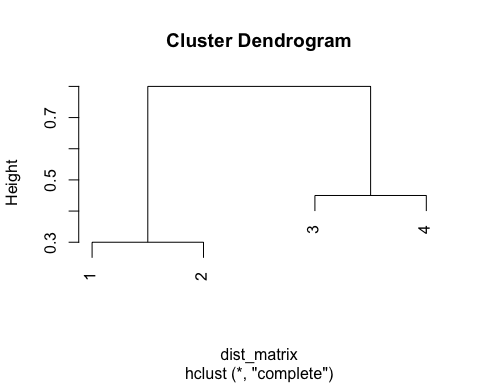
Clusters, Hierarchy, K-means

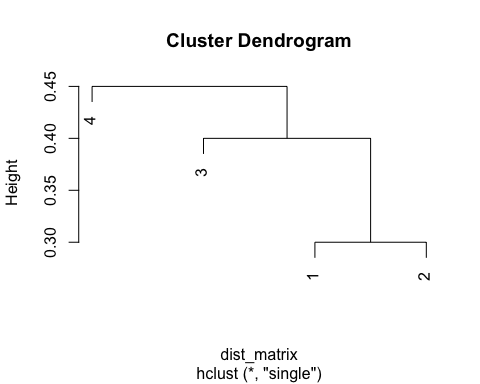
1. (19pts) Suppose that we have four observations, for which we compute a dissimilarity matrix, given by
2. On the basis of this dissimilarity matrix, sketch the dendrogram that results from hierarchically clustering these four observations using complete linkage. Be sure to indicate on the plot the height at which each fusion occurs, as well as the observations corresponding to each leaf in the dendrogram. (5pts: 2 for code, 3 for plot)

dist\_matrix = 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))  
clust.complete = hclust(dist\_matrix, method="complete")  
plot(clust.complete)



1. Repeat (a), this time using single linkage clustering. (5pts: 2 for code, 3 for plot)

clust.single = hclust(dist\_matrix, method="single")  
plot(clust.single)



1. Suppose that we cut the dendogram obtained in (a) such that two clusters result. Which observations are in each cluster? (3pts)

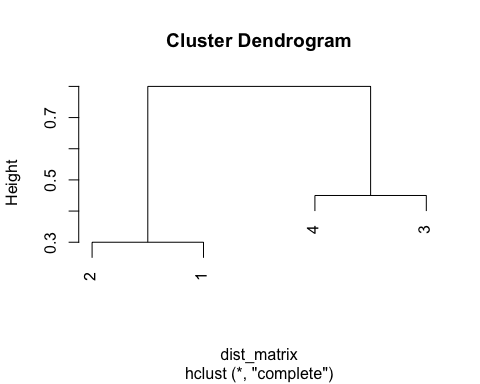
* The clusters would be grouped observation (1,2) and observation (3,4)

1. Suppose that we cut the dendogram obtained in (b) such that two clusters result. Which observations are in each cluster? (3pts)

* The clusters of would be observations (1,2,3) and observation 4

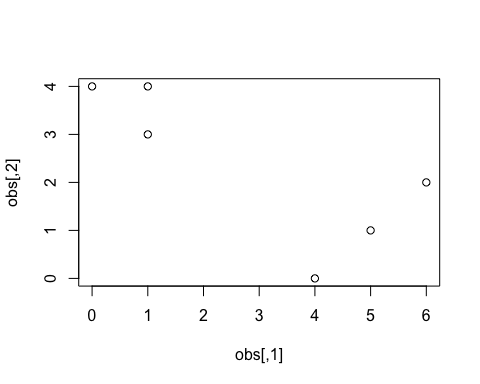
1. It is mentioned in the chapter that at each fusion in the dendrogram, the position of the two clusters being fused can be swapped without changing the meaning of the dendrogram. Draw a dendrogram that is equivalent to the dendrogram in (a), for which two or more of the leaves are repositioned, but for which the meaning of the dendrogram is the same. (3pts: 1 for code, 2 for plot)

plot(clust.complete, labels=c(2,1,4,3))



1. 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. (29pts)
2. Plot the observations. (2pts)

obs = cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))  
plot(obs)



1. 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. (3pts)

set.seed(1)  
cluster\_labels = sample(2, nrow(obs), replace=T)  
cluster\_labels

## [1] 1 2 1 1 2 1

1. Compute the centroid for each cluster. (3pts)

centroid\_1 = c(mean(obs[cluster\_labels==1, 1]), mean(obs[cluster\_labels==1, 2]))  
centroid\_2 = c(mean(obs[cluster\_labels==2, 1]), mean(obs[cluster\_labels==2, 2]))  
print(centroid\_1)

## [1] 2.50 2.25

print(centroid\_2)

## [1] 3.5 2.5

1. Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each. (8pts)

euc\_dist\_func = function(a, b) { (sqrt((a[1] - b[1])^2 + (a[2]-b[2])^2))}  
  
cluster\_assigner = function(obs, centroid\_1, centroid\_2) {  
 labels = rep(NA, nrow(obs))  
 for(i in 1:nrow(obs)) {  
 dist\_centroid\_1 = euc\_dist\_func(obs[i, ], centroid\_1)  
 dist\_centroid\_2 = euc\_dist\_func(obs[i, ], centroid\_2)  
 if (dist\_centroid\_1 > dist\_centroid\_2) {  
 labels[i] = 1   
 } else {  
 labels[i] = 2   
 }  
 }  
 return(labels)  
}  
new\_cluster\_labels = cluster\_assigner(obs, centroid\_1, centroid\_2)  
new\_cluster\_labels

## [1] 2 2 2 1 1 1

1. Repeat (c) and (d) until the answers obtained stop changing.(8pts)

last\_labels = rep(-1, 6)  
new\_cluster\_labels = cluster\_labels  
count=0  
while (!all(last\_labels == new\_cluster\_labels)) {  
 count=count+1  
 last\_labels = cluster\_assigner(obs, centroid\_1, centroid\_2)  
 centroid1 = c(mean(obs[last\_labels==1, 1]), mean(obs[last\_labels==1, 2]))  
 centroid2 = c(mean(obs[last\_labels==2, 1]), mean(obs[last\_labels==2, 2]))  
 print(centroid1)  
 print(centroid2)  
 new\_cluster\_labels = cluster\_assigner(obs, centroid\_1, centroid\_2)  
}

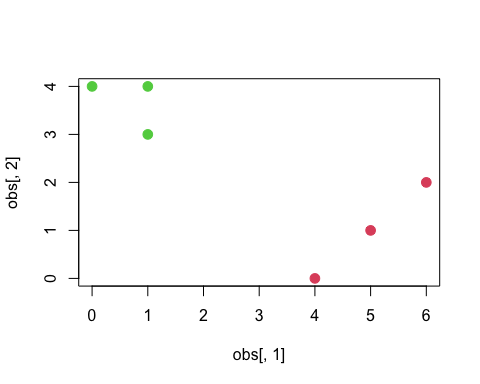
## [1] 5 1  
## [1] 0.6666667 3.6666667

new\_cluster\_labels

## [1] 2 2 2 1 1 1

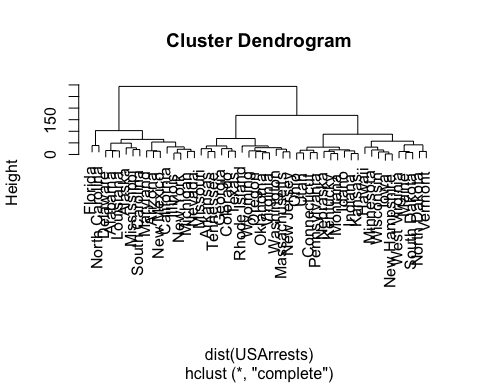
1. In your plot from (a), color the observations according to the cluster labels obtained. (5pts)

plot(obs[,1], obs[,2], col=(new\_cluster\_labels+1), pch=20, cex=2)



1. Consider the USArrests data. We will now perform hierarchical clustering on the states. (21pts)
2. Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states. (pts 6, 2 for code 4 for graph)

library(ISLR)  
cluster.complete = hclust(dist(USArrests), method="complete")  
plot(cluster.complete)



1. Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters? (pts 5)

cutree(cluster.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(cluster.complete, 3))

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

1. Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one. (pts 5)

scaled.USArrests = scale(USArrests)  
scaled.cluster.complete = hclust(dist(scaled.USArrests), method="complete")  
table(cutree(scaled.cluster.complete, 3))

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

1. 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. (pts 5)

* Scaling markedly changed the height of the dendrogram as well as the count of membership for each cluster. The data should be scaled as there are unit differences in the variables, such as percentages as well as rates. Standardization allows for these variables to be of the same magnitude.

1. In this problem, you will generate simulated data, and then perform PCA and K-means clustering on the data. (31pts)
2. 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. (5pts)

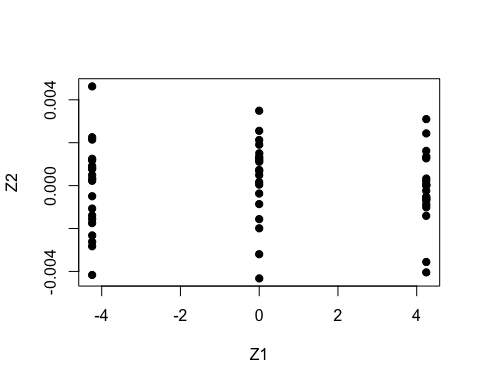
set.seed(1)  
x = matrix(rnorm(20\*3\*50, mean=0, sd=0.001), ncol=50)  
x[1:20, 1] = x[1:20, 1]+3  
x[1:20, 2] = x[1:20, 2]+3  
  
x[41:60, 1] = x[41:60, 1]-3  
x[41:60, 2] = x[41:60, 2]-3

1. 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.(5pts)

pca.out = prcomp(x)  
summary(pca.out)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6  
## Standard deviation 3.493 0.001895 0.001835 0.001697 0.001695 0.001622  
## Proportion of Variance 1.000 0.000000 0.000000 0.000000 0.000000 0.000000  
## Cumulative Proportion 1.000 1.000000 1.000000 1.000000 1.000000 1.000000  
## PC7 PC8 PC9 PC10 PC11 PC12  
## Standard deviation 0.001587 0.00153 0.001452 0.001423 0.0014 0.001346  
## Proportion of Variance 0.000000 0.00000 0.000000 0.000000 0.0000 0.000000  
## Cumulative Proportion 1.000000 1.00000 1.000000 1.000000 1.0000 1.000000  
## PC13 PC14 PC15 PC16 PC17 PC18  
## Standard deviation 0.001314 0.001284 0.001266 0.001246 0.001178 0.001114  
## Proportion of Variance 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000  
## Cumulative Proportion 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000  
## PC19 PC20 PC21 PC22 PC23 PC24  
## Standard deviation 0.001091 0.001046 0.001009 0.0009883 0.0009723 0.0009587  
## Proportion of Variance 0.000000 0.000000 0.000000 0.0000000 0.0000000 0.0000000  
## Cumulative Proportion 1.000000 1.000000 1.000000 1.0000000 1.0000000 1.0000000  
## PC25 PC26 PC27 PC28 PC29  
## Standard deviation 0.0009316 0.0008798 0.0008642 0.000859 0.000763  
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.000000 0.000000  
## Cumulative Proportion 1.0000000 1.0000000 1.0000000 1.000000 1.000000  
## PC30 PC31 PC32 PC33 PC34  
## Standard deviation 0.0007566 0.0007045 0.0006894 0.0006739 0.0006147  
## 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  
## PC35 PC36 PC37 PC38 PC39  
## Standard deviation 0.000594 0.0005893 0.0005278 0.0005073 0.0004977  
## Proportion of Variance 0.000000 0.0000000 0.0000000 0.0000000 0.0000000  
## Cumulative Proportion 1.000000 1.0000000 1.0000000 1.0000000 1.0000000  
## PC40 PC41 PC42 PC43 PC44  
## Standard deviation 0.0004737 0.0003959 0.0003837 0.0003726 0.0003099  
## 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  
## PC45 PC46 PC47 PC48 PC49  
## Standard deviation 0.0002831 0.0002641 0.0002139 0.0001925 0.0001572  
## 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  
## PC50  
## Standard deviation 0.0001181  
## Proportion of Variance 0.0000000  
## Cumulative Proportion 1.0000000

plot(pca.out$x[,1:2], 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. (4pts, 1 for code, 3 for interpretation)

km.out3 =kmeans(x, 3, nstart=20)  
table(km.out3$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

We see that k-means is achieving 100% accuracy in determining the clusters.

1. Perform K-means clustering with K = 2. Describe your results. (4pts, 1 for code, 3 for interpretation)

km.out2 = kmeans(x, 2, nstart=20)  
km.out2$cluster

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

table(km.out2$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

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

We see that Group 2 and 3 were merged into a single group.

1. Now perform K-means clustering with K = 4, and describe your results.4pts, 1 for code, 3 for interpretation)

km.out4 = kmeans(x, 4, nstart=20)  
km.out4$cluster

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

table(km.out4$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

##   
## 1 2 3  
## 1 10 0 0  
## 2 10 0 0  
## 3 0 0 20  
## 4 0 20 0

We see that group 1 is split equally into 2 groups.

1. 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. (4pts, 1 for code, 3 for interpretation)

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

Utilizing the dimensions PCA determined to be the most different, we see that k-means performed perfectly again.

1. 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. (5pts, 1 for code, 4 for interpretation)

x.scaled = scale(x)  
km.out.scaled =kmeans(x.scaled, 3, nstart=20)  
table(km.out.scaled$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

##   
## 1 2 3  
## 1 6 3 4  
## 2 0 8 16  
## 3 14 9 0

We see a lower accuracy for kmeans with the scaled data, this occurs because the distances between the clusters will become closer together. In the original data set the X1 variable for cluster 1 was shifted by positive 3 and cluster 3 was shifted by negative 3. Scaling will decreases this spread and k-means is less efficient.