Assignment 4

Please show all the steps and keep the final answer 3-4 decimal places. You should use R for the applied questions.

Unit 07 Unsupervised learning

1. PCA

(a). What is Principal Component Analysis (PCA)? PCA is a technique which is used to reduce the number of variables in a dataset while trying to retain as much information (variance) as possible. Variables called principle components are created which are ordered by how much variance they retain from the data.

(b). Why is PCA considered an unsupervised technique? PCA does not use any target variables during the process. It only involves a set of features variables, no response.

(c). Describe the steps of PCA. 1. Standardize data so that variables with larger values don't control the analysis

2. Compute the covariance matrix $S = \frac{1}{n-1}X^TX$
3. Calculate eigenvectors and eigenvalues which are derived from the covariance matrix.
4. We pick the largest k eigenvalues $\lambda_1, \ldots, \lambda_n$ and their corresponding eigenvectors $V_1, \ldots, V_n$ that capture most of the variance from the data (such as 95% or any other appropriate number)
5. Project the data onto the selected principle components to get a lower-dimensional representation

(d). Describe the role of the sample covariance matrix in PCA. The sample covariance matrix shows how different features in the dataset vary together. In PCA, it is used to identify correlation patterns between features. This information is used to understand the overall structure of the data, and identifying the principle components.

(e). Can you explain the concept of eigenvalues and eigenvectors in PCA? Eigenvalues tell us how much variance is captured by each PC, a larger eigenvalue means the direction explains more of the data's spread.

Eigenvectors represent the 'weight' that each variable has on a PC. It can be also seen as the direction in which the data varies the most.

(f). What is the variance explained by a principal component? How is PCA used for dimensionality reduction? The variance explained by a PC refers to how much of the total variability/"information" the PC captures from the original data. A higher variance of a PC means it was able to understand more of the data's patterns and overall structure.

PCA reduces dimensions by selecting the top k PC's, which are able to explain most of the original data's varaiance. Since not every PC is picked, dimentionality is reduced without losing too much information.

(h). How do you determine the number of principal components to use? You should choose enough components for when the variances are added up, should equal the threshold you have chosen appropriate for the task. One can also create a plot, with the PCs on the x axis and the variances on the Y axis, and examine where the variance drops significantly, and use any PC's before that.

(g). What is meant by "loadings" and "PC scores" in the context of PCA? loadings show the "weight" that each variable has on a PC. They come from eigenvectors, and as mentioned above can be seen as the direction in which the data varies the most.

PC scores are the coordinates of the transformed data on the PC space. They are found by projecting the original data onto the PCs.

Textbook questions 1, 2 (a)-(d), 3, 9 on pages 546-549

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

Firstly, we can see that: $(x_{ij}-x_{i'j})^2 = ((x_{ij}-\bar{x}_{kj}) - (x_{i'j}-\bar{x}_{kj}))^2 = (x_{ij}-\bar{x}_{kj})^2 - 2(x_{ij}-\bar{x}_{kj})(x_{i'j}-\bar{x}_{kj}) + (x_{i'j}-\bar{x}_{kj})^2$

Now, let's rewrite the LHS of the equation given our expansion above:

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

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

We can drop the middle term, because the sum equals to 0 because the sum of deviations from the mean is 0 for all of the data points in the cluster.

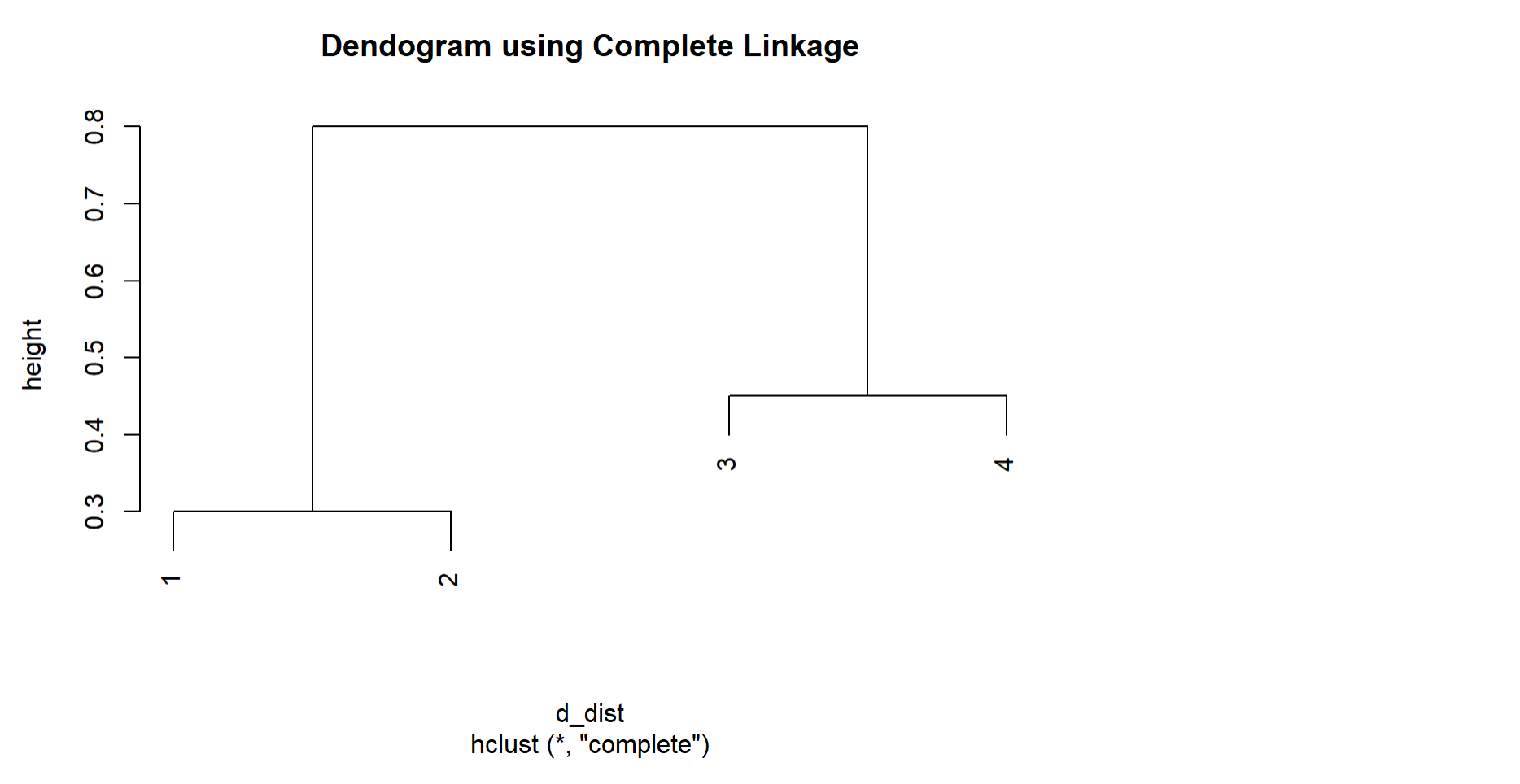
$$= 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2$$

Therefore LHS = RHS

b. K-means clustering minimizes the objective (12.17) at each iteration, which sums pairwise squared distances within each cluster. Using our identity in (12.18), this is equivalent to minimizing the total squared distance from points to the cluster means, as in (12.12). The algorithm changes between assigning points to the closest centroid and updating the centroid to the mean of the assigned points at each step, reducing the objective. Therefore, (12.12) decreases (12.17) at each iteration.

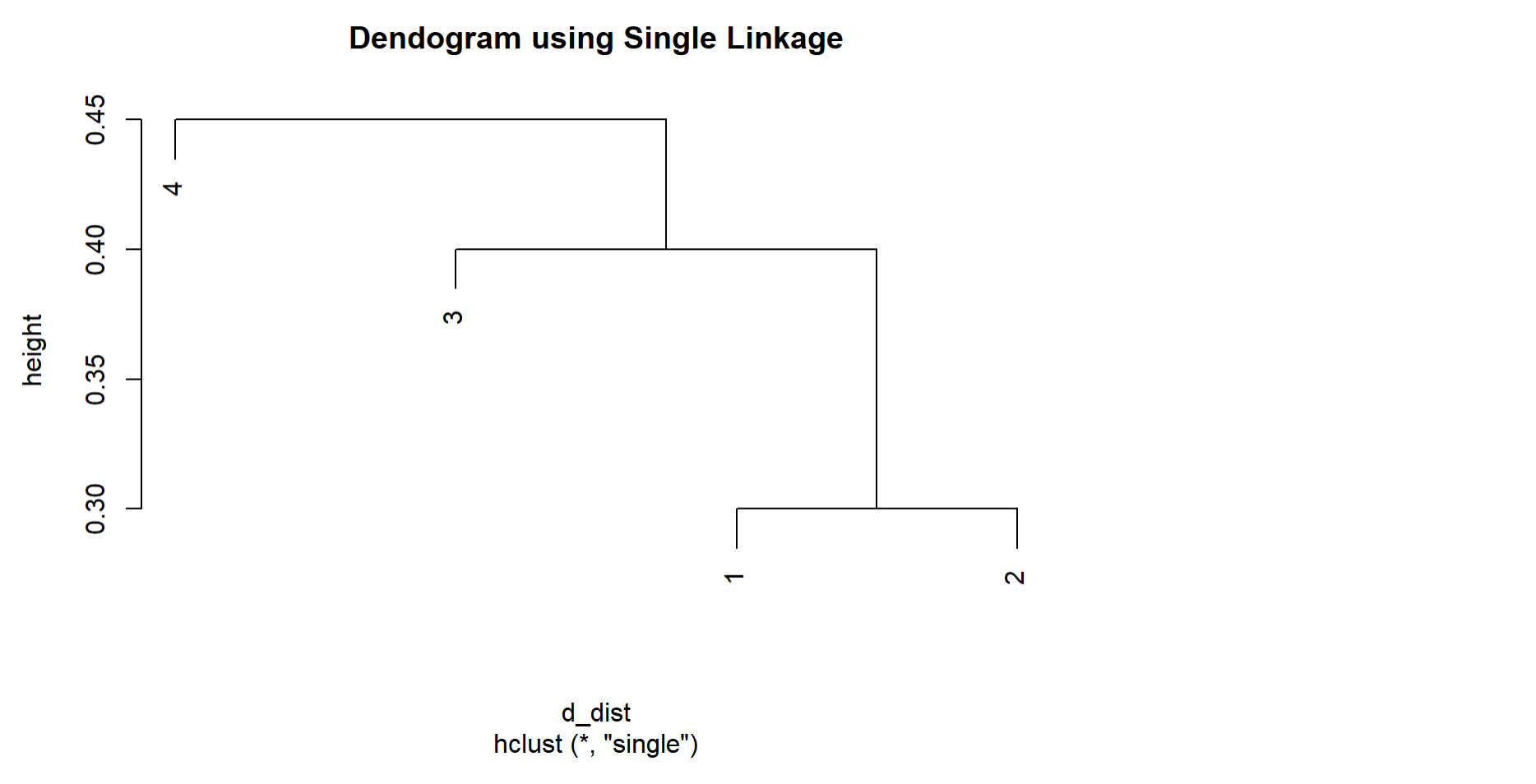
2.(a)

```
d <- 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),nrow=4,byrow=TRUE)

d_dist <- as.dist(d)
hc <- hclust(d_dist,method="complete")

plot(hc, main="Dendogram using Complete Linkage", ylab="height")
```



b.

```
hc_single <- hclust(d_dist,method="single")
plot(hc_single, main="Dendogram using Single Linkage", ylab="height")
```



c.

```
cluster_c = cutree(hc, k=2)
cluster_c
```

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

We see that Observations 1 and 2 are of cluster 1, and observations 3 and 4 are of cluster 2

d.

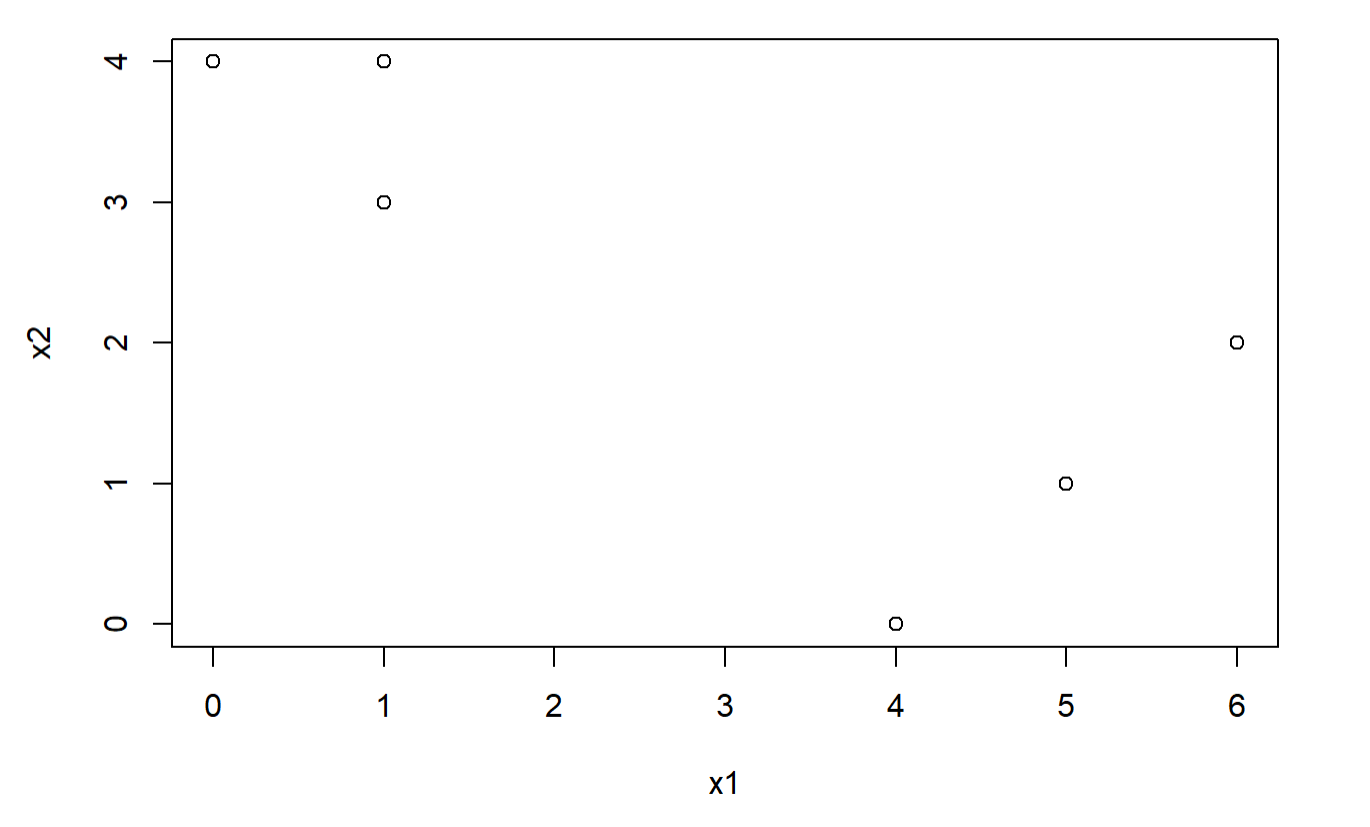
```
cluster_d = cutree(hc_single,k=2)
cluster_d
```

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

We see that observations 1,2, and 3 are of cluster 1, and observation 4 is of cluster 2

3.(a)

```
x1 <- c(1,1,0,5,6,4)
x2 <- c(4,3,4,1,2,0)

plot(x1,x2)
```



b.

```
df <- data.frame(x1,x2)

set.seed(42)
df$cluster <- sample(c(1,2), size=nrow(df), replace=TRUE)

df$cluster
```

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

c.

```
centroids <- aggregate(. ~ cluster,data=df,FUN=mean)
centroids
```

```
##   cluster   x1 x2
## 1       1 1.75  3
## 2       2 5.00  1
```

d.

```
closest_centroid <- function(row,centroids){
  distances <- apply(centroids,1, function(centroid) sqrt(sum((row-centroid)^2)))
  # returns the closest centroid
  return(which.min(distances))
}

new_cluster1 <- apply(df[, 1:2],1,closest_centroid,centroids=centroids)
```

```
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
## Warning in row - centroid: longer object length is not a multiple of shorter
## object length
```

```
df$cluster_2 <- new_cluster1

df
```

```
##   x1 x2 cluster cluster_2
## 1  1  4       1         2
## 2  1  3       1         2
## 3  0  4       1         2
## 4  5  1       1         1
## 5  6  2       2         1
## 6  4  0       2         1
```

e.

```
# New Centroids
centroids_2 <- aggregate(. ~ cluster_2,data=df,FUN=mean)
centroids_2
```

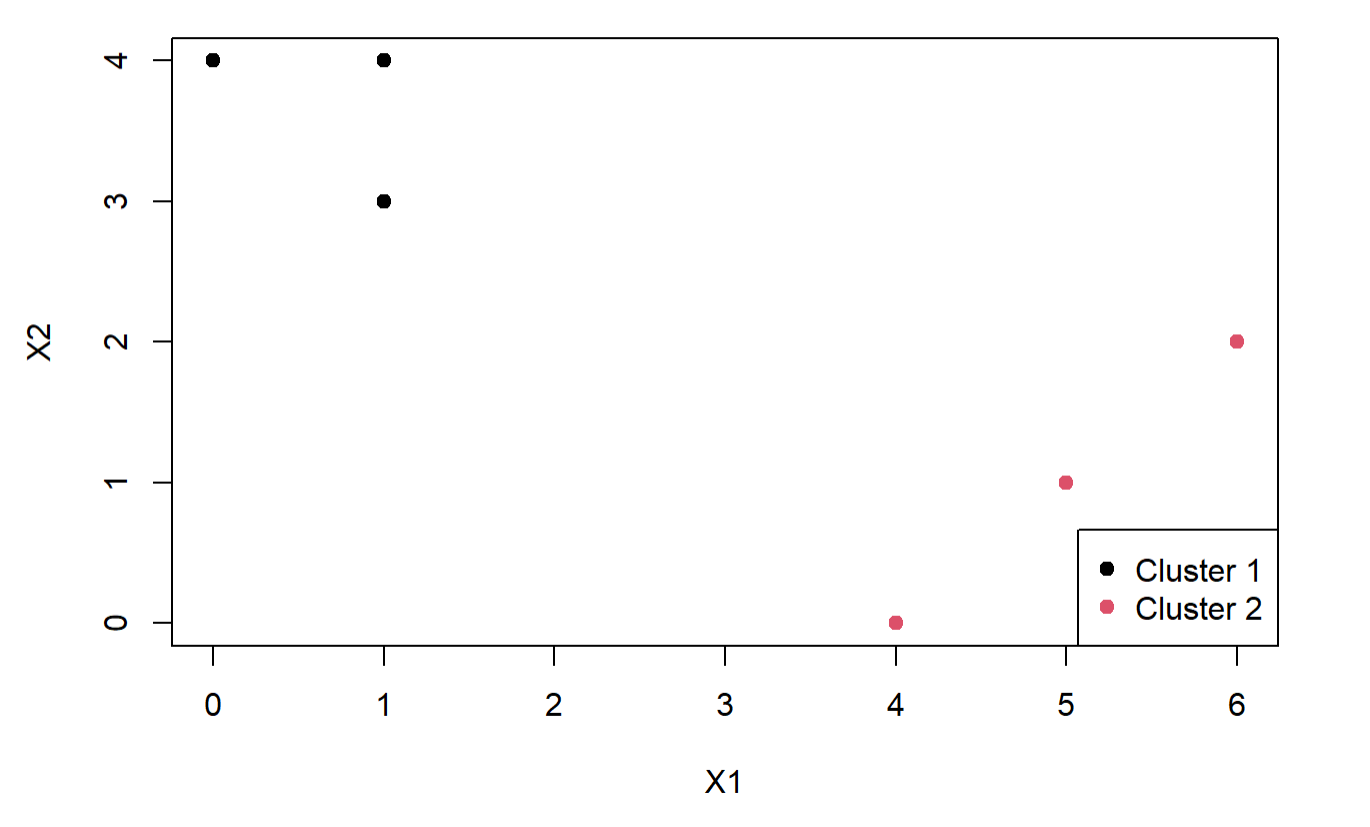
```
##   cluster_2        x1       x2  cluster
## 1         1 5.0000000 1.000000 1.666667
## 2         2 0.6666667 3.666667 1.000000
```

```
new_cluster2 <- apply(df[, 1:2],1,closest_centroid,centroids=centroids_2)
df$cluster_3 <- new_cluster2

df
```

```
##   x1 x2 cluster cluster_2 cluster_3
## 1  1  4       1         2         1
## 2  1  3       1         2         1
## 3  0  4       1         2         1
## 4  5  1       1         1         2
## 5  6  2       2         1         2
## 6  4  0       2         1         2
```

f.

```
plot(df$x1, df$x2,
     col = df$cluster_3,
     pch = 19,
     xlab = "X1", ylab = "X2",)
legend("bottomright", legend = c("Cluster 1", "Cluster 2"),
       col = c(1, 2), pch = 19)
```



9.(a)

```
data("USArrests")
set.seed(42)
hc_complete <- hclust(dist(USArrests), method="complete")

hc_complete
```

```
##
## Call:
## hclust(d = dist(USArrests), method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 50
```

b.

```
three_cut <- cutree(hc_complete,3)

split(rownames(USArrests),three_cut)
```

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

Here, we see the organized list of which cluster each state belongs to

c.

```
hc_scaled_complete <- hclust(dist(scale(USArrests)), method="complete")

hc_scaled_complete
```

```
##
## Call:
## hclust(d = dist(scale(USArrests)), method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 50
```

d.

```
three_cut_scale <- cutree(hc_scaled_complete,3)

split(rownames(USArrests),three_cut_scale)
```

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

```
sapply(USArrests, range) # min and max values for each feature
```

```
##      Murder Assault UrbanPop Rape
## [1,]    0.8      45       32  7.3
## [2,]   17.4     337       91 46.0
```

Heriarchal clustering using euclidean distance is sensitive to the scale of variables, so scaling for this dataset is appropriate. This is because we can see that the range of values for each feature are dramatically different. Assault has a large gap and without scaling, it may be a lot more dominant when calculating the euclidean distance. After scaling, all the values contribute equally to the dissimilarity computation.