

## Ch.10 Exercises: Unsupervised Learning

```
library(e1071)
library(ISLR)
require(caTools)
```

```
## Loading required package: caTools
```

```
## Warning: package 'caTools' was built under R version 3.6.2
```

```
require(plotrix)
```

```
## Loading required package: plotrix
```

```
library(tidyr)
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.6.2
```

### Conceptual

#### 1. (a)

We have the following equation:

$$\frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^p (x_{ij} - \bar{x}_{kj})^2 \quad (10.12)$$

Where,  $\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$  is the mean for feature  $j$  in cluster  $C_k$ .

Expanding the LHS over  $i$ :

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

Further expanding a single term from the summation:

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

Substituting the final term from above in the summation:

$$2 \frac{|C_k|}{|C_k|} \sum_{i \in C_k} \sum_{j=1}^p (x_{ij} - \bar{x}_j)^2 - \underbrace{\frac{2}{|C_k|} \sum_{i' \in C_k, i' \neq i} \sum_{j=1}^p (x_{1j} - \bar{x}_j)(x_{i'j} - \bar{x}_j) - \frac{2}{|C_k|} \sum_{i' \in C_k, i' \neq i} \sum_{j=1}^p (x_{2j} - \bar{x}_j)(x_{i'j} - \bar{x}_j) + \dots}_{\text{This term is equal to zero because } \sum_{i' \in C_k} x_{i'j} - |C_k| \bar{x}_j = 0}$$

The remaining term is:

$$2 \frac{|C_k|}{|C_k|} \sum_{i \in C_k} \sum_{j=1}^p (x_{ij} - \bar{x}_j)^2 \quad \text{Proving 10.12}$$

(b)

- At end of each iteration the K-Means algorithm assigns the observations to the closest (Euclidean distance) centroid. Minimising the RHS of the identity will reduce the euclidean distance and so the LHS (10.11) will also be reduced. In other words the identity shows that minimizing the sum of the squared Euclidean distance for each cluster is the same as minimizing the within-cluster variance for each cluster.

## 2. (a)

Using algorithm 10.2 and complete linkage (maximal intercluster dissimilarity) will result in the following dendrogram:

Initial dissimilarity matrix:

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

Treating each observation as its own cluster and fusing those most similar, will result in clusters 1 and 2 being fused at height 0.3 and giving us the following matrix:

$$\begin{bmatrix} & 0.5 & 0.8 \\ 0.5 & & 0.45 \\ 0.8 & 0.45 & \end{bmatrix}$$

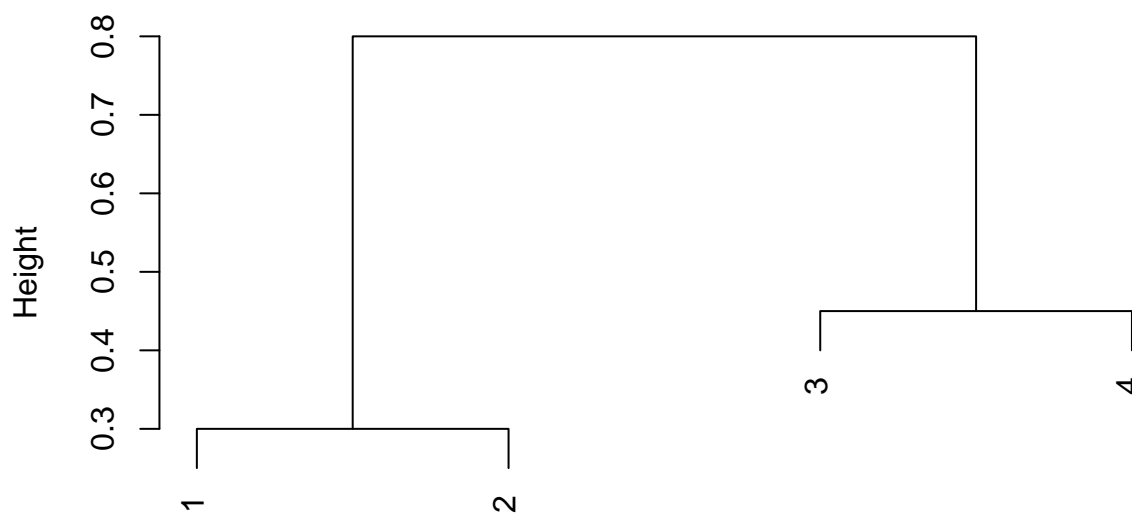
As before, cluster 3 and 4 will be fused at 0.45, leaving the highest dissimilarity value of 0.8 at which both clusters (1 & 2) and (3 & 4) will be fused.

$$\begin{bmatrix} & 0.8 \\ 0.8 & \end{bmatrix}$$

Dendrogram sketch:

```
m = as.dist(matrix(c(0, 0.3, 0.4, 0.7,
                    0.3, 0, 0.5, 0.8,
                    0.4, 0.5, 0.0, 0.45,
                    0.7, 0.8, 0.45, 0.0), nrow = 4))
plot(hclust(m, method = "complete"))
```

## Cluster Dendrogram



m  
hclust (\*, "complete")

(b)

Repeating (a) with single linkage (minimal intercluster dissimilarity)

Like in (a), cluster 1 and 2 are the most similar and are fused at height 0.3. After this first fusion we are left with the matrix below:

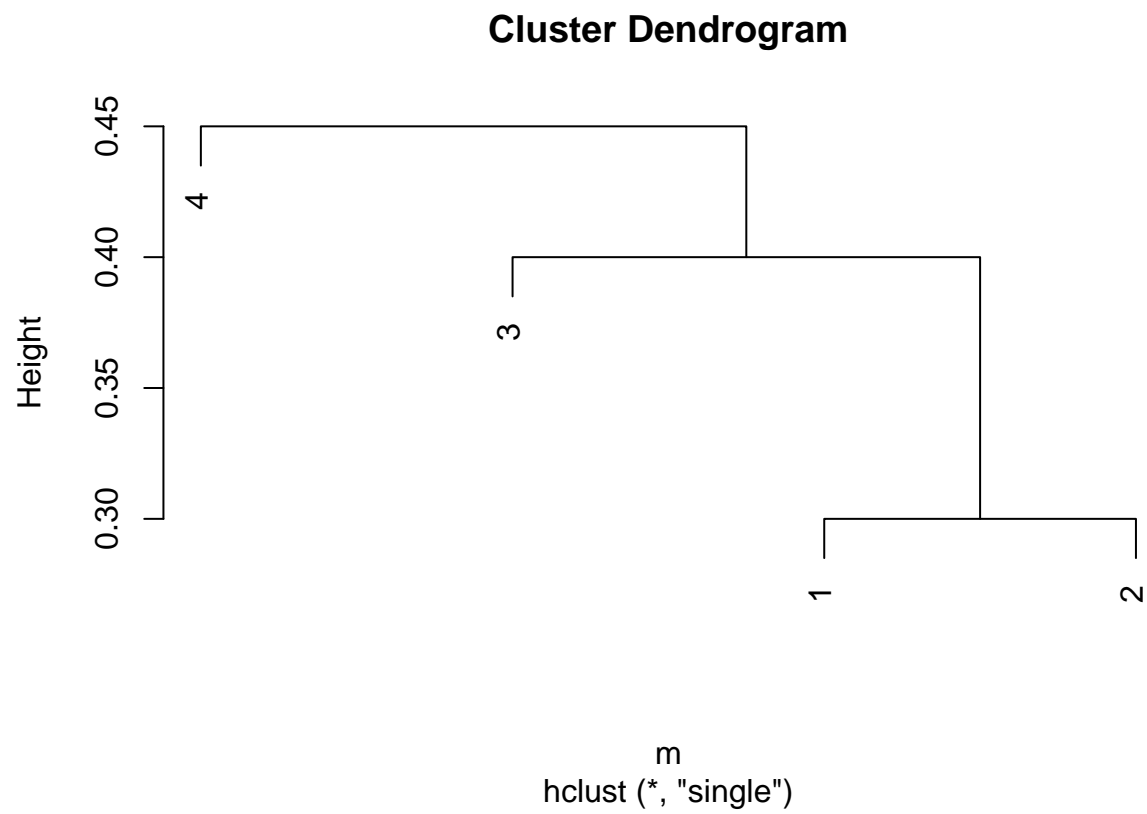
$$\begin{bmatrix} & 0.4 & 0.7 \\ 0.4 & & 0.45 \\ 0.7 & 0.45 & \end{bmatrix}$$

The lowest dissimilarity value is 0.4, between cluster (1 & 2) and 3, so we fuse them together. Leaving us the following matrix:

$$\begin{bmatrix} & 0.45 \\ 0.45 & \end{bmatrix}$$

Finally we are left with 0.45 at which the cluster ((1, 2) & 3) will be fused with 4.

```
m = as.dist(matrix(c(0, 0.3, 0.4, 0.7,
                    0.3, 0, 0.5, 0.8,
                    0.4, 0.5, 0.0, 0.45,
                    0.7, 0.8, 0.45, 0.0), nrow = 4))
plot(hclust(m, method = "single"))
```



(c)

- (1,2) & (3,4)

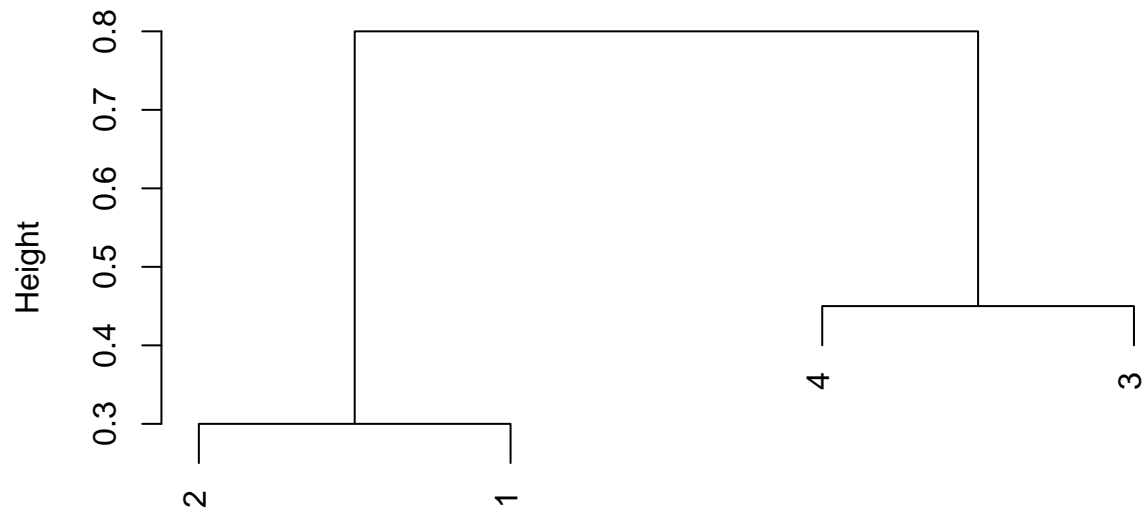
(d)

- ((1,2),3) & 4

(e)

```
plot(hclust(m, method = "complete"), labels = c(2,1,4,3))
```

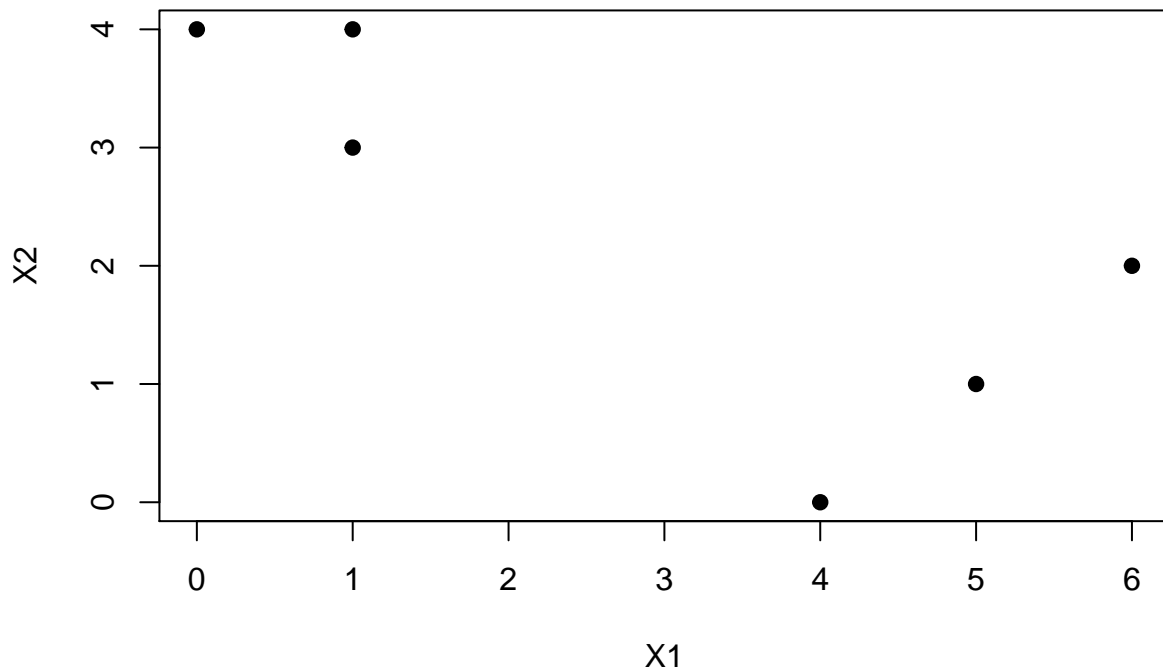
## Cluster Dendrogram



m  
hclust (\*, "complete")

3. (a)

```
df = data.frame(x1 = c(1, 1, 0, 5, 6, 4), x2 = c(4, 3, 4, 1, 2, 0))  
colnames(df)=c('X1','X2')  
plot(df, pch = 19, col = "black")
```



(b)

```
set.seed(3)

# Assigning observations randomly to two clusters
cluster = sample(2, nrow(df), replace=T)

# Table of assigned clusters
cbind(df, cluster)
```

```
##   X1 X2 cluster
## 1  1  4       1
## 2  1  3       2
## 3  0  4       2
## 4  5  1       1
## 5  6  2       2
## 6  4  0       2
```

(c)

The cluster centroids are simply the mean of the observations assigned to each cluster.

So for cluster 1, we can calculate it as follows:

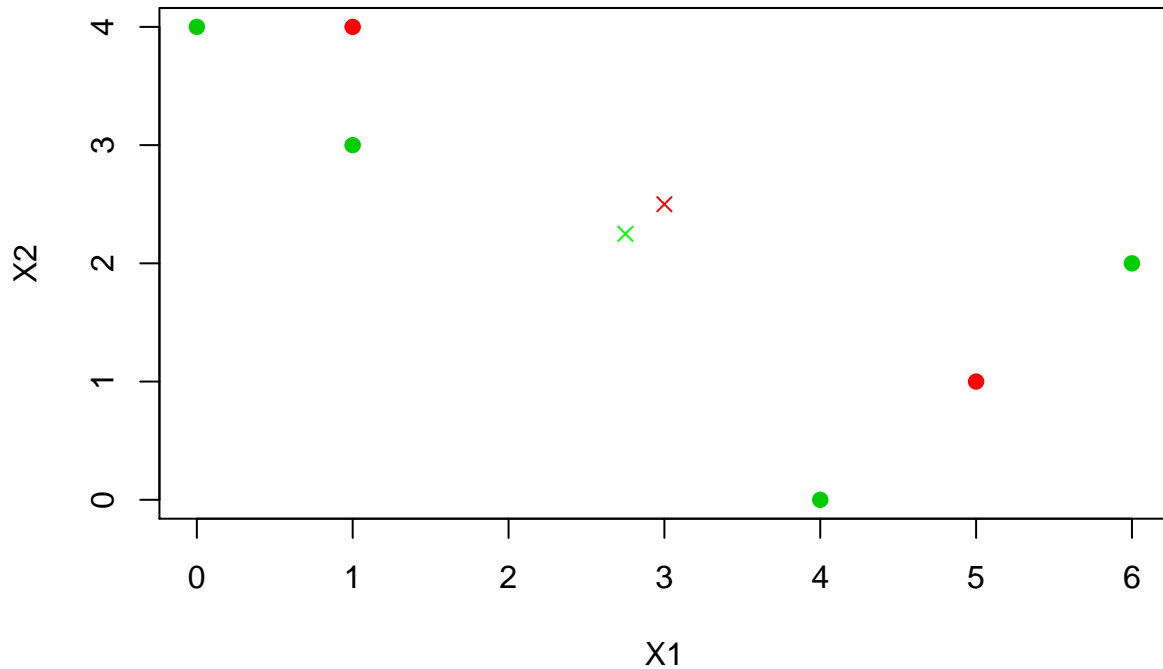
$$\bar{x}_{11} = \frac{1}{2}(1 + 5) = 3 \quad \text{and} \quad \bar{x}_{12} = \frac{1}{2}(4 + 1) = 2.5$$

And for cluster 2:

$$\bar{x}_{21} = \frac{1}{4}(1 + 0 + 6 + 4) = 2.75 \quad \text{and} \quad \bar{x}_{22} = \frac{1}{4}(3 + 4 + 2 + 0) = 2.25$$

```
# Computing centroids and plotting them alongside the clusters
c1 = c(mean(df[cluster==1, 1]), mean(df[cluster==1, 2])) #Centroid 1
c2 = c(mean(df[cluster==2, 1]), mean(df[cluster==2, 2])) #Centroid 2

plot(df, pch = 19, col = (cluster+1))
points(c1[1], c1[2], col="red", pch=4)
points(c2[1], c2[2], col="green", pch=4)
```



(d)

```
# Function to calculate the euclidean distance between two points
euclidean = function(a, b) {
  (sqrt((a[1] - b[1])^2 + (a[2] - b[2])^2))
}

# Function that loops over all observations and assigns them to the closest centroid, by calling the euclidean
new_labels = function(df, c1, c2) {
  labels = rep(NA, nrow(df))
  for (i in 1:nrow(df)) {
    if (euclidean(df[i,], c1) < euclidean(df[i,], c2)) {
      labels[i] = 1
    }
  }
}
```

```

    } else {
      labels[i] = 2
    }
  }
  return(labels)
}

```

```

#Assigning observations to their closest centroids
new_cluster = new_labels(df, c1, c2)

```

```

#Table of original and new clusters
cbind(df, cluster, new_cluster)

```

```

##   X1 X2 cluster new_cluster
## 1  1  4      1           2
## 2  1  3      2           2
## 3  0  4      2           2
## 4  5  1      1           1
## 5  6  2      2           1
## 6  4  0      2           2

```

(e) (f)

```

#Running the new labels function until the assigned clusters stop changing.
final_cluster = rep(-1, 6)
while (!all(final_cluster == cluster)) {
  final_cluster = cluster
  c1 = c(mean(df[cluster==1, 1]), mean(df[cluster==1, 2]))
  c2 = c(mean(df[cluster==2, 1]), mean(df[cluster==2, 2]))
  cluster = new_labels(df, c1, c2)
}

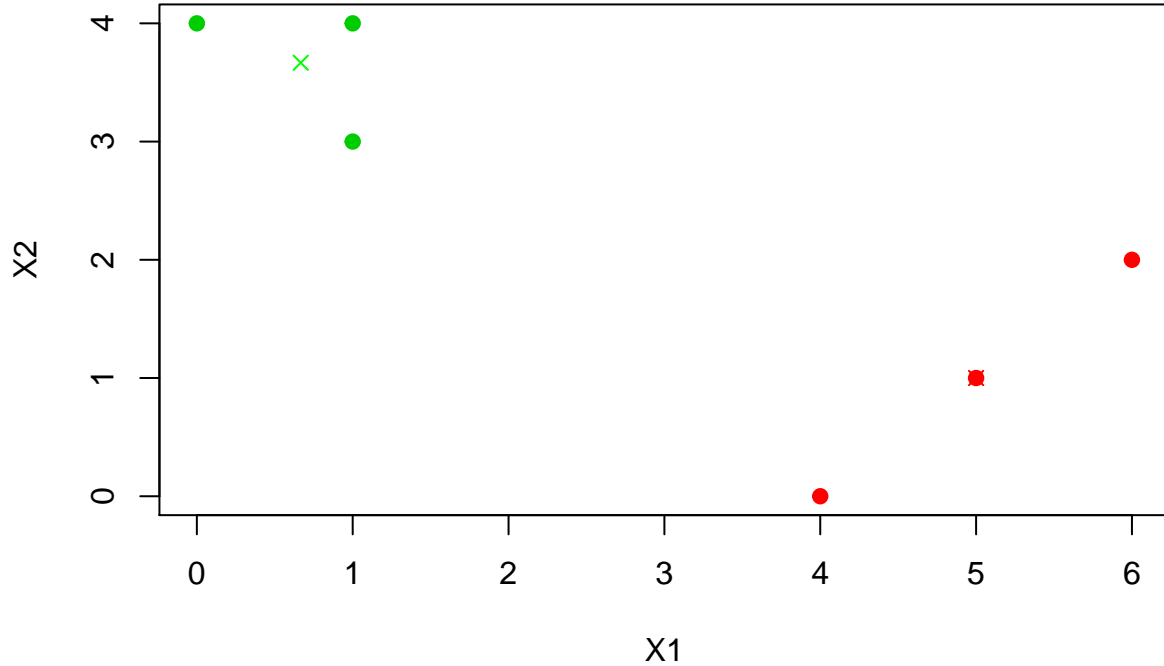
```

```

#Plot of the final cluster assignments
plot(df, pch = 19, col = (cluster+1))
points(c1[1], c1[2], col="red", pch=4)
points(c2[1], c2[2], col="green", pch=4)

```





#### 4. (a)

Not enough information to tell.

Since complete linkage uses the highest inter-cluster dissimilarity and single linkage uses the lowest, in most of the cases the fusion with complete linkage will occur higher on the tree.

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.

In some special cases (when the inter-cluster distances are all the same) the fusion will occur at the same height for both linkage methods.

For example, if all inter-cluster distances 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)

They would fuse at the same height because linkage type does not affect leaf-to-leaf fusion. 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.

#### 5.

- For the figure on the left, we have unscaled variables and the number of socks will have the biggest impact on the resulting clusters. The observations would likely be split between a cluster with customers purchasing ‘most socks and computer’, and ‘least socks and computer’.

- For the central figure, the variables have been scaled, and so the number of computers will have a much bigger impact than before. We would likely have clusters split between the number of computers purchased.
- For the figure on the right, the price of computers has an overwhelmingly large impact. The clusters will be split between ‘purchased computer’, and ‘no computer purchased’.

6.

(a)

- PCA finds, in the data space, the dimension (or direction) with the largest variance out of the overall variance. This is our first principal component, which explains 10% of the total variance in this data set. Then it would find the dimension with the second largest variance (second principal component), orthogonal to the first one, out of the remaining 90% variance and so on.
- Another way of looking at it is that 90% of the variance in the data is not contained in the first principal component, and other components are required to explain more of the variance. In a situation where there exist a whole bunch of independent variables, PCA helps you figure out which linear combinations of these variables matter the most. Ideally, we would want a few principle components that explain most of the variability in the data set.

(b)

- Given the flaw shown in pre-analysis of a time-wise linear trend amongst the tissue samples’ first principal component, I would advise the researcher to include the machine used (A vs B) as a feature of the data set. This should enhance the PVE of the first principal component before applying the two-sample t-test.
- The use of machine A during earlier and then B at later times respectively, could have been responsible for a time-wise linear trend amongst the tissue samples first principle component. As such I would advise the researcher to include the machine used (A vs B) as a new feature for the data set. This should enhance the PVE (proportion of variance explained) for the first principle component, before applying the two-sample t-test.

(c)

A toy data set to show that adding a feature for the machine (A vs B) used, improves the PVE for the first principle component. This is a very basic simulation, so don’t take the results for granted.

```
set.seed(101)

Control = matrix(rnorm(50 * 1000), ncol = 50)
Treatment = matrix(rnorm(50 * 1000), ncol = 50)

X = cbind(Control, Treatment)
X[1, ] = seq(-18, 18 - .36, .36) # linear trend in one dimension
pr.out = prcomp(scale(X))
summary(pr.out)$importance[, 1]
```

##	Standard deviation	Proportion of Variance	Cumulative Proportion
##	3.174779	0.100790	0.100790

- From the summary, we can observe that 10.08% of the variance is explained by the first principle component.

Adding in machine used (A vs B) using 10 vs 0 encoding.

```
set.seed(101)

X = rbind(X, c(rep(10, 50), rep(0, 50)))
pr.out = prcomp(scale(X))
summary(pr.out)$importance[, 1]

##      Standard deviation Proportion of Variance Cumulative Proportion
##      3.425154                0.117320                0.117320
```

- We can see the PVE explained increased to 11.7% after adding in a feature for the machine used.

## Applied

### 7.

As per the HINT, using 'cor' and 'dist' functions to calculate the correlation and distance of the USArrest observations.

```
scaledArrests = scale(USArrests)

corr_dis = as.dist(1-cor(t(scaledArrests))) #correlation based distance
euclidean_dis = dist(scaledArrests)^2 #squared euclidean distance

summary(corr_dis/euclidean_dis)
```

```
##      Min.  1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.000086 0.069135 0.133943 0.234193 0.262589 4.887686
```

If the quantities are approximately proportional then  $euclidean\_dis \approx K \cdot corr\_dis$  for a constant  $K$ .

```
summary(corr_dis-0.1339*euclidean_dis)
```

```
##      Min.  1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.569956 -0.459715  0.000393 -0.059043  0.557616  1.808901
```

If  $K = 0.1339$  then they are approximately equal, with a mean difference of around 0.05 only.

### 8. (a)

```
pr.out = prcomp(scaledArrests)
pr.var = pr.out$sdev^2

pve = pr.var/sum(pr.var)
pve
```

```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

- PVE calculated using the prcomp() function. As expected, the first principal component explains 62% of the variance in this dataset.

### (b)

```

loadings = pr.out$rotation #principle component loadings

num = apply((as.matrix(scaledArrests) %*% loadings)^2, 2, sum) #Summation of squares of scorings (numer
denom = sum(apply(as.matrix(scaledArrests)^2, 2, sum)) #Variability in the data (denominator in the for

pve2 = num/denom
pve2

```

```

##          PC1          PC2          PC3          PC4
## 0.62006039 0.24744129 0.08914080 0.04335752

```

- As expected, the resulting PVE's are exactly the same as when using prcomp().

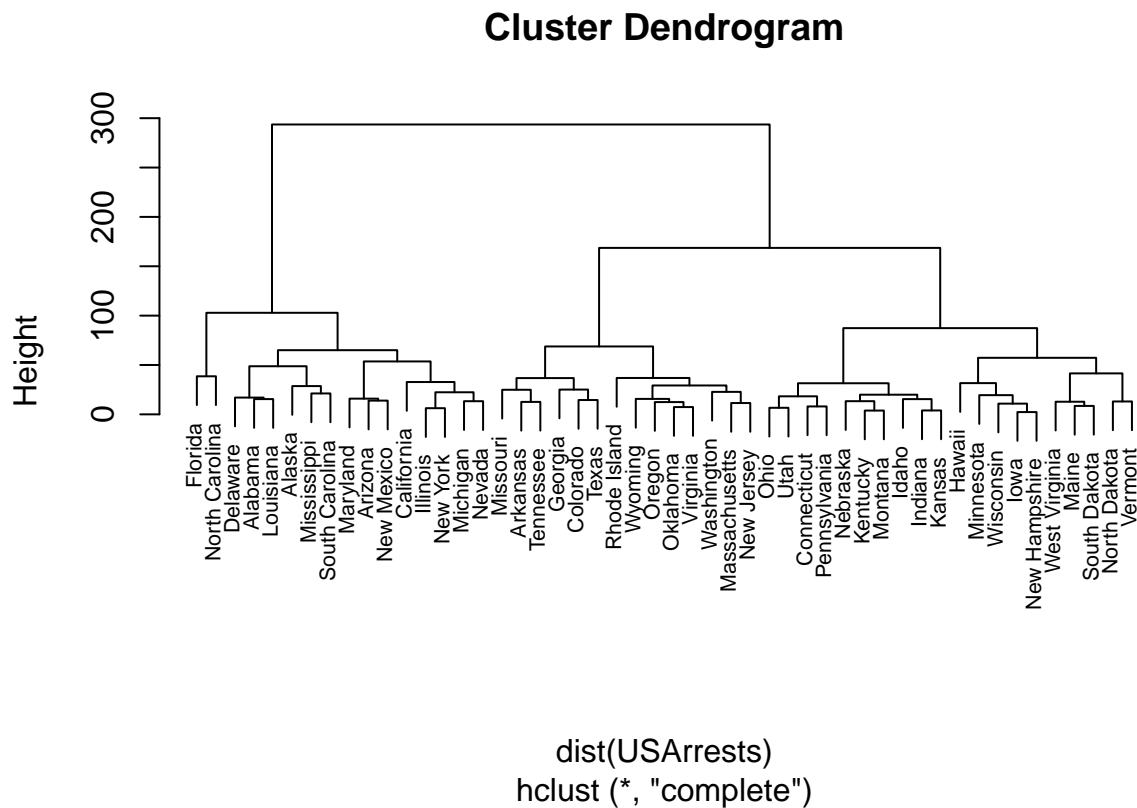
9. (a)

```

set.seed(1)

hc.complete = hclust(dist(USArrests), method = 'complete')
plot(hc.complete, cex = 0.7)

```



(b)

```

hc.cut = cutree(hc.complete, k=3) #Cutting tree with K=3, so we get 3 clusters.

```

```
tibble(
  states = rownames(USArrests),
  cluster = hc.cut
)
```

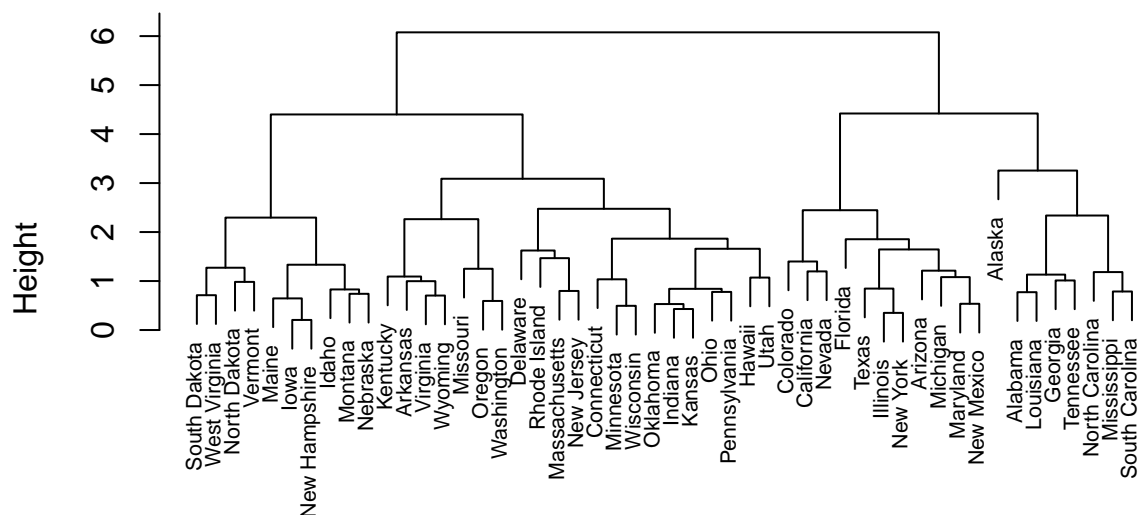
```
## # A tibble: 50 x 2
##   states      cluster
##   <chr>      <int>
## 1 Alabama          1
## 2 Alaska            1
## 3 Arizona            1
## 4 Arkansas           2
## 5 California         1
## 6 Colorado           2
## 7 Connecticut        3
## 8 Delaware           1
## 9 Florida            1
##10 Georgia            2
## # ... with 40 more rows
```

(c)

*# As I have already scaled the US Arrests dataset, I'm simply going to use the existing variable 'scale'*

```
hc.complete.scaled = hclust(dist(scaledArrests), method = 'complete')
plot(hc.complete.scaled, cex = 0.7)
```

## Cluster Dendrogram



```
dist(scaledArrests)
hclust (*, "complete")
```

(d)

- The scaled dendrogram has greatly reduced height, and the clusters obtained are somewhat different. However, the bushiness of the tree doesn't appear to be affected.
- As a general rule, variables with different measurement units should be scaled before computing the inter-observation dissimilarities. Which applies to this dataset.

10.

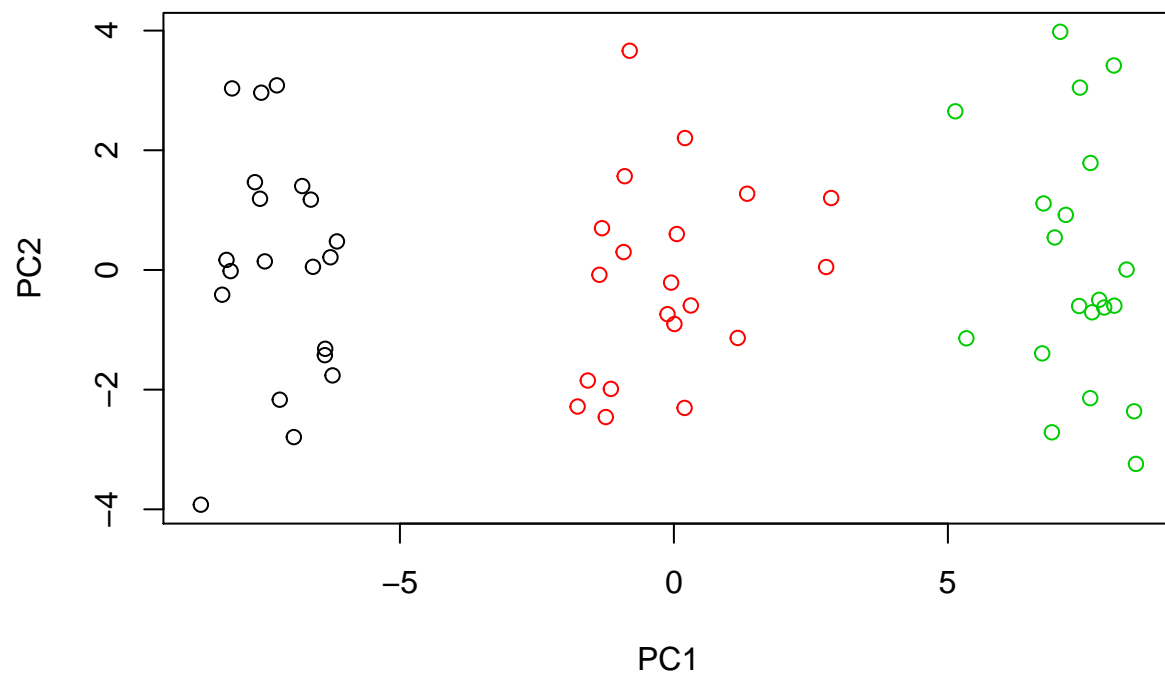
(a)

```
set.seed(1)
# Simulated data with mean shift to create three distinct classes.
# Increasing the difference in means for each class would create greater separation between the classes
simulated.data = matrix(c(rnorm(20 * 50, mean = 1),
                           rnorm(20 * 50, mean = 2),
                           rnorm(20 * 50, mean = 3)), ncol = 50, byrow = TRUE)

class = unlist(lapply(1:3,function(x){rep(x,20)}))
```

(b)

```
pr.out2 = prcomp(simulated.data)
plot(pr.out2$x[,1:2],col=class)
```



(c)

```
kmeans.out = kmeans(simulated.data, 3, nstart = 60)
table(class, kmeans.out$cluster)
```

```
##
## class  1  2  3
##      1  0  0 20
##      2 20  0  0
##      3  0 20  0
```

- All observations are correctly classified. This is expected as the observations in the three classes are well separated.

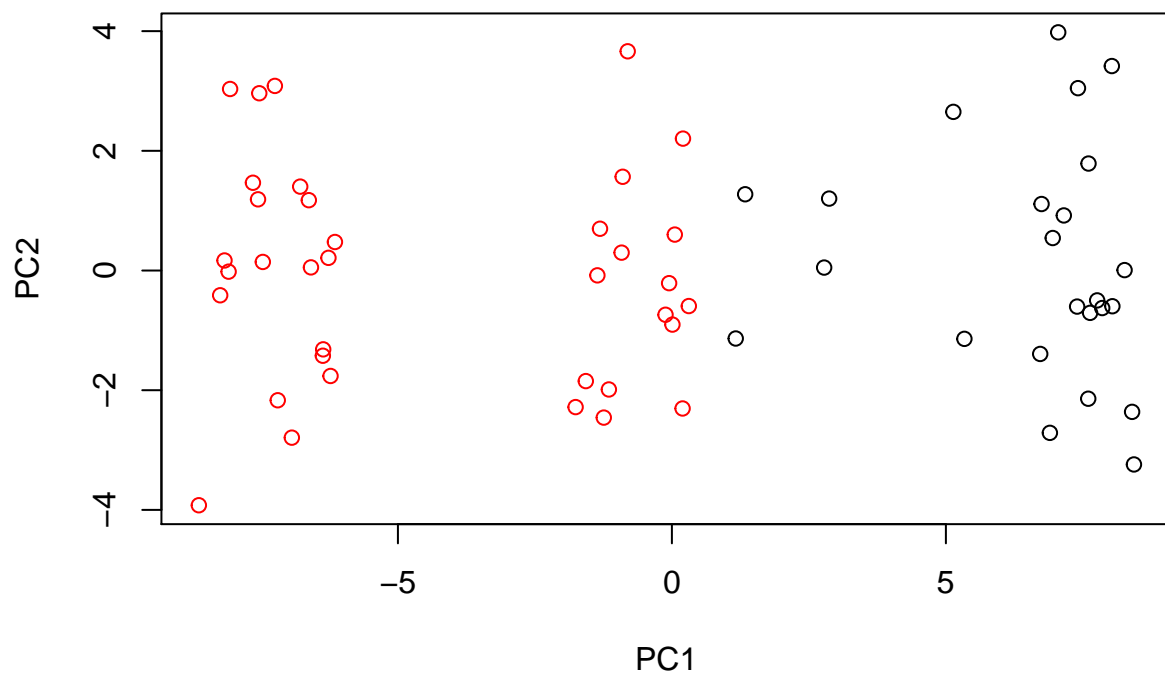
(d)

```
kmeans.out = kmeans(simulated.data, 2, nstart = 60)
table(kmeans.out$cluster)
```

```
##
##  1  2
## 24 36
```

- The observations are now assigned to two classes. One class is assigned much more of the observations than the other. A plot of the newly assigned cluster is shown below.

```
plot(pr.out2$x[,1:2], col=kmeans.out$cluster)
```



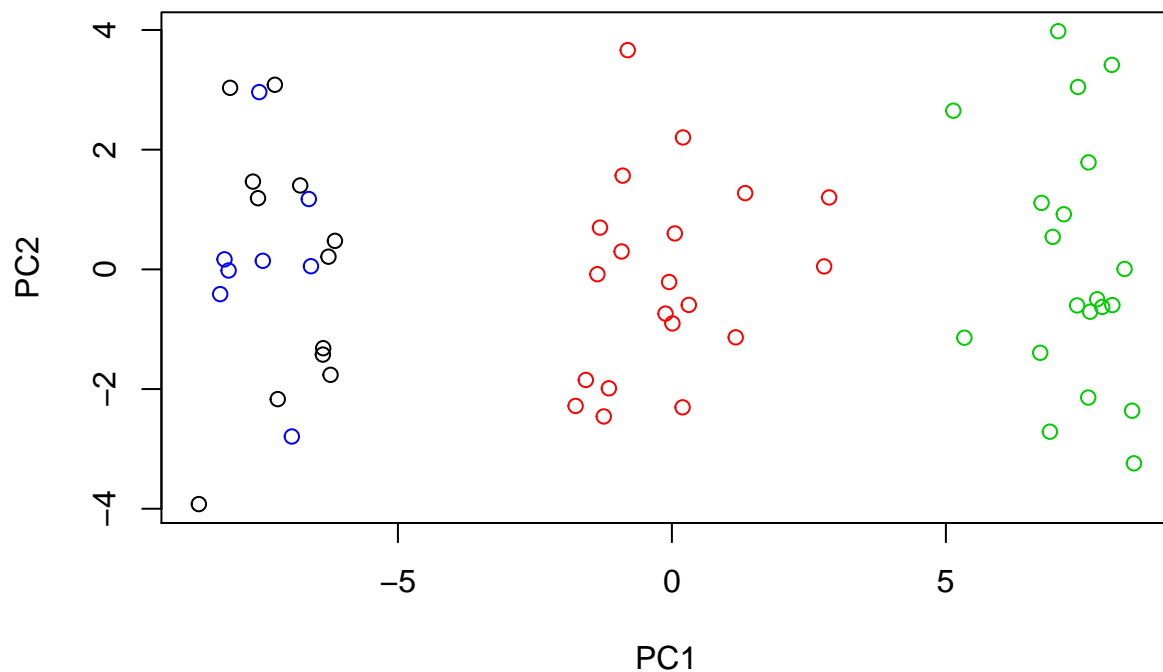
(e)

```
kmeans.out = kmeans(simulated.data, 4, nstart = 60)
table(kmeans.out$cluster)
```

```
##
##  1  2  3  4
## 12 20 20  8
```

- We already know that there are three distinct classes, and so when using  $K = 4$ , we could see observations being assigned into more classes or clusters than is necessary (or is supported by the dataset). That is what has happened in this case. In other words, when using 4 clusters it becomes more difficult to determine the difference between the new found clusters and the actual class values.
- The plot below shows the newly assigned clusters.

```
plot(pr.out2$x[,1:2], col=kmeans.out$cluster)
```



(f)

```
kmeans.out = kmeans(pr.out2$x[,1:2], 3, nstart = 60)
table(kmeans.out$cluster)
```

```
##
##  1  2  3
## 20 20 20
```



```
table(class)
```

```
## class
##  1  2  3
## 20 20 20
```

- As expected, and like in part (c), the clusters obtained have a perfect mapping to the original classes.

(g)

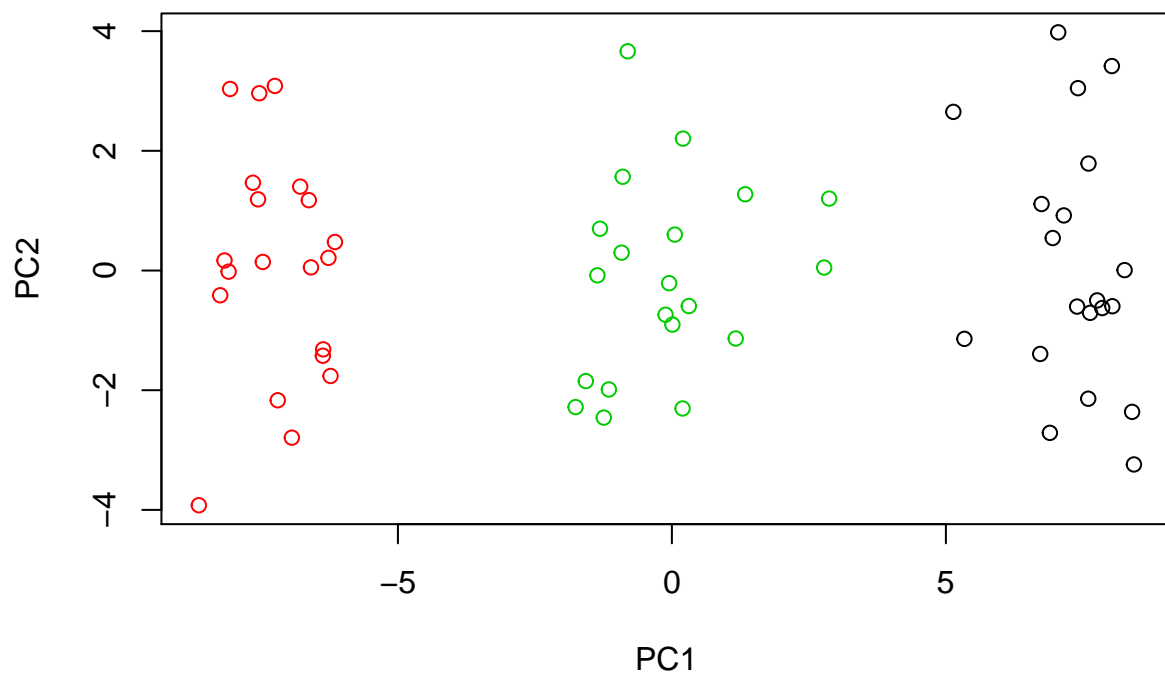
```
set.seed(1)
kmeans.out = kmeans(scale(simulated.data), 3, nstart = 60)
table(kmeans.out$cluster)
```

```
##
##  1  2  3
## 20 20 20
```

```
table(class)
```

```
## class
##  1  2  3
## 20 20 20
```

```
plot(pr.out2$x[,1:2], col=kmeans.out$cluster)
```



- The results are exactly like in part (b), where the assigned clusters are perfectly mapped to the original classes. Likely because the simulated data set I created was very well separated. Datasets with overlapping observations would likely result in a different outcome.

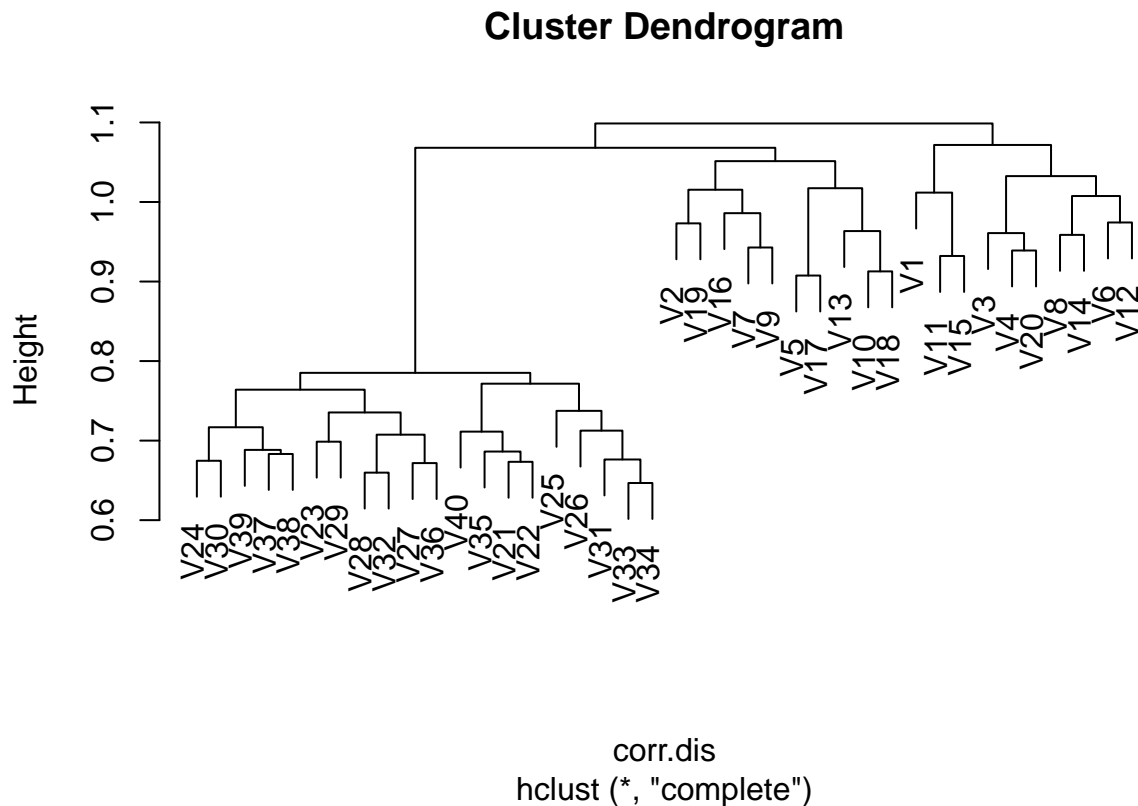
11. (a)

```
set.seed(1)
gene.data = read.csv("Ch10Ex11.csv", header = F)
```

(b)

```
# Using complete linkage and correlation based distance
```

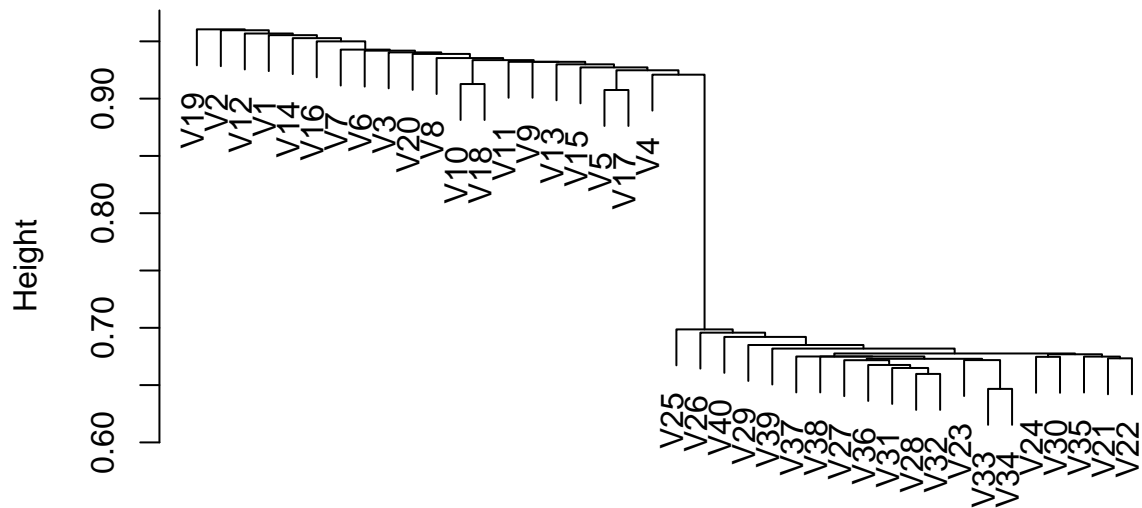
```
corr.dis = as.dist(1 - cor(gene.data))
hc.complete = hclust(corr.dis, method = "complete")
plot(hc.complete)
```



```
# Using single linkage and correlation based distance
```

```
hc.single = hclust(corr.dis, method = "single")
plot(hc.single)
```

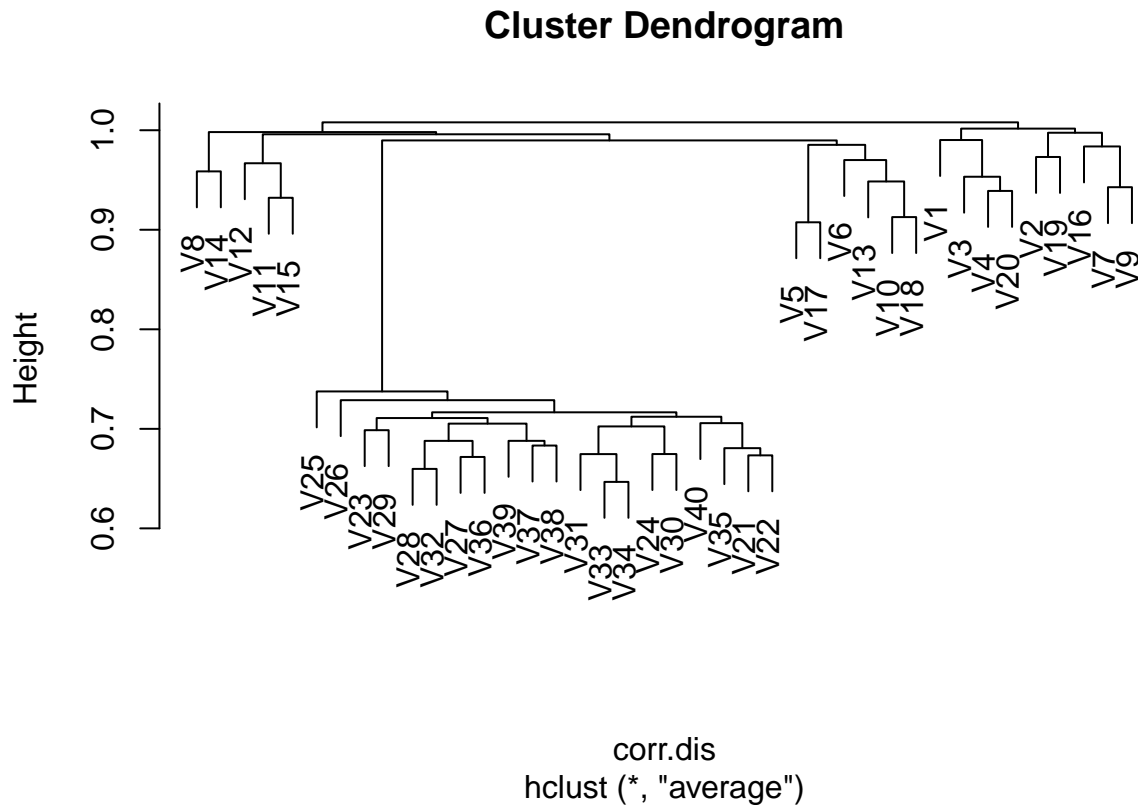
## Cluster Dendrogram



corr.dis  
hclust (\*, "single")

```
# Using average linkage and correlation based distance
```

```
hc.average = hclust(corr.dis, method = "average")  
plot(hc.average)
```



- The resulting trees are quite different, and so are impacted by the type of linkage. Single linkage results in a very unbalanced tree and average has three clusters, whereas complete gives the most balanced tree where the samples are split into two roughly 50/5 groups. As we already know that samples are equally split between healthy and diseased groups, complete linkage will likely give the best results.

(c)

- We could use PCA to see which genes differ the most between the healthy and diseased group. We will examine the absolute values of the total loadings for each gene as it characterizes the weight of each gene.

```
pr.gene = prcomp(t(gene.data))
summary(pr.gene)
```

```
## Importance of components:
##               PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation  11.9409  6.06818  5.93476  5.83115  5.75209  5.70031  5.63448
## Proportion of Variance  0.1267  0.03271  0.03129  0.03021  0.02939  0.02887  0.02821
## Cumulative Proportion  0.1267  0.15939  0.19068  0.22089  0.25029  0.27915  0.30736
##               PC8      PC9      PC10     PC11     PC12     PC13     PC14
## Standard deviation   5.57726  5.54943  5.50625  5.48852  5.46025  5.40230  5.33441
## Proportion of Variance 0.02764  0.02736  0.02694  0.02676  0.02649  0.02593  0.02528
## Cumulative Proportion 0.33499  0.36236  0.38929  0.41605  0.44254  0.46847  0.49375
##               PC15     PC16     PC17     PC18     PC19     PC20     PC21
```

```
## Standard deviation      5.27756 5.21594 5.20000 5.15140 5.11600 5.05591 5.03836
## Proportion of Variance 0.02475 0.02417 0.02402 0.02358 0.02325 0.02271 0.02255
## Cumulative Proportion 0.51850 0.54267 0.56669 0.59027 0.61352 0.63623 0.65878
##          PC22      PC23      PC24      PC25      PC26      PC27      PC28
## Standard deviation      5.01868 4.95965 4.91393 4.86397 4.81796 4.80811 4.73485
## Proportion of Variance 0.02238 0.02185 0.02145 0.02102 0.02062 0.02054 0.01992
## Cumulative Proportion 0.68116 0.70301 0.72447 0.74548 0.76611 0.78665 0.80656
##          PC29      PC30      PC31      PC32      PC33      PC34      PC35
## Standard deviation      4.70098 4.65564 4.61621 4.56733 4.53032 4.49528 4.36502
## Proportion of Variance 0.01963 0.01926 0.01893 0.01853 0.01823 0.01795 0.01693
## Cumulative Proportion 0.82620 0.84545 0.86439 0.88292 0.90115 0.91910 0.93603
##          PC36      PC37      PC38      PC39      PC40
## Standard deviation      4.35858 4.26700 4.20277 4.13922 5.251e-15
## Proportion of Variance 0.01688 0.01618 0.01569 0.01522 0.000e+00
## Cumulative Proportion 0.95291 0.96909 0.98478 1.00000 1.000e+00
```

```
total.load = apply(pr.gene$rotation, 1, sum)
index = order(abs(total.load), decreasing = TRUE)
index[1:10]
```

```
## [1] 865 68 911 428 624 11 524 803 980 822
```

- These are the 10 genes that differ most across the two groups.