K-Means Clustering

Alireza Sheikh-Zadeh, Ph.D.

K-Means Clustering

The most commonly used implementation of the k-means clustering is the one that tries to find the partition of the n observations (rows of data) into k groups that minimizes the within-group sum of squares (WGSS) over all variables.

$$WGSS = \sum_{j=1}^{q} \sum_{l=1}^{k} \sum_{i \in G_l} (x_{ij} - \overline{x}_j^{(l)})^2$$

Where $\overline{x}_j^{(l)} = \frac{1}{n_i} \sum_{i \in G_l} x_{ij}$ is the mean of the individuals in group G_l on variable j. And, q is the number of variables, and k is the number of clusters.

What is the within-group sum square? If you look at the above formula, all it is doing with WGSS is to looking for clusters that minimize the squared Euclidean distances of the points from centroids. The goal is to partition data into groups to get the minimum within the group sum squares.

For example, if the following observations are grouped into two groups, then WGSS can be computed as follows,

	Obs	X_1	X_2	X
Group 1	1	2	5	1
	2	2	3	3
Group 2	3	6	8	2
	4	4	10	6

Group means:

Group 1: $\bar{x}^{(1)} = \{2,4,2\}$

Group 2: $\bar{x}^{(2)} = \{5,9,4\}$

$$WGSS = \sum_{j=1}^{q}\sum_{l=1}^{k}\sum_{i\in G_l}(x_{ij} - \overline{x}_j^{(l)})^2$$
$$= ((2-2)^2 + (2-2)^2 + (6-5)^2 + (4-5)^2)$$
$$+ ((5-4)^2 + (3-4)^2 + (8-9)^2 + (10-9)^2)$$
$$+ ((1-2)^2 + (3-2)^2 + (2-4)^2 + (6-4)^2) = 16$$

It isn't straightforward to develop distinct clusters that minimize the within-groups sum of squares (WGSS). For example, you have a dataset with 15 observations, and you want to partition those observations into three subsets. How many possible ways can you do that? 2,375,101 (see Table 6.2 in EH Textbook).

The whole point is to find the optimal partition of the data into groups that minimizes the WGSS, but you cannot do it, even if with great computers. Searching through all possible partitions expense huge running time. Instead, we try to develop an approximate solution method to do the grouping more efficiently. The essential heuristic steps in the K-means algorithm are listed on page 176 EH textbook.

Note:

- In K-Means, we put observations (the data frame) into groups. In contrast, in hierarchical clustering (HC), we can put distance matrix into groups. This makes the HC more flexible because any even hypothetical distance matrix (e.g., 1-cor(data)) can be used for grouping.
- K-means clustering is more useful for larger datasets than hierarchical clustering (the main reason is having faster computation time).
- In the K-means algorithm, because of a random initial assignment, sometimes you do get different clustering results even on the same dataset. That is the limitation of this methodology.
- Standardization (scaling) is important, and there are different ways to do that:
  - Converting to z-score (the most common).
  - Dividing by standard deviation
  - Dividing by range
  - Log transformation

Example: Crime Data

```
crime <- read.csv("https://rb.gy/wu8kvo", row.names = "STATE")
crime.s <- scale(crime)
km <- kmeans(crime.s, centers = 4) # Applying kmeans for k=4 clusters
table(km$cluster)

##
##  1  2  3  4
## 10 15 18  7

km$cluster

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

km$tot.withinss

## [1] 135.0884

# Which states are in group 1:
subset(crime.s, km$cluster == 1)

##                MURDER     RAPE       ROBBERY   ASSAULT  BURGLARY   LARCENY
## ALASKA      0.8679081 2.403986 -3.089128e-01 0.7251650 0.0920233 0.9622587
## ARIZONA     0.5317101 0.786830  1.596857e-01 1.0074507 2.4376970 2.4742946
## CALIFORNIA  1.0489378 2.199518  1.843924e+00 1.4632971 1.9597290 1.1413446
## COLORADO   -0.2958542 1.511762  5.275468e-01 0.8139403 1.4875419 1.6970619
## FLORIDA     0.7127398 1.288706  7.222302e-01 2.3719977 1.3134200 1.6106874
## MARYLAND    0.1437893 0.842594  1.901649e+00 1.4722744 0.2499585 0.6976249
## MICHIGAN    0.4799873 1.223648  1.559822e+00 0.6314022 0.5336870 0.6718641
## NEVADA      2.1609773 2.171636  2.252532e+00 1.4333729 2.6851212 2.1232863
## NEW YORK    0.8420467 0.340718  3.944693e+00 1.0752790 1.0084177 0.1525150
## OREGON     -0.6579136 1.316588  9.055042e-05 0.7540918 0.7966041 1.1500234
##                    AUTO
## ALASKA      1.94304471
## ARIZONA     0.32045392
## CALIFORNIA  1.47870866
## COLORADO    0.51487525
## FLORIDA    -0.13509180
## MARYLAND    0.26357534
## MICHIGAN    0.86855661
## NEVADA      0.93939630
## NEW YORK    1.90426386
## OREGON      0.05881245
```

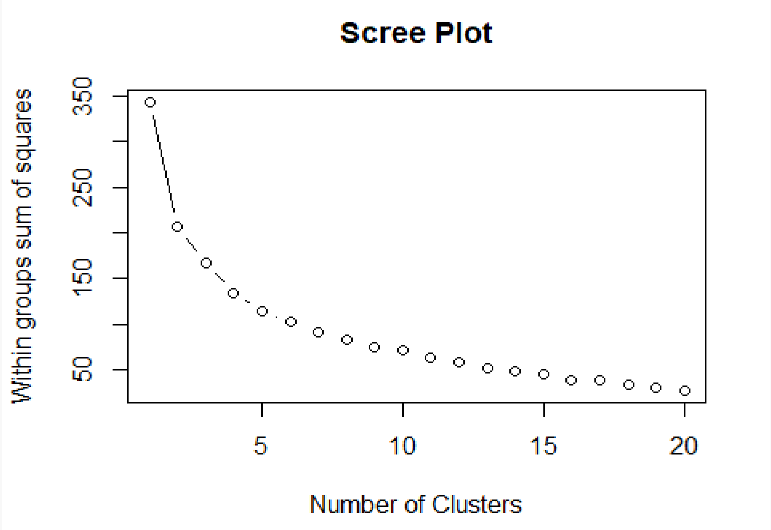
```
# To find a more reliable result, increase nstart.
km <- kmeans(crime.s, centers = 4, nstart = 10)
table(km$cluster)
```

```
##
##  1  2  3  4
##  7 15 18 10
```

How to decide the number of clusters?

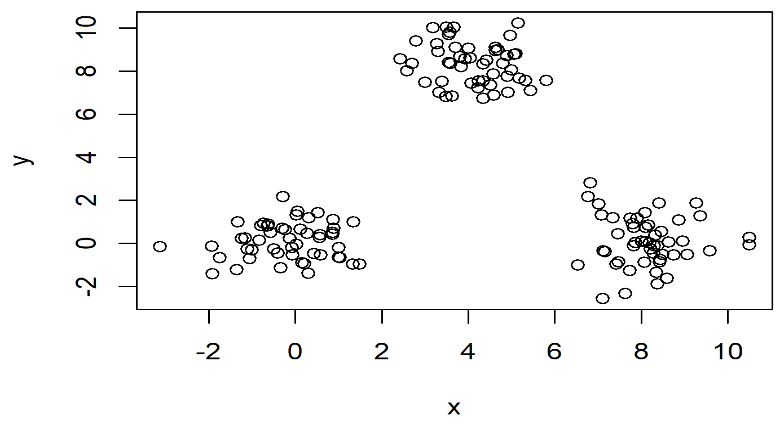
The next chunk of code creates a scree plot for deciding the appropriate number of clusters.

```
plot.wgss = function(mydata, maxc) {
  wss = numeric(maxc)
  for (i in 1:maxc)
    wss[i] = kmeans(mydata, centers=i, nstart = 10)$tot.withinss
  plot(1:maxc, wss, type="b", xlab="Number of Clusters",
  ylab="Within groups sum of squares", main="Scree Plot")
}

plot.wgss(crime.s, 20) # Elbow test.
```



Example: Simulation Data

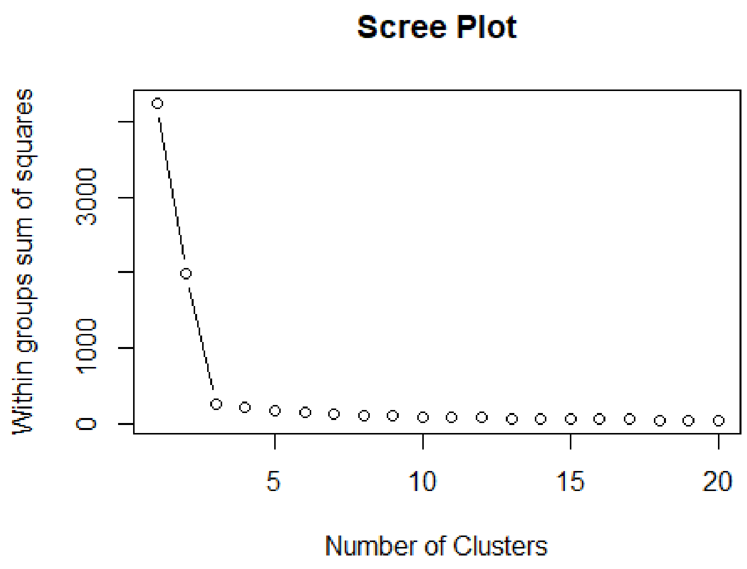
```
# Cluster 1 data are from a bivariate normal distribution with mu1 = (4,8)
# and identity covariance matrix (that means zero correlation).
# Cluster 2 data are from a bivariate normal distribution with mu2 = (8,0)
# and identity covariance matrix.
# Cluster 3 data are from a bivariate normal distribution with mu3 = (0,0)
# and identity covariance matrix.
x = rnorm(50,4,1) # We did not use mvrnorm because we assumed x and y are ind
ependent of each other.
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")
plot(cdata[,2:3])
```



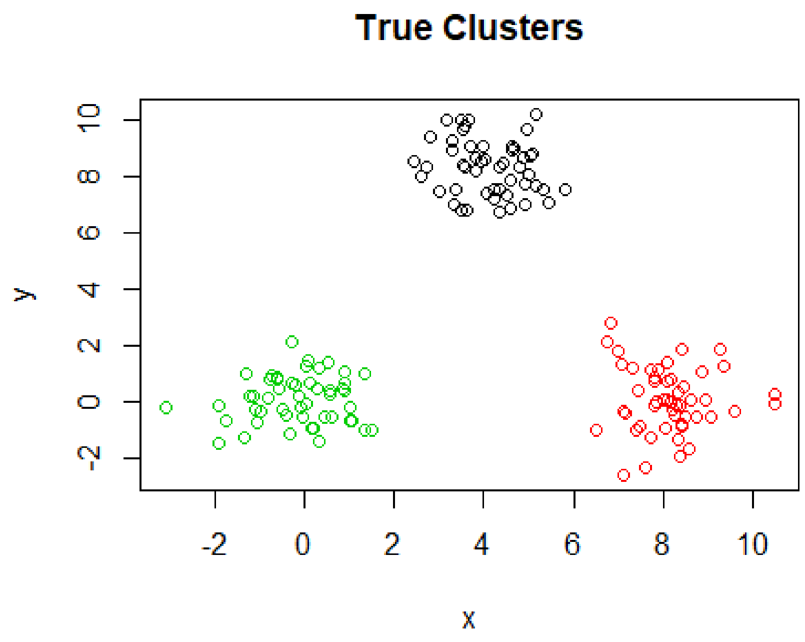
```
# So we know the true ground.
```

- How many number of cluster is appropriate?

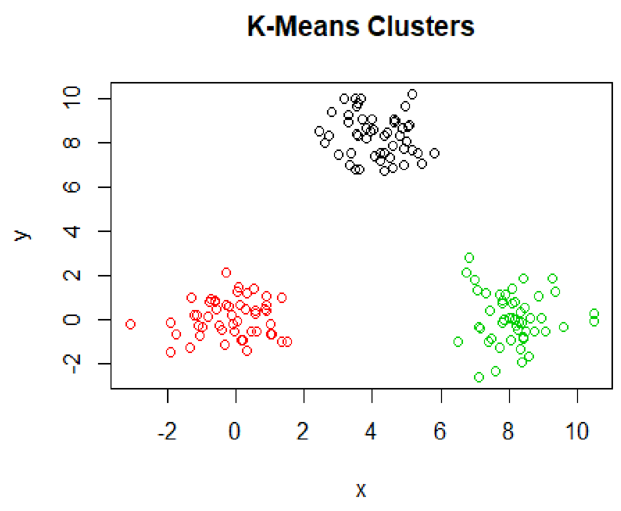
```
plot.wgss(cdata[,2:3], 20)
```



```
km2 <- kmeans(cdata[,2:3], 3)


# data is color-coded based on the true clusters (from simulation)
plot(cdata[,2:3], col = cdata$True.Cluster, main = "True Clusters")
```



```r
# data is color coded based on the Kmeans clusters
plot(cdata[,2:3], col = km2$cluster, main = "K-Means Clusters")
```



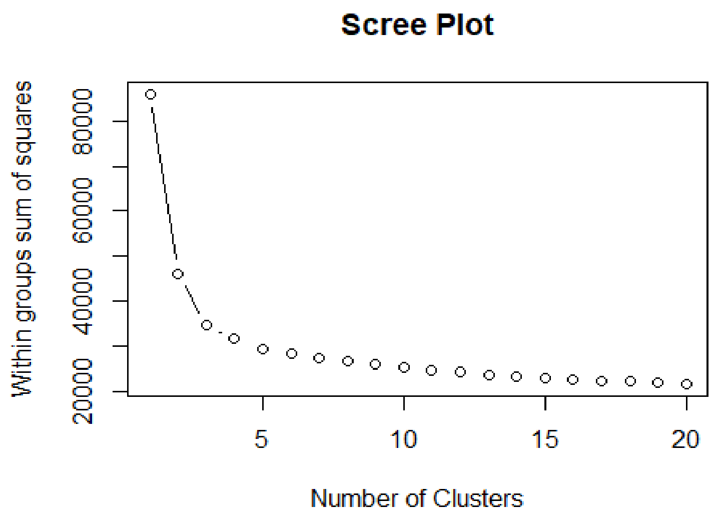
Student activity: Change the stdev of the simulation code from 1 to 3, and redo the k-means clustering. Does K-means detects true clusters?

Example: TTU Students Evaluation Data. Comparing K-means vs hierarchical clustering.

```r
data <- read.csv("https://bit.ly/3ed5Bia")
evals <- data[,3:18]     # select variables to use
evals <- na.omit(evals) # do listwise cleaning for missing values
# Do we need to standardize this data?
```

- What is the appropriate number of clusters?

```r
plot.wgss(evals, 20)
```



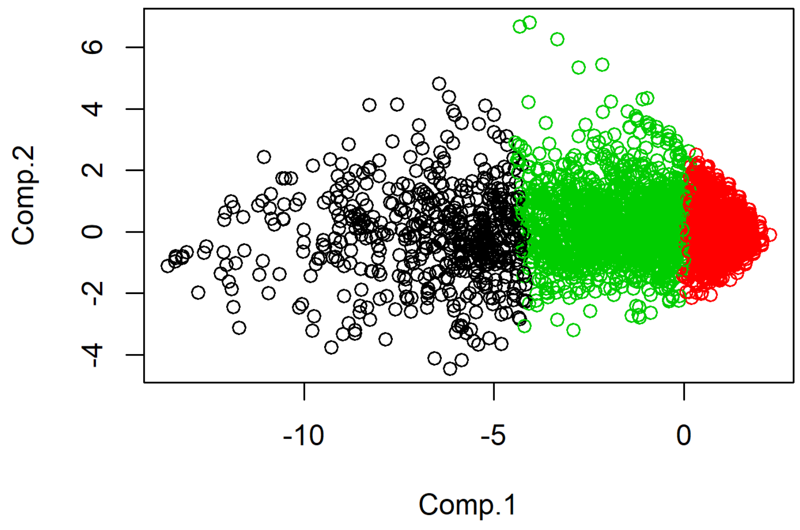
Let's pick k=3 and apply k-means.

```
km3 <- kmeans(evals, 3, nstart = 10)
table(km3$cluster)

##
##    1    2    3
##  635 4479 2293
```

What do these three clusters mean?

In student evaluation data, what do these three clusters mean?

```
# Principal components may help (it's better than MDS because we can attach

names to PCs)
pca <- princomp(evals)
# pca$loadings[,1:3] # how you name pc1, pc2, and pc3?
plot(pca$scores[, 1:2], col = km3$cluster)
```



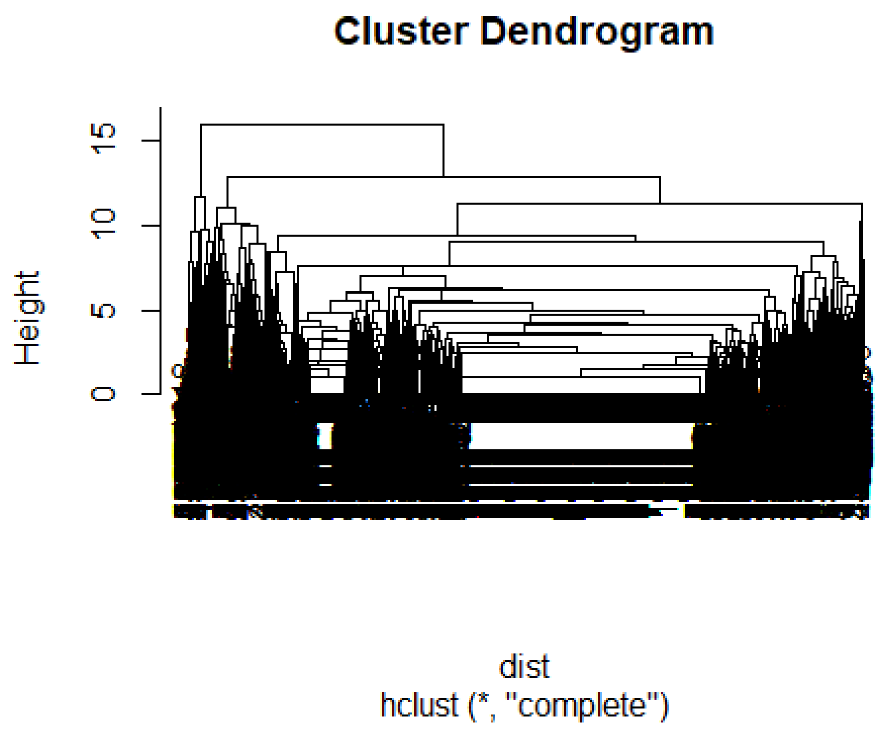
```
# It seems like PC1 (which is interpreted as the overall rating) explains the
discrimination between the clusters.
```

```
# Another way to find out what these clusters mean is to look at the cluster
centroids.
km3$centers

##     RESP_1   RESP_2   RESP_3   RESP_4   RESP_5   RESP_6   RESP_7   RESP_8
## 1 2.507087 3.212598 2.425197 3.074016 3.040945 3.118110 2.332283 2.636220
## 2 4.881223 4.826747 4.890824 4.947533 4.953561 4.955794 4.857334 4.924760
## 3 3.986917 4.044919 3.952464 4.225469 4.255124 4.299608 3.853903 4.038814
##     RESP_9  RESP_10  RESP_11  RESP_12  RESP_13  RESP_14  RESP_15  RESP_16
## 1 2.507087 3.248819 2.289764 2.540157 2.634646 2.992126 2.792126 2.622047
## 2 4.890377 4.978790 4.854878 4.845055 4.850413 4.927662 4.895736 4.869614
## 3 3.920628 4.394679 3.838203 3.802878 3.810292 4.053205 3.973397 3.873964
```

We see cluster 2 contains students who rated the instructor and course high (as highlighted). And cluster 1 explains the lowest rating. As it is clear, the numerical label of the clusters is arbitrary.

Now, let's do hierarchical clustering.

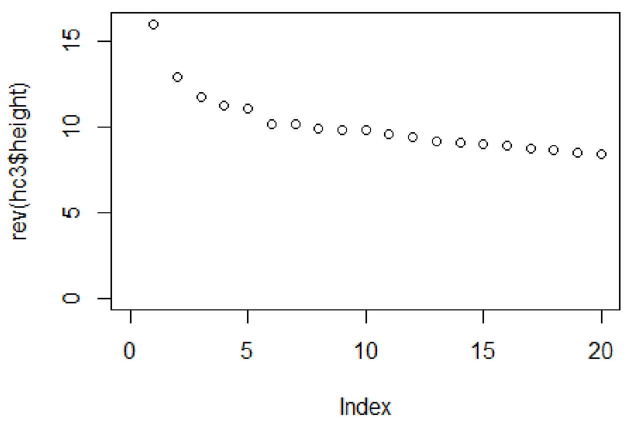
```
dist <- dist(evals)
hc3 <- hclust(dist, "complete")
plot(hc3) # How many clusters can you pick?
```



```
# For large datasets, dendrograms are junk.
```

- You remember how to make scree plot for h-clustering.

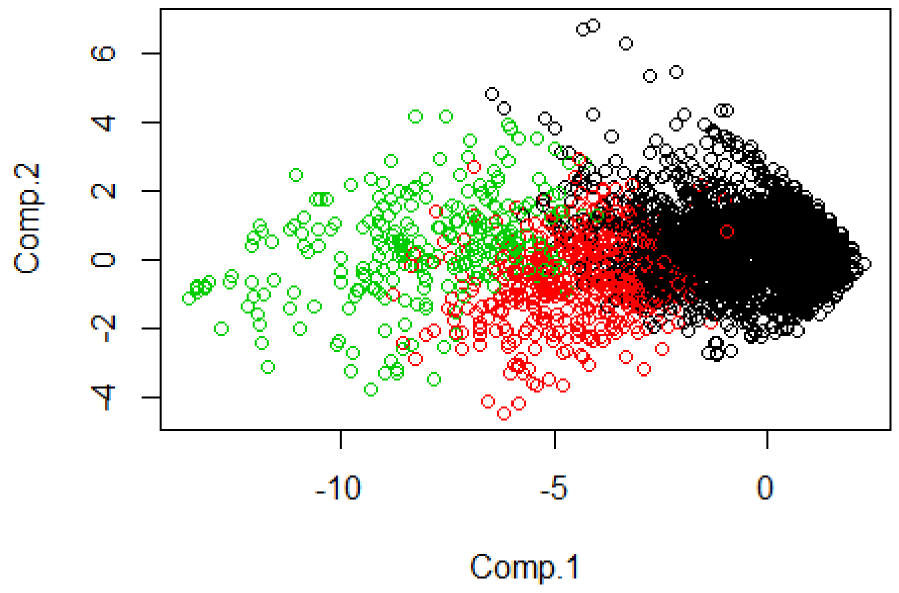
```
plot(rev(hc3$height), xlim = c(0,20))
```



```
# 2 or 3?
```

We repeat the principal component plotting to see the difference with k-means.

```
hc3.clust <- cutree(hc3, 3)
plot(pca$scores[, c(1:2)], col = hc3.clust)
```



```
# We can check the centroids of the clusters as well, the same as k-means.
colMeans(subset(evals, hc3.clust ==1))
```

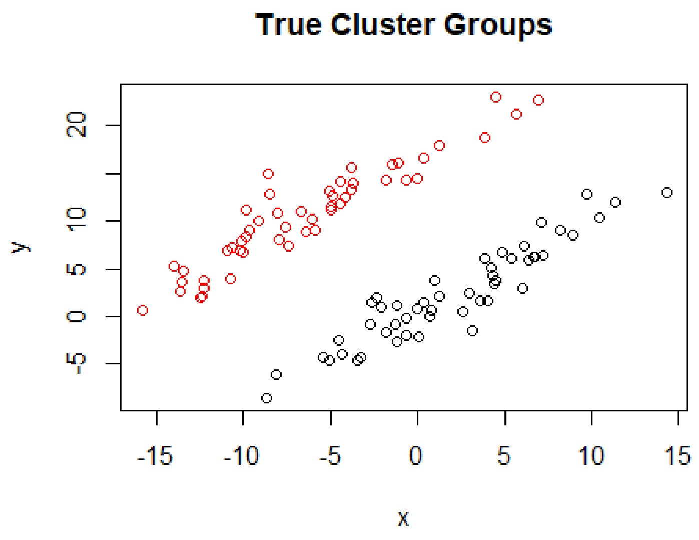
```
##    RESP_1   RESP_2   RESP_3   RESP_4   RESP_5   RESP_6   RESP_7   RESP_8 
## 4.614901 4.591752 4.612295 4.742143 4.755787 4.760693 4.564464 4.663345 
##    RESP_9  RESP_10  RESP_11  RESP_12  RESP_13  RESP_14  RESP_15  RESP_16 
## 4.605396 4.802238 4.545608 4.521999 4.523992 4.663345 4.623180 4.569523

colMeans(subset(evals, hc3.clust ==2))

##    RESP_1   RESP_2   RESP_3   RESP_4   RESP_5   RESP_6   RESP_7   RESP_8 
## 3.157718 3.538591 3.083893 3.456376 3.427852 3.592282 2.867450 3.233221 
##    RESP_9  RESP_10  RESP_11  RESP_12  RESP_13  RESP_14  RESP_15  RESP_16 
## 3.048658 3.796980 3.122483 3.348993 3.436242 3.550336 3.347315 3.233221

colMeans(subset(evals, hc3.clust ==3))

##    RESP_1   RESP_2   RESP_3   RESP_4   RESP_5   RESP_6   RESP_7   RESP_8 
## 2.125000 3.031250 2.031250 2.805556 2.812500 2.920139 2.052083 2.246528 
##    RESP_9  RESP_10  RESP_11  RESP_12  RESP_13  RESP_14  RESP_15  RESP_16 
## 2.180556 2.958333 1.694444 1.878472 2.003472 2.534722 2.291667 2.170139
```

It seems k-means gave us more reasonable clusters. Remember, since we don't have a ground truth, we can't say which one is necessarily better.

Example: Oddly shaped data

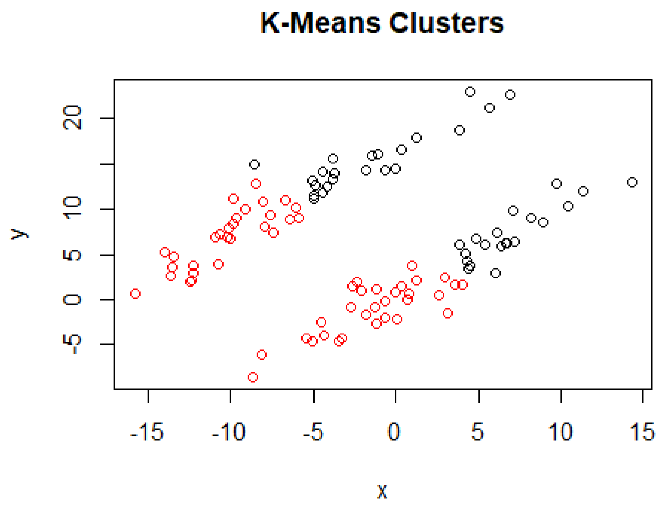
While k-means clustering works well with spherical clusters as in the previous example, hierarchical clustering can work much better than the k-means clustering for oddly-shaped clusters. Consider the following data.

```
## Example with elongated (exhibiting high correlation) multinormal clusters.
# Here, there are two true groups, with mean vectors (8,8) and (-2,14), with
# identical covariance matrices:
# 37 35
# 35 37 (correlation = 35/(37^.5 * 37^.5) = .946.
library(MASS)

set.seed(123)
dmat1 = mvrnorm(50, c(2, 2), matrix(c(30, 28, 28, 30), nrow = 2))
dmat2 = mvrnorm(50, c(-5, 12), matrix(c(30, 28, 28, 30), nrow = 2))
cL1 = rep(1, 50)
cL2 = rep(2, 50)
dmat1 = cbind(cL1,dmat1)
dmat2 = cbind(cL2,dmat2)
cdata = rbind(dmat1,dmat2)
cdata = data.frame(cdata)
colnames(cdata) <- c("True.Cluster", "x","y")
head(cdata)

##   True.Cluster         x          y
## 1            1 -1.271572 -0.7649352
## 2            1  0.789003  0.7319095
## 3             1 10.436772 10.3510307
## 4             1  1.011097  3.7483016
## 5             1  2.922007  2.4704648
## 6             1  9.719437 12.7523782

plot(cdata[,2:3], col = cdata[,1], main="True Cluster Groups")
```



We know that we have two clusters in this data. First, we apply k-means clustering.

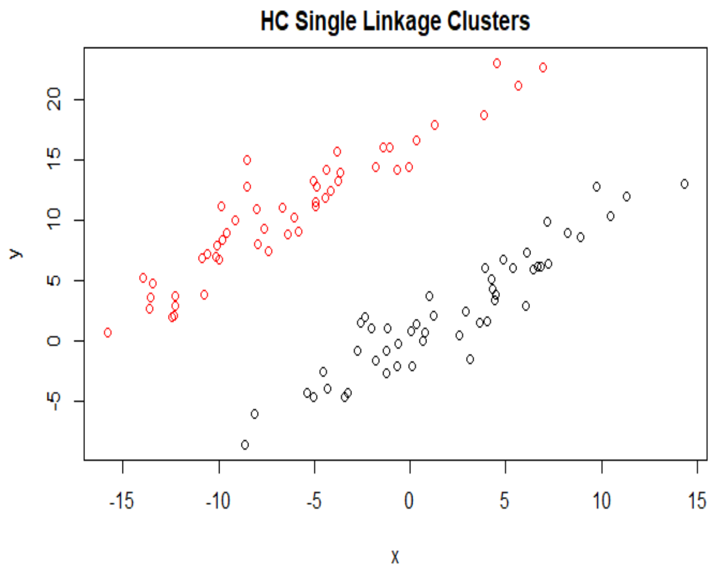
```
km4 <- kmeans(cdata[,2:3], 2)
plot(cdata[,2:3], col = km4$cluster, main="K-Means Clusters")
```



```
# This is the weakness of kmeans clustering. Kmeans doesn't work well for non
-spherical clusters.
```

How about hierarchical clustering?

```
dist2 <- dist(cdata[,2:3])
hc4 <- hclust(dist2, "single")
plot(cdata[,2:3], col = cutree(hc4, 2))
```


HC Single Linkage Clusters
y
x