

```
## clusters3
##  1  2  3
## 16 14 20
```

The Table summarizes number of states in each cluster and which states belongs to which cluster.

c)

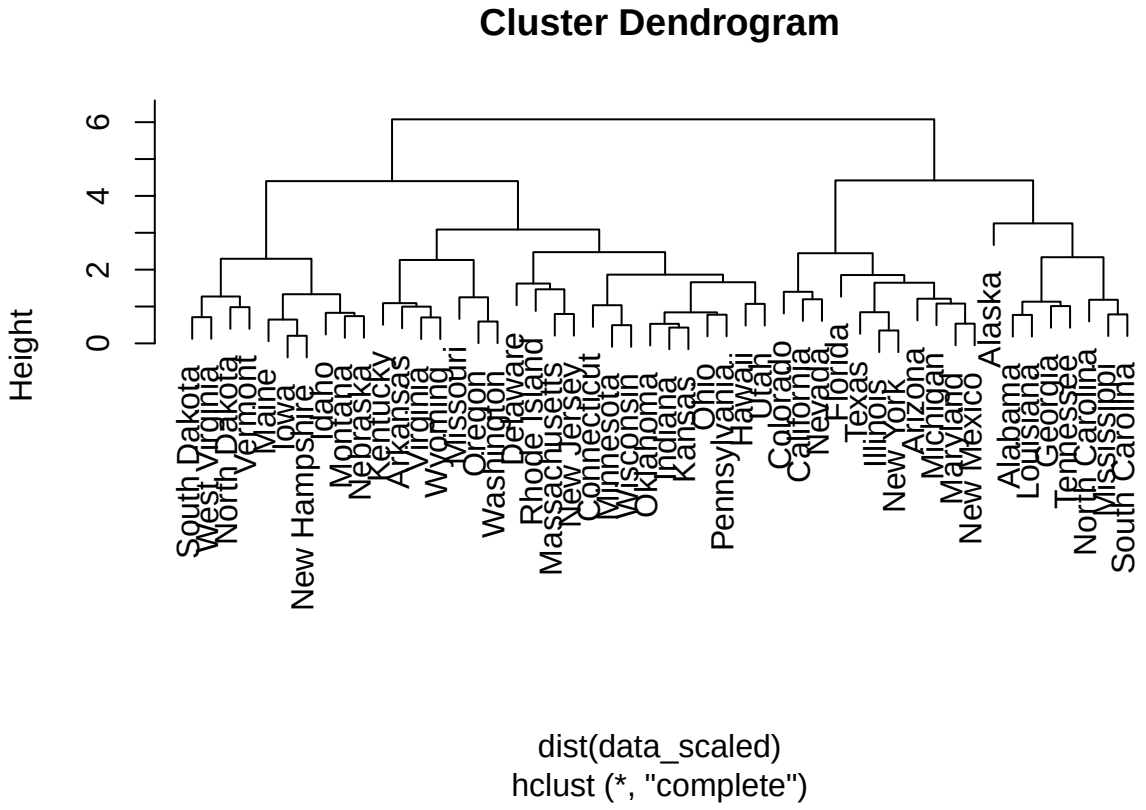
Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```r
# Scale the data
data_scaled = scale(USArrests)

# Set seed for reproducibility
set.seed(2100)

# Create the complete linkage cluster based on scaled data.
# 'dist()' computes the Euclidean distance between the observations.
complete_cluster_scaled = hclust(dist(data_scaled), method="complete")

# Plot the complete linkage cluster with scaled data
plot(complete_cluster_scaled)
```



d)

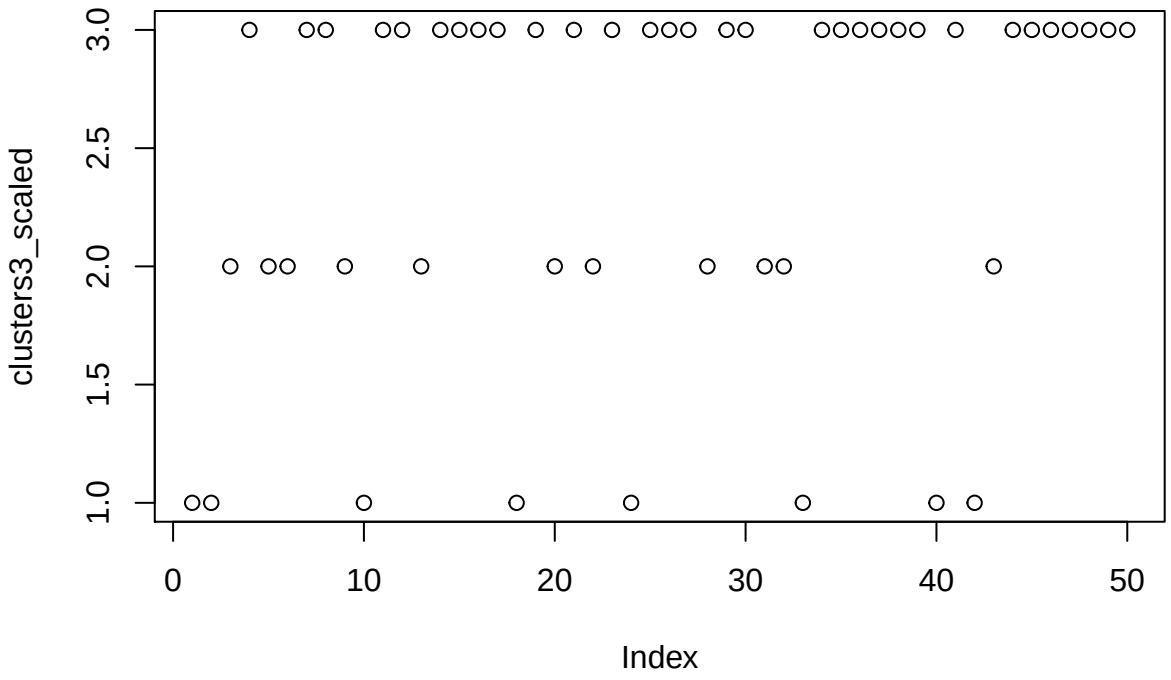
What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer.

```
# Call the cutree function and specify the number of clusters.
clusters3_scaled = cutree(complete_cluster_scaled, 3)
clusters3_scaled
```

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

```
##               3               3               3               3               3
```

```
plot(clusters3_scaled)
```




```
table(clusters3_scaled)
```

```
## clusters3_scaled
##  1  2  3
##  8 11 31
```

```
rbind(clusters3, clusters3_scaled)
```

```
##                 Alabama Alaska Arizona Arkansas California Colorado
## clusters3             1      1       1        2          1        2
## clusters3_scaled      1      1       2        3          2        2
##                 Connecticut Delaware Florida Georgia Hawaii Idaho Illinois
## clusters3                 3        1       1       2      3     3        1
## clusters3_scaled          3        3       2       1      3     3        2
##                 Indiana Iowa Kansas Kentucky Louisiana Maine Maryland
## clusters3             3    3      3        3         1     3        1
## clusters3_scaled      3    3      3        3         1     3        2
##                 Massachusetts Michigan Minnesota Mississippi Missouri Montana
## clusters3                   2        1         3           1        2       3
## clusters3_scaled            3        2         3           1        3       3
##                 Nebraska Nevada New Hampshire New Jersey New Mexico New York
## clusters3              3      1             3          2          1        1
## clusters3_scaled       3      2             3          3          2        2
##                 North Carolina North Dakota Ohio Oklahoma Oregon Pennsylvania
## clusters3                    1            3    3        2      2            3
## clusters3_scaled             1            3    3        3      3            3
##                 Rhode Island South Carolina South Dakota Tennessee Texas Utah
## clusters3                  2              1            3         2     2    3
## clusters3_scaled           3              1            3         1     2    3
##                 Vermont Virginia Washington West Virginia Wisconsin Wyoming
```

```
## clusters3              3        2          2             3         3       2
## clusters3_scaled       3        3          3             3         3       3
```

Scaling affects the clusters. Scaling should be done if the units of measure of variables are different, which is the case for the `USArrests` data set.

```
# Before scaling
mean = apply(data, 2, mean)
sd = apply(data, 2, sd)
IQR = apply(data, 2, IQR)
show(round(rbind(mean, sd, IQR), 3))
```

```
##      Murder Assault UrbanPop   Rape
## mean  7.788 170.760   65.540 21.232
## sd    4.356  83.338   14.475  9.366
## IQR   7.175 140.000   23.250 11.100
```

```
# After scaling
mean = apply(data_scaled, 2, mean)
sd = apply(data_scaled, 2, sd)
IQR = apply(data_scaled, 2, IQR)
show(round(rbind(mean, sd, IQR), 3))
```

```
##      Murder Assault UrbanPop  Rape
## mean  0.000    0.00    0.000 0.000
## sd    1.000    1.00    1.000 1.000
## IQR   1.647    1.68    1.606 1.185
```

Exercise 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.

```
# Set seed for reproducibility
set.seed(2021)

# Number of classes
n_classes = 3

# Number of obs in each class
n_obs = 20

# Number of variables
n_var = 50

# Generate the data
# Can increase the mean to separate the clusters of PC more.
data =
  matrix(sapply(1:3,
                function(x) rnorm(n_obs*n_var, mean = 12*sqrt(x))),
         ncol=n_var) #, byrow = TRUE) yield completely different values.
dim(data)
```

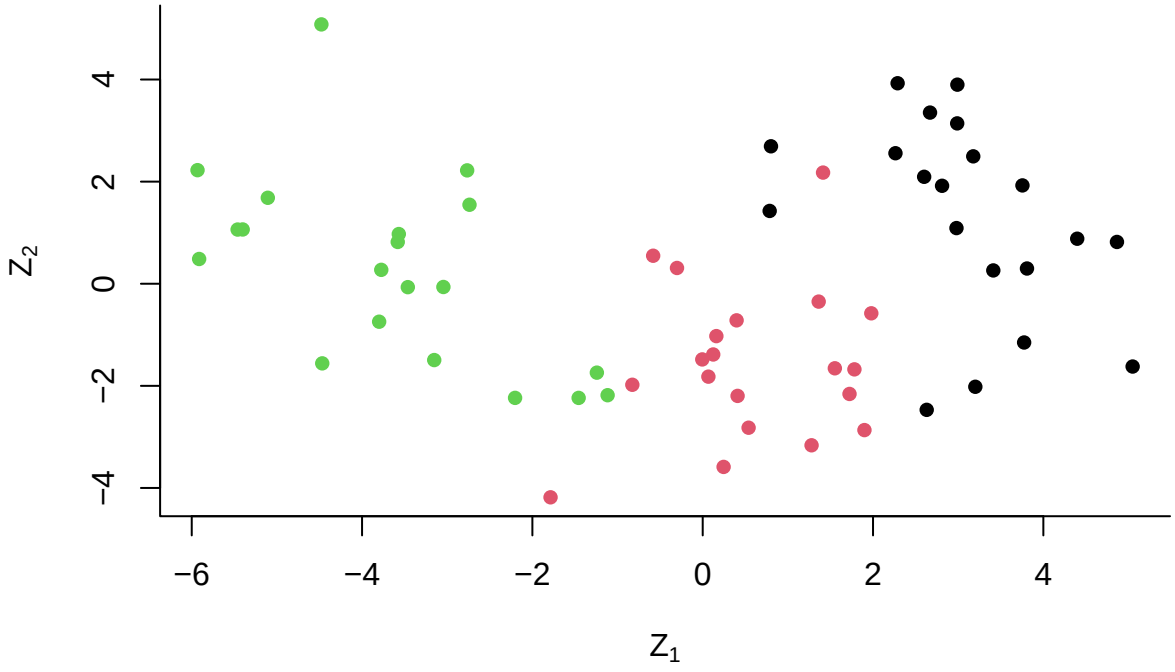
```
## [1] 60 50
```

```
# Generate the true labels
true_labels = rep(1:3, each = n_obs)
```

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, the 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.

```
pr_results = prcomp(data)
plot(pr_results$x[, 1:2],
     col = true_labels,
     bty = "l",
     xlab = expression(Z[1]),
     ylab = expression(Z[2]),
     pch = 16)
```



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?

```
K = 3
km_results <- kmeans(data, K)
table(true_labels, km_results$cluster)
```

```
##
## true_labels  1  2  3
##           1 18  2  0
##           2  1 19  0
##           3  0  4 16
```

Quite good classifications. Miss a little bit, as expected as there is some overlap between the clusters.

d)

Perform K-means clustering with $K = 2$. Describe your results.

```
K = 2
km_results_2 <- kmeans(data, K)
table(true_labels, km_results_2$cluster)
```

```
##
## true_labels  1  2
##           1 20  0
##           2 18  2
##           3  0 20
```

See that the red observations are now absorbed into the two other classes, mainly into the black class. We perfectly cluster the green and black observations.

e)

Now perform K-means clustering with $K = 4$, and describe your results.

```
K = 4
km_results_4 <- kmeans(data, K)
table(true_labels, km_results_4$cluster)
```

```
##
## true_labels  1  2  3  4
##           1  0  2 18  0
##           2 16  3  1  0
##           3  1  3  0 16
```

We see that especially the first cluster have been split into two classes. The third cluster have also somewhat been split. Would have gotten other results if we increased the *mean* in *a)*. Try that yourself!

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 60×2 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.

```
K = 3
km_results_pc = kmeans(pr_results$x[, 1:2], K)
table(true_labels, km_results_pc$cluster)
```

```
##
## true_labels  1  2  3
##           1  2 18  0
##           2 19  1  0
##           3  4  0 16
```

The observations are nearly perfectly clustered once again. Not perfect as there is some overlap between them, would change with different mean.

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)*?

```
K = 3
km_results_scale = kmeans(scale(data), K)
table(true_labels, km_results_scale$cluster)
```

```
##
## true_labels  1  2  3
##           1  4 11  5
##           2 11  5  4
##           3  6  2 12
```

We may see that we have worse results than with unscaled data, as scaling affects the distance between the observations.