Cluster Analysis - Hierarchical Clustering

Alireza Sheikh-Zadeh, Ph.D.

Cluster Analysis

Cluster analysis is a generic term for a wide range of numerical methods with the common goal of uncovering or discovering groups or clusters of observations that are homogeneous and separated from other groups.

The purpose of cluster analysis is to put objects (observations) into groups that the groups are dissimilar to each other, where the objects in the group are similar or close together.

Hierarchical Clustering

This class of clustering methods produces a hierarchical cluster of data, including n observations. In hierarchical clustering (HC), the data are not partitioned into a particular number of groups at a single step. Instead, hierarchical clustering consists of a series of partitions that may run from a single cluster containing all individuals to n clusters. In the end, each cluster includes only a single object.

Here are some essential points about hierarchical clustering:

- It uses **distance matrix** as input similar to multidimensional scaling.
- It's suggested to use standardized Euclidean distance, especially when the units of variables are different.
- The hierarchical clustering does not have the error of reducing the dimensions like multidimensional scaling. It uses the entire original data.

The HC algorithm is explained on page 166 of the EH Textbook. As part of its algorithm, we need a logic for measuring the between-group distances. There are different ways to do that, such as:

- Single linkage: Choosing the smallest distance between two groups
- Complete linkage: Choosing the maximum distance between two groups
- Average linkage: Measuring the average distance between all possible pairs

One difficulty with the cluster analysis is that you do not know which linkage method is the best.

```
# Suppose d is a distance matrix.
```

```r
d <- matrix(c(0, 1, 2, 8,
              1, 0, 7, 9,
              2, 7, 0, 6,
              8, 9, 6, 0), ncol = 4, byrow = T)
d
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    1    2    8
## [2,]    1    0    7    9
## [3,]    2    7    0    6
## [4,]    8    9    6    0
```

```r
class(d)
```

```
## [1] "matrix"
```

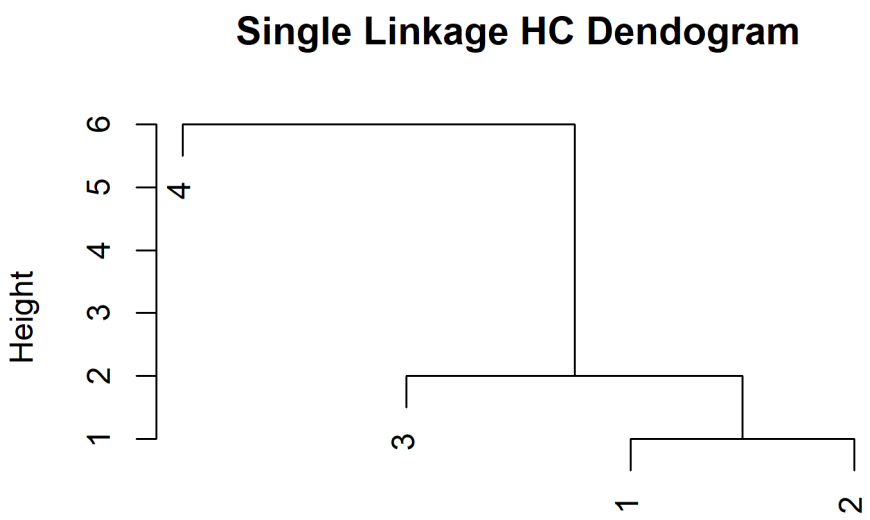
```r
d = as.dist(d) # Change the class from matrix to distance
class(d)
```

```
## [1] "dist"
```

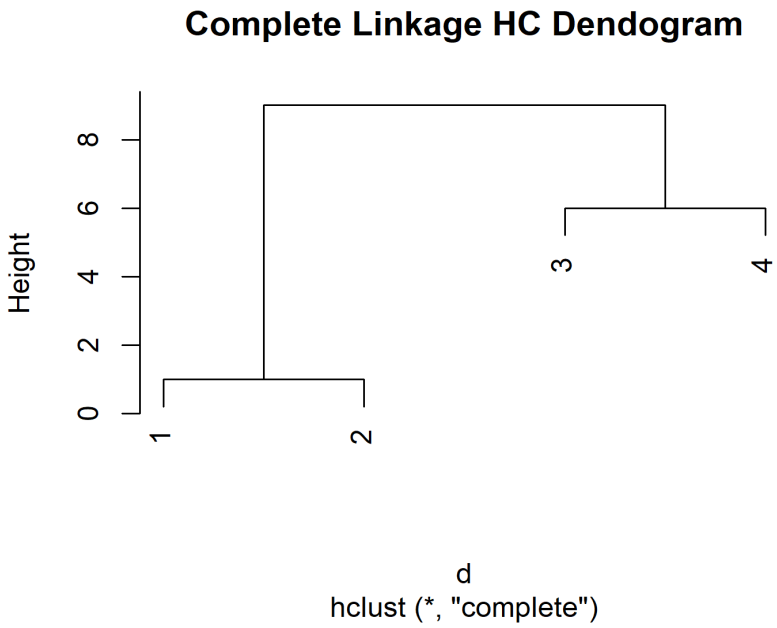
```r
d
```

```
##   1 2 3
## 2 1
## 3 2 7
## 4 8 9 6
```

```r
hc <- hclust(d, "single")
plot(hc, main = "Single Linkage HC Dendogram")
```



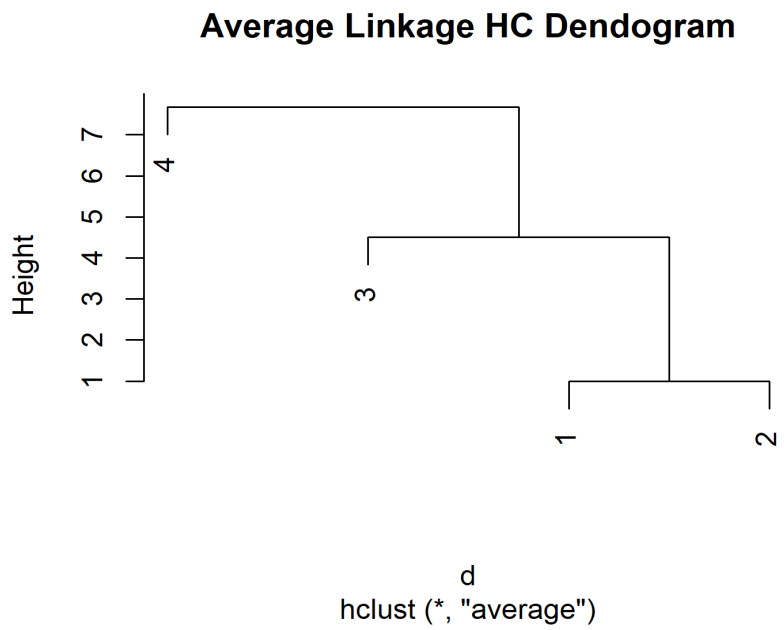

```
hc <- hclust(d, "complete")
plot(hc, main = "Complete Linkage HC Dendogram")
```




```
hc$height
```

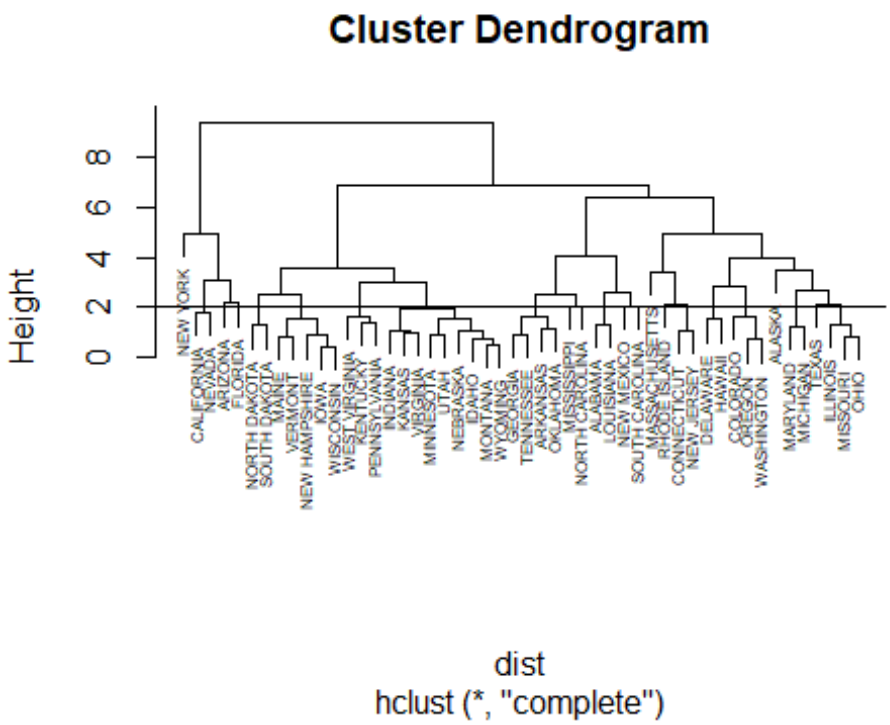
```
## [1] 1 6 9
```

```r
hc <- hclust(d, "average")
plot(hc, main = "Average Linkage HC Dendogram")
```



Example: Perform H-clustering on Crime Data.

```r
crime <- read.csv("https://rb.gy/wu8kvo", row.names = "STATE")
crime.s <- scale(crime)
dist <- dist(crime.s) # distance matrix

hc1 <- hclust(dist, "complete")

plot(hc1, cex = 0.5) # dendrogram
abline(h=2) # This line is useful to determine clusters. If h=8, we divide th
e data into two clusters.
```

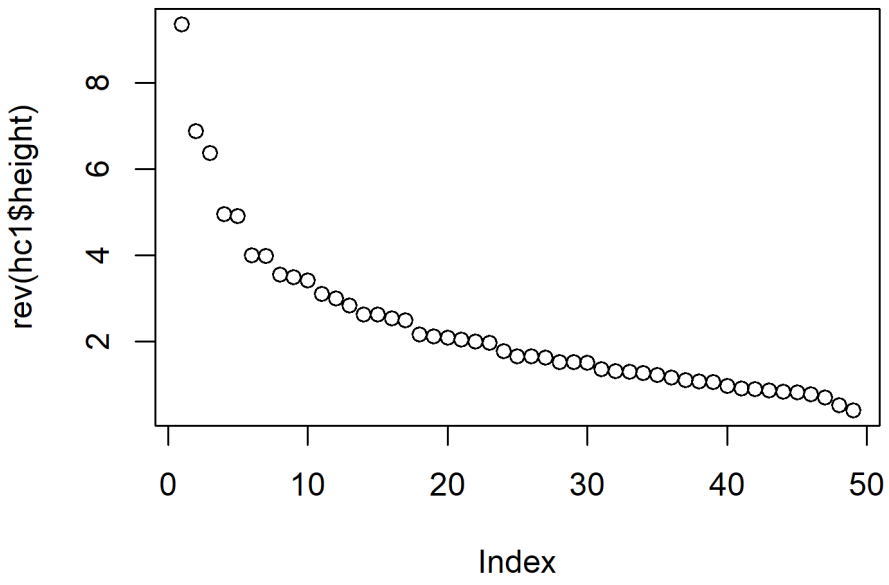



```
# We can check the height of each observation in the dendrogram.

hc1$height

##  [1] 0.4019679 0.5317103 0.7095824 0.7783094 0.8339243 0.8418636 0.8723435
##  [8] 0.9015398 0.9206936 0.9805798 1.0583261 1.0791260 1.1093078 1.1717383
## [15] 1.2366533 1.2742897 1.3033898 1.3188218 1.3695801 1.5152381 1.5254681
## [22] 1.5354740 1.6328410 1.6622884 1.6659082 1.7850846 1.9754183 2.0050489
## [29] 2.0519489 2.0986564 2.1198852 2.1634311 2.4986739 2.5470918 2.6290261
## [36] 2.6383934 2.8436425 3.0091330 3.1146639 3.4163569 3.4922256 3.5595422
## [43] 3.9902108 4.0069255 4.9138853 4.9566281 6.3773302 6.8836840 9.3518725
```

The number of clusters

Construct a scree plot of the inter-cluster distances (heights). The drop up point help you decide the number of clusters.

```
# hc1 is the hierarchical clustering object of the crime data. hc1 contains

following information.
names(hc1)

## [1] "merge"       "height"      "order"       "labels"      "method"
## [6] "call"        "dist.method"

plot(rev(hc1$height))
```




```
# The drop up point help you decide the number of clusters. "Elbow Test"
```

To group objects, we cut the dendrogram at some height, giving a partition with a particular number of groups.

```
# How to get 4 cluster solution?
ct <- cutree(hc1, 4)
ct
```

```
##        ALABAMA         ALASKA        ARIZONA       ARKANSAS     CALIFORNIA
##              1              2              3              1              3
##       COLORADO    CONNECTICUT       DELAWARE        FLORIDA        GEORGIA
##              2              2              2              3              1
##         HAWAII          IDAHO       ILLINOIS        INDIANA           IOWA
##              2              4              2              4              4
##         KANSAS       KENTUCKY      LOUISIANA          MAINE       MARYLAND
##              4              4              1              4              2
##  MASSACHUSETTS       MICHIGAN      MINNESOTA    MISSISSIPPI       MISSOURI
##              2              2              4              1              2
##        MONTANA       NEBRASKA         NEVADA  NEW HAMPSHIRE     NEW JERSEY
##              4              4              3              4              2
##     NEW MEXICO       NEW YORK NORTH CAROLINA   NORTH DAKOTA           OHIO
##              1              3              1              4              2
##       OKLAHOMA         OREGON   PENNSYLVANIA   RHODE ISLAND SOUTH CAROLINA
##              1              2              4              2              1
##   SOUTH DAKOTA      TENNESSEE          TEXAS           UTAH        VERMONT
##              4              1              2              4              4
```

```
##       VIRGINIA     WASHINGTON  WEST VIRGINIA      WISCONSIN        WYOMING
##              4              2              4              4              4

# The best way to summarize the clustering information is the table of counts
.

table(ct)

## ct
##  1  2  3  4
## 10 16  5 19

# You can find the content of each group, e.g., group 1:
cluster1 = subset(rownames(crime), ct==1)
cluster1

## [1] "ALABAMA"        "ARKANSAS"       "GEORGIA"

## [4] "LOUISIANA"      "MISSISSIPPI"    "NEW MEXICO"

## [7] "NORTH CAROLINA" "OKLAHOMA"       "SOUTH CAROLINA"

## [10] "TENNESSEE"

# Then by looking at the average z-score value of the group data, we can find
a meaning for that group.
crime.s = scale(crime)

index1 = match(cluster1, rownames(crime))

colMeans(crime.s[index1, ])

##     MURDER       RAPE    ROBBERY    ASSAULT   BURGLARY    LARCENY
##  1.0360071  0.2329076 -0.2438296  0.7039187 -0.1490187 -0.7846000
##       AUTO
## -0.6172153

cluster2 = subset(rownames(crime), ct==2)

index2 = match(cluster2, rownames(crime))
colMeans(crime.s[index2, ])

##      MURDER        RAPE     ROBBERY     ASSAULT    BURGLARY     LARCENY
## -0.08411415  0.38196016  0.46692042  0.11277712  0.48865351  0.58975150
##        AUTO
##  0.84373687

cluster3 = subset(rownames(crime), ct==3)

index3 = match(cluster3, rownames(crime))
colMeans(crime.s[index3, ])
```

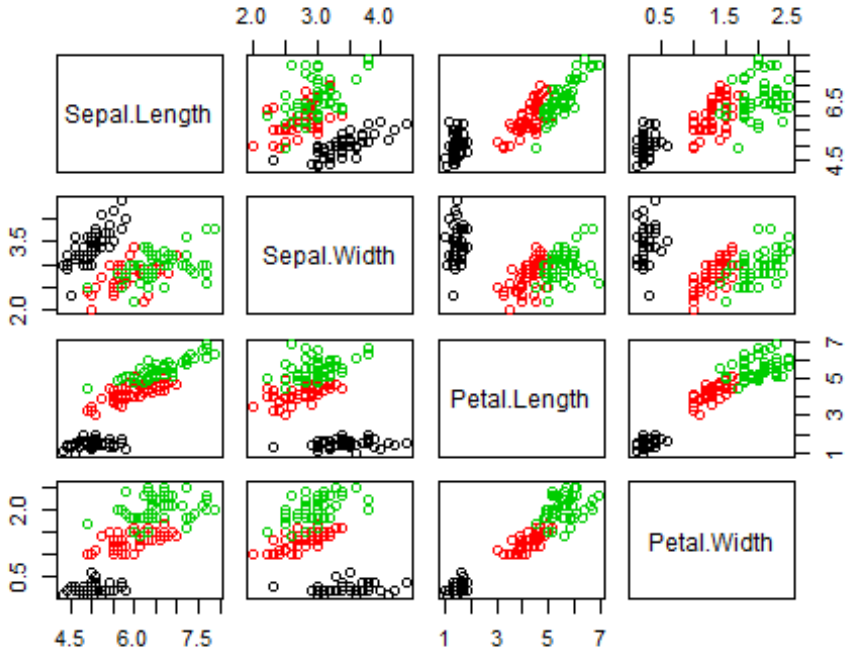
```
##    MURDER      RAPE   ROBBERY   ASSAULT  BURGLARY   LARCENY      AUTO
## 1.0592823 1.3574816 1.7846130 1.4702795 1.8808770 1.5004256 0.9015462

cluster4 = subset(rownames(crime), ct==4)

index4 = match(cluster4, rownames(crime))
colMeans(crime.s[index4, ])

##     MURDER       RAPE    ROBBERY    ASSAULT   BURGLARY    LARCENY
## -0.7531924 -0.8014656 -0.7344997 -0.8523694 -0.8280344 -0.4785343
##       AUTO
## -0.6229141
```

To better understand clustering methods, it is instructive to use examples that you know the actual clusters.

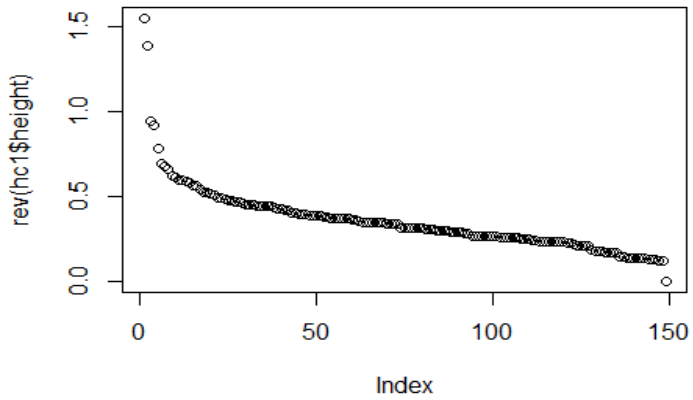
Practice 1: What linkage method can find the best clusters in the data below. Use the Iris data.

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris Setosa, Versicolor, and Virginia.

```
head(iris)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa

mydata1 = iris[,-5]

plot(mydata1, col = iris$Species)
```




```
mydata1.s = scale(mydata1)

# First, we test the single linkage:

hc1 = hclust(dist(mydata1.s), "single")

plot(rev(hc1$height)) # we have 3 clusters
```



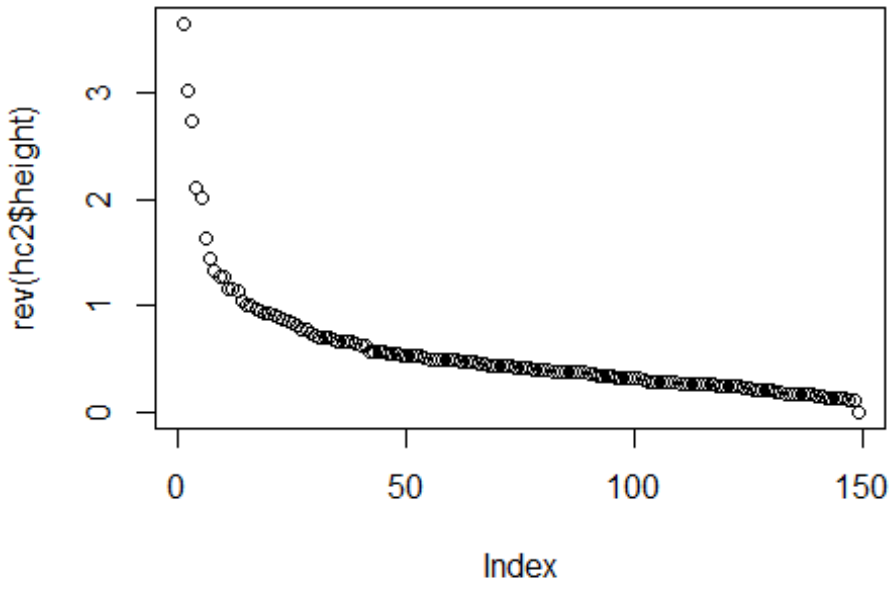
```r
ct1 = cutree(hc1, 3)
table(ct1)
```

```
## ct1
##   1   2   3
##  49   1 100
```

```r
# By looking at the contingency table for H-clusters and true-clusters (given species), we can check the H-clustering outcome's quality.

table(ct1, iris$Species)
```

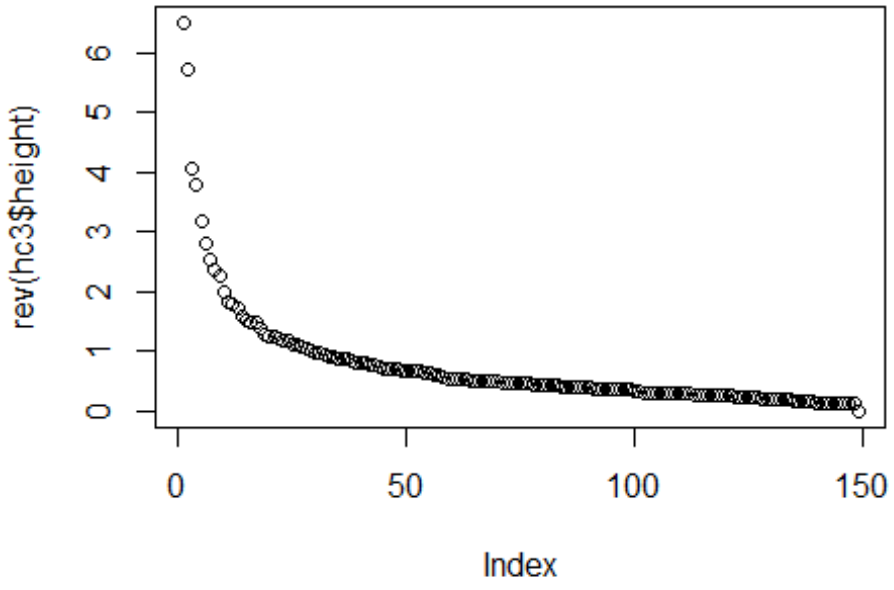
```
##
## ct1 setosa versicolor virginica
##   1     49          0         0
##   2      1          0         0
##   3      0         50        50
```

```r
# Second, we test the average linkage:

hc2 = hclust(dist(mydata1.s), "average")

plot(rev(hc2$height)) # Shows 4 groups, but we know that there are 3 groups in our data.
```



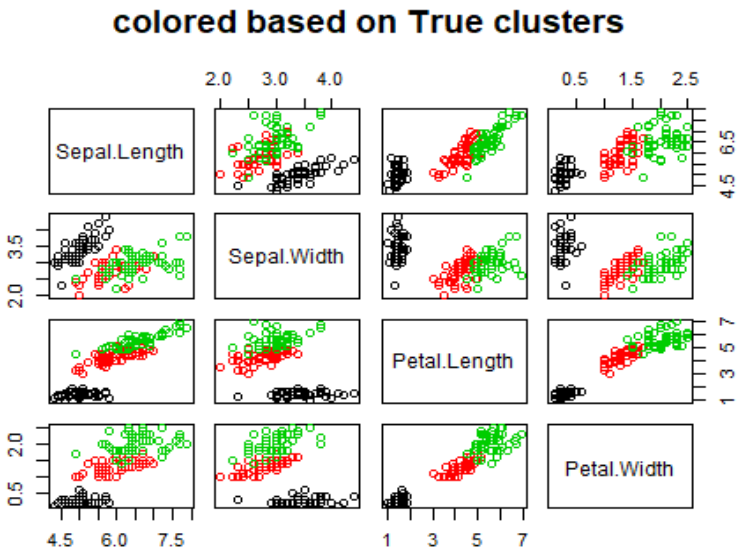

```r
ct2 = cutree(hc2, 3)
table(ct2)
```

```
## ct2
##  1  2  3
## 50 97  3

table(ct2, iris$Species)

##
## ct2 setosa versicolor virginica
##   1     50          0         0
##   2      0         50        47
##   3      0          0         3

# Still not good!

# Third, we test the complete linkage (which is the default):

hc3 = hclust(dist(mydata1.s), "complete")

plot(rev(hc3$height)) # Shows 3 groups
```

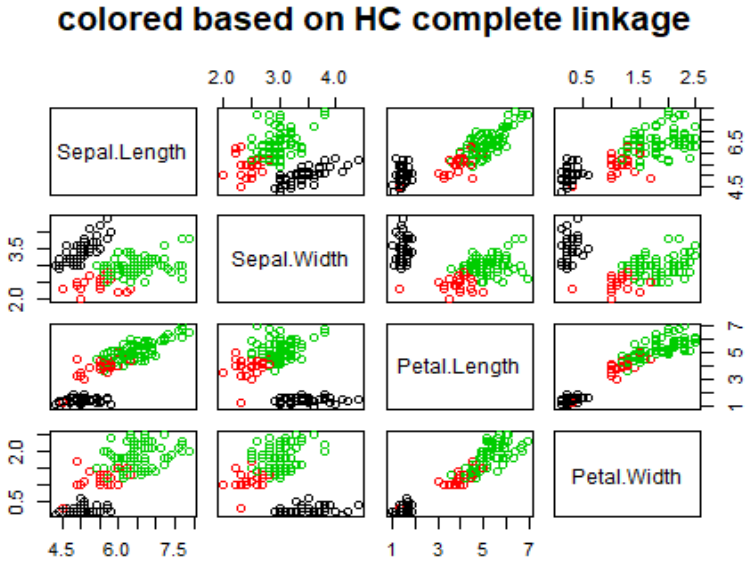



```
ct3 = cutree(hc3, 3)
table(ct3)

## ct3
##  1  2  3
## 49 24 77

table(ct3, iris$Species)
```

```
##
## ct3 setosa versicolor virginica
##   1     49          0         0
##   2      1         21         2
##   3      0         29        48

# The complete linkage performed better than the single and average linkage.

plot(mydata1, col = iris$Species, main = "colored based on True clusters")
```




```
plot(mydata1, col = ct3, main = "colored based on HC complete linkage")
```



Practice 2: Simulation Data

```
# Cluster 1 data are from a bivariate normal distribution with mu1 = (4,8)
# and identity covariance matrix (that means zero correlation).
# Cluster 2 data are from a bivariate normal distribution with mu2 = (8,0)
# and identity covariance matrix.
# Cluster 3 data are from a bivariate normal distribution with mu3 = (0,0)
# and identity covariance matrix.
x = rnorm(50,4,1)
y = rnorm(50,8,1)
cL1 = rep(1, 50)
dmat1 = cbind(cL1,x,y)

x = rnorm(50,8,1)
y = rnorm(50,0,1)
cL2 = rep(2, 50)
dmat2 = cbind(cL2,x,y)

x = rnorm(50,0,1)
y = rnorm(50,0,1)
cL3 = rep(3, 50)
dmat3 = cbind(cL3,x,y)

cdata = rbind(dmat1,dmat2,dmat3)
cdata = data.frame(cdata)
colnames(cdata) <- c("True.Cluster", "x","y")
head(cdata)

##   True.Cluster        x        y
## 1            1 5.329783 9.261687
## 2            1 2.706888 8.069425
## 3            1 4.596425 8.732543
## 4            1 2.041393 8.124605

plot(cdata[,2:3], main = "True Clusters", col = cdata$True.Cluster)
```

```
# If you see a different plot in your machine is because of randomness. I did
not set a seed.

true.cluster.tbl = table(cdata$True.Cluster)
true.cluster.tbl

##
##  1  2  3
## 50 50 50
```

According to the scatterplot only, why are there three clusters?

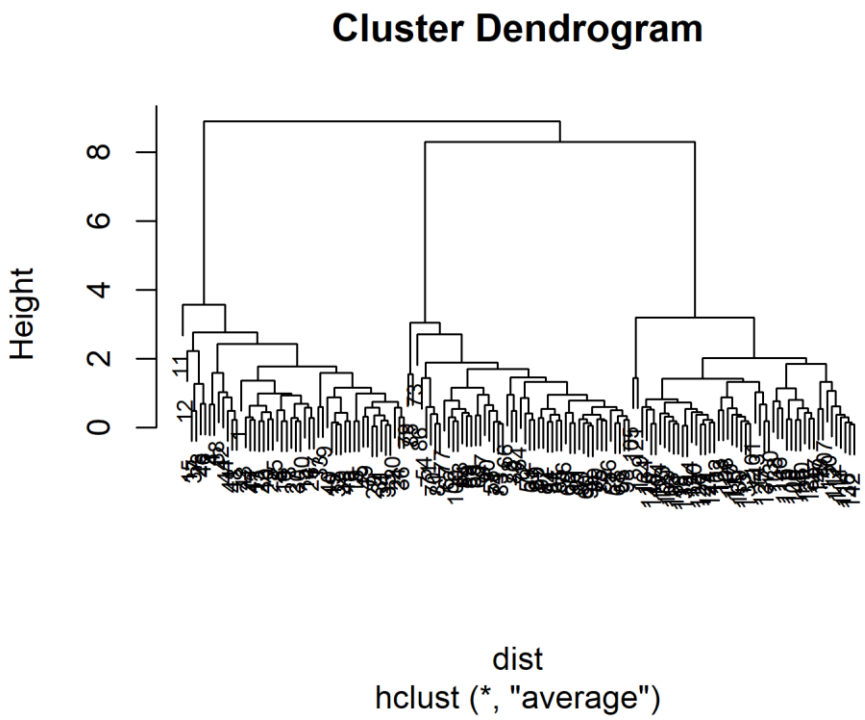
According to the code, what explains why there are three clusters? (Do not perform any clustering algorithms; just read the code.)

Now that you know there are three clusters, you can see how well the clustering methods work.

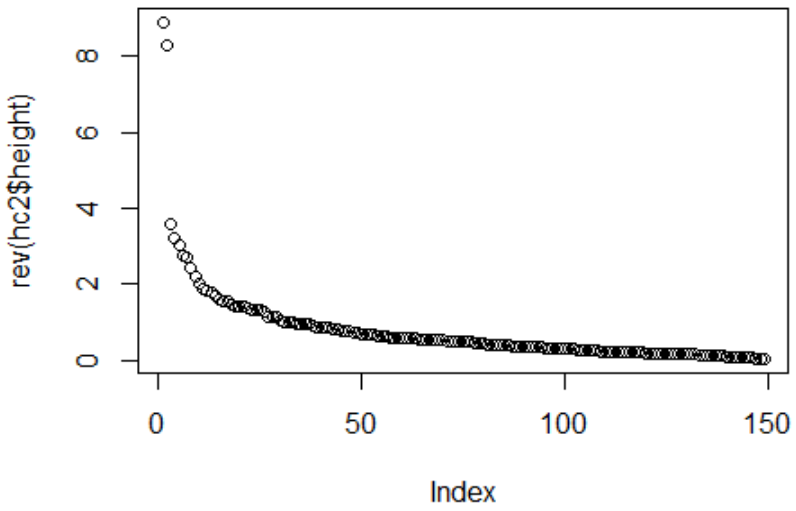
Note: You cannot use the "True.Cluster" variable in the analysis, because in practice, this is another latent variable, and you won't have it in your data set.

How can you see that there are three clusters using the dendrogram of the hclust program? (Use average linkage)

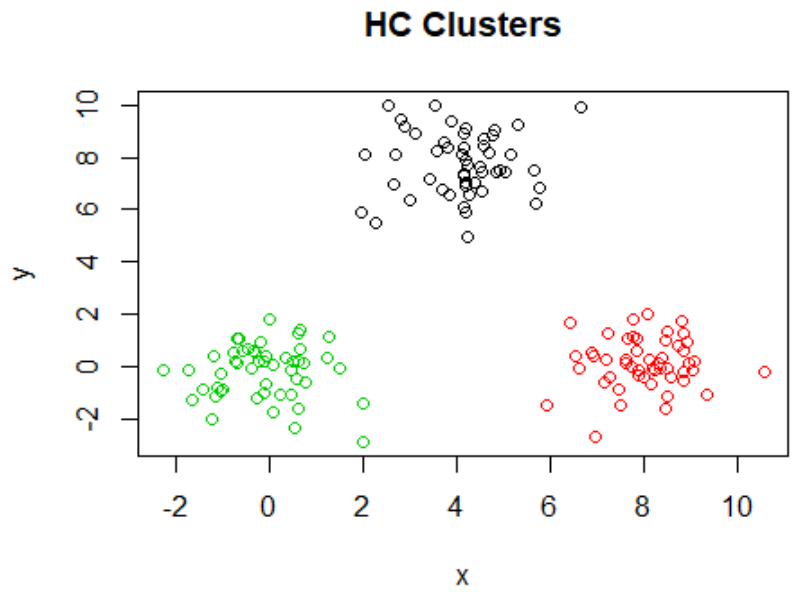
```
dist <- dist(cdata[,2:3])
hc2 <- hclust(dist, "average")
plot(hc2, cex = 0.7)
```



```
plot(rev(hc2$height))
```




```
ct <- cutree(hc2, 3)

# Color code each cluster based on the HC results
plot(cdata[,2:3], col = ct, main = "HC Clusters")
```




```
hc.cluster.tbl = table(ct)
hc.cluster.tbl
```

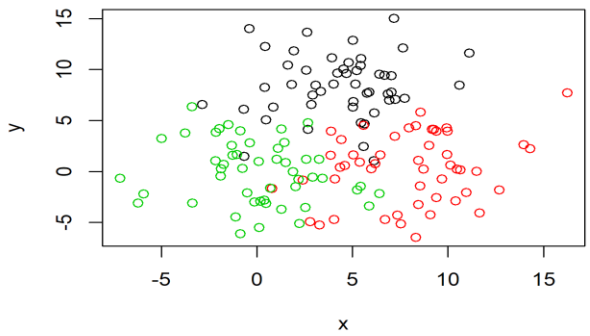
```
## ct
##  1  2  3
## 50 50 50
```

Now change all the standard deviations in the simulation code from 1.0 to 3.0. Here, there are really three clusters as before, but they are poorly separated, so your answers should contrast the poorly-separated case from the well-separated case.

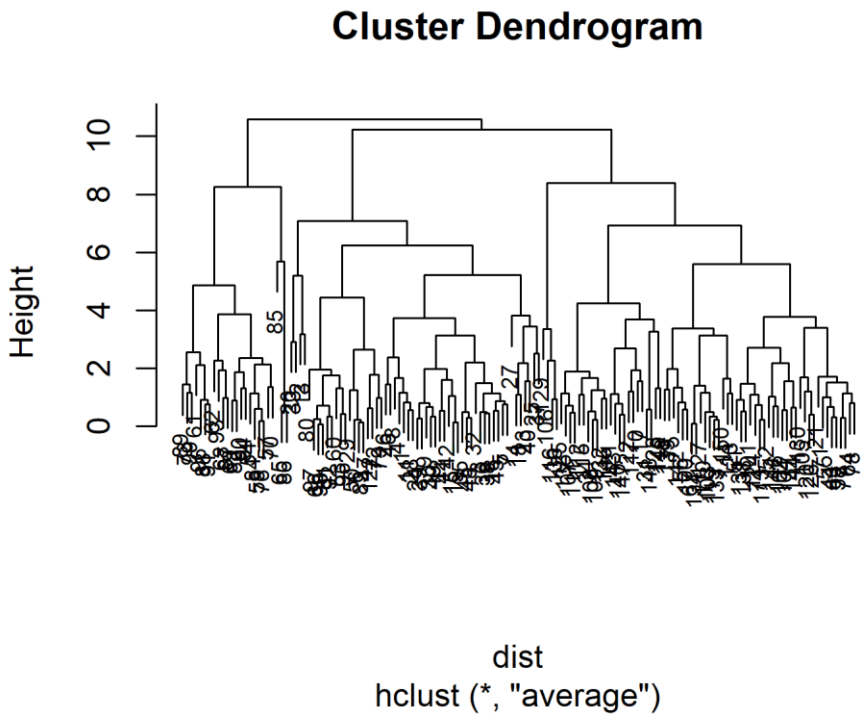
```
x = rnorm(50,4,3)
y = rnorm(50,8,3)
cL1 = rep(1, 50)
dmat1 = cbind(cL1,x,y)

x = rnorm(50,8,3)
y = rnorm(50,0,3)
cL2 = rep(2, 50)
dmat2 = cbind(cL2,x,y)

x = rnorm(50,0,3)
y = rnorm(50,0,3)
cL3 = rep(3, 50)
dmat3 = cbind(cL3,x,y)

cdata = rbind(dmat1,dmat2,dmat3)
cdata = data.frame(cdata)
colnames(cdata) <- c("True.Cluster", "x","y")
head(cdata)
```

```
##   True.Cluster         x         y
## 1            1  3.076790  8.482838
## 2            1 11.117174 11.617221
## 3            1  4.204709  9.657216
## 4            1  4.006982  8.588914
```

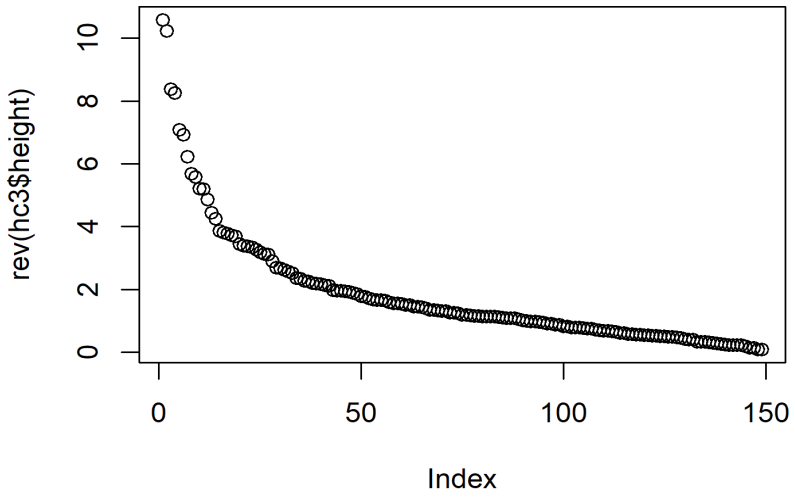
```
#plot(cdata[,2:3])
plot(cdata[,2:3], col=cdata$True.Cluster)
```



```r
dist <- dist(cdata[,2:3])
hc3 <- hclust(dist, "average")
plot(hc3, cex = 0.7)
```




```r
plot(rev(hc3$height))
```



```r
hc3 <- hclust(dist, "average")
ct <- cutree(hc3, 3)
plot(cdata[,2:3], col = ct, main = "HC Clusters")
```


```r
# remember the table of the true clusters
table(cdata$True.Cluster)
```

```
##
##  1  2  3
## 50 50 50
```

```r
# The table of HC clusters
table(ct)
```

```
## ct
##  1  2  3
## 56 70 24
```

```r
# contingency table
tb <- table(ct, cdata$True.Cluster)
tb
```

```
##
## ct   1  2  3
##   1 43 12  1
##   2  7 14 49
##   3  0 24  0
```

```r
chisq.test(tb) #H0: rows and columns are independent
```

```
##
##  Pearson's Chi-squared test
##
## data:  tb
## X-squared = 142.22, df = 4, p-value < 2.2e-16
```

```r
# Larger X-squared value means a better match between the true clusters and H
-clusters. You can test various linkages and see which one has a higher X-squ
ared value. Intuitively, a higher X-squared value means the clustering result
, and the true clusters are more independent of each other (consequently smal
ler p-value).
```