```
#########################################################
######################                                  #
# R example code for cluster analysis:                  #
######################                                  #
#########################################################


#####################################################################
#####################################################################
############
############           Hierarchical Clustering
############
#####################################################################
#####################################################################


# This is the "foodstuffs" data set from HW 2:

##########################################
###  Foodstuffs example
##########################################

food <- read.table("http://www.stat.sc.edu/~hitchcock/foodstuffs.txt", header=T)

attach(food)

# The hclust function requires that a distance object be input:

# Let's first scale the data by dividing each variable by its standard deviation:

std <- sd(food[,-1]) # finding standard deviations of variables
food.std <- sweep(food[,-1],2,std,FUN="/")

# Calculating pairwise Euclidean distances between the (standardized) objects:

dist.food <- dist(food.std)

# Single linkage:

food.single.link <- hclust(dist.food, method='single')

# Plotting the single linkage dendrogram:

plclust(food.single.link, labels=Food, ylab="Distance")

windows() # opening new window while keeping previous one open

# complete linkage:

food.complete.link <- hclust(dist.food, method='complete')

# Plotting the complete linkage dendrogram:

plclust(food.complete.link, labels=Food, ylab="Distance")

windows() # opening new window while keeping previous one open

# Average linkage:
```

```
food.avg.link <- hclust(dist.food, method='average')

# Plotting the average linkage dendrogram:

plclust(food.avg.link, labels=Food, ylab="Distance")

# Note the complete linkage algorithm is slightly less prone to forming
# "outlier-only" clusters here.

# Cutting the complete-linkage dendrogram to form k=2 clusters here:

cut.2 <- cutree(food.complete.link, k=2)
cut.2     # printing the "clustering vector"

food.2.clust <- lapply(1:2, function(nc) Food[cut.2==nc])
food.2.clust   # printing the clusters in terms of the Food labels

# Suppose we preferred a 5-cluster solution:

cut.5 <- cutree(food.complete.link, k=5)

# Equivalently, in this case:
cut.5 <- cutree(food.complete.link, h=3.5)
# h specifies the height at which the dendrogram should be cut

cut.5   # printing the "clustering vector"

food.5.clust <- lapply(1:5, function(nc) Food[cut.5==nc])
food.5.clust   # printing the clusters in terms of the Food labels


############## Visualization of Clusters:

### Via the scatterplot matrix:

pairs(food[,-1], panel=function(x,y) text(x,y,cut.5))

# Cluster 1 seems to be the high-fat, high-energy foods (beef, ham, pork)
# Cluster 2 foods seem to have low iron (more white meats than red meats)
# Cluster 4 foods have low protein (the clams)
# Cluster 5 is a high-calcium outlier (canned sardines)

### Via a plot of the scores on the first 2 principal components,
### with the clusters separated by color:

food.pc <- princomp(food[,-1],cor=T)

# Setting up the colors for the 5 clusters on the plot:
my.color.vector <- rep("green", times=nrow(food))
my.color.vector[cut.5==2] <- "blue"
my.color.vector[cut.5==3] <- "red"
my.color.vector[cut.5==4] <- "orange"
my.color.vector[cut.5==5] <- "brown"

# Plotting the PC scores:

par(pty="s")
plot(food.pc$scores[,1], food.pc$scores[,2], ylim=range(food.pc$scores[,1]),
     xlab="PC 1", ylab="PC 2", type ='n', lwd=2)
text(food.pc$scores[,1], food.pc$scores[,2], labels=Food, cex=0.7, lwd=2,
     col=my.color.vector)
```

```
# What would this plot look like for the 2-cluster solution?
# For the 3-cluster solution?


##########################################
###  Cars example
##########################################

# The mtcars data set is built into R:

help(mtcars)

# We will focus on the variables that are continuous in nature rather than
discrete:

cars.data <- mtcars[,c(1,3,4,5,6,7)]

# Standardizing by dividing through by the sample range of each variable

samp.range <- function(x){
myrange <- diff(range(x))
return(myrange)
}
my.ranges <- apply(cars.data,2,samp.range)
cars.std <- sweep(cars.data,2,my.ranges,FUN="/")


# Getting distance matrix:

dist.cars <- dist(cars.std)

# Single linkage:

cars.single.link <- hclust(dist.cars, method='single')

# Plotting the single linkage dendrogram:

plclust(cars.single.link, labels=row.names(cars.data), ylab="Distance")

windows() # opening new window while keeping previous one open

# complete linkage:

cars.complete.link <- hclust(dist.cars, method='complete')

# Plotting the complete linkage dendrogram:

plclust(cars.complete.link, labels=row.names(cars.data), ylab="Distance")

windows() # opening new window while keeping previous one open

# Average linkage:

cars.avg.link <- hclust(dist.cars, method='average')

# Plotting the average linkage dendrogram:

plclust(cars.avg.link, labels=row.names(cars.data), ylab="Distance")
```

```
# Average Linkage dendrogram seems to indicate two major clusters,
# Single Linkage dendrogram may indicate three.

# Single Linkage Solution:

cut.3 <- cutree(cars.single.link, k=3)
cut.3     # printing the "clustering vector"

cars.3.clust <- lapply(1:3, function(nc) row.names(cars.data)[cut.3==nc])
cars.3.clust   # printing the clusters in terms of the car names


# Cluster 1 seems to be mostly compact cars, Cluster 2 is sports cars, Cluster 3
is large luxury sedans

############# Visualization of Clusters:

### Via the scatterplot matrix:

pairs(cars.data, panel=function(x,y) text(x,y,cut.3))

# Cluster 1 cars tend to have high mileage, low displacement, low horsepower, low
weight.
# Cluster 3 cars tend to have low mileage, high weight.

### Via a plot of the scores on the first 2 principal components,
### with the clusters separated by color:

cars.pc <- princomp(cars.data,cor=T)

carnames <- abbreviate(row.names(cars.data))

# Setting up the colors for the 3 clusters on the plot:
my.color.vector <- rep("green", times=nrow(cars.data))
my.color.vector[cut.3==2] <- "blue"
my.color.vector[cut.3==3] <- "red"

# Plotting the PC scores:

par(pty="s")
plot(cars.pc$scores[,1], cars.pc$scores[,2], ylim=range(cars.pc$scores[,1]),
     xlab="PC 1", ylab="PC 2", type ='n', lwd=2)
text(cars.pc$scores[,1], cars.pc$scores[,2], labels=carnames, cex=0.7, lwd=2,
     col=my.color.vector)

# It's clear that PC1 is more important in separating the clusters than PC2.

######################    Ward's method:  ####################################
##
## To perform Ward's method of hierarchical clustering, alter the above code to:
##
##         cars.ward <- hclust(dist.cars, method='ward')
##
## and proceed as with the linkage methods.
##
#############################################################################



#################################################################
#################################################################
```

```
############
############           Partitioning Clustering
############
#####################################################################
#####################################################################



############################
##
##  K-means clustering
##
############################

#########################################
###  Foodstuffs example
#########################################

# Consider the food.std data frame given above.

# A K-means clustering with k = 5:

# Note that the stability of the result can be improved by increasing the maximum
number
# of iterations and using multiple random starts:

food.k5 <- kmeans(food.std, centers=5, iter.max=100, nstart=25)
food.k5

# Let's try k=4:

food.k4 <- kmeans(food.std, centers=4, iter.max=100, nstart=25)
food.k4

# Printing the clustering vector for the 4-cluster solution:

food.k4$cluster

food.k4.clust <- lapply(1:4, function(nc) Food[food.k4$cluster==nc])
food.k4.clust   # printing the clusters in terms of the Food labels


############# Visualization of Clusters:

### Via the scatterplot matrix:

pairs(food[,-1], panel=function(x,y) text(x,y,food.k4$cluster))

# Cluster 1 foods tend to be high in calcium. (this comment does not reflect all
runs of the algorithm)
# Cluster 4 foods tend to be high in fat. (this comment does not reflect all runs
of the algorithm)


### Via a plot of the scores on the first 2 principal components,
### with the clusters separated by color:

food.pc <- princomp(food[,-1],cor=T)

# Setting up the colors for the 5 clusters on the plot:
my.color.vector <- rep("green", times=nrow(food))
```

```
my.color.vector[food.k4$cluster==2] <- "blue"
my.color.vector[food.k4$cluster==3] <- "red"
my.color.vector[food.k4$cluster==4] <- "orange"

# Plotting the PC scores:

par(pty="s")
plot(food.pc$scores[,1], food.pc$scores[,2], ylim=range(food.pc$scores[,1]),
     xlab="PC 1", ylab="PC 2", type ='n', lwd=2)
text(food.pc$scores[,1], food.pc$scores[,2], labels=Food, cex=0.7, lwd=2,
     col=my.color.vector)

# Cluster 1 is the "canned seafood" cluster.  (this comment does not reflect all
runs of the algorithm)
# Cluster 2 is the clams cluster.  (this comment does not reflect all runs of the
algorithm)

## NOTE:  The default for the kmeans function in R is the Hartigan-Wong (1979)
algorithm.
## The MacQueen algorithm (1967) can be used by altering the code to, say:
##                kmeans(food.std, centers=4,algorithm="MacQueen")
## You can try it in this case -- I don't think the MacQueen algorithm produces as
good of a result.



############################
##
##  K-medoids clustering
##
############################


###########################################
###  Cars example
###########################################


# Consider the cars.data and cars.std data frames we created above.

# Let's cluster the cars into k groups using the K-medoids approach.

# The function "pam" is in the "cluster" package.

# Loading the "cluster" package:

library(cluster)

# K-medoids directly on the (standardized) data matrix:
cars.kmed.3 <- pam(cars.std, k=3, diss=F)

# Or you can do K-medoids by inputting the distance matrix:
# cars.kmed.3 <- pam(dist.cars, k=3, diss=T)

cars.kmed.3$clustering  # printing the "clustering vector"

cars.kmed.3$silinfo$avg.width  #printing the average silhouette width

### A little function to calculate the average silhouette width
### for a variety of choices of k:
```

```
my.k.choices <- 2:8
avg.sil.width <- rep(0, times=length(my.k.choices))
for (ii in (1:length(my.k.choices)) ){
avg.sil.width[ii] <- pam(cars.std, k=my.k.choices[ii])$silinfo$avg.width
}
print( cbind(my.k.choices, avg.sil.width) )

# A LARGE average silhouette width indicates that the observations are properly
clustered.

# Maybe k=2 is the best choice of k here?


cars.3.clust <- lapply(1:3, function(nc) row.names(cars.data)
[cars.kmed.3$clustering==nc])
cars.3.clust   # printing the clusters in terms of the car names


# Cluster 1 seems to be mostly compact cars, Cluster 2 is sports cars, Cluster 3
is large luxury sedans

############# Visualization of Clusters:

## Built-in plots available with the pam function:

# The "clusplot":

plot(cars.kmed.3, which.plots=1)

# The clusplot (in the "cluster" library) can actually be used with
# any clustering partition by entering the data set and the clustering vector,
e.g.:

clusplot(food[,-1], food.k4$cluster)

# The "silhouette plot":

plot(cars.kmed.3, which.plots=2)

# This shows which observations are "best clustered."


#######################################################
#######################################################
#
# Choosing the number of clusters k using the average silhouette width criterion.
#
#######################################################
#######################################################

# When using pam, the output will give you the average silhouette width (see above
code).

# We can also get the average silhouette width when using other algorithms:

### With a hierarchical method (Complete linkage here):

dist.food <- dist(food.std)
food.complete.link <- hclust(dist.food, method='complete')

summary(silhouette(cutree(food.complete.link, k=2), dist(food.std)))$avg.width
```

```
summary(silhouette(cutree(food.complete.link, k=3), dist(food.std)))$avg.width
summary(silhouette(cutree(food.complete.link, k=4), dist(food.std)))$avg.width
summary(silhouette(cutree(food.complete.link, k=5), dist(food.std)))$avg.width

### With k-means:

summary(silhouette(kmeans(food.std, centers=2, iter.max=100, nstart=25)$cluster,
dist(food.std)))$avg.width
summary(silhouette(kmeans(food.std, centers=3, iter.max=100, nstart=25)$cluster,
dist(food.std)))$avg.width
summary(silhouette(kmeans(food.std, centers=4, iter.max=100, nstart=25)$cluster,
dist(food.std)))$avg.width
summary(silhouette(kmeans(food.std, centers=5, iter.max=100, nstart=25)$cluster,
dist(food.std)))$avg.width

# In each case, we might choose the value of k associated with the LARGEST average
silhouette width.

################################
################################
##
##  Plotting the WSS for several choices of k
##
################################
################################

# This is a recommended method for choosing k in K-means clustering.

# For the cars data, let's consider letting k vary up to 5.

### CODE FOR WSS PLOT BEGINS HERE ###
##
#Enter name of the data matrix to be clustered here:
my.data.matrix <- cars.std

my.k.choices <- 2:5
n <- length(my.data.matrix[,1])
wss1 <- (n-1)*sum(apply(my.data.matrix,2,var))
wss <- numeric(0)
for(i in my.k.choices) {
  W <- sum(kmeans(my.data.matrix,i)$withinss)
  wss <- c(wss,W)
}
wss <- c(wss1,wss)
plot(c(1,my.k.choices),wss,type='l',xlab='Number of clusters', ylab='Within-groups
sum-of-squares', lwd=2)
##
### CODE FOR WSS PLOT ENDS HERE ###

# For what value of k does the elbow of the plot occur?


################################################################
################################################################
############
############           Model-based Clustering
############
################################################################
################################################################

# Consider the built-in USArrests data set in r:
```

```
help(USArrests)

# We will perform a model-based clustering of the 50 states based on these 4
variables:

# Loading the mclust package:
# May need to install the mclust package first?
# If so, type at the command line:  install.packages("mclust", dependencies=T)
# while plugged in to the internet.

library(mclust)

# The R function Mclust performs model-based clustering for a range of models
# and a variety of values of k:

arrest.clus <- Mclust(USArrests)

# By default, the models considered are:
# "EII": spherical, equal volume
# "VII": spherical, unequal volume
# "EEI": diagonal, equal volume and shape
# "VEI": diagonal, varying volume, equal shape
# "EVI": diagonal, equal volume, varying shape
# "VVI": diagonal, varying volume and shape
# "EEE": ellipsoidal, equal volume, shape, and orientation
# "EEV": ellipsoidal, equal volume and equal shape
# "VEV": ellipsoidal, equal shape
# "VVV": ellipsoidal, varying volume, shape, and orientation

# Plotting the BIC values:

plot(arrest.clus, data=USArrests, what="BIC")

# Hit ENTER to see the BIC plot.

# The best solution is VEI with 3 clusters.

# The clustering vector:

clus.vec.3 <- arrest.clus$classification
clus.vec.3

arrest.3.clust <- lapply(1:3, function(nc) row.names(USArrests)[clus.vec.3==nc])
arrest.3.clust   # printing the clusters in terms of the state names

# This gives the probabilities of belonging to each cluster for every object:

round(arrest.clus$z,2)


# Visualizing the clusters:

## Via a scatterplot matrix:

plot(arrest.clus, data=USArrests, what="classification")

# Hit ENTER to see a scatterplot matrix with the points separated by cluster.
# Hit ENTER again to see a scatterplot of the first two variables with objects
separated by cluster.
```

```
### Via a plot of the scores on the first 2 principal components,
### with the clusters separated by color:

arrests.pc <- princomp(USArrests,cor=T)

# Setting up the colors for the 5 clusters on the plot:
my.color.vector <- rep("blue", times=nrow(USArrests))
my.color.vector[arrest.clus$classification==2] <- "red"
my.color.vector[arrest.clus$classification==3] <- "green"

# Plotting the PC scores:

par(pty="s")
plot(arrests.pc$scores[,1], arrests.pc$scores[,2],
ylim=range(arrests.pc$scores[,1]),
     xlab="PC 1", ylab="PC 2", type ='n', lwd=2)
text(arrests.pc$scores[,1], arrests.pc$scores[,2], labels=row.names(USArrests),
     cex=0.7, lwd=2, col=my.color.vector)

# Reviewing the PCA:

summary(arrests.pc,loadings=T)

# Note PC1 is an overall "lack-of-crime" index and PC2 is a "rural" index.

## Note: We could also specifically request the best, say, 2-cluster solution
(according to BIC)
## if we wanted to, for example:

# arrest.clus.2 <- Mclust(USArrests, G=2)


#####################################################################
#####################################################################
############
############           Clustering Binary Data
############
#####################################################################
#####################################################################

# We can read the ACT math test items data from the Internet:

ACTitems <- read.table("http://www.stat.sc.edu/~hitchcock/ACTitems.txt", header=F)

# The first column is a set of labels for the 60 test items.
# The next 60 columns are the scores (0=incorrect, 1=correct) on the items for 55
male students.

# We wish to cluster the items into groups based on the students' scores on them.

# Just using squared Euclidean distance (counting total mismatches):

dist.items <- dist(ACTitems[,-1], method='euclidean')^2

dist.items.2 <- dist(ACTitems[,-1], method='binary') # This distance measure
ignores 0-0 matches altogether

dist.items.3 <- dist(1 - ACTitems[,-1], method='binary') # This distance measure
ignores 1-1 matches altogether
```

```
#### Complete linkage clustering of the 60 test items:

items.complete.link <- hclust(dist.items, method='complete')

# Plotting the complete linkage dendrogram:

plclust(items.complete.link, labels=ACTitems[,1], ylab="Distance")

# Single linkage:

items.sing.link <- hclust(dist.items, method='single')
plclust(items.sing.link, labels=ACTitems[,1], ylab="Distance")

# Single linkage really breaks down when there are a lot of ties among the
distances!

#### K-medoids clustering of the 60 test items:

library(cluster)

### A little function to calculate the average silhouette width
### for a variety of choices of k:

my.k.choices <- 2:8
avg.sil.width <- rep(0, times=length(my.k.choices))
for (ii in (1:length(my.k.choices)) ){
avg.sil.width[ii] <- pam(dist.items, k=my.k.choices[ii])$silinfo$avg.width
}
print( cbind(my.k.choices, avg.sil.width) )

# Maybe 2 clusters, or at most 3, should be used.

items.kmed.2 <- pam(dist.items, k=2, diss=T)
items.2.clust <- lapply(1:2, function(nc) ACTitems[,1]
[items.kmed.2$clustering==nc])
items.2.clust

items.kmed.3 <- pam(dist.items, k=3, diss=T)
items.3.clust <- lapply(1:3, function(nc) ACTitems[,1]
[items.kmed.3$clustering==nc])
items.3.clust
```