Model-Based Clustering

Alireza Sheikh-Zadeh, Ph.D.

Model-based Clustering

If data comes from the same distribution, it belongs to the same cluster. Model-based clustering is an effective and flexible procedure because it can model different data patterns.

The big picture is if we have every observation coming from a specific distribution, that distribution has lots of parameters. The parameters are different mean vectors for different groups, other elements of the covariance matrices, and the probability p (what proportion of data belong to each group). What we need to do is to estimate those parameters.

Finite mixture densities often provide a sensible statistical model for the clustering process, and cluster analyses based on *finite mixture models* are also known as model-based clustering methods.

Mixture distribution (page 186):

$$f(x) = \sum_{j=1}^{c} p_j \, g_j(x|\hat{\theta})$$

- c is the number of clusters
- g is the distribution like multivariate normal distribution with mean and covariance matrix. In theory, g can be any distribution. The model-based clustering in R assumes that the combined data distribution is a mixture of multivariate normal distributions. It has its own limitation; for example, what if the data is not normally distributed.
- p refers to the proportion of data comes from g distribution. p is the key parameter for assigning items into groups.

I provided a step-by-step numerical example (one dimensional) about how the mixture model works in this document's appendix. If you are interested, please don't miss it. I'll be glad to clarify it more if you request.

Other notes:

- In the data, we may see the data is distributed as a mixture of multiple multivariate normal distributions. In that case, model-based clustering will be the best choice.
- To decide the number of clusters, we measure the BIC (Bayes Inf. Criteria), which is based on the trade-off between the simplicity (less number of parameters, i.e., less number of groups) of the model and the quality of the model (larger loglikelihood, i.e., the better fit). The following equation shows how to measure the BIC.

BIC = Loglikelihood - 0.5(number of parameters)(log n)

In the appendix, I measured the BIC for our numerical example. In R, the Mclust function measures the BIC for the different number of clusters. The larger BIC is better.

- You can set the number of clusters based on your need or let the model-based clustering algorithm propose the best number of clusters.
- Finite mixture modeling can be seen as a form of latent variable analysis, with *subpopulation* being a latent categorical variable and the latent classes described by the different components of the mixture density. Consequently, cluster analysis based on such models is also often referred to as *latent class cluster analysis*.

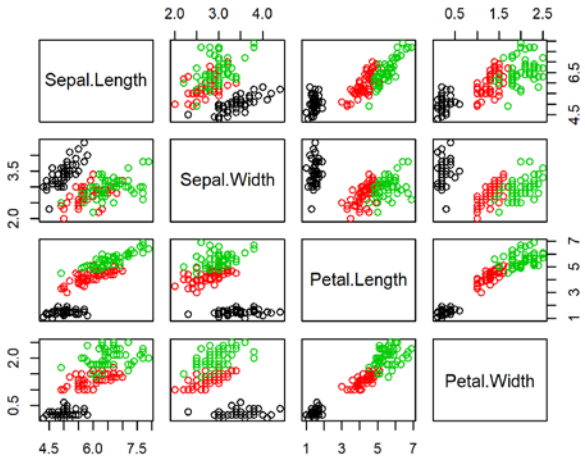
Example: The Iris data

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 *virginica*.

```
data(iris)
str(iris)
```

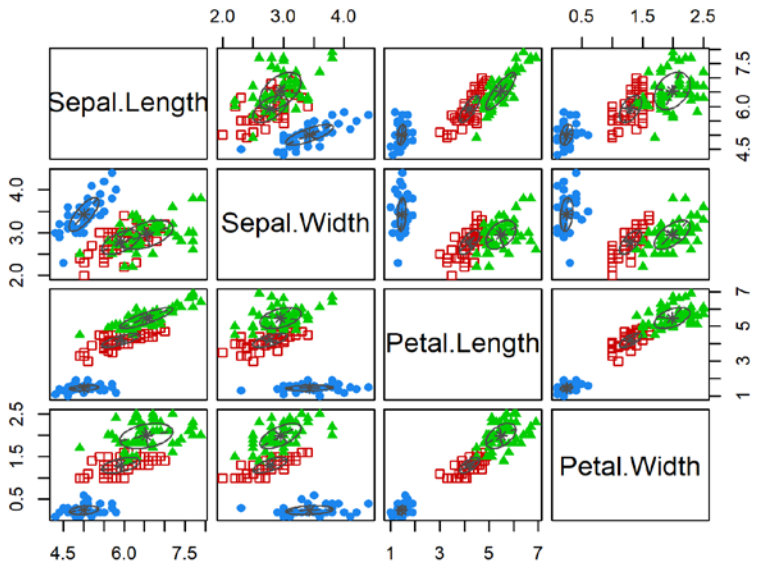
```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1
...
```

```
plot(iris[,1:4], col = iris$Species) # color-coded by true clusters
```



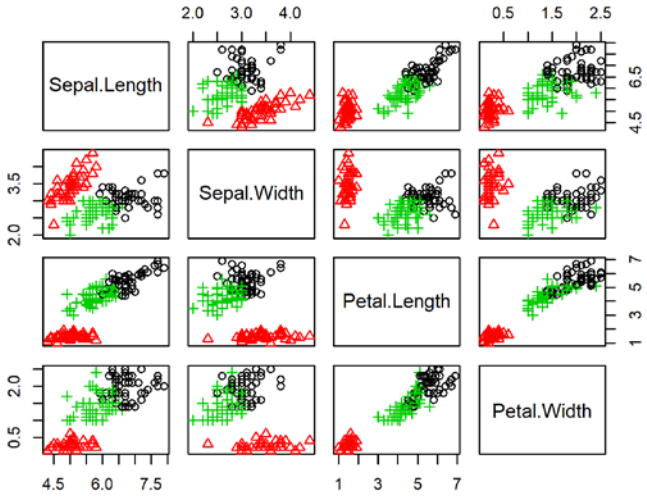

```
# first install mclust package
library(mclust)
```

```r
# Since we know there are 3 species, we pick the number of clusters = 3
mc <- Mclust(iris[,1:4], 3)

table(mc$classification)

##
##  1  2  3
## 50 45 55

table(mc$classification, iris$Species)

##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         45         0
##   3      0          5        50

plot(mc, what = "classification")
```

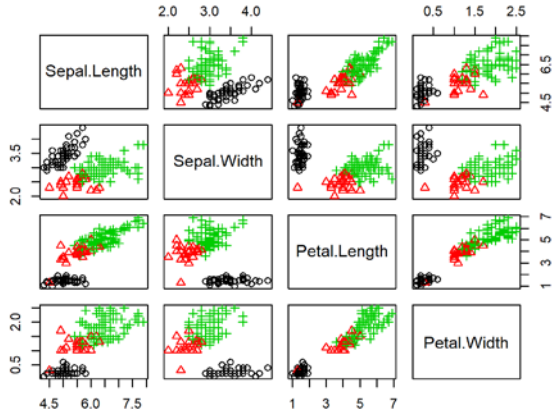



```r
summary(mc)

## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 3 components:
##
## loglikelihood n df       BIC       ICL
##       -186.074 150 38 -562.5522 -566.4673
##
## Clustering table:
```

```
##  1  2  3
## 50 45 55

# We can compare the result with k-means and H-clustering
km <- kmeans(scale(iris[,1:4]), 3, nstart = 10)
table(km$cluster, iris$Species)

##
##     setosa versicolor virginica
##   1      0         11        36
##   2     50          0         0
##   3      0         39        14

plot(iris[,1:4], col = km$cluster, pch = km$cluster)
```




```
hc <- hclust(dist(scale(iris[,1:4])))
hc.clust <- cutree(hc, 3)
table(hc.clust, iris$Species)

##
## hc.clust setosa versicolor virginica
##        1     49          0         0
##        2      1         21         2
##        3      0         29        48

plot(iris[,1:4], col = hc.clust, pch = hc.clust)
```

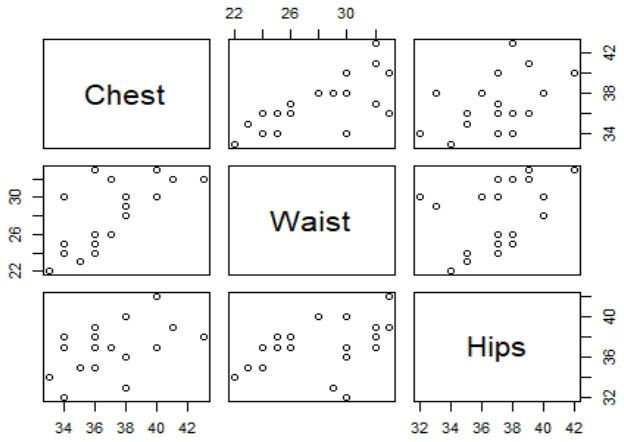



```
# By looking at the contingency tables, We can admit that the Mclust caught
the true clusters better than the kmeans and h-clust.
```

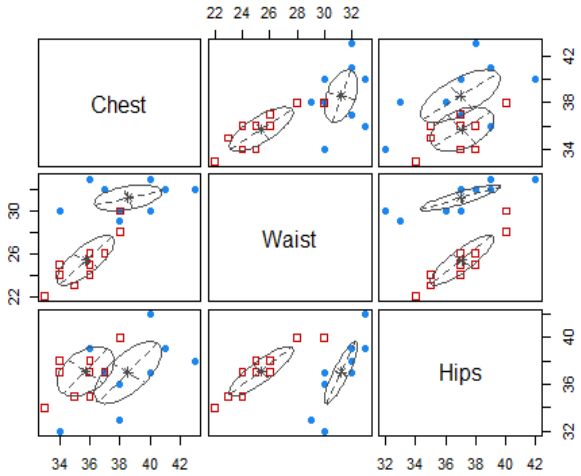
Example: The body measurement data

What is the suggested number of clusters by Mclust?

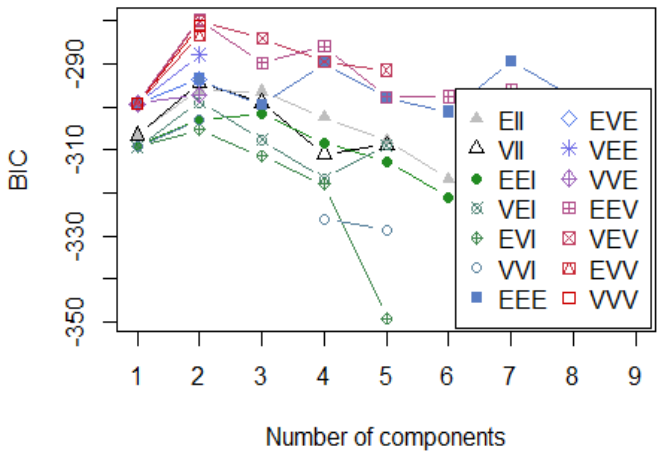
```
measure <-read.csv("https://bit.ly/3habmxj")
head(measure)

##   Chest Waist Hips Gender
## 1    34    30   32      1
## 2    37    32   37      1
## 3    38    30   36      1
## 4    36    33   39      1
## 5    38    29   33      1
## 6    43    32   38      1

plot(measure[,1:3])
```



```r
# we have to expected clusters; male and female. Mclust optimally recognizes
that
mc <- Mclust(measure[,1:3])

table(mc$classification, measure$Gender)

##
##      0  1
##   1  0  9
##   2 10  1

plot(mc, what = "classification")
```



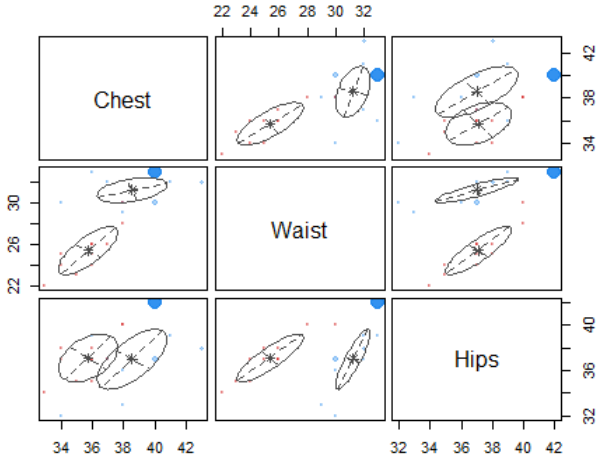

```r
# The number of clusters is based on maximum BIC
plot(mc, what = "BIC")
```



```
mc$modelName
```

```
## [1] "EEV"
```

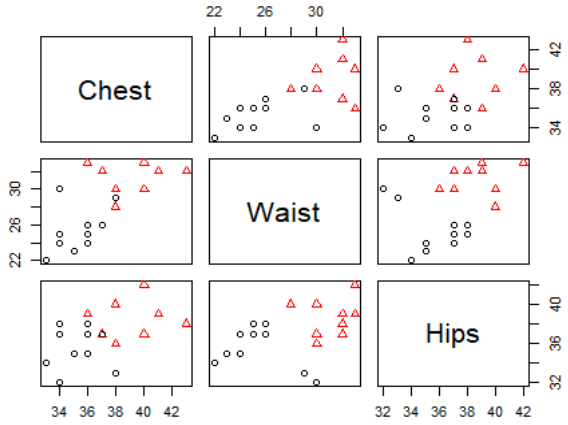
```
# Find uncertain points, darker and larger points are more uncertain
plot(mc, what = "uncertainty")
```




```
# Comparing Mclust to kmeans
km <- kmeans(scale(measure[,1:3]), 2)
table(km$cluster, measure[,4]) # km results vs true clusters
```

```
##
##     0 1
##   1 9 2
##   2 1 8
```

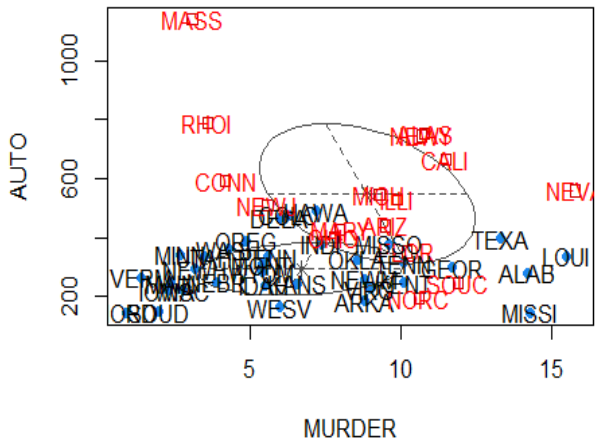
```
# We see that Mclust is more accurate.

plot(measure[,1:3], col = km$cluster, pch = km$cluster)
```



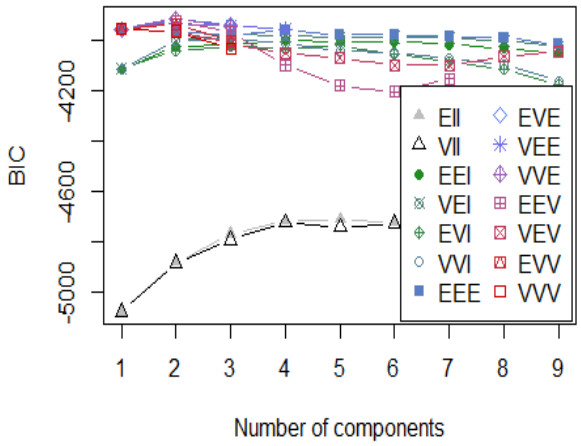
```
# you can try hierarchical as well.
```

Example: The crime data

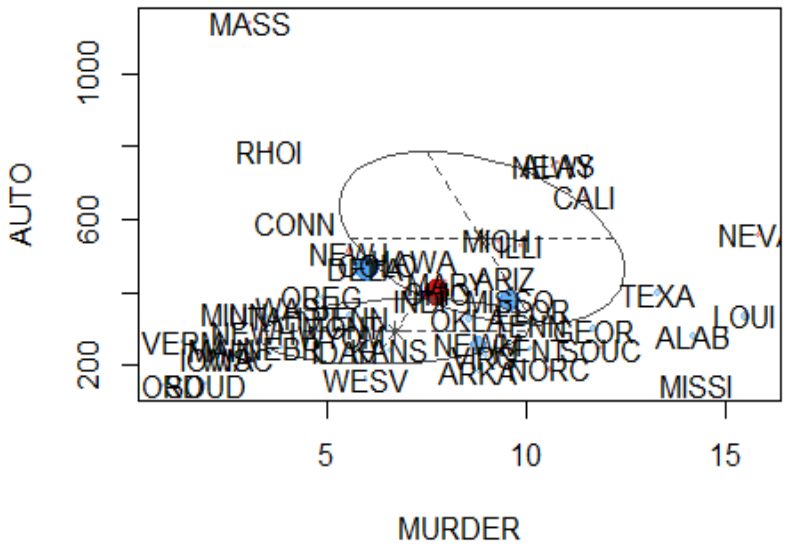
```
crime <- read.csv("https://rb.gy/wu8kvo", row.names = "STATE")
library(mclust)
mc <- Mclust(crime)
summary(mc)

## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVE (ellipsoidal, equal orientation) model with 2 components:
##
## loglikelihood n df       BIC       ICL
##       -1859.765 50 50 -3915.131 -3918.316
##
## Clustering table:
##  1  2
## 34 16

plot(mc, what = "classification" , dimens = c(1,7)) # for only two variables
text(mc$data[,c(1,7)], labels = abbreviate(rownames(crime)), col = mc$classif
ication)
```




```
# check the BIC plot if you think the number of clusters doesn't make sense
plot(mc, what = "BIC") # 3 cluster is the is the second best option
```




```
plot(mc, what = "uncertainty", dimens = c(1,7))
text(mc$data[,c(1,7)], labels = abbreviate(rownames(crime)))
```




```
clust.data = cbind(rownames(crime), mc$classification, mc$uncertainty)
clust.data[order(mc$uncertainty),]
```

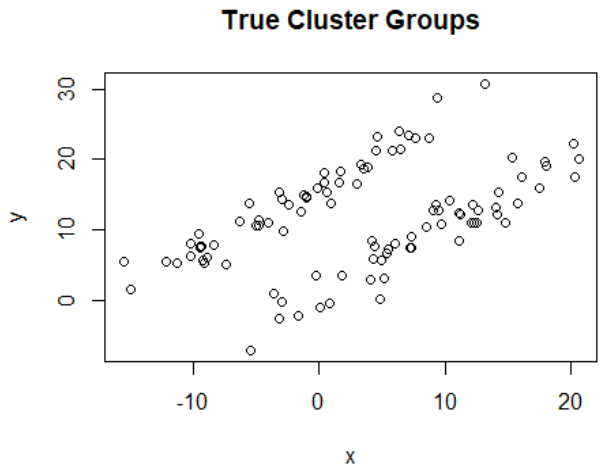
```
##                [,1]             [,2] [,3]
## ALASKA         "ALASKA"         "2"  "0"
## MASSACHUSETTS  "MASSACHUSETTS"  "2"  "0"
## NEW YORK       "NEW YORK"       "2"  "0"
## RHODE ISLAND   "RHODE ISLAND"   "2"  "0"
```

```
## MARYLAND       "MARYLAND"       "2"  "9.95455394781075e-08"
## SOUTH CAROLINA "SOUTH CAROLINA" "2"  "1.60553232486649e-07"
## NORTH DAKOTA   "NORTH DAKOTA"   "1"  "2.2473706213777e-06"
…

…

## COLORADO       "COLORADO"       "1"  "0.00555882524038454"
## OKLAHOMA       "OKLAHOMA"       "1"  "0.00633780181816612"
## ALABAMA        "ALABAMA"        "1"  "0.00786359556206728"
## PENNSYLVANIA   "PENNSYLVANIA"   "1"  "0.00809086256118285"
## GEORGIA        "GEORGIA"        "1"  "0.00926222829023071"
## NEW JERSEY     "NEW JERSEY"     "2"  "0.00974075475321445"
## MAINE          "MAINE"          "1"  "0.0107988180443978"
## INDIANA        "INDIANA"        "1"  "0.0109803227961517"
## LOUISIANA      "LOUISIANA"      "1"  "0.0292316886013233"
## NEW MEXICO     "NEW MEXICO"     "1"  "0.150466857007152"
## MISSOURI       "MISSOURI"       "1"  "0.268821187876607"
## DELAWARE       "DELAWARE"       "1"  "0.329529963159755"
## OHIO           "OHIO"           "2"  "0.438829514994426"
```

Example: Simulation I

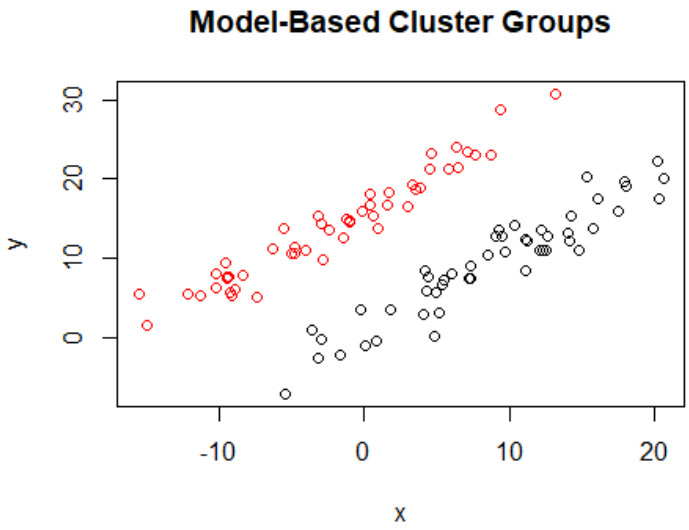
```
set.seed(12345)
## Example with elongated (exhibiting high correlation) multinormal clusters.
# Here, there are two true groups, with mean vectors (8,8) and (-2,14), with
# identical covariance matrices:
# 37 35
# 35 37 (correlation = 35/(37^.5 * 37^.5) = .946.
a = rnorm(50,8,6)
b = rnorm(50,0,1)
x = a-b
y = a+b
cL1 = rep(1, 50)
dmat1 = cbind(cL1,x,y)
a = rnorm(50,6,6)
b = rnorm(50,8,1)
x = a-b
y = a+b
cL2 = rep(2, 50)
dmat2 = cbind(cL2,x,y)
cdata = rbind(dmat1,dmat2)
cdata = data.frame(cdata)
colnames(cdata) <- c("True.Cluster", "x","y")
head(cdata)

##   True.Cluster         x         y
## 1            1 12.053559 10.972787
## 2            1 10.309103 14.204489
## 3            1  7.290590  7.397770
## 4            1  4.927354  5.630680
```

```
## 5            1 12.306301 10.964348
## 6            1 -3.185690 -2.629782

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

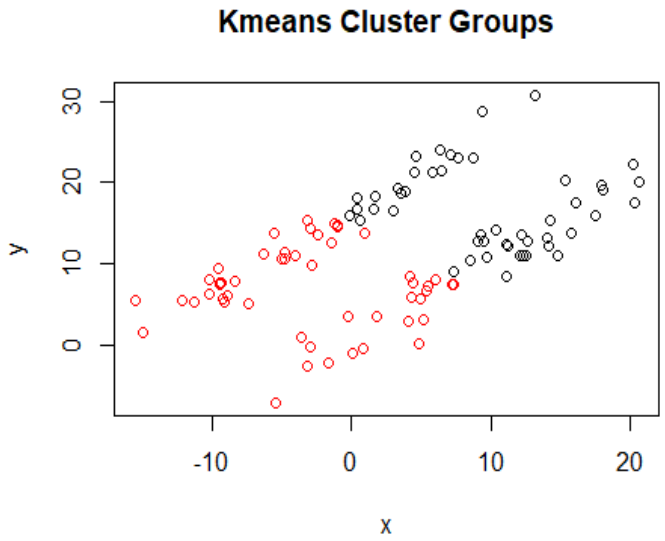


We know that we have two clusters in this data. Model-based clustering works better than alternative methods.

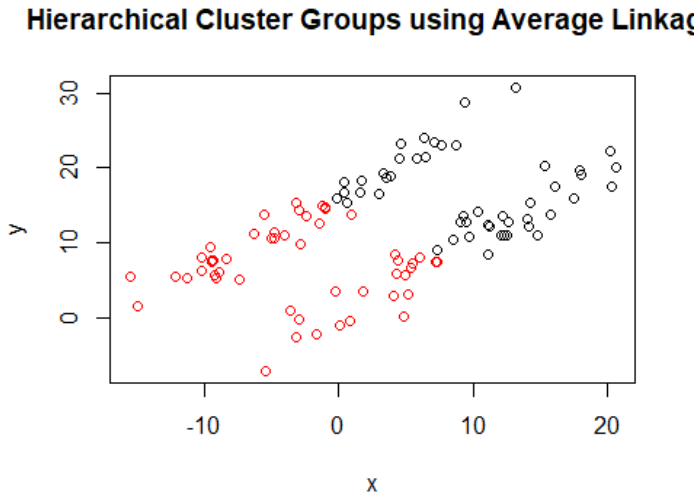
```
mc2 <- Mclust(cdata[,2:3])
summary(mc2)

## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EEE (ellipsoidal, equal volume, shape and orientation) model with 2
## components:
##
## loglikelihood n df       BIC       ICL
##       -612.3408 100  8 -1261.523 -1261.523
##
## Clustering table:
##  1  2
## 50 50

plot(cdata[,2:3], main="Model-Based Cluster Groups", col = mc2$classification
)
```



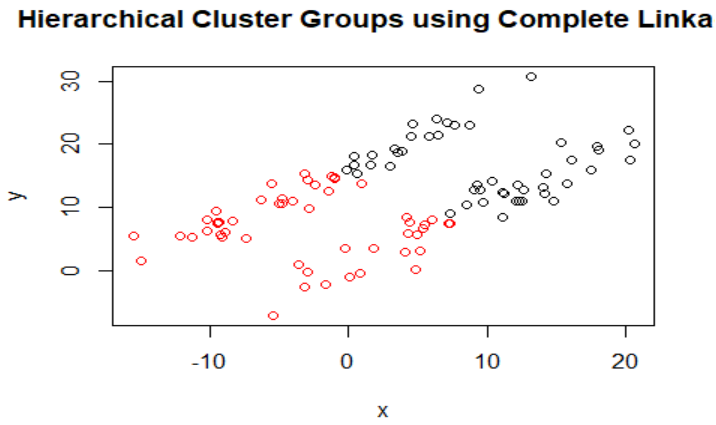

```
km2 <- kmeans(scale(cdata[,2:3]), 2)
plot(cdata[,2:3], main="Kmeans Cluster Groups", col = km2$cluster)
```



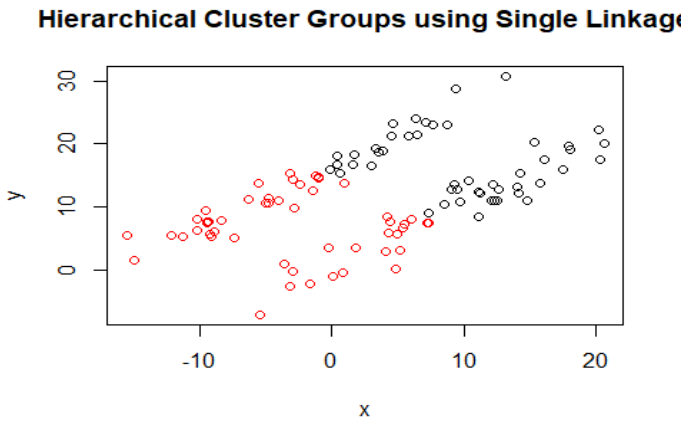

```
hc2 <- hclust(dist(scale(cdata[,2:3])), "average")
hc2.cluster= cutree(hc2,2)
plot(cdata[,2:3], main="Hierarchical Cluster Groups using Average Linkage", c
ol = km2$cluster)
```




```
hc2 <- hclust(dist(scale(cdata[,2:3])), "complete")
hc2.cluster= cutree(hc2,2)
plot(cdata[,2:3], main="Hierarchical Cluster Groups using Complete Linkage",
col = km2$cluster)
```




```
hc2 <- hclust(dist(scale(cdata[,2:3])), "single")
hc2.cluster= cutree(hc2,2)
plot(cdata[,2:3], main="Hierarchical Cluster Groups using Single Linkage", co
l = km2$cluster)
```



Example: Simulation II

As always, simulation is a great way to understand complex methods because the data analysis targets are known precisely, without ambiguity.

For simplicity, let's consider univariate data.

To put some context on it, suppose you can observe data on Y = sales % changes for a collection of firms. Unknown to you, some of these are private manufacturing firms, some are private service firms, and some are public firms.

A model for the data is as follows:

- $y_1 \sim N(10, 5^2)$ for private manufacturing firms
- $y_2 \sim N(30, 10^2)$ for private service firms
- $y_3 \sim N(50, 15^2)$ for public firms

The proportions of data for each of the three groups are also the unknown numbers p_1, p_2, and p_3, where $p_1 + p_2 + p_3 = 1$.

Hence, the model for the combined data is a mixture model:

$$Y = p_1 y_1 + p_2 y_2 + p_3 y_3$$

This model states as follows: A "future" data value Y will come to you. That value may be from one of the three groups, with probabilities p_1, p_2, and p_3. For example, if $(p_1, p_2, p_3) = (0.10, 0.30, 0.60)$, then 10% of such "future" Y will come from group 1, 30% of such "future" Y will come from group 2, and 60% of such "future" Y will come from group 3.

Further, the distribution of possible "future" data values within one of the three groups is normal, with mean and standard deviation specific to that group.

```
set.seed(123456)
n = 1000 # Sample size
True.group = sample(c(1,2,3), n, c(.1,.35,.55), replace=T)
m.group =ifelse(True.group==1,10, ifelse(True.group==2,30,50))
```

```
sd.group =ifelse(True.group==1,5, ifelse(True.group==2,10,15))
data1=data.frame(rnorm(n,m.group,sd.group), True.group)
colnames(data1) = c("Y", "True.group")
table(data1$True.group)

##
##   1   2   3
##  84 362 554

# Check that the simulation is working correctly by computing the estimates o
f the parameters
mean(subset(data1$Y, data1$True.group==1))

## [1] 9.208442

mean(subset(data1$Y, data1$True.group==2))

## [1] 29.75134

mean(subset(data1$Y, data1$True.group==3))

## [1] 50.68711

sd(subset(data1$Y, data1$True.group==1))

## [1] 4.515029

sd(subset(data1$Y, data1$True.group==2))

## [1] 10.50442

sd(subset(data1$Y, data1$True.group==3))

## [1] 14.72132

# plot(data1$Y, col = data1$True.group)
```

Now, as is usual, assume the true group labels are unknown. Assume three clusters, and use *kmeans* clustering.

```
km <- kmeans(data1$Y, 3)
table(km$cluster)

##
##   1   2   3
## 276 432 292

mean(subset(data1$Y, km$cluster==1))

## [1] 17.13237

mean(subset(data1$Y, km$cluster==2))

## [1] 38.98158
```

```r
mean(subset(data1$Y, km$cluster==3))

## [1] 61.83416

# or you can do km$center
```

Not a satisfactory result!

Again, assume the true group labels are unknown. And use *hierarchical* clustering.

```r
hc <- hclust(dist(data1$Y), "average")
hc.cluster <- cutree(hc, 3)
table(hc.cluster)

## hc.cluster
##   1   2   3
## 225 748  27

mean(subset(data1$Y, hc.cluster==1))

## [1] 14.97967

mean(subset(data1$Y, hc.cluster==2))

## [1] 45.46946

mean(subset(data1$Y, hc.cluster==3))

## [1] 83.05797
```

Not a satisfactory result!

Again, assume the true group labels are unknown. And use *model-based* clustering.

```r
library("mclust")
# You don't have to be worried about standardization in model-based clusterin
g (why?)
mc <- Mclust(data1$Y, 3)
table(mc$classification)
##   1   2   3
## 132 333 535

mean(subset(data1$Y, mc$classification==1))

## [1] 10.31276

mean(subset(data1$Y, mc$classification==2))

## [1] 28.82633

mean(subset(data1$Y, mc$classification==3))

## [1] 53.577

# or you can do mc$parameters$mean
```

APPENDIX: Mixture model in one dimension, step by step explanation

Observations: 1, 2, 4, 8, 9, 11. We want to group them in 2 groups.

```
# For one-dimensional Normal dist, we need to estimate mean and sd
x = c(1,2,4,8,9,11)

# Model-based clustering algorithm:
# Starts with two randomly placed normal distributions: Norm1(mu1, sd1), Norm
2(mu2, sd2).
set.seed(333)
mu1 = runif(1, min(x), max(x)) # this will be the initial mu1
mu2 = runif(1, min(x), max(x)) # this will be the initial mu2
sd1 = sd2 = sqrt(var(x)/2) # divided by 2 because we suppose to cluster into
2 groups # This is a kind of arbitrary estimation for sd1, and sd2
# for each point, does each point came from the above mentioned normal distri
butions?
# find the likelihoods: g1 = p(x | norm1) and g2 = p(x | norm2)
g1 = dnorm(x, mu1, sd1)
g2 = dnorm(x, mu2, sd2)
# Then find p1, p2 for each point. To do so, we need to perform the bayesian
formula (equation 6.2 on page 186 EH textbook)
#p1 = p(norm1|x) = p(x|norm1)p(norm1)/p(x) and p(x) = p(x|norm1)p(norm1) + p(
x|norm1)*p(norm2)
#p2 = p(norm2|x) = p(x|norm2)p(norm2)/p(x) and p(x) = p(x|norm1)p(norm1) + p(
x|norm1)*p(norm2)
# it's reasonable to assume p(norm1) = p(norm2) = 0.5 (because we have two gr
oups)
pnorm1 = 0.5
pnorm2 = 0.5
p1 = g1*pnorm1/(g1*pnorm1 + g2*pnorm2)
p2 = g2*pnorm2/(g1*pnorm1 + g2*pnorm2)
# what happened? which observation is in which group? Print out p1 and p2 to
understand it better.
label = ifelse(p1>p2, 1, 2)
label

## [1] 2 2 1 1 1 1

# In each step, we can measure the mixture density (equation 6.1 in page 186
EH textbook)
fx = p1*g1 + p2*g2
# Also the loglikelihood #(equation 6.3 in page 187 EH textbook)
sum(log(fx))

## [1] -15.41429

# We want to maximize the loglikelihod

# Then algorithm reestimate mu1, sd1 and mu2, sd2 (The idea is to perform the
weighted average)
```

```r
mu1 = sum(p1*x)/sum(p1) #(equation 6.6 in page 187 EH textbook)
sd1 = sqrt(sum(p1*(x-mu1)^2)/sum(p1))
mu2 = sum(p2*x)/sum(p2) #(equation 6.7 in page 187 EH textbook)
sd2 = sqrt(sum(p2*(x-mu2)^2)/sum(p2))

# Then follow the same process to update p1, p2
g1 = dnorm(x, mu1, sd1)
g2 = dnorm(x, mu2, sd2)

# estimating p(norm1) and p(norm2) depends on the proportion of p1 and p2 fro
m the total (equation 6.5 in page 187 EH textbook)
pnorm1 = sum(p1)/(sum(p1) + sum(p2))
pnorm2 = 1-pnorm1
p1 = g1*pnorm1/(g1*pnorm1 + g2*pnorm2)
p2 = g2*pnorm2/(g1*pnorm1 + g2*pnorm2)
# what happened? which observation is in which group?
label = ifelse(p1>p2, 1, 2)
label
```

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

```r
data.frame(x,label, p1, p2)
```

```
##    x label        p1         p2
## 1  1     2 0.1754930 0.82450703
## 2  2     2 0.2337308 0.76626918
## 3  4     2 0.4420369 0.55796309
## 4  8     1 0.9318397 0.06816032
## 5  9     1 0.9713360 0.02866405
## 6 11     1 0.9962165 0.00378353
```

```r
# In each step, we can measure the mixture density
fx = p1*g1 + p2*g2 # the
# Also, the loglikelihood
sum(log(fx))
```

```
## [1] -13.5216
```

```r
# We want to maximize loglikelihod -13.52 is better than -15.41. If we contin
ue, we may find a better loglikelihood estimate.

# we can also measure the BIC:
# In our example, the number of parameters is 5 (because we have mu1, mu2, sd
1, sd2, and pnorm1, we ignore pnorm2 because pnorm2 = 1-pnorm1)
# given the loglikelihood:
L = sum(log(fx))
n = length(x) # number of observations
BIC = L - .5 * 5 * log(n)
BIC # the larger is better
```

```
## [1] -18.001
```