Model-based Classification

The course so far has been "variable focused". We have examined relations between predictor variables and dependent variables. We focused on differen types of dependent variables like binary and counts.

We can also take a "unit focus" where we classify subjects or cases as well as a "variable focus".

K-means

There are some ways of classifing cases that are exploratory, such as k-means clustering. The idea underlying k-means clustering is to see if there are groups of subjects who follow a similar pattern in their means.

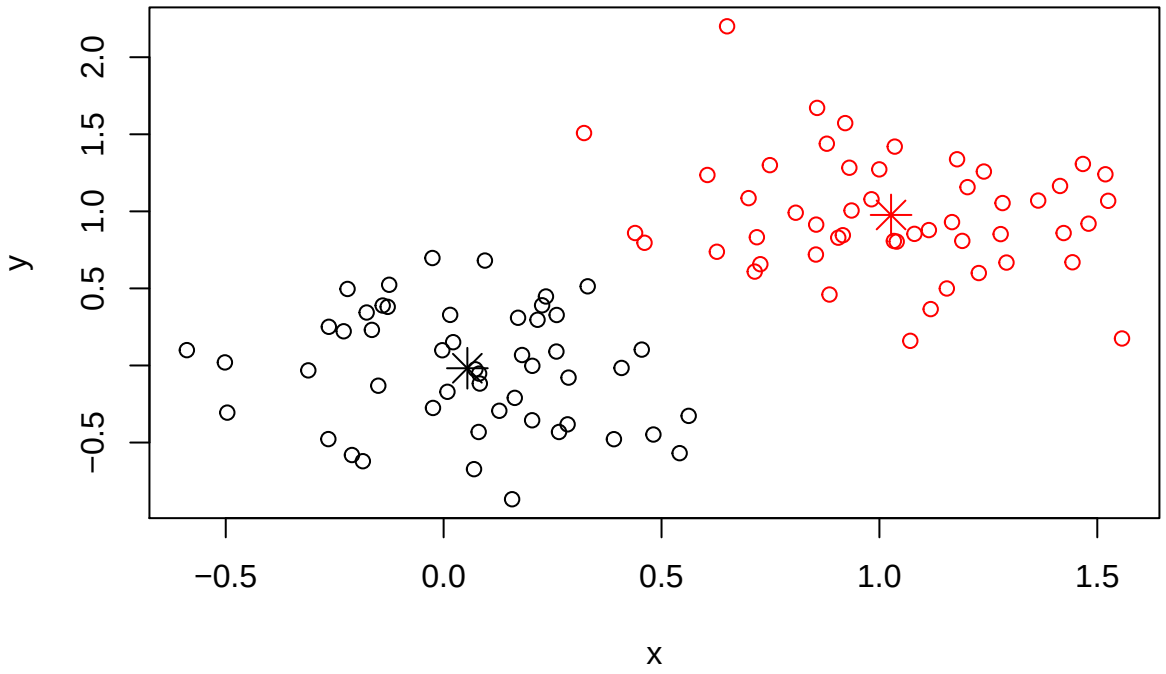
A simple example from the k-means help file. There are two variables, x and y. Suppose there are two types of subjects: some are low on both x and y, and others tend to be high on both x and y.

```
# a 2-dimensional example
data <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
colnames(data) <- c("x", "y")

#ask for 2 clusters
cl <- kmeans(data, 2)
cl$centers
```

```
##         x       y
## 1 0.05448 -0.0179
## 2 1.02676  0.9766
```

```
plot(data, col = cl$cluster)
points(cl$centers, col = 1:2, pch = 8, cex = 2)
```



The output of k-means clustering finds sets of means that minimize the within group sum of squares. This is related to ANOVA with the difference being that in ANOVA you already know the groups in advance, with k-means you are trying to find groups (subjects with similar patterns) in your data.

In this example the red and black asterisks represent the "centers" (each have a mean on one variable x and a mean on one variable y). These centers are defined to minimize the within group sum of squares.

```
cl$withinss
```

```
## [1] 10.64 11.54
```

In this example we are treating x and y as two variables. No variable has special status as dependent variable or predictor.

This is a type of exploratory data analyis. There are some ways of attaching test of significance (such as bootstrapping) and measures of fit. K-means is a fine way to go if you are exploring your data to find groups of subjects who may have common patters in the way they respond to the various measures.

However, k-means clustering is somewhat limited in its ability to fit into the broader statistical framework of typical research questions. We would also like a technique that can fit into the way we do statistics, such as one variable is given status as a dependent variable and others are predictors.

Model-based Clustering

I want to introduce an example of a class of techniques that do a nice job of finding different types of subjects who have different models. An example is some groups of subjects may have different slopes and intercept in a regression context. Some subjects may not have changed at all over three times, whereas some subjects may have increased the scores on the dependent variable across the three times. That would get reflected in a subgroup of subjects having a slope close to zero and another subgroup having a positive slope.

We call this model-based clustering to distinguish it from the exploratory type of clustering done by k-means. With k-means one classifies common patterns across a set of variables, in model-based clustering one fits a statistical model (such as lm or glm) and clusters on the bases of subjects who have common parameter values on those models.

A nice package in R to do these kinds of model-based clustering is flexmix. Here are a couple of examples from the help file. The first fits a linear and quadratic regression to normally distributed data and asks for k=2.

```
library(flexmix)
```

```
## Loading required package: lattice
```

```
data("NPreg", package = "flexmix")

## mixture of two linear regression models.
ex1 <- flexmix(yn~x+I(x^2), data=NPreg, k=2,
                   control=list(verb=5, iter=100))
```

```
## Classification: weighted
##    5 Log-likelihood :    -722.6286
##   10 Log-likelihood :    -646.5500
##   15 Log-likelihood :    -642.5453
## converged
```

```
ex1
```

```
##
## Call:
## flexmix(formula = yn ~ x + I(x^2), data = NPreg, k = 2, control = list(verb = 5,
##     iter = 100))
##
## Cluster sizes:
##   1   2
## 100 100
##
## convergence after 15 iterations
```

```
summary(ex1)
```

```
##
## Call:
## flexmix(formula = yn ~ x + I(x^2), data = NPreg, k = 2, control = list(verb = 5,
##     iter = 100))
##
##        prior size post>0 ratio
## Comp.1 0.506  100    141 0.709
## Comp.2 0.494  100    145 0.690
##
## 'log Lik.' -642.5 (df=9)
## AIC: 1303   BIC: 1333
```

The cases are clustered into those who have common βs. The output includes a vector of grouping codes that you can then use in other analyses, such as in plots or subsequent regressions and ANOVAS.

Flexmix use a type of object called slot. slotNames() gives names of slots that you can use for the object.

```
slotNames(ex1)
```

```
##  [1] "posterior"   "weights"     "iter"        "cluster"     "logLik"
##  [6] "df"          "control"     "group"       "size"        "converged"
## [11] "k0"          "model"       "prior"       "components"  "concomitant"
## [16] "formula"     "call"        "k"
```

```
slot(ex1,"cluster")
```

```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [36] 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [71] 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1
## [106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
## [141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2
## [176] 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
slot(ex1, "size")
```

```
##   1   2
## 100 100
```

You can get the separate parameters for each class.

```
parameters(ex1,component=1)
```

```
##                   Comp.1
## coef.(Intercept) 14.7168
## coef.x            9.8469
## coef.I(x^2)      -0.9684
## sigma             3.4795
```

```
parameters(ex1,component=2)
```

```
##                    Comp.2
## coef.(Intercept) -0.21002
## coef.x            4.81817
## coef.I(x^2)       0.03612
## sigma             3.47676
```

You can also fit glm models within the flexmix package. In this example we have one one predictor.

```
ex3 <- flexmix(yp~x, data=NPreg, k=2, model=FLXMRglm(family="poisson"))
summary(ex3)
```

```
##
## Call:
## flexmix(formula = yp ~ x, data = NPreg, k = 2, model = FLXMRglm(family = "poisson"))
##
##        prior size post>0 ratio
## Comp.1 0.459   87    200 0.435
## Comp.2 0.541  113    198 0.571
##
## 'log Lik.' -440.6 (df=5)
## AIC: 891.3   BIC: 907.8
```

```
slot(ex3,"size")
```

```
##   1   2
##  87 113
```

```
parameters(ex3,component=1)
```

```
##                   Comp.1
## coef.(Intercept) 1.25246
## coef.x           0.06431
```

```
parameters(ex3,component=2)
```

```
##                   Comp.2
## coef.(Intercept)  1.8577
## coef.x           -0.1607
```

You can also have more than one dependent variable, which we will discuss later in the workshop. We can set up a system of equations where variables can have different distributions.

```
ex2 <- flexmix(yn~x, data=NPreg, k=2,
               model=list(FLXMRglm(yn~.+I(x^2)),
                          FLXMRglm(yp~., family="poisson")))
ex2
```

```
##
## Call:
## flexmix(formula = yn ~ x, data = NPreg, k = 2, model = list(FLXMRglm(yn ~
##     . + I(x^2)), FLXMRglm(yp ~ ., family = "poisson")))
##
## Cluster sizes:
##   1   2
##  96 104
##
## convergence after 25 iterations
```

```
summary(ex2)
```

```
##
## Call:
## flexmix(formula = yn ~ x, data = NPreg, k = 2, model = list(FLXMRglm(yn ~
##     . + I(x^2)), FLXMRglm(yp ~ ., family = "poisson")))
##
##        prior size post>0 ratio
## Comp.1 0.493   96    139 0.691
## Comp.2 0.507  104    137 0.759
##
## 'log Lik.' -1045 (df=13)
## AIC: 2116   BIC: 2158
```

```
## for Gaussian responses
parameters(ex2, component=1, model=1)
```

```
##                   Comp.1
## coef.(Intercept) 14.5906
## coef.x            9.9156
## coef.I(x^2)      -0.9758
## sigma             3.4110
```

```
parameters(ex2, component=2, model=1)
```

```
##                    Comp.2
## coef.(Intercept) -0.14125
## coef.x            4.73436
## coef.I(x^2)       0.04251
## sigma             3.42692
```

```
## for Poisson responses
parameters(ex2, component=1, model=2)
```

```
##                    Comp.1
## coef.(Intercept) 1.03772
## coef.x           0.09108
```

```
parameters(ex2, component=2, model=2)
```

```
##                    Comp.2
## coef.(Intercept)  1.9391
## coef.x           -0.1809
```

Fancy Model-based Clustering

There are some programs you should consider if you want to get serious into model-based clustering. The gold standard is the program Mplus. It is a very nice program but it is expensive. It can do many things, including IRT analyses, all of SEM, latent growth curve, growth mixture models, multiple grou analyses, and handle all the generalized linear distributions (normal, poisson, binomial, negative binomial, etc; with several link options) as well as extensions like zero-inflated models, multinomial models, and ordinal regressions.

There are two groups working on bringing all of MPlus features into R without having to buy Mplus. One group is the openMX project http://openmx.psyc.virginia.edu/. This tool is very flexible but requires quite a bit of programming. It is a great tool if you are a developer of new methods and want to quickly program new techniques. There is a graphical user interface that can be used with openMX http://onyx.brandmaier.de/. I won't talk about those two features because they require quite intensive installation procedures (e.g., openMX cannot be installed through the regular package feature), and while onyx makes openMX easier to use we should devote an entire course to just using both.

A second alternative to Mplus is the lavaan package in R. This is easier to use and the syntax is very similar to Mplus. The development team is making good progress at making lavaan be almost as comprehensive as Mplus. They still have to add categorical latent variables, which would give the ability to do model-based clustering. I'll talk about lavaan in this workshop.