Assignment 5

Exercise 1

In this exercise, we will predict the number of applications received using the other variables in the College data set.

(a) Split the data set into a training set and a test set.

```
library(ISLR)

sum(is.na(College))
```

```
[1] 0
```

```
summary(College$Apps)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
     81     776    1558    3002    3624   48094 
```

```
# one extreme outlier is removed before the split (the amount of
# applications has probably been writen wrong)

College2 = College[-which.max(College$Apps), ]
summary(College2$Apps)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
     81     776    1558    2944    3603   21804 
```

```
set.seed(12345)
# 50-50 split so that we can compare the results with assignment 1 ex. 1
train = sample(1:nrow(College2), floor(nrow(College2)/2), replace = F)

trainCollege = College2[train, ]
testCollege = College2[-train, ]
```

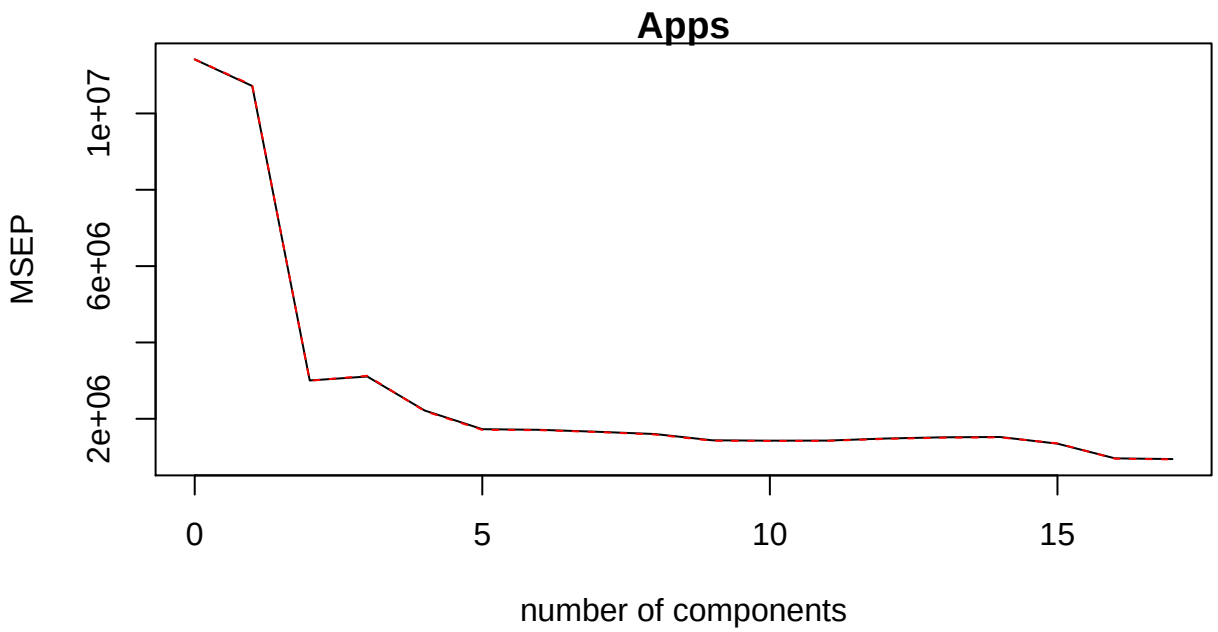
(e) Fit a PCR model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

```
library(pls)
```

```
Attaching package: 'pls'
```

```
The following object is masked from 'package:stats':

    loadings
```

```
# scale = TRUE, X is scaled by dividing each variable by its sample standard
# deviation CV = 10-fold by default
set.seed(31)
pcr.fit = pcr(Apps ~ ., data = trainCollege, scale = T, validation = "CV")

# MSEP = mean squared error of prediction
validationplot(pcr.fit, val.type = "MSEP")
```




```
# value of M can be found in that column for which the CV is the smallest
summary(pcr.fit)
```

```
Data:   X dimension: 388 17
    Y dimension: 388 1
Fit method: svdpc
Number of components considered: 17

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV            3379     3274     1733     1763     1488     1314     1308
adjCV         3379     3277     1731     1768     1483     1307     1304
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV        1288     1265     1198      1194      1195      1217      1231
adjCV     1286     1260     1193      1191      1192      1213      1226
       14 comps  15 comps  16 comps  17 comps
CV         1234      1161     981.4     972.0
adjCV      1230      1165     977.4     967.5

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X      30.841    56.13    63.48    69.80    75.40    80.24    84.21
Apps    8.872    74.32    74.45    82.45    86.03    86.04    86.43
      8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
X       87.29    90.33     92.95     95.17     96.96     98.07     98.84
Apps    87.53    88.34     88.47     88.55     88.59     88.59     88.68
      15 comps  16 comps  17 comps
X        99.41     99.89    100.00
Apps     90.05     92.81     93.07
```

```
# ncomp = the PCR components for which predictions are sought
pcr.pred = predict(pcr.fit, testCollege, ncomp = pcr.fit$ncomp)
```

```
# test MSE
mean((testCollege[, "Apps"] - data.frame(pcr.pred))^2)
```

```
[1] 1181488
```

```
# test RMSE
sqrt(mean((testCollege[, "Apps"] - data.frame(pcr.pred))^2))
```

```
[1] 1086.963
```

The value of M selected by 10-fold cross-validation is 17 ($RMSEP = 972.0$) for which the test error rate is 1 181 488 and RMSE is 1086.96. All 17 attributes are included in the model which means that the model is the same we got with OLS so the PCR didn't do any dimensionality reduction. If the number of components is reduced manually into 10 ($RMSEP = 1194$) for example, we get

```
pcr.pred2 = predict(pcr.fit, testCollege, ncomp = 10)

# test MSE
mean((testCollege[, "Apps"] - data.frame(pcr.pred2))^2)
```

```
[1] 1355909
```

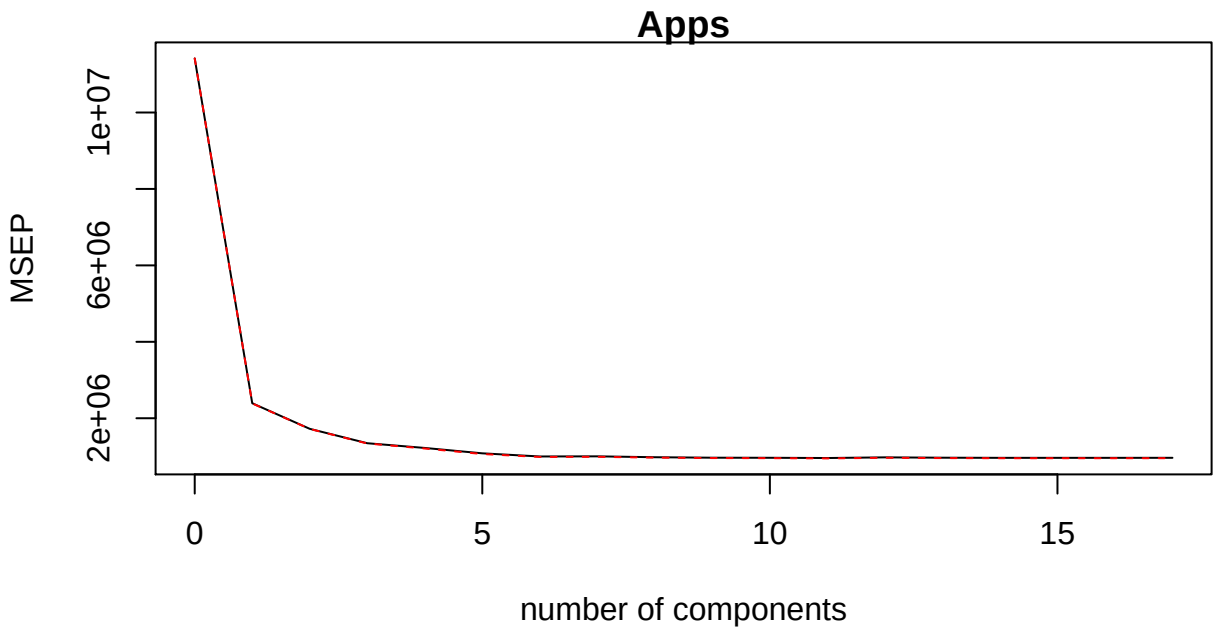
```
# test RMSE
sqrt(mean((testCollege[, "Apps"] - data.frame(pcr.pred2))^2))
```

```
[1] 1164.435
```

which is more useful in a sense that we're trying to reduce dimensions but it comes with the cost of increasing the test error.

(f) Fit a PLS model on the training set, with M chosen by crossvalidation. Report the test error obtained, along with the value of M selected by cross-validation.

```
set.seed(54)
pls.fit = plsr(Apps ~ ., data = trainCollege, scale = T, validation = "CV")

validationplot(pls.fit, val.type = "MSEP")
```



```r
# value of M can be found in that column for which the CV is the smallest
summary(pls.fit)
```

```
Data:   X dimension: 388 17
    Y dimension: 388 1
Fit method: kernelpls
Number of components considered: 17

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV            3379     1547     1313     1160     1106     1040    999.7
adjCV         3379     1545     1312     1157     1099     1031    993.4
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV      1001.4    989.8    983.4     982.3     979.1     988.0     983.6
adjCV    995.3    984.7    978.7     977.4     974.3     982.5     978.4
       14 comps  15 comps  16 comps  17 comps
CV        981.7     981.7     981.9       982
adjCV     976.7     976.7     976.9       977

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X       26.03    45.03    61.94    65.27    68.54    72.43    78.12
Apps    79.64    85.81    89.23    90.91    92.26    92.86    92.91
      8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
X       81.63    84.03     85.64     88.04     91.04     93.03     94.80
Apps    92.94    92.98     93.02     93.04     93.06     93.06     93.06
      15 comps  16 comps  17 comps
X        96.42     98.72    100.00
Apps     93.07     93.07     93.07
```

```r
pls.pred = predict(pls.fit, testCollege, ncomp = )

# test MSE
mean((testCollege[, "Apps"] - data.frame(pls.pred))^2)
```

```
[1] 1289324
```

```r
# test RMSE
sqrt(mean((testCollege[, "Apps"] - data.frame(pls.pred))^2))
```

```
[1] 1135.484
```

CV error minimizes with 11 components ($RMSEP = 979.1$), the test error for this model is 1 289 324 and $RMSE = 1135.48$.

(g) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

In the first exercise of Assignment 1 we derived the test RMSE for OLS, Ridge and Lasso which were 1086.963, 1077.797 and 1101.812, respectively. Lasso had only 4 attributes where as Ridge and OLS had all 18 attributes, intercept included. For PCR with 10 components the test RMSE is 1164.435 and for PLS with 11 components $RMSE = 1135.484$.

The best test error is achieved when all attributes are included in the model but it makes the model harder to

interpret and we get pretty close to the same test error rate with less saturated model. The best performing model, when both parsimony and test error rate is considered, seems to be Lasso, though it might be too sparse. Both PCR and PLS get close to the same test error but they both have more than double the number of predictors than in Lasso. Overall, the differences between the models are relatively small.

Exercise 2

We will now try to predict per capita crime rate in the Boston data set.

(a) Try out PCR. Present and discuss results.

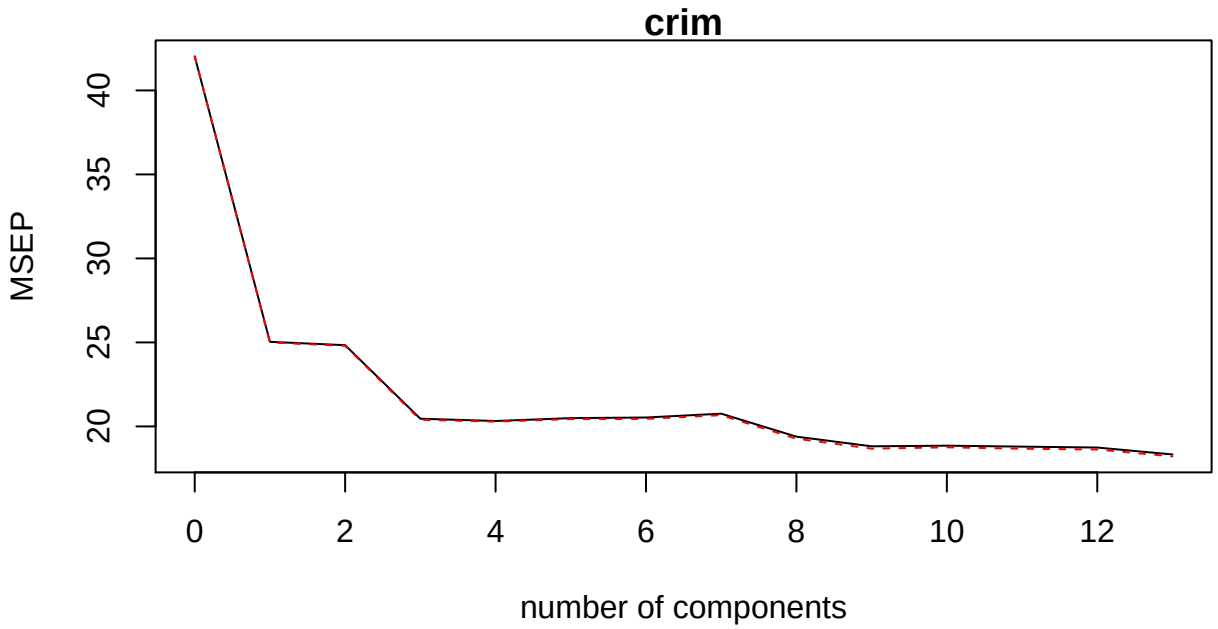
```
library(MASS)

sum(is.na(Boston))
```

```
[1] 0
```

```
# if the data is split we can use separate test set to calculate the final
# test set error
set.seed(12345)
train = sample(1:nrow(Boston), size = floor(nrow(Boston)/2), replace = F)
trainBoston = Boston[train, ]
testBoston = Boston[-train, ]

set.seed(67)
pcr.fit.Boston = pcr(crim ~ ., data = trainBoston, scale = TRUE, validation = "CV")

summary(pcr.fit.Boston)
```

```
Data:   X dimension: 253 13
    Y dimension: 253 1
Fit method: svdpc
Number of components considered: 13

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           6.483    5.003    4.984    4.523    4.508    4.527    4.531
adjCV        6.483    4.999    4.979    4.516    4.504    4.521    4.522
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       4.556    4.403    4.338     4.342     4.336     4.329     4.282
adjCV    4.546    4.390    4.322     4.331     4.321     4.316     4.268

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X       48.33    60.85     70.4    77.19    83.48    88.42    91.52
crim    41.72    42.46     53.3    53.39    53.71    54.80    54.80
      8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
X       93.86    95.56     97.15     98.46     99.50    100.00
crim    58.10    59.30     59.47     60.14     60.16     61.16
```

```
validationplot(pcr.fit.Boston, val.type = "MSEP")
```




```
# printing out test RMSE for different models
for (i in c(13, 12, 11, 9)) {
    print(paste("The test RMSE for PCR with", i, "components is"))
    pcr.pred.Boston = predict(pcr.fit.Boston, testBoston, ncomp = i)
    print(sqrt(mean((testBoston[, "crim"] - data.frame(pcr.pred.Boston))^2)))
}
```

```
[1] "The test RMSE for PCR with 13 components is"
[1] 8.189385
[1] "The test RMSE for PCR with 12 components is"
[1] 8.286668
[1] "The test RMSE for PCR with 11 components is"
[1] 8.303782
[1] "The test RMSE for PCR with 9 components is"
[1] 8.31295
```

The PCR model with 13 components has the lowest CV error (4.282), the second lowest CV error is for model with 12 components (4.329) and the fourth lowest is for a model with 9 components (4.338).

(b) Propose a model that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, crossvalidation, or some other reasonable alternative, as opposed to using training error.

In exercise 2 of Assignment 1, the test RMSE for subset selection, Ridge and Lasso were 8.175, 8.264 and 8.213, respectively.Subset selection had 10 parameters, Ridge had 14 parameters and Lasso had only 3 parameters. PCR with 9 components has test RMSE of 8.313. The 10 parameter model formed by subset selection seems to perform best if test RMSE is considered and I'd choose it for the final model. Lasso might be too sparse and thus not perform well in general.

(c) Does your chosen model involve all of the features in the data set? Why or why not?

The final model involves only 9 features because the intercept is included in 10 parameters. It' doesn't include all the features because the method is developed to choose the n most explanatory variable according to used criterion. A more sparse model is easier to interpret and not all variables have the same power to explain the

output so they might not even be so interesting. The features included in the model are *zn, chas, nox, dis, rad, ptratio, black, lstat* and *medv*.

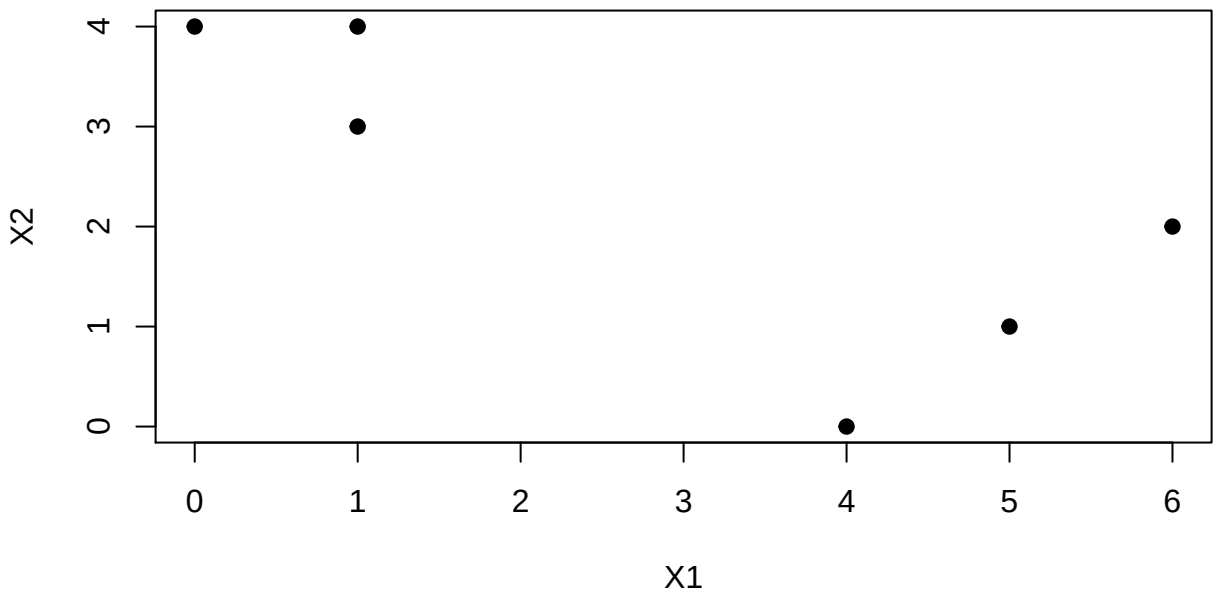
Exercise 3

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.

Obs.	X1	X2
1	1	4
2	1	3
3	0	4
4	5	1
5	6	2
6	4	0

(a) Plot the observations.

```
# storing the data given
X = as.data.frame(cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0)))
colnames(X) <- c("X1", "X2")

plot(X[, 1], X[, 2], xlab = "X1", ylab = "X2", pch = 19)
```



(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(23)
labels = sample(2, nrow(X), replace = T)
labels
```

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

(c) Compute the centroid for each cluster.

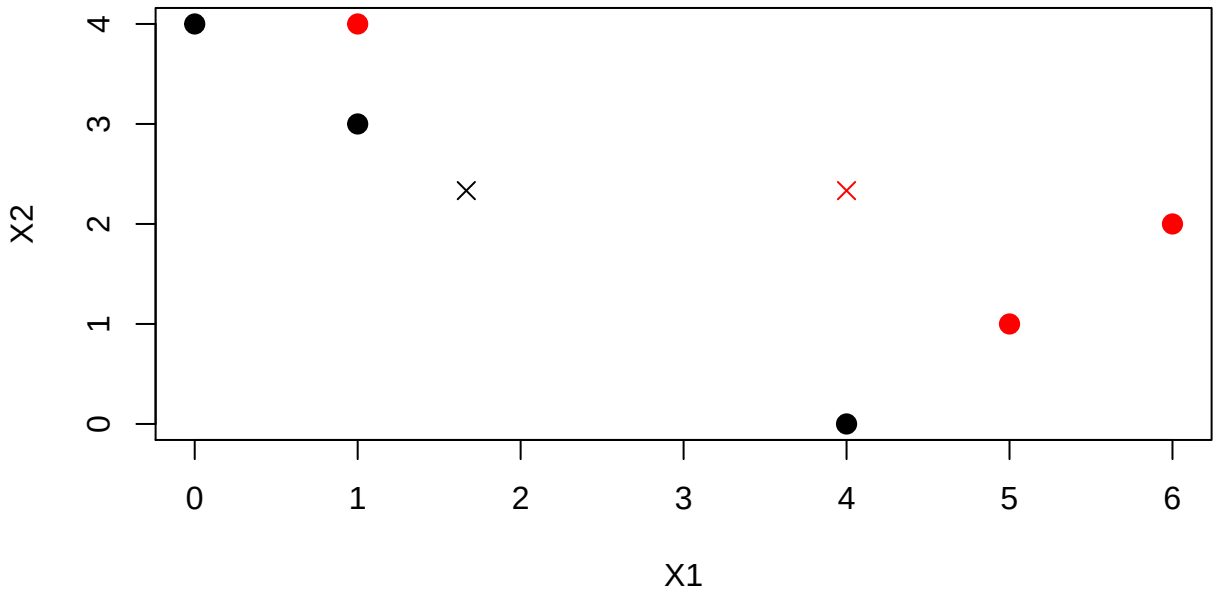
```
centroid1 = c(mean(X[labels == 1, 1]), mean(X[labels == 1, 2]))
centroid2 = c(mean(X[labels == 2, 1]), mean(X[labels == 2, 2]))

centroid1
```

```
[1] 1.666667 2.333333
```

```
centroid2
```

```
[1] 4.000000 2.333333
```

```
plot(X[, 1], X[, 2], col = (labels), pch = 20, cex = 2, xlab = "X1", ylab = "X2")
points(centroid1[1], centroid1[2], col = 1, pch = 4, cex = 1.2)
points(centroid2[1], centroid2[2], col = 2, pch = 4, cex = 1.2)
```



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

```
euclid = function(a, b) {
    # calculates euclidean distance
    return(sqrt((a[1] - b[1])^2 + (a[2] - b[2])^2))
}

assign_labels = function(x, centroid1, centroid2) {
    # used to store the predicted labels
    labels = rep(NA, nrow(x))
    for (i in 1:nrow(x)) {
        # if the distance to other centroid is less than to other, choose the
        # centroid for which the distance is smaller
        if (euclid(x[i, ], centroid1) < euclid(x[i, ], centroid2)) {
            labels[i] = 1
        } else {
            labels[i] = 2
        }
```

```
    }
    return(labels)
}

labels = assign_labels(X, centroid1, centroid2)
labels
```

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

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

```
# vector consisting of -1 for 6 times
last_labels = rep(-1, 6)

# while all labels do not differ from each other
while (!all(last_labels == labels)) {
    last_labels = labels
    # calculate new centroids
    centroid1 = c(mean(X[labels == 1, 1]), mean(X[labels == 1, 2]))
    centroid2 = c(mean(X[labels == 2, 1]), mean(X[labels == 2, 2]))
    # print out new centroids
    print(centroid1)
    print(centroid2)
    # form new labels
    labels = assign_labels(X, centroid1, centroid2)
}
```

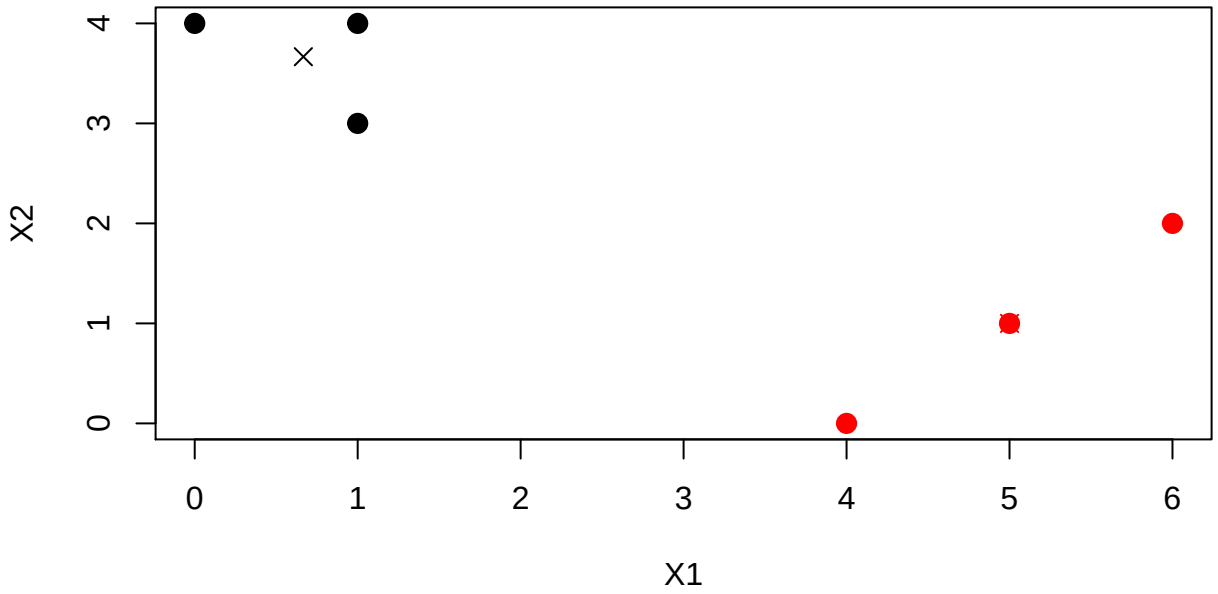
```
[1] 0.6666667 3.6666667
[1] 5 1
```

```
# final labels
labels
```

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

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

```
plot(X[, 1], X[, 2], col = (labels), pch = 20, cex = 2, xlab = "X1", ylab = "X2")
points(centroid1[1], centroid1[2], col = 1, pch = 4, cex = 1.2)
points(centroid2[1], centroid2[2], col = 2, pch = 4, cex = 1.2)
```

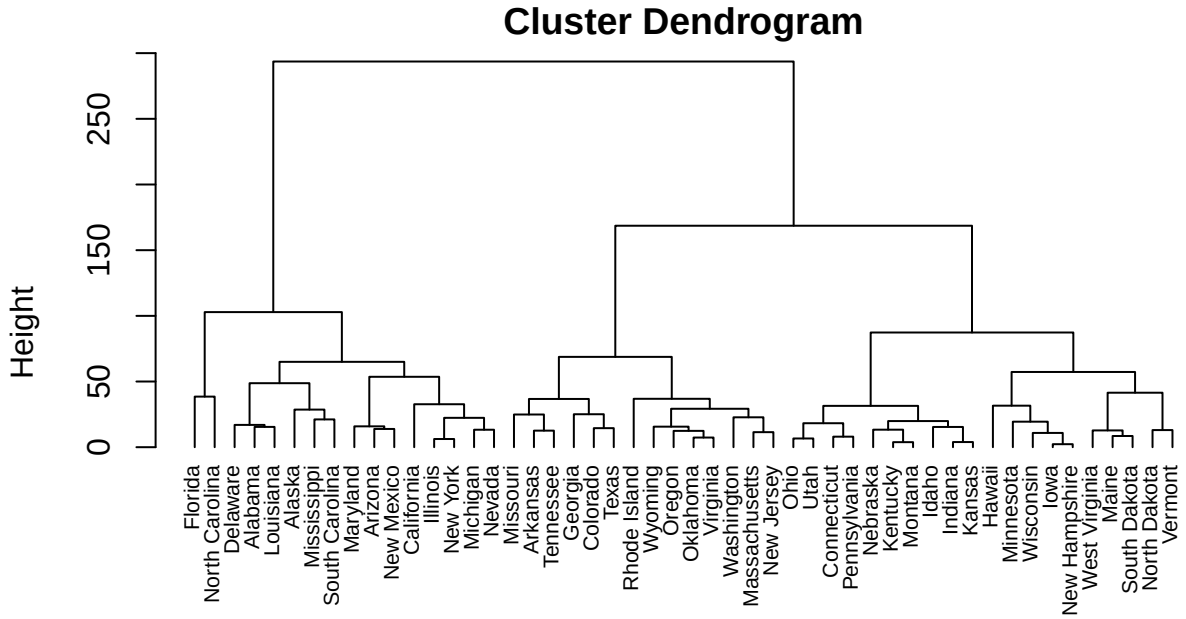


Exercise 4

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")
par(mar = c(0, 4, 1, 1))
plot(hc.complete, hang = -1, cex = 0.7, xlab = "", sub = "")
```



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

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

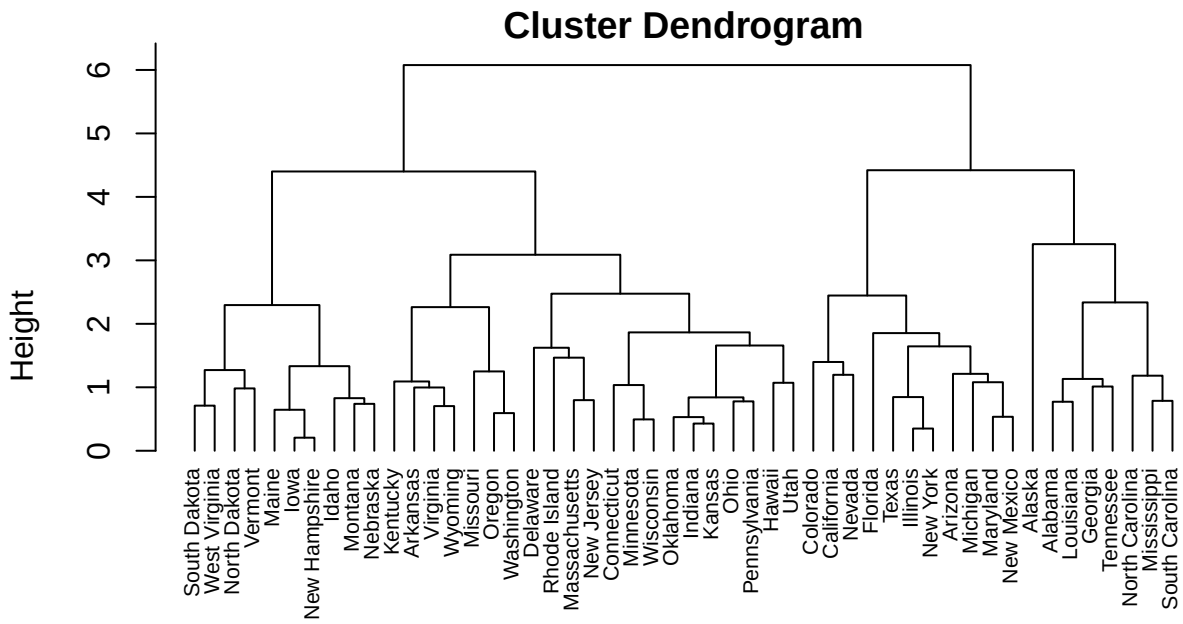
```
       Alabama         Alaska        Arizona       Arkansas     California 
             1              1              1              2              1 
      Colorado    Connecticut       Delaware        Florida        Georgia 
             2              3              1              1              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), dnn = "Not Scaled")
```

```
Not Scaled
 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.

```
dsc = scale(USArrests)
hc.s.complete = hclust(dist(dsc), method = "complete")
plot(hc.s.complete, hang = -1, cex = 0.7, xlab = "", sub = "")
```



(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.

```
cutree(hc.s.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(hc.s.complete, 3), dnn = "Scaled")
```

```
Scaled
 1  2  3
 8 11 31
```

```r
table(cutree(hc.s.complete, 3), cutree(hc.complete, 3), dnn = c("Scaled", "Not Scaled"))
```

```
      Not Scaled
Scaled  1  2  3
     1  6  2  0
     2  9  2  0
     3  1 10 20
```

Scaling the variables affects the max height of the dendogram but it doesn't really impact the bushiness of the tree obtained. However, it does affect the clusters obtained from cutting the dendogram into 3 clusters. The data should be standardized because the data measured has different units (*UrbanPop* compared to other three columns).

Exercise 5

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.

```r
X <- matrix(nrow = 60, ncol = 0)
for (i in 1:50) {
    X <- cbind(X, rnorm(60, mean = rep(1:3, each = 20), sd = 0.2))
}
# y <- rnorm(60, mean = rep(c(1, 2, 1), each = 4), sd = 0.2)
data <- data.frame(X)
```

(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.

```r
pca.out = prcomp(data)  # defalut n-1 -> 3-1 = 2

# the first five components
summary(pca.out)$importance[, 1:5]
```

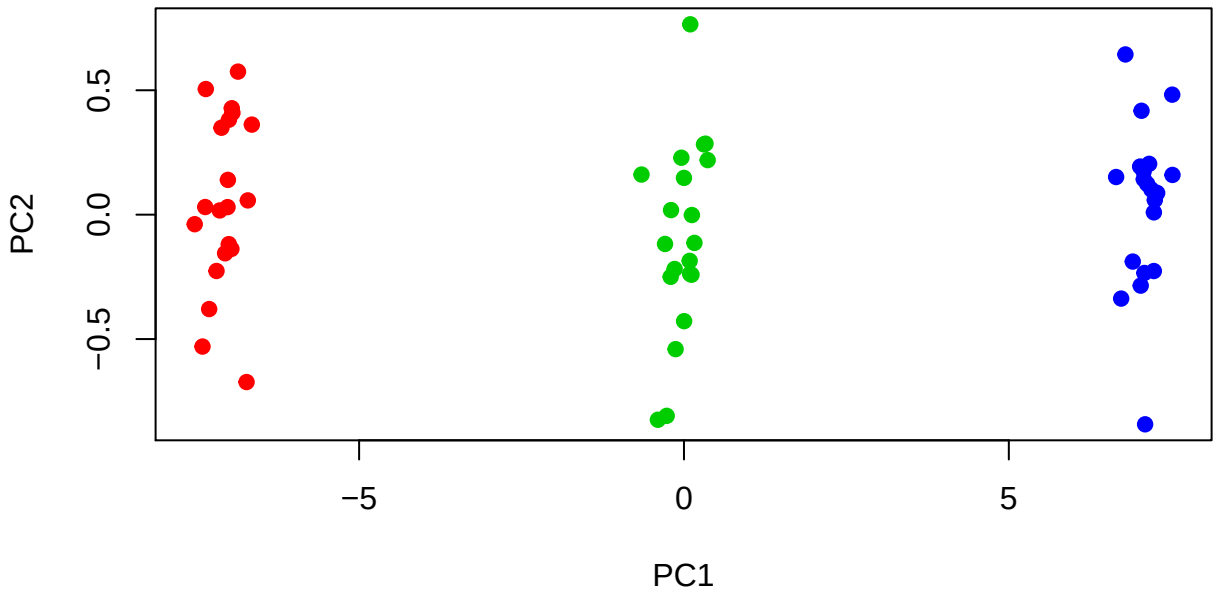
```
                            PC1       PC2       PC3       PC4       PC5
Standard deviation     5.83879 0.3545953 0.3427769 0.3302125 0.3249011
Proportion of Variance 0.94659 0.0034900 0.0032600 0.0030300 0.0029300
Cumulative Proportion  0.94659 0.9500800 0.9533500 0.9563700 0.9593000
```

```r
# few loadings in the first two components
head(pca.out$x[, 1:2])
```

```
           PC1         PC2
[1,] -7.194684 -0.22632615
[2,] -7.118689  0.34932894
[3,] -7.530297 -0.03832284
```

```
[4,] -6.959825  0.42769403
[5,] -7.359571  0.50491949
[6,] -6.949726  0.40908044
```

```
par(mar = c(4, 4, 1, 1))  # graphic setting
plot(pca.out$x[, 1:2], col = rep(2:4, each = 20), xlab = "PC1", ylab = "PC2",
    pch = 19)
```



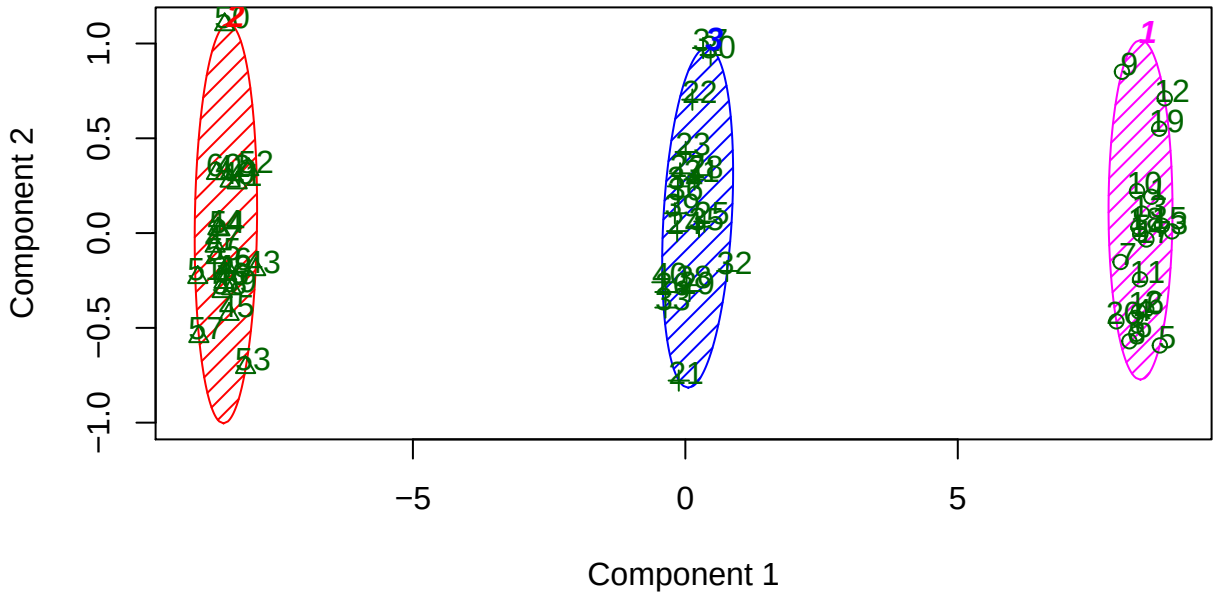
The first two principal components explain 95% of the total variance.

(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.

```
set.seed(289)
km.out = kmeans(data, 3, nstart = 20)
table(km.out$cluster, dnn = NULL)
```

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

```
library(cluster)
clusplot(data, km.out$cluster, color = TRUE, shade = TRUE, labels = 2, lines = 0,
    main = NULL)
```



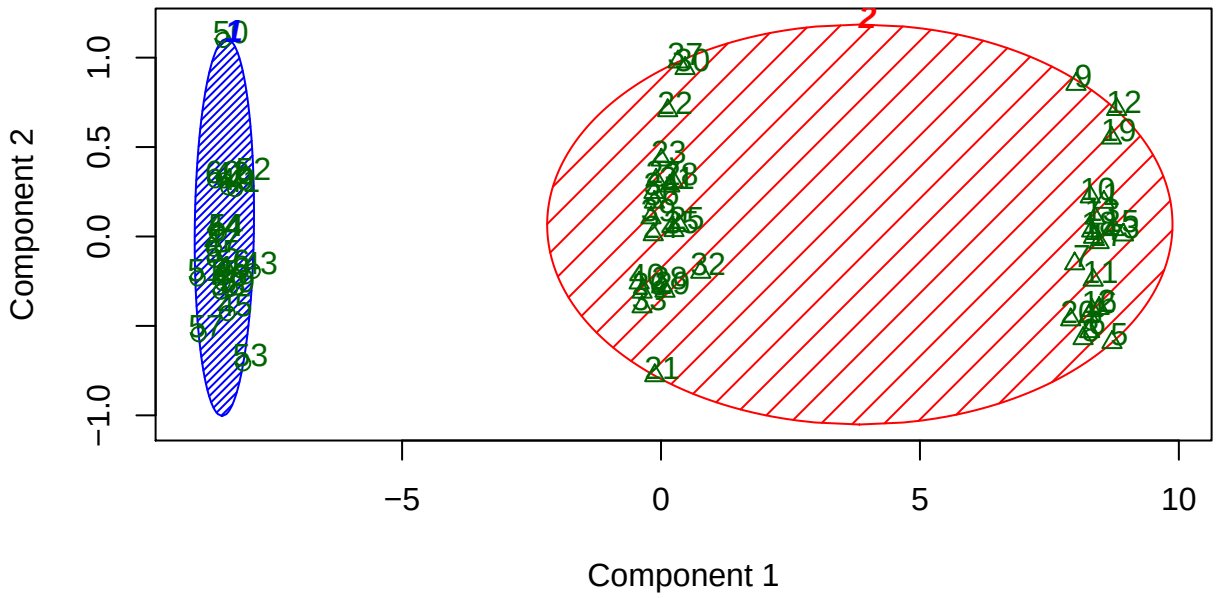
All 60 observations are classified into 3 class with correct amount of observations in each class and the clusters are the same as the true class labels.

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

```
set.seed(486)
km.out = kmeans(data, 2, nstart = 20)
table(km.out$cluster, dnn = NULL)
```

```
 1  2
20 40
```

```
clusplot(data, km.out$cluster, color = TRUE, shade = TRUE, labels = 2, lines = 0,
    main = NULL)
```



Two clusters are united into one but the two groups in that cluster seem so distant from each other, that

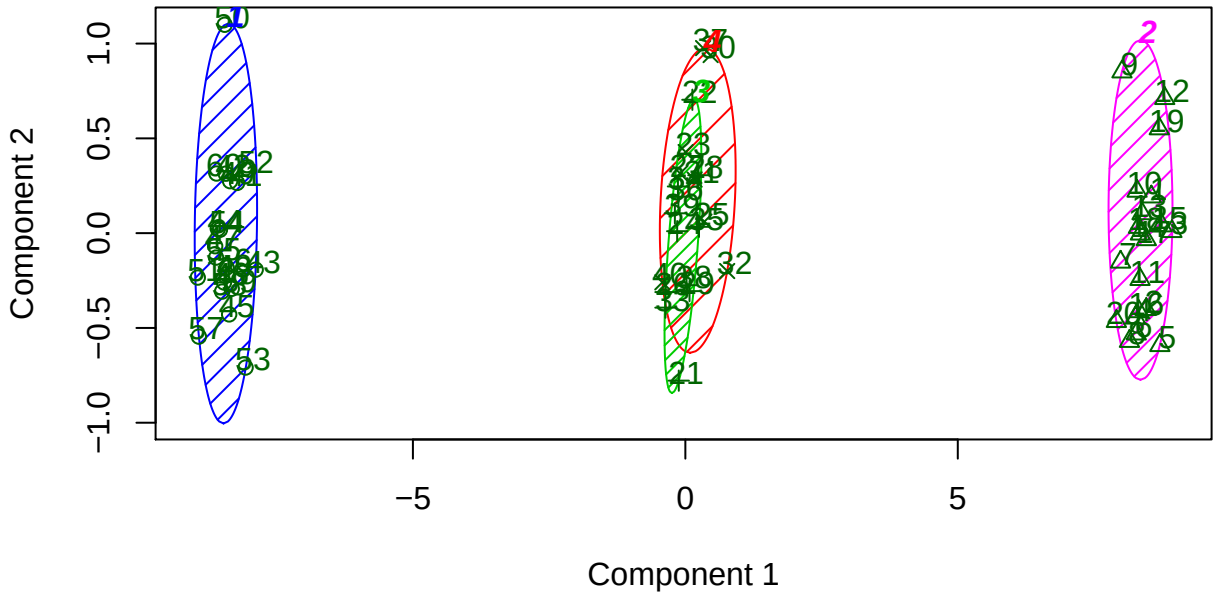
adding one cluster and separating them could be a good idea.

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

```
set.seed(516)
km.out = kmeans(data, 4, nstart = 20)
table(km.out$cluster, dnn = NULL)
```

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

```
clusplot(data, km.out$cluster, color = TRUE, shade = TRUE, labels = 2, lines = 0,
    main = NULL)
```



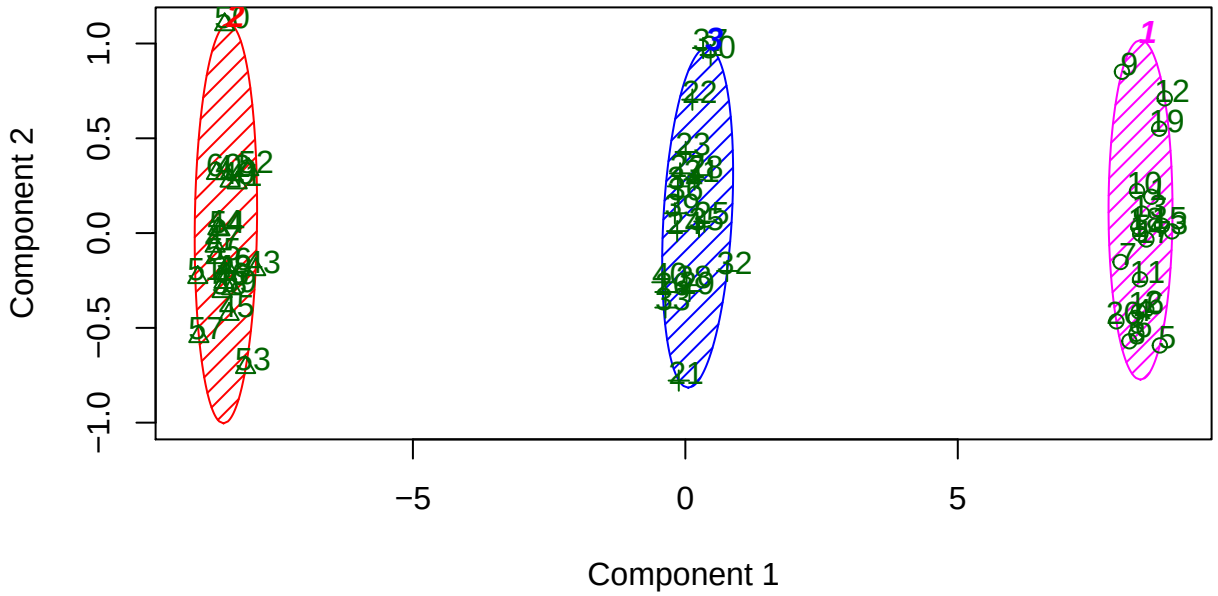
One cluster is split into two different overlapping clusters which means that there may be too many clusters and one could be united into one.

(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 $\times$ 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.

```
set.seed(289)
km.out = kmeans(pca.out$x[, 1:2], 3, nstart = 20)
table(km.out$cluster, dnn = NULL)
```

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

```
clusplot(data, km.out$cluster, color = TRUE, shade = TRUE, labels = 2, lines = 0,
    main = NULL)
```



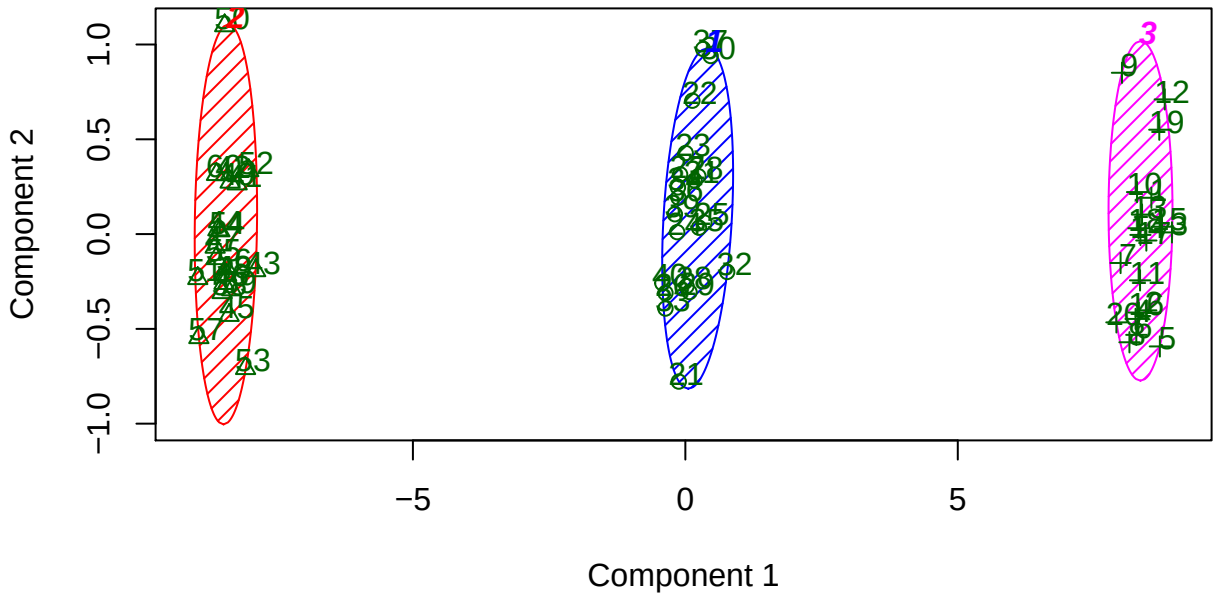
The result is same as in (c).

(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

```
set.seed(273)
km.out = kmeans(scale(data), 3, nstart = 20)
table(km.out$cluster, dnn = NULL)
```

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

```
clusplot(data, km.out$cluster, color = TRUE, shade = TRUE, labels = 2, lines = 0,
    main = NULL)
```



In this data, scaling doesn't affect the clusterization because each cluster is simulated from a normal distribution with same parameters, which means that the distance of the observations doesn't change if they're scaled. Scaling is usually recommended if the data has several different numerical attributes and/or they have different units, in the latter case the attributes are not really comparable in anyway so standardization is recommended.