# ch12exercises

Lauren Temple

3/26/2022

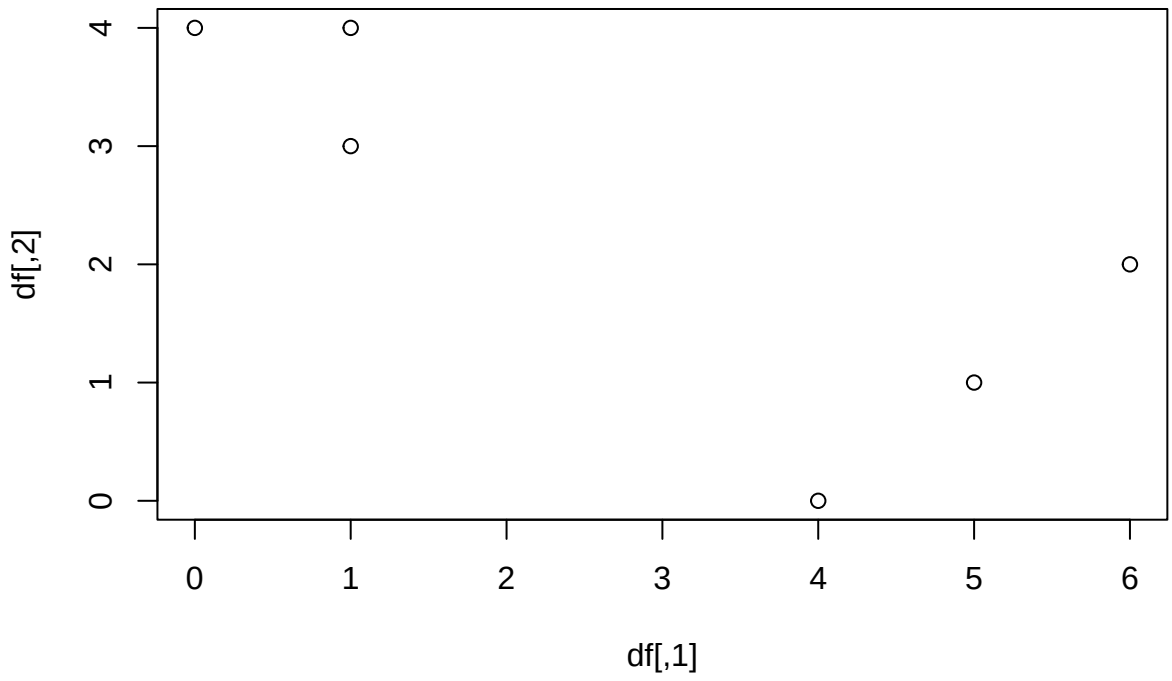
```
pacman::p_load(e1071, ROCR, ISLR2, caTools)
```

12.3 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.

(a) Plot the observations.

```
df <- matrix(c(1,1,0,5,6,4,4,3,4,1,2,0), nrow=6)
```

```
plot(df)
```




(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(100)
labels <- sample(2, nrow(df), replace = TRUE)
labels
```

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

(c) Compute the centroid for each cluster.

```
cent1 <- c(mean(df[labels == 1, 1]), mean(df[labels == 1, 2]))
cent2 <- c(mean(df[labels == 2, 1]), mean(df[labels == 2, 2]))

cent1
```

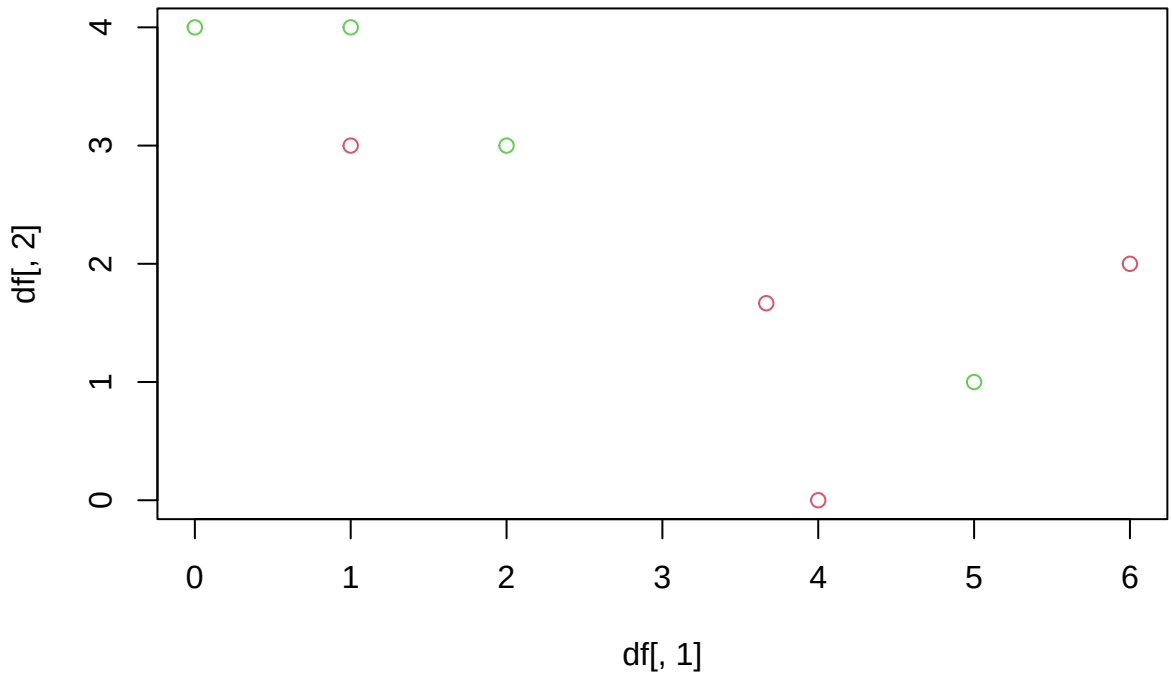
```
## [1] 3.666667 1.666667
```

```
cent2
```

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

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

```
plot(df[,1], df[,2], col=(labels+1))
points(cent1[1], cent1[2], col=2)
points(cent2[1], cent2[2], col=3)
```



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

```
euclid = function(a, b) {
  return(sqrt((a[1] - b[1])^2 + (a[2]-b[2])^2))
}
assign_labels = function(df, cent1, cent2) {
  labels = rep(NA, nrow(df))
  for (i in 1:nrow(df)) {
    if (euclid(df[i,], cent1) < euclid(df[i,], cent2)) {
      labels[i] = 1
    } else {
      labels[i] = 2
    }
  }
  return(labels)
}
labels = assign_labels(df, cent1, cent2)
labels
```

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

```
last_labels = rep(-1, 6)
while (!all(last_labels == labels)) {
  last_labels = labels
  cent1 = c(mean(df[labels==1, 1]), mean(df[labels==1, 2]))
  cent2 = c(mean(df[labels==2, 1]), mean(df[labels==2, 2]))
  print(cent1)
  print(cent2)
  labels = assign_labels(df, cent1, cent2)
}
```

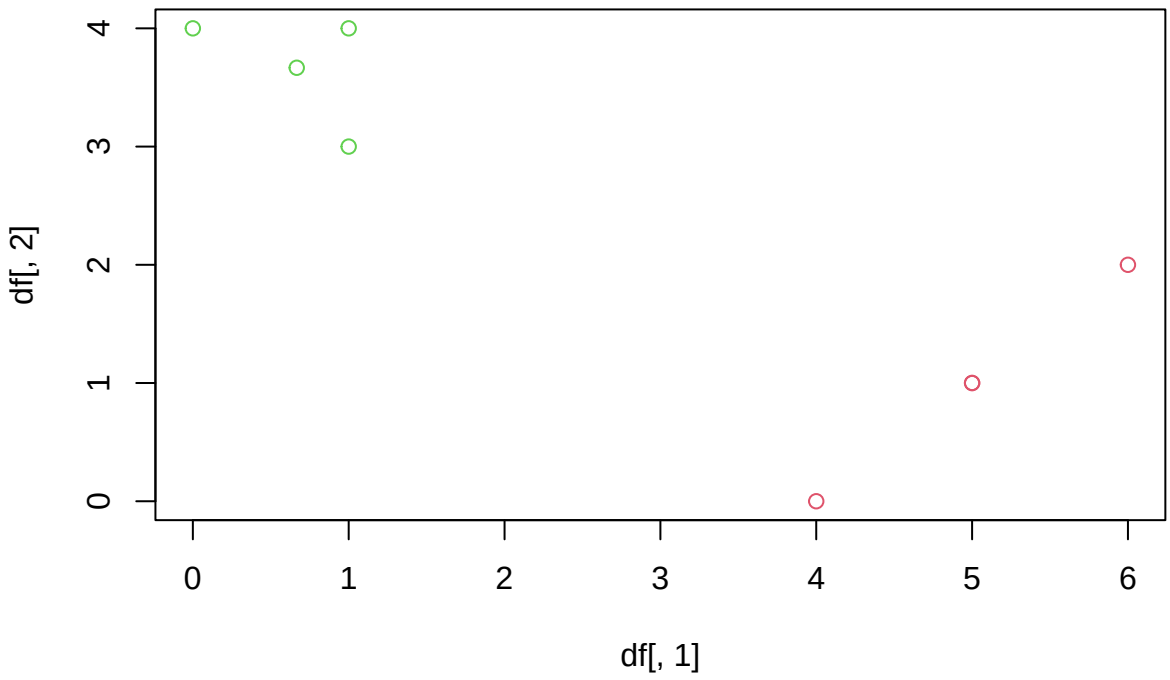
```
## [1] 5 1
## [1] 0.6666667 3.6666667
```

```
labels
```

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

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

```
plot(df[,1], df[,2], col=(labels+1))
points(cent1[1], cent1[2], col=2)
points(cent2[1], cent2[2], col=3)
```




12.5 In words, describe the results that you would expect if you performed K-means clustering of the eight shoppers in Figure 12.16, on the basis of their sock and computer purchases, with K = 2. Give three answers, one for each of the variable scalings displayed. Explain.

Using clusters based on two-dimensional euclidean distance - least socks and computers vs more socks and computers - purchased computer vs no computer purchase - purchased computer vs no computer purchase

12.6 We saw in Section 12.2.2 that the principal component loading and score vectors provide an approximation to a matrix, in the sense of (12.5). Specifically, the principal component score and loading vectors solve the optimization problem given in (12.6).

Now, suppose that the M principal component score vectors zim, m = 1,...,M, are known. Using (12.6), explain that the first M principal component loading vectors jm, m = 1,...,M, can be obtaining by performing M separate least squares linear regressions. In each regression, the principal component score vectors are the predictors, and one of the features of the data matrix is the response.

12.8 In Section 12.2.3, a formula for calculating PVE was given in Equation 12.10. 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:

```
data(USArrests)
df <- USArrests
X <- data.matrix(scale(USArrests))
```

(a) Using the sdev output of the prcomp() function, as was done in Section 12.2.3.

```
pr.out <- prcomp(X)
pr.var <- pr.out$sdev^2
pve <- pr.var / sum(pr.var)
pve
```

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

(b) By applying Equation 12.10 directly. That is, use the prcomp() function to compute the principal component loadings. Then, use those loadings in Equation 12.10 to obtain the PVE. These two approaches should give the same results.

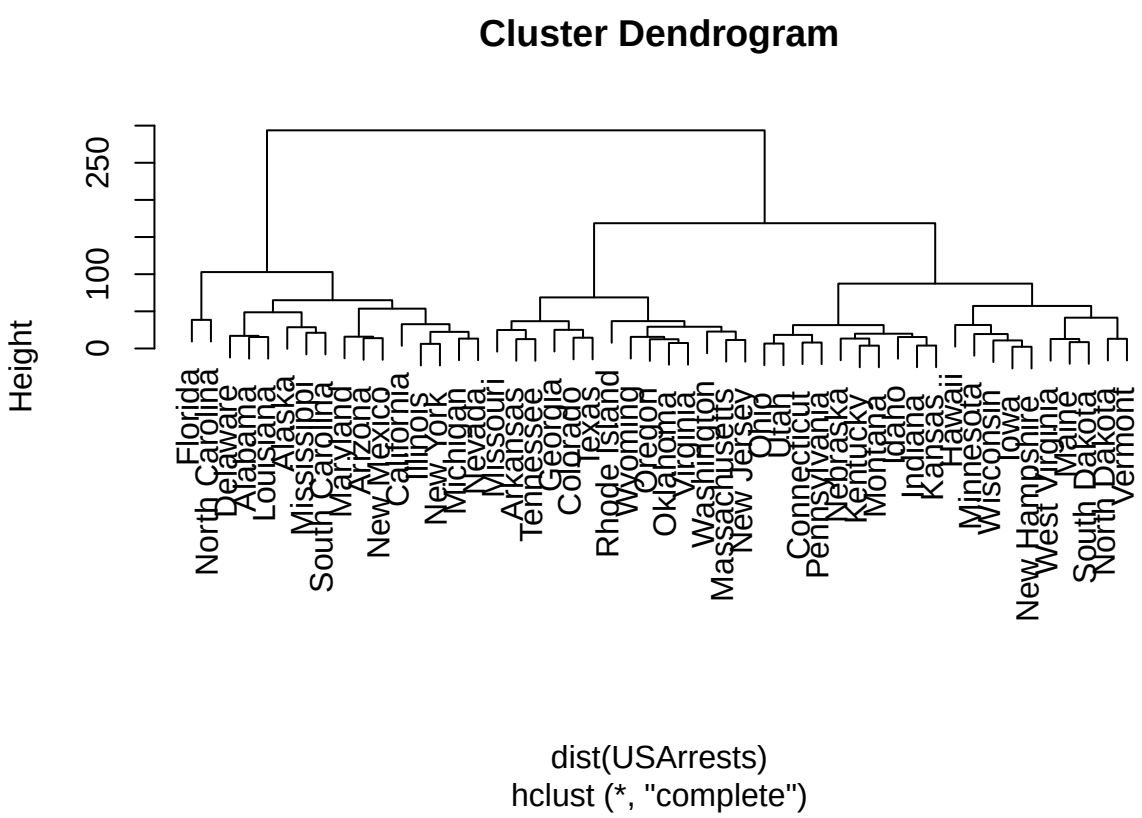
```
loadings = pr.out$rotation
pve2 = rep(NA, 4)
dmean = apply(X, 2, mean)
dsdev = sqrt(apply(X, 2, var))
dsc = sweep(X, MARGIN=2, dmean, "-")
dsc = sweep(dsc, MARGIN=2, dsdev, "/")
for (i in 1:4) {
  proto_x = sweep(dsc, MARGIN=2, loadings[,i], "*")
  pc_x = apply(proto_x, 1, sum)
  pve2[i] = sum(pc_x^2)
}
pve2 = pve2/sum(dsc^2)
pve2
```

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

12.9 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.

```
hc.complete <- hclust(dist(USArrests), method= "complete"); plot(hc.complete)
```




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

```
cutree(hc.complete, 3)
```

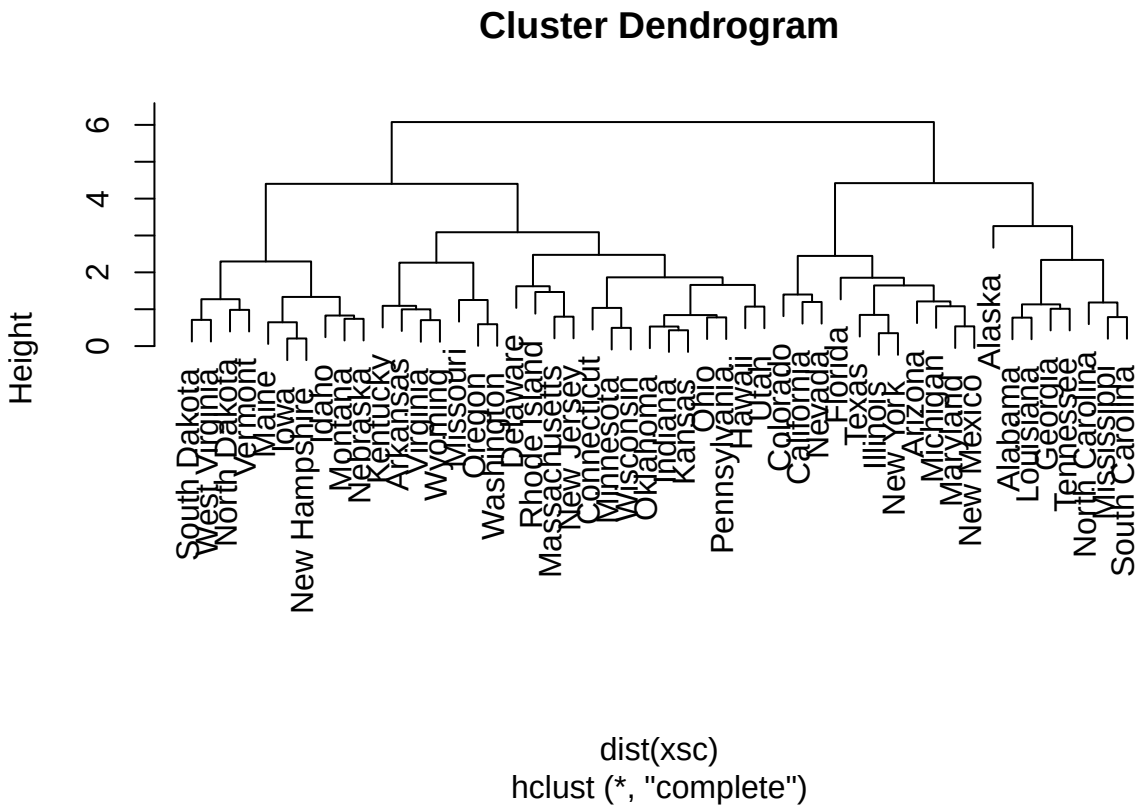
```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              2              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              1              1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

```
table(cutree(hc.complete, 3))
```

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

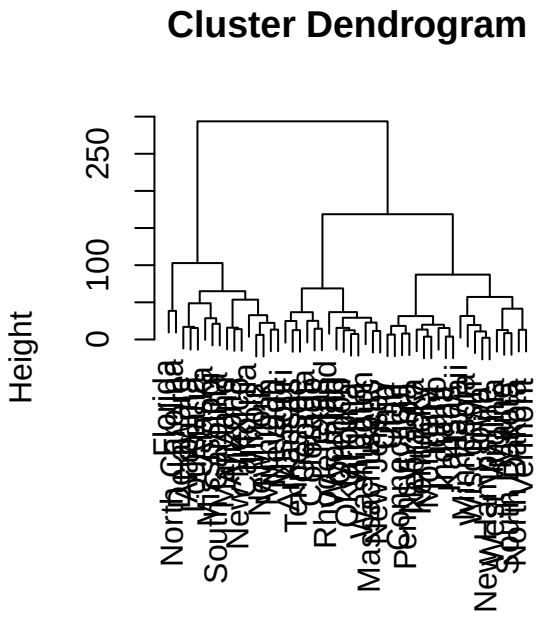
(c) Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

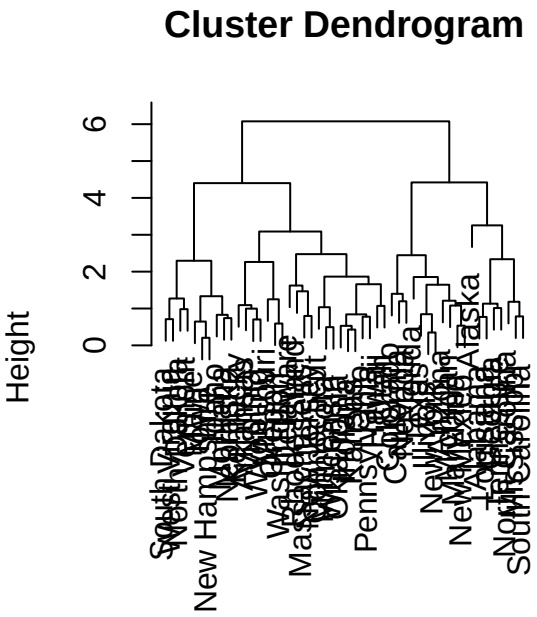
```
xsc <- scale(USArrests)
s.hc.complete <- hclust(dist(xsc), method= "complete")
plot(s.hc.complete)
```




(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.

```
par(mfrow = c(1,2))
plot(hc.complete)
plot(s.hc.complete)
```




```
cutree(s.hc.complete, 3)
```

```
##        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
```

```
table(cutree(s.hc.complete, 3))
```

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

```
table(cutree(s.hc.complete, 3), cutree(hc.complete, 3))
```

```
##
##      1  2  3
##   1  6  2  0
##   2  9  2  0
##   3  1 10 20
```

Scaling the variables effects the max height of the dendogram.

12.10 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.

```
set.seed(42)
x <- matrix(rnorm(20*3*50, mean = 0, sd=0.001), ncol = 50)
x[1:20, 2] = 1
x[21:40, 1] = 2
x[21:40, 2] = 2
x[41:60, 1] = 1
```

(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.

```
pca.out <- prcomp(x)
summary(pca.out)
```

```
## Importance of components:
##                           PC1    PC2      PC3      PC4      PC5      PC6
## Standard deviation     1.008 0.5822 0.001838 0.001756 0.001681 0.001634
## Proportion of Variance 0.750 0.2500 0.000000 0.000000 0.000000 0.000000
## Cumulative Proportion  0.750 1.0000 0.999970 0.999970 0.999970 0.999970
##                           PC7      PC8      PC9     PC10    PC11     PC12
## Standard deviation     0.001585 0.001546 0.001493 0.001398 0.00137 0.001318
## Proportion of Variance 0.000000 0.000000 0.000000 0.000000 0.00000 0.000000
## Cumulative Proportion  0.999980 0.999980 0.999980 0.999980 0.99998 0.999980
##                           PC13     PC14     PC15     PC16     PC17     PC18
## Standard deviation     0.001289 0.001255 0.001208 0.001184 0.001146 0.001131
## Proportion of Variance 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
## Cumulative Proportion  0.999980 0.999990 0.999990 0.999990 0.999990 0.999990
##                           PC19     PC20      PC21      PC22      PC23
```

```
## Standard deviation     0.001112 0.001061 0.0009968 0.0009732 0.0009321
## Proportion of Variance 0.000000 0.000000 0.0000000 0.0000000 0.0000000
## Cumulative Proportion  0.999990 0.999990 0.9999900 0.9999900 0.9999900
##                            PC24      PC25      PC26      PC27      PC28
## Standard deviation     0.0009079 0.0008839 0.0008404 0.0008306 0.0007873
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## Cumulative Proportion  0.9999900 0.9999900 1.0000000 1.0000000 1.0000000
##                            PC29      PC30     PC31      PC32      PC33
## Standard deviation     0.0007483 0.0007442 0.000688 0.0006564 0.0006051
## Proportion of Variance 0.0000000 0.0000000 0.000000 0.0000000 0.0000000
## Cumulative Proportion  1.0000000 1.0000000 1.000000 1.0000000 1.0000000
##                            PC34      PC35      PC36      PC37      PC38
## Standard deviation     0.0005916 0.0005614 0.0005073 0.0004997 0.0004527
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## Cumulative Proportion  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##                            PC39      PC40      PC41      PC42      PC43
## Standard deviation     0.0004246 0.0004031 0.0003531 0.0003352 0.0003243
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## Cumulative Proportion  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##                           PC44      PC45      PC46     PC47      PC48      PC49
## Standard deviation     0.00031 0.0002541 0.0002386 0.000196 0.0001804 0.0001425
## Proportion of Variance 0.00000 0.0000000 0.0000000 0.000000 0.0000000 0.0000000
## Cumulative Proportion  1.00000 1.0000000 1.0000000 1.000000 1.0000000 1.0000000
##                            PC50
## Standard deviation     0.0001034
## Proportion of Variance 0.0000000
## Cumulative Proportion  1.0000000
```

```
#center = mean
#scale = sd
```

```
pca.out$x[,1:2]
```

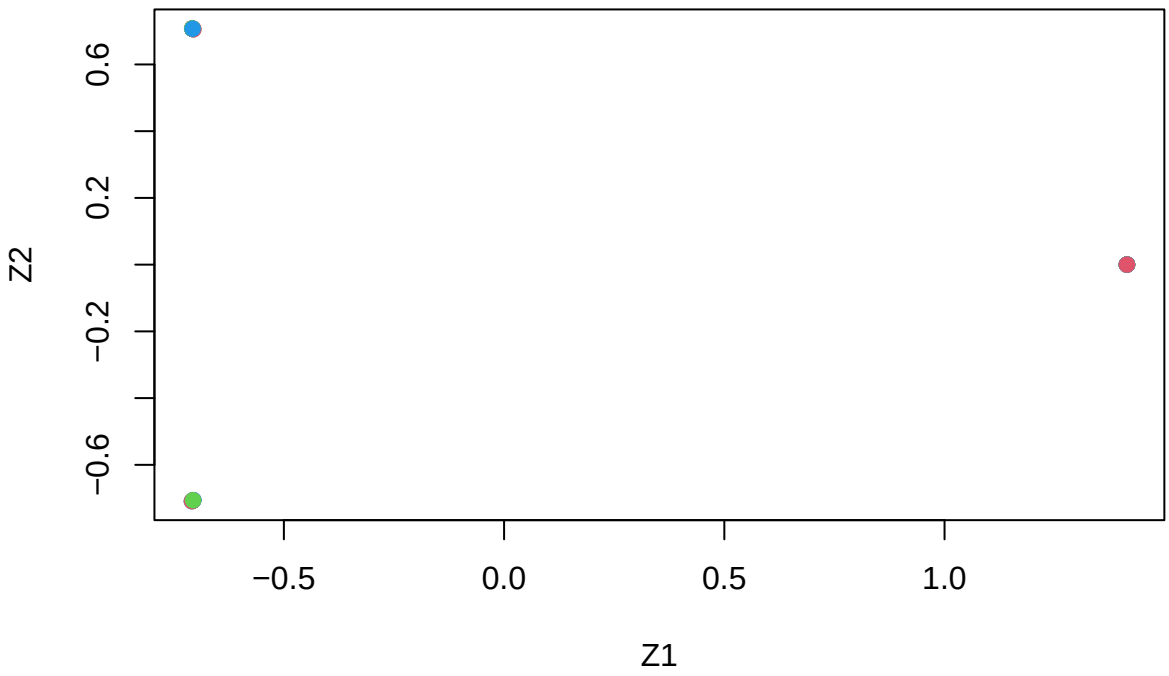
```
##              PC1           PC2
##  [1,] -0.7061235 -7.062436e-01
##  [2,] -0.7074932 -7.076085e-01
##  [3,] -0.7068371 -7.069567e-01
##  [4,] -0.7066456 -7.067688e-01
##  [5,] -0.7068080 -7.069248e-01
##  [6,] -0.7071683 -7.072873e-01
##  [7,] -0.7060238 -7.061442e-01
##  [8,] -0.7071622 -7.072776e-01
##  [9,] -0.7056660 -7.057851e-01
## [10,] -0.7071383 -7.072540e-01
## [11,] -0.7061699 -7.062879e-01
## [12,] -0.7054764 -7.055951e-01
## [13,] -0.7080748 -7.081951e-01
## [14,] -0.7072898 -7.074079e-01
## [15,] -0.7071858 -7.073086e-01
## [16,] -0.7066445 -7.067601e-01
## [17,] -0.7072937 -7.074142e-01
## [18,] -0.7089719 -7.090905e-01
## [19,] -0.7088174 -7.089374e-01
```

```
## [20,] -0.7061612 -7.062738e-01
## [21,]  1.4141667  8.259491e-05
## [22,]  1.4141654  8.098864e-05
## [23,]  1.4141659  8.285926e-05
## [24,]  1.4141649  8.300148e-05
## [25,]  1.4141655  8.397797e-05
## [26,]  1.4141655  8.340420e-05
## [27,]  1.4141655  8.316826e-05
## [28,]  1.4141649  8.117997e-05
## [29,]  1.4141648  8.447030e-05
## [30,]  1.4141653  8.390736e-05
## [31,]  1.4141643  8.475419e-05
## [32,]  1.4141648  8.417797e-05
## [33,]  1.4141646  8.287956e-05
## [34,]  1.4141663  8.397035e-05
## [35,]  1.4141646  8.278485e-05
## [36,]  1.4141663  8.103362e-05
## [37,]  1.4141639  8.293366e-05
## [38,]  1.4141656  8.376207e-05
## [39,]  1.4141642  8.697669e-05
## [40,]  1.4141664  8.182318e-05
## [41,] -0.7063684  7.061515e-01
## [42,] -0.7064793  7.062623e-01
## [43,] -0.7079267  7.077126e-01
## [44,] -0.7059111  7.056969e-01
## [45,] -0.7076902  7.074766e-01
## [46,] -0.7071448  7.069343e-01
## [47,] -0.7075169  7.073030e-01
## [48,] -0.7073043  7.070919e-01
## [49,] -0.7070861  7.068692e-01
## [50,] -0.7071357  7.069194e-01
## [51,] -0.7072358  7.070198e-01
## [52,] -0.7071419  7.069254e-01
## [53,] -0.7075622  7.073461e-01
## [54,] -0.7075751  7.073611e-01
## [55,] -0.7083942  7.081763e-01
## [56,] -0.7074901  7.072745e-01
## [57,] -0.7075819  7.073662e-01
## [58,] -0.7053079  7.050943e-01
## [59,] -0.7081804  7.079702e-01
## [60,] -0.7071212  7.069053e-01
```

```
pr.out$rotation
```

```
##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
## Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
## UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
## Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

```
plot(pca.out$x[,1:2], col=2:4, xlab="Z1", ylab="Z2", pch=19)
```




(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.

```
km.out <- kmeans(x, 3, nstart = 20)
table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))
```

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

Three equal clusters of 20 observations.

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

```
km.out <- kmeans(x, 2, nstart = 20)
km.out$cluster
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

The third class gets clustered in with class 1 or 2.

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

```
km.out <- kmeans(x, 4, nstart = 20)
km.out$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [39] 4 4 2 2 2 2 3 3 2 2 3 3 3 3 3 3 3 3 2 2 2
```

One of the previous clusters is split into two so that there are four total.

(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.

```
km.out = kmeans(pca.out$x[,1:2], 3, nstart=20)
table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))
```

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

Three equal clusters of 20 observations.

(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.

```
km.out = kmeans(scale(x), 3, nstart=20)
km.out$cluster
```

```
##  [1] 2 2 2 3 2 3 3 2 3 2 1 1 1 3 3 1 3 2 1 1 1 3 3 1 3 1 3 3 1 3 2 2 1 3 3 3 1 1
## [39] 3 2 3 3 1 1 3 2 3 1 2 2 2 2 2 2 2 2 2 1 1 1
```

```
table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))
```

```
##
##      1  2  3
##   1  6  7  6
##   2  7  3 10
##   3  7 10  4
```

The results are not good as in part b, scaling the observations effects the distance between them.