# Classification

May 20, 2016

## Contents


**1 Logistic Regression** **2**
- 1.1 The logistic models . . . 2
- 1.2 Generalized linear models . . . 6

**2 Cluster analysis** **8**
- 2.1 What is cluster analysis . . . 8
- 2.2 Why do cluster analysis . . . 8
- 2.3 How to do cluster analysis . . . 8
- 2.4 Hierarchical Clustering . . . 9
- 2.5 K-Means Clustering . . . 16

**3 Discriminant Analysis** **25**
- 3.1 Distance . . . 25
- 3.2 Discriminant function . . . 28
- 3.3 Fisher discriminant analysis . . . 30

The linear regression model discussed in Regression assumes that the response variable Y is quantitative. But in many situations, the response variable is instead qualitative. For example, eye color is qualitative, taking on values blue, brown, or green. Often qualitative variables are referred to as categorical ; we will use these terms interchangeably. In this chapter, we study approaches for predicting qualitative responses, a process that is known as classification. Predicting a qualitative response for an observation can be referred to as classifying that observation, since it involves assigning the observation to a category, or class. On the other hand, often the methods used for classification first predict the probability of each of the categories of a qualitative variable, as the basis for making the classification. In this sense they also behave like regression methods.

Often, an analyst's goal is to classify an item into a category or maybe to estimate the probability that an item belongs to a certain category. Models that describe this relationship are called classification models.

# 1 Logistic Regression

In previous chapter, we explored linear models that can be used to predict a normally distributed response variable from a set of continuous and/or categorical predictor variables. But there are many situations in which it's unreasonable to assume that the dependent variable is normally distributed (or even continuous). For example: The outcome variable may be categorical. Binary variables (for example, yes/ no, passed/failed, lived/died) and polytomous variables (for example, poor/ good/excellent, republican/democrat/independent) are clearly not normally distributed.

## 1.1 The logistic models

Logistic regression is applied to situations in which the response variable is dichotomous (0,1). We talked of using a linear regression model to represent these probabilities:

$$p(Y = 1|X) = \beta_0 + \beta_1 X$$

Instead of modelling the expected value of the response directly as a linear function of explanatory variables, a suitable transformation is modelled. In this case the most suitable transformation is the logistic or logit function of $p$ leading to the model

$$\text{logit}(p) = \log(\frac{p}{1-p}) = \log(\Omega) = \beta_0 + \beta_1 X \tag{1}$$

where $p = \mu_Y$ is the conditional mean of $Y$ (that is, the probability that $Y = 1$ given a set of $X$ values), $(p/(1-p))$ is the odds that $Y = 1$, and $\log(p/(1-p))$ is the log odds, or logit. In this case, $\log(p/(1-p))$ is the link function, the probability distribution is binomial, and the logistic regression model can be fit using

```
glm(Y~X1+X2+X3, family=binomial(link="logit"), data=mydata)
```

Equation (1) can be rewritten as

$$p(X) = \frac{\exp(\beta_0 + \beta_1 X)}{1 + \exp(\beta_0 + \beta_1 X)} \tag{2}$$

The logit function can take any real value, but the associated probability always lies in the required [0, 1] interval. In a logistic regression model, the parameter $j$ associated with explanatory variable $x_j$ is such that $\exp(j)$ is the odds that the response variable takes the value one when $x_j$ increases by one, conditional on the other explanatory variables remaining constant. The parameters of the logistic regression model (the vector of regression coefficients ) are estimated by maximum likelihood.

$$L(\beta_0, \beta) = \Pi_{i:y_i=1} p(x_i) \Pi_{i':y'_i=0} (1 - p(x_{i'}))$$

The estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are chosen to maximize this likelihood function.

## An Example

Faraway gives an example of predicting a binary-valued variable: test from the dataset pima is true if the patient tested positive for diabetes. The predictors are diastolic blood pressure and body mass index (BMI). Faraway uses linear regression, so let's try logistic regression instead: (install.packages(faraway))

```
library("faraway")
data(pima, package="faraway")
head(pima)

##   pregnant glucose diastolic triceps insulin  bmi diabetes age test
## 1        6     148        72      35       0 33.6    0.627  50    1
## 2        1      85        66      29       0 26.6    0.351  31    0
## 3        8     183        64       0       0 23.3    0.672  32    1
## 4        1      89        66      23      94 28.1    0.167  21    0
## 5        0     137        40      35     168 43.1    2.288  33    1
## 6        5     116        74       0       0 25.6    0.201  30    0

b <- factor(pima$test)
head(b)

## [1] 1 0 1 0 1 0
## Levels: 0 1

m <- glm(b ~ diastolic + bmi, family=binomial(link="logit"), data=pima)
summary(m)

##
## Call:
## glm(formula = b ~ diastolic + bmi, family = binomial(link = "logit"),
##     data = pima)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9128  -0.9180  -0.6848   1.2336   2.7417
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.629553   0.468176  -7.753 9.01e-15 ***
## diastolic   -0.001096   0.004432  -0.247    0.805
## bmi          0.094130   0.012298   7.654 1.95e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 993.48  on 767  degrees of freedom
## Residual deviance: 920.65  on 765  degrees of freedom
## AIC: 926.65
##
## Number of Fisher Scoring iterations: 4
```

The summary of the resulting model, m, shows that the respective p-values for the diastolic and the bmi variables are 0.805 and (essentially) zero. We can therefore conclude that only the bmi variable is significant:

```
m.red <- glm(b ~ bmi, family=binomial, data=pima)
summary(m.red)

##
## Call:
## glm(formula = b ~ bmi, family = binomial, data = pima)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9209  -0.9178  -0.6838   1.2351   2.7244
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.68641    0.40896  -9.014  < 2e-16 ***
## bmi          0.09353    0.01205   7.761 8.45e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 993.48  on 767  degrees of freedom
## Residual deviance: 920.71  on 766  degrees of freedom
## AIC: 924.71
##
## Number of Fisher Scoring iterations: 4
```

In the call to predict, we set type="response" so that predict returns a probability. Otherwise, it returns log-odds, which I don't find as useful. Let's use the model to calculate the probability that someone with an average BMI (32.0) will test positive for diabetes: (Using Equation 2)

```
#1
newdata <- data.frame(bmi=32.0)
predict(m.red, type="response", newdata=newdata)

##         1
## 0.3332689

#2
E<-exp(-3.68641+0.09353*32)
E/(1+E)

## [1] 0.333266
```

According to this model, the probability is about 33.3%. The same calculation for someone in the 90th percentile gives a probability of 54.9%:

```
newdata <- data.frame(bmi=quantile(pima$bmi, .90))
predict(m.red, type="response", newdata=newdata)

##       90%
## 0.5486215
```

## Exercise

```
SoftDrink<-read.table(file="SoftDrink.txt",header=TRUE)
head(SoftDrink)

##   Brand Price Calories Fat Vitamin Fruits Age Choice
## 1     1  2.50      463  28       9     13  34      1
## 2     2  5.00      926  57       4      6  22      0
## 3     3  2.28      310  19      13     18  23      1
## 4     4  4.00     1080  66       4      6  40      0
## 5     5  2.50      154   9      15     21  36      1
## 6     6  3.97      221  14       5      7  45      1

SoftDrink$Choice<-as.factor(SoftDrink$Choice)
Fit<-glm(Choice~.-Brand,data=SoftDrink,family=binomial(link="logit"))
summary(Fit)

##
## Call:
## glm(formula = Choice ~ . - Brand, family = binomial(link = "logit"),
##     data = SoftDrink)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8461  -0.0433   0.0020   0.1114   3.2400
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 20.851906   8.081177   2.580  0.00987 **
## Price        0.423373   0.327125   1.294  0.19559
## Calories    -0.025519   0.008767  -2.911  0.00361 **
## Fat         -0.080466   0.048667  -1.653  0.09825 .
## Vitamin     -0.940737   0.460753  -2.042  0.04118 *
## Fruits       0.376950   0.159685   2.361  0.01825 *
## Age          0.013747   0.015796   0.870  0.38415
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 403.427  on 291  degrees of freedom
## Residual deviance:  79.962  on 285  degrees of freedom
```

```
## AIC: 93.962
##
## Number of Fisher Scoring iterations: 8

Fit<-glm(Choice~.-Brand-Price-Fat-Age-Vitamin,data=SoftDrink,family=binomial(link="logit"))
summary(Fit)

##
## Call:
## glm(formula = Choice ~ . - Brand - Price - Fat - Age - Vitamin,
##     family = binomial(link = "logit"), data = SoftDrink)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.6579  -0.1009   0.0019   0.1018   2.9622
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  8.577449   3.439845   2.494   0.0126 *
## Calories    -0.017091   0.003644  -4.690 2.74e-06 ***
## Fruits       0.354141   0.186476   1.899   0.0575 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 403.427  on 291  degrees of freedom
## Residual deviance:  88.498  on 289  degrees of freedom
## AIC: 94.498
##
## Number of Fisher Scoring iterations: 8

coef(Fit)

## (Intercept)    Calories      Fruits
##   8.5774486  -0.0170909   0.3541405

exp(coef(Fit))

##  (Intercept)     Calories       Fruits
## 5310.5389762    0.9830543    1.4249554
```

## 1.2 Generalized linear models

Generalized linear models extend the linear model framework to include dependent variables that are decidedly non-normal. Generalized linear models are typically fit in R through the glm() function (although other specialized functions are available). The form of the function is similar to lm() but includes additional parameters. The basic format of the function is

```
glm(formula, family=family(link=function), data=)
```

where the probability distribution (family) and corresponding default link function (function) are given in the following table.

| Family | Default link function |
|---|---|
| binomial | (link = "logit") |
| gaussian | (link = "identity") |
| gamma | (link = "inverse") |
| inverse.gaussian | (link = "1/mu^2") |
| poisson | (link = "log") |
| quasi | (link = "identity", variance = "constant") |
| quasibinomial | (link = "logit") |
| quasipoisson | (link = "log") |

Figure 1: glm() parameter

The glm() function allows you to fit a number of popular models, including logistic regression, Poisson regression, and survival analysis (not considered here).

# 2 Cluster analysis

## 2.1 What is cluster analysis

Identifying groups of individuals or objects that are similar to each other but different from individuals in other groups can be intellectually satisfying, profitable, or sometimes both.[1]

The term cluster analysis does not identify a particular statistical method or model, as do discriminant analysis, factor analysis, and regression. You often don't have to make any assumptions about the underlying distribution of the data. Using cluster analysis, you can also form groups of related variables, similar to what you do in factor analysis. There are numerous ways you can sort cases into groups. The choice of a method depends on, among other things, the size of the data file. Methods commonly used for small data sets are impractical for data files with thousands of cases.

## 2.2 Why do cluster analysis

- You need to identify people with similar patterns of past purchases so that you can tailor your marketing strategies.
- You've been assigned to group television shows into homogeneous categories based on viewer characteristics. This can be used for market segmentation.
- You want to cluster skulls excavated from archaeological digs into the civilizations from which they originated. Various measurements of the skulls are available.
- You're trying to examine patients with a diagnosis of depression to determine if distinct subgroups can be identified, based on a symptom checklist and results from psychological tests.

## 2.3 How to do cluster analysis

You start out with a number of cases and want to subdivide them into homogeneous groups. First, you choose the variables on which you want the groups to be similar. Next, you must decide whether to standardize the variables in some way so that they all contribute equally to the distance or similarity between cases. Finally, you have to decide which clustering procedure to use, based on the number of cases and types of variables that you want to use for forming clusters.

For hierarchical clustering, you choose a statistic that quantifies how far apart (or similar) two cases are. Then you select a method for forming the groups. Because you can have as many clusters as you do cases (not a useful solution!), your last step is to determine how many clusters you need to represent your data. You do this by looking at how similar clusters are when you create additional clusters or collapse existing ones.

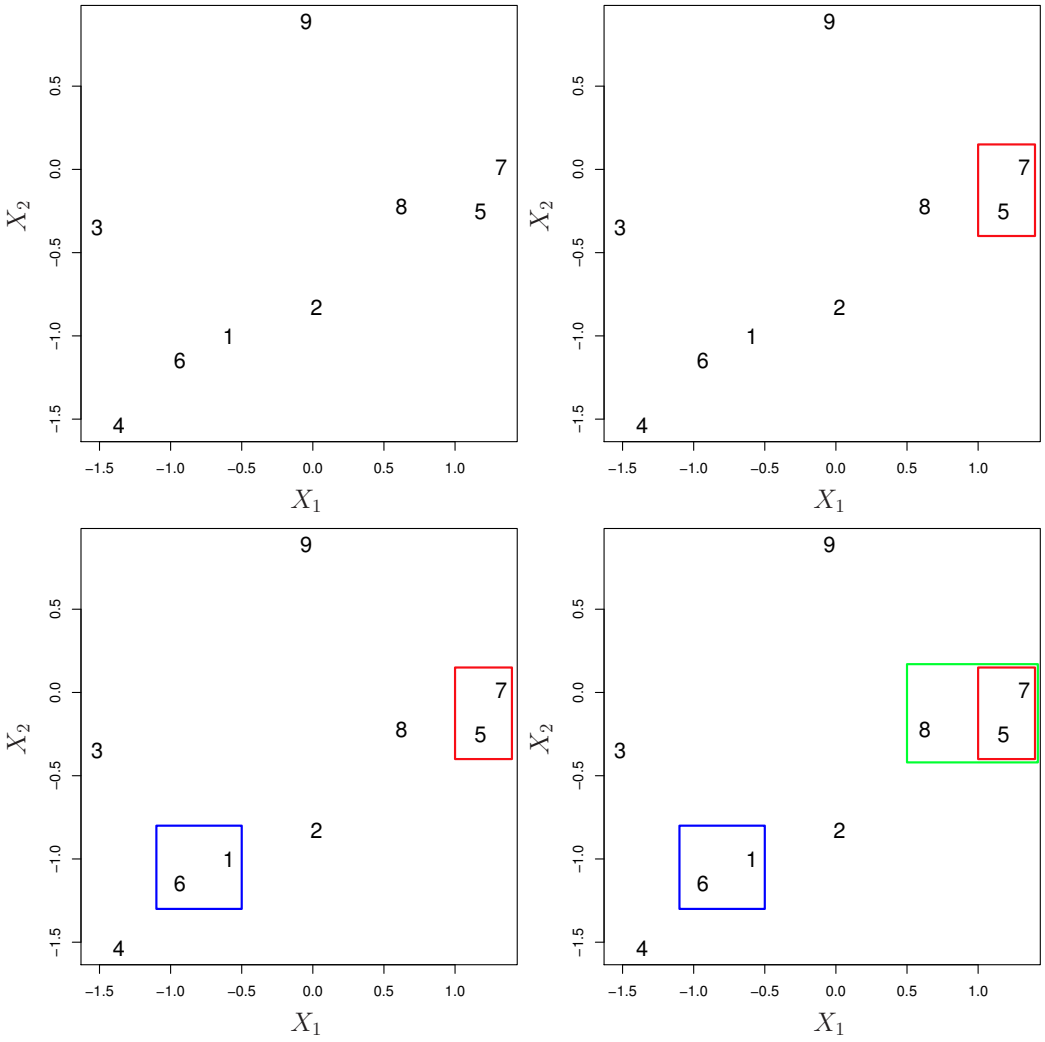
In k-means clustering, you select the number of clusters you want. The algorithm iteratively estimates the cluster means and assigns each case to the cluster for which its distance to the cluster mean is the smallest.

In two-step clustering, to make large problems tractable, in the first step, cases are assigned to "preclusters." In the second step, the preclusters are clustered using the hierarchical clustering algorithm. You can specify the number of clusters you want or let the algorithm decide based on preselected criteria.

[1] Although both cluster analysis and discriminant analysis classify objects (or cases) into categories, discriminant analysis requires you to know group membership for the cases used to derive the classification rule. The goal of cluster analysis is to identify the actual groups. For example, if you are interested in distinguishing between several disease groups using discriminant analysis, cases with known diagnoses must be available. Based on these cases, you derive a rule for classifying undiagnosed patients. In cluster analysis, you don't know who or what belongs in which group. You often don't even know the number of groups.

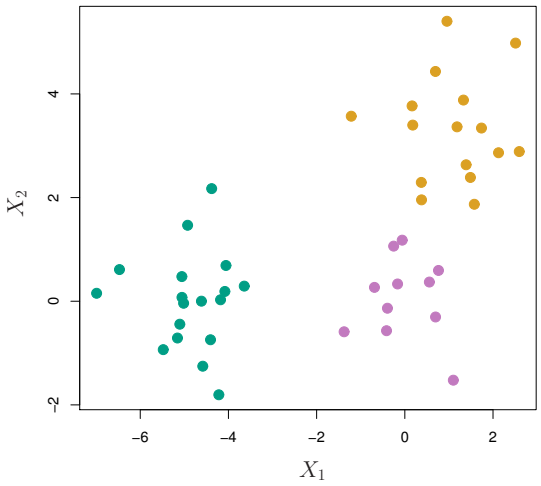
## 2.4 Hierarchical Clustering

There are numerous ways in which clusters can be formed. Hierarchical clustering is one of the most straightforward methods. Agglomerative hierarchical clustering starts with each case (in this example, each judge) being a cluster. At the next step, the two judges who have the smallest value for the distance measure (or largest value if you are using similarities) are joined into a single cluster. At the second step, either a third case is added to the cluster that already contains two cases or two other cases are merged into a new cluster. At every step, either individual cases are added to existing clusters, two individuals are combined, or two existing clusters are combined.

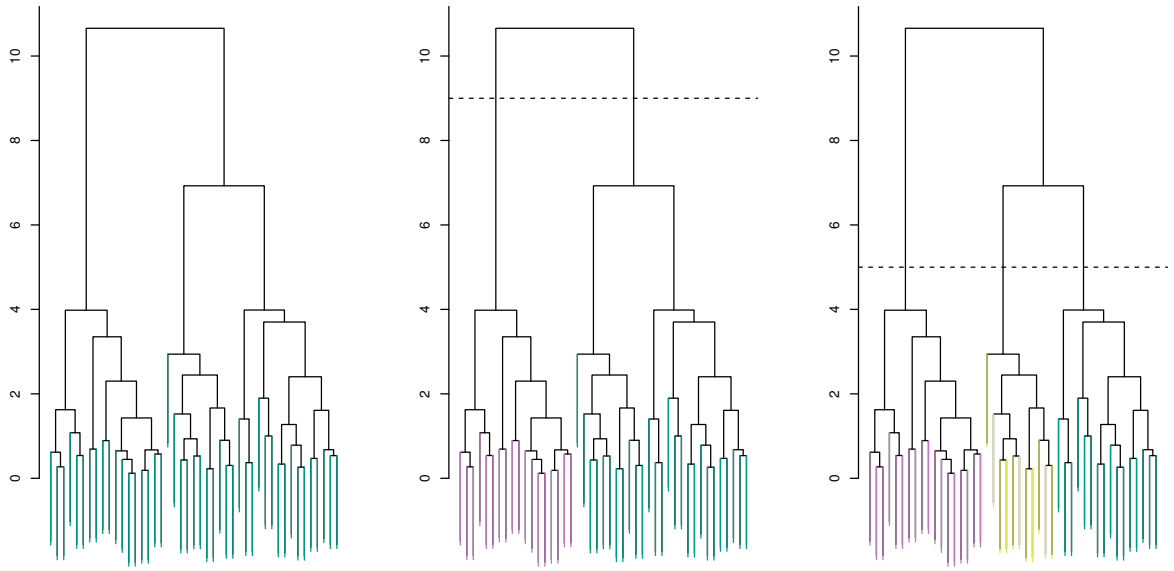



# Interpreting a Dendrogram

We begin with the simulated data set consisting of 45 observations in two-dimensional space. The data were generated from a three-class model; the true class labels for each observation are shown in distinct colors. However, suppose that the data were observed without the class labels, and that we wanted to perform hierarchical clustering of the data.




Each leaf of the dendrogram represents one of the 45 observations. However, as we move up the tree, some leaves begin to fuse into branches. These correspond to observations that are similar to each other. As we move higher up the tree, branches themselves fuse, either with leaves or other branches. The earlier (lower in the tree) fusions occur, the more similar the groups of observations are to each other. On the other hand, observations that fuse later (near the top of the tree) can be quite different. In fact, this statement can be made precise: for any two observations, we can look for the point in the tree where branches containing those two observations are first fused. The height of this fusion, as measured on the vertical axis, indicates how different the two observations are. Thus, observations that fuse at the very bottom of the tree are quite similar to each other, whereas observations that fuse close to the top of the tree will tend to be quite different.




## Linkage (distance) between Cluster Pairs

These methods define the distance between two clusters at each stage of the procedure. If cluster A has cases 1 and 2 and if cluster B has cases 5, 6, and 7, you need a measure of how different or similar the two clusters are.

**Nearest neighbor (single linkage).**

If you use the nearest neighbor method to form clusters, the distance between two clusters is defined as the smallest distance between two cases in the different clusters.

**Furthest neighbor (complete linkage).**

If you use a method called furthest neighbor (also known as complete linkage), the distance between two clusters is defined as the distance between the two furthest points.

**Average method.**

Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the average of these dissimilarities.

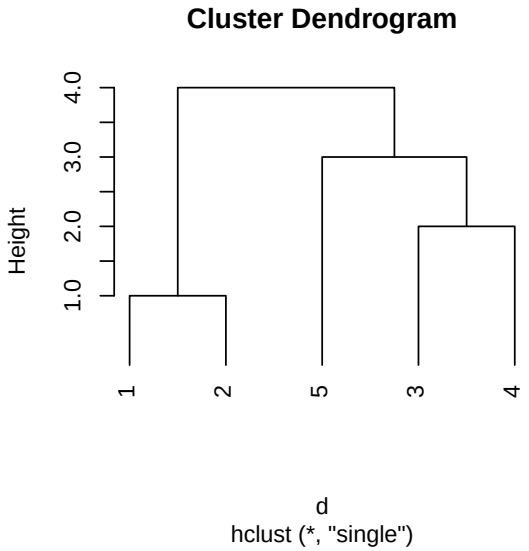
**Centroid method.**

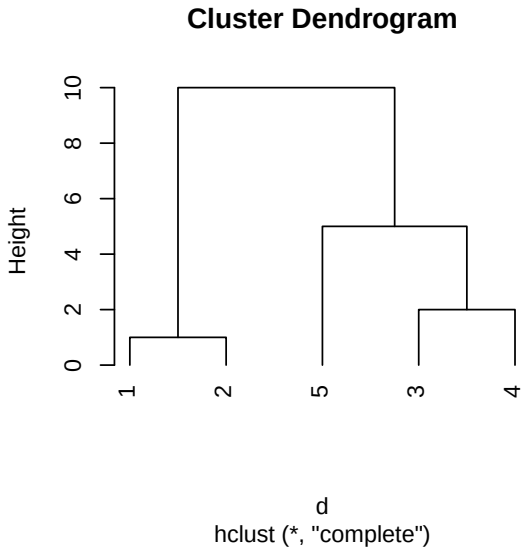
This method calculates the distance between two clusters as the sum of distances between cluster means for all of the variables. In the centroid method, the centroid of a merged cluster is a weighted combination of the centroids of the two individual clusters, where the weights are proportional to the sizes of the clusters. One disadvantage of the centroid method is that the distance at which clusters

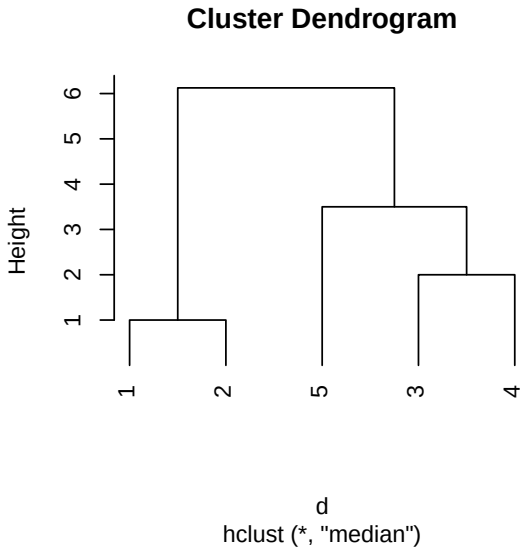
are combined can actually decrease from one step to the next. This is an undesirable property because clusters merged at later stages are more dissimilar than those merged at early stages.

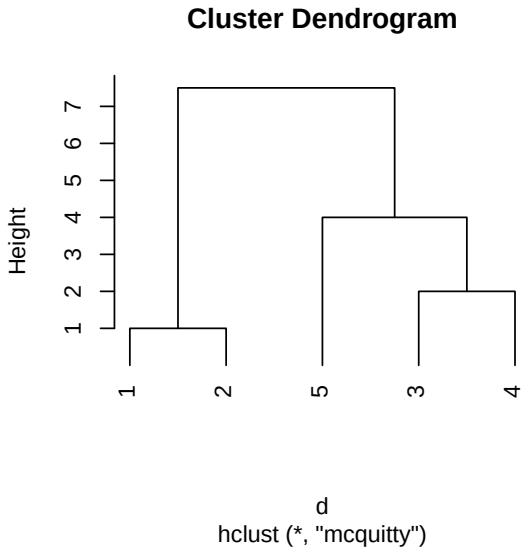
## Example 1

```
#Calculate
x<-c(1,2,6,8,11); dim(x)<-c(5,1)
d<-dist(x)
hc1<-hclust(d, "single"); hc2<-hclust(d, "complete")
hc3<-hclust(d, "median"); hc4<-hclust(d, "mcquitty")
#Plot
opar <- par(mfrow = c(2, 2))
plot(hc1,hang=-1); plot(hc2,hang=-1)
plot(hc3,hang=-1); plot(hc4,hang=-1)
```



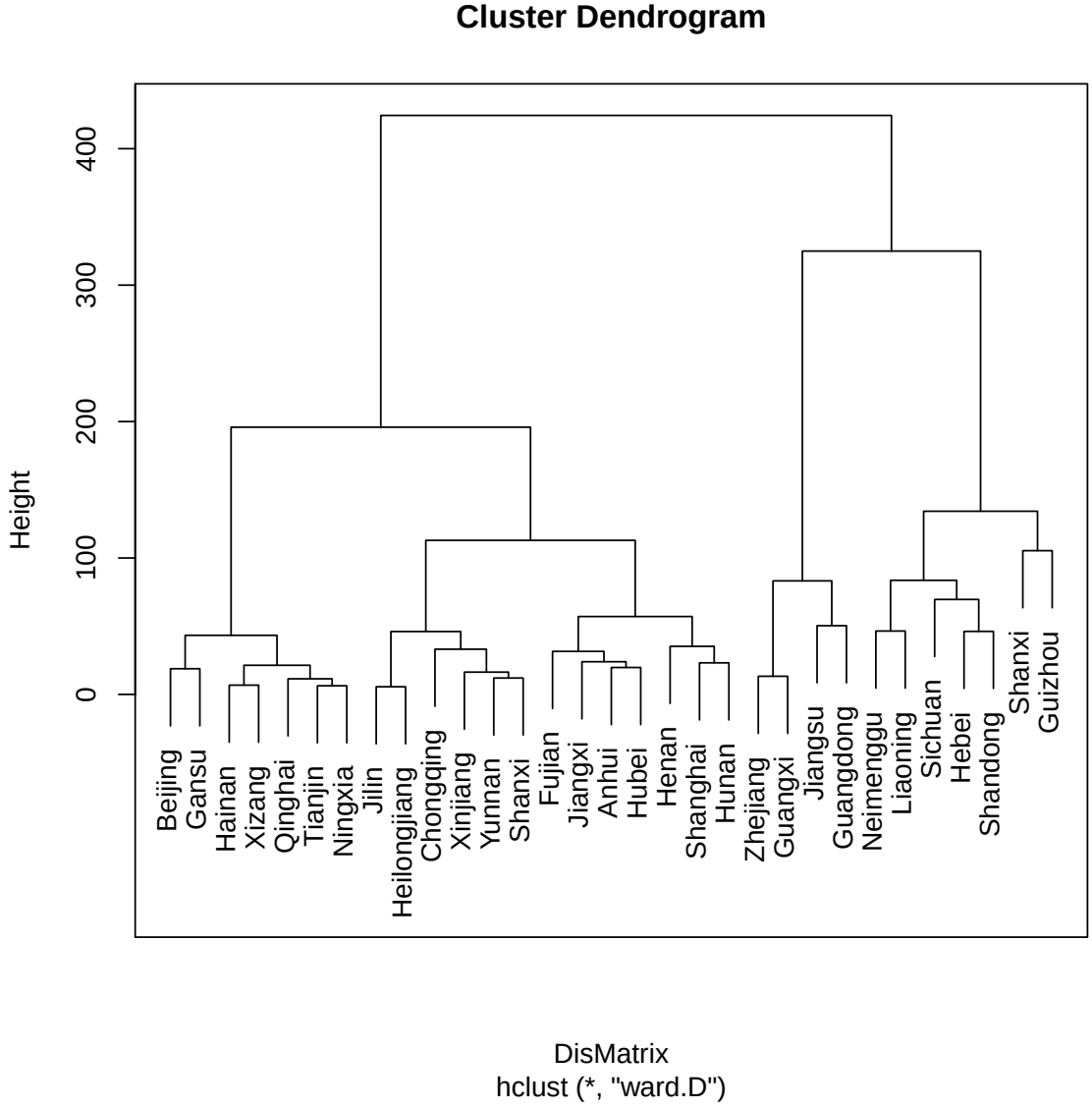

```
par(opar)
```

## Example 2

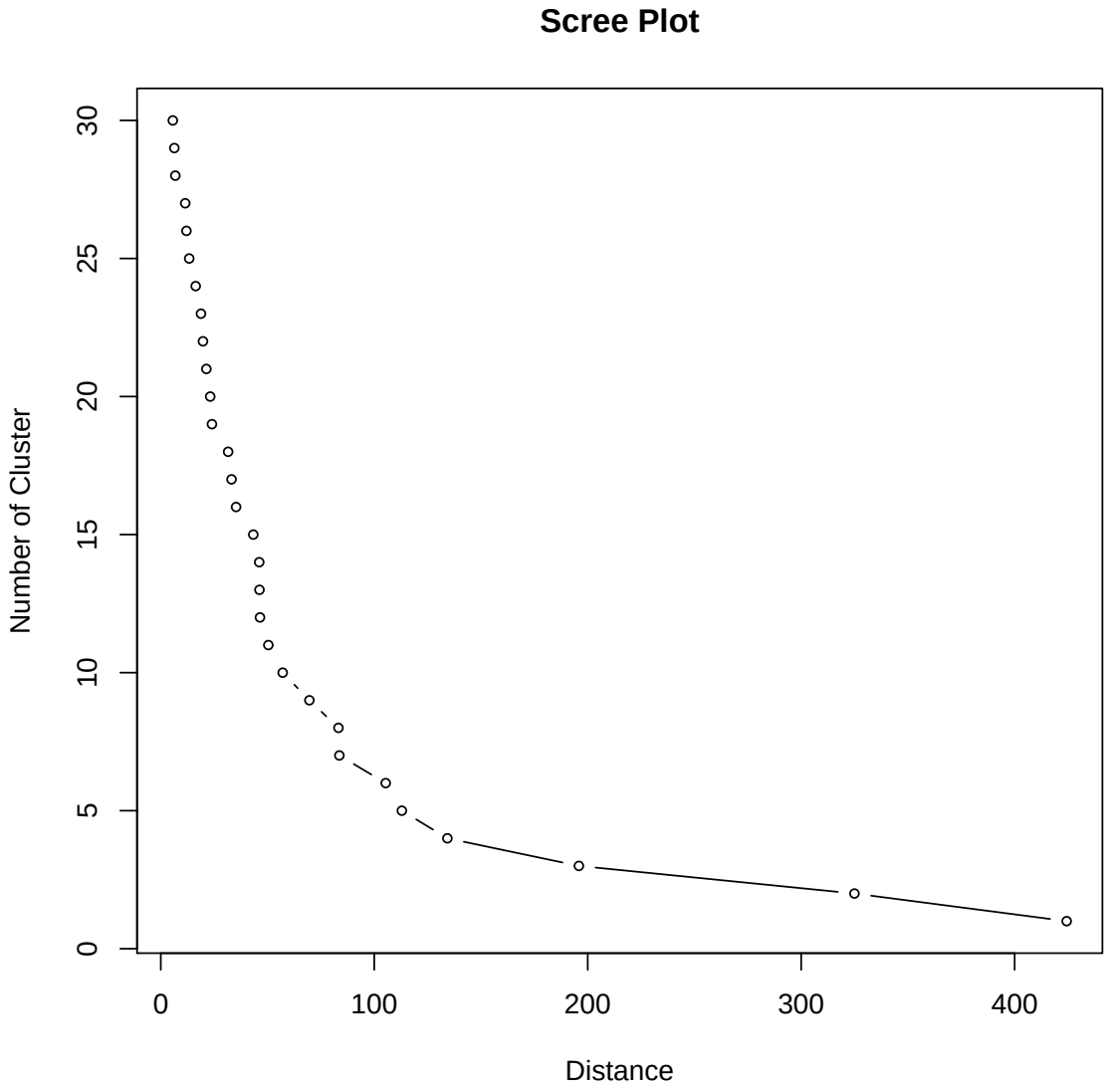
```
PoData<-read.table(file="PollutionData1.txt",header=TRUE)
CluData<-PoData[,2:7]
head(CluData)
```

```
##       x1    x2     x3     x4     x5    x6
## 1 21.76 15.01  12.23   0.02  10.69  3.09
## 2  7.47  4.19   4.80   0.00  11.44  7.68
## 3 21.92 43.49  69.43   9.40 100.00 45.79
```

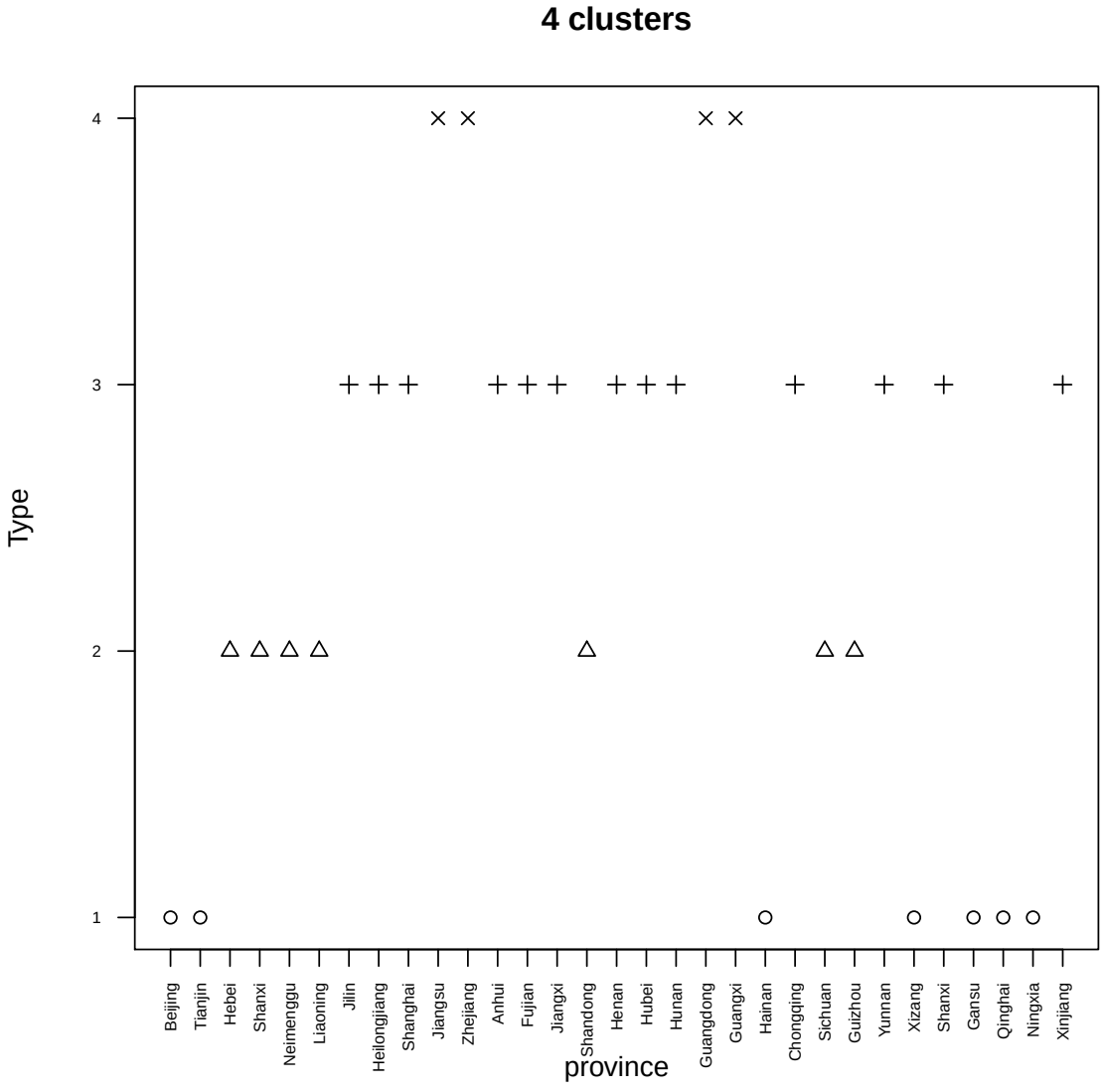
```
## 4 13.79 58.94  93.45 100.00  44.60 15.04
## 5  7.44 37.97  69.87   2.16  37.87  9.02
## 6 27.90 36.64 100.00   1.08  49.84 35.21

###Hierarchical Clustering
DisMatrix<-dist(CluData,method = "euclidean")
CluR<-hclust(d=DisMatrix,method="ward")
#Graph_1
plot(CluR,labels=PoData[,1])
box()
```




```
#Graph_2
plot(CluR$height,30:1,type="b",cex=0.7,xlab="Distance",ylab="Number of Cluster",main="Scree Plot")
```

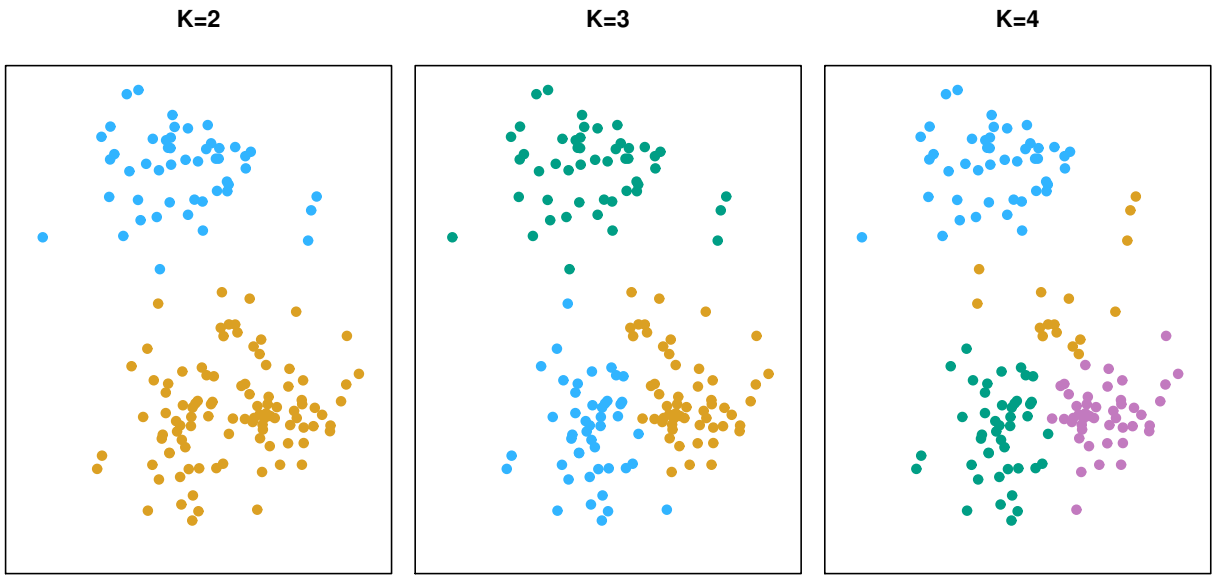



```
#Graph_3
PoData$memb<-cutree(CluR,k=4)
table(PoData$memb)

##
##  1  2  3  4
##  7  7 13  4

plot(PoData$memb,pch=PoData$memb,ylab="Type",xlab="province",main="4 clusters",axes=FALSE)
par(las=2)
axis(1,at=1:31,labels=PoData$province,cex.axis=0.6)
axis(2,at=1:4,labels=1:4,cex.axis=0.6)
box()
```




## 2.5 K-Means Clustering

Hierarchical clustering requires a distance or similarity matrix between all pairs of cases. That's a humongous matrix if you have tens of thousands of cases trapped in your data file. Even today's computers will take pause, as will you, waiting for results.




**The K-means algorithm:**

1. Selects K centroids (K rows chosen at random)
2. Assigns each data point to its closest centroid
3. Recalculates the centroids as the average of all data points in a cluster (i.e., the centroids are p-length mean vectors, where p is the number of variables)
4. Assigns data points to their closest centroids
5. Continues steps 3 and 4 until the observations are not reassigned or the maximum number of iterations (R uses 10 as a default) is reached.

R uses an efficient algorithm by Hartigan and Wong (1979) that partitions the observations into k groups such that the sum of squares of the observations to their assigned cluster centers is a minimum. This means that in steps 2 and 4, each observation is assigned to the cluster with the smallest value of:

$$SS(k) = \sum_{i=1}^{n} \sum_{j=0}^{p} (x_{ij} - \bar{x}_{kj})^2 \tag{3}$$

Where $k$ is the cluster, $x_{ij}$ is the value of the jth variable for the ith observation, and $\bar{x}_{kj}$ is the mean of the jth variable for the kth cluster.

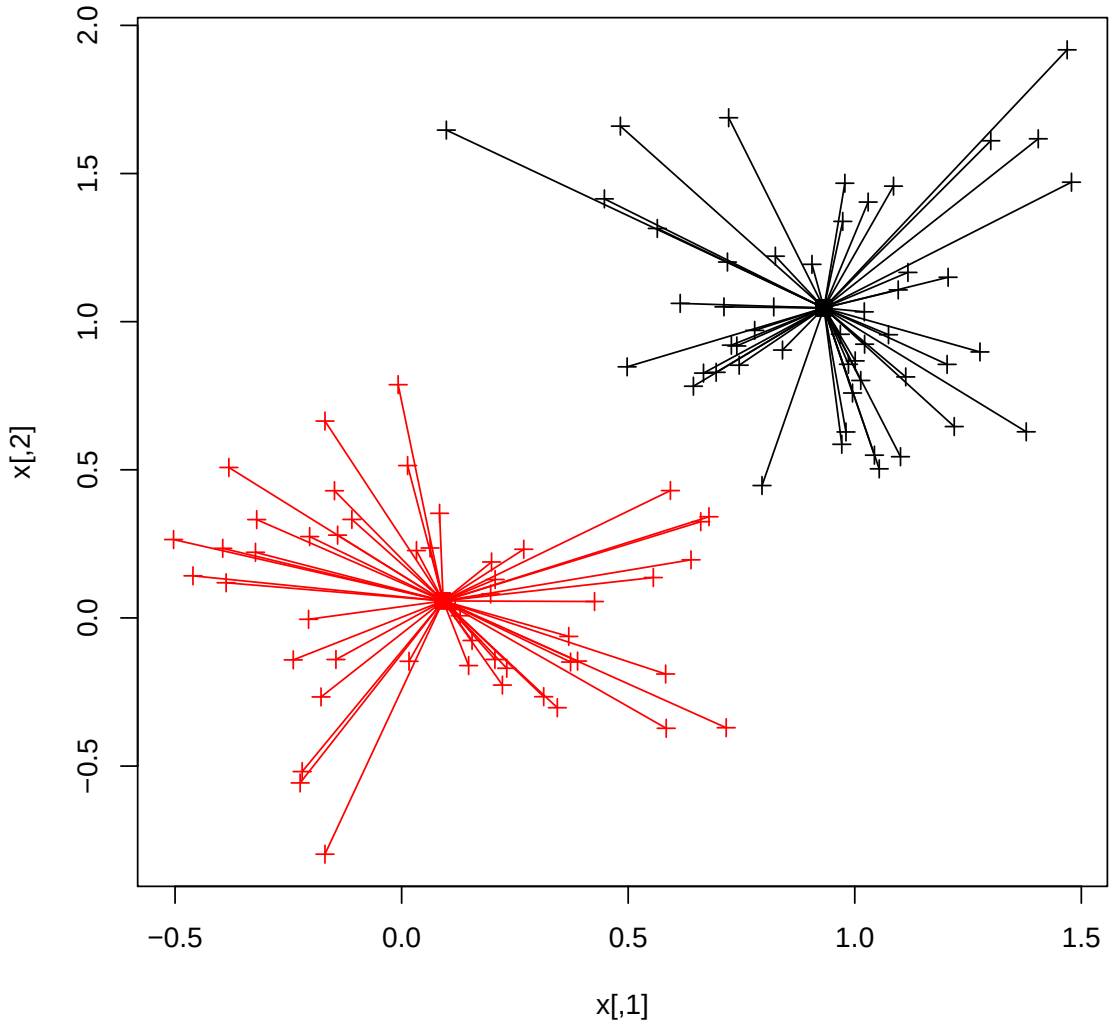

Because the K-means algorithm finds a local rather than a global optimum, the results obtained will depend on the initial (random) cluster assignment of each observation in Step 1 of choosing initial configurations. For this reason, it is important to run the algorithm multiple times from different random initial configurations. Then one selects the best solution, i.e. that for which the objective (3) is smallest.

The format of the K-means function in R is kmeans(x, centers) where x is a numeric dataset (matrix or data frame) and centers is the number of clusters to extract. The function returns the cluster memberships, centroids, sums of squares (within, between, total), and cluster sizes.

Since K-means cluster analysis starts with k randomly chosen centroids, a different solution can be obtained each time the function is invoked. Use the set.seed() function to guarantee that the results are reproducible. Additionally, this clustering approach can be sensitive to the initial selection of centroids. The kmeans() function has an nstart option that attempts multiple initial configurations and reports on the best one.

```
x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
cl <- kmeans(x, 2)
plot(x, col = cl$cluster, pch=3, lwd=1)
points(cl$centers, col = 1:2, pch = 7, lwd=3)
segments( x[cl$cluster==1,][,1], x[cl$cluster==1,][,2],
```

```
          cl$centers[1,1], cl$centers[1,2])
segments( x[cl$cluster==2,][,1], x[cl$cluster==2,][,2],
          cl$centers[2,1], cl$centers[2,2],
          col=2)
```




Whenever you use a statistical procedure that calculates distances, you have to worry about the impact of the different units in which variables are measured. Variables that have large values will have a large impact on the distance compared to variables that have smaller values.

```
newiris <- iris
newiris

##     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
## 7           4.6         3.4          1.4         0.3     setosa
## 8           5.0         3.4          1.5         0.2     setosa
## 9           4.4         2.9          1.4         0.2     setosa
## 10          4.9         3.1          1.5         0.1     setosa
## 11          5.4         3.7          1.5         0.2     setosa
## 12          4.8         3.4          1.6         0.2     setosa
## 13          4.8         3.0          1.4         0.1     setosa
## 14          4.3         3.0          1.1         0.1     setosa
## 15          5.8         4.0          1.2         0.2     setosa
## 16          5.7         4.4          1.5         0.4     setosa
## 17          5.4         3.9          1.3         0.4     setosa
## 18          5.1         3.5          1.4         0.3     setosa
## 19          5.7         3.8          1.7         0.3     setosa
## 20          5.1         3.8          1.5         0.3     setosa
## 21          5.4         3.4          1.7         0.2     setosa
## 22          5.1         3.7          1.5         0.4     setosa
## 23          4.6         3.6          1.0         0.2     setosa
## 24          5.1         3.3          1.7         0.5     setosa
## 25          4.8         3.4          1.9         0.2     setosa
## 26          5.0         3.0          1.6         0.2     setosa
## 27          5.0         3.4          1.6         0.4     setosa
## 28          5.2         3.5          1.5         0.2     setosa
## 29          5.2         3.4          1.4         0.2     setosa
## 30          4.7         3.2          1.6         0.2     setosa
## 31          4.8         3.1          1.6         0.2     setosa
## 32          5.4         3.4          1.5         0.4     setosa
## 33          5.2         4.1          1.5         0.1     setosa
## 34          5.5         4.2          1.4         0.2     setosa
## 35          4.9         3.1          1.5         0.2     setosa
## 36          5.0         3.2          1.2         0.2     setosa
## 37          5.5         3.5          1.3         0.2     setosa
## 38          4.9         3.6          1.4         0.1     setosa
## 39          4.4         3.0          1.3         0.2     setosa
## 40          5.1         3.4          1.5         0.2     setosa
## 41          5.0         3.5          1.3         0.3     setosa
## 42          4.5         2.3          1.3         0.3     setosa
## 43          4.4         3.2          1.3         0.2     setosa
## 44          5.0         3.5          1.6         0.6     setosa
## 45          5.1         3.8          1.9         0.4     setosa
## 46          4.8         3.0          1.4         0.3     setosa
## 47          5.1         3.8          1.6         0.2     setosa
## 48          4.6         3.2          1.4         0.2     setosa
## 49          5.3         3.7          1.5         0.2     setosa
## 50          5.0         3.3          1.4         0.2     setosa
## 51          7.0         3.2          4.7         1.4 versicolor
## 52          6.4         3.2          4.5         1.5 versicolor
## 53          6.9         3.1          4.9         1.5 versicolor
```

```
## 54          5.5         2.3          4.0         1.3 versicolor
## 55          6.5         2.8          4.6         1.5 versicolor
## 56          5.7         2.8          4.5         1.3 versicolor
## 57          6.3         3.3          4.7         1.6 versicolor
## 58          4.9         2.4          3.3         1.0 versicolor
## 59          6.6         2.9          4.6         1.3 versicolor
## 60          5.2         2.7          3.9         1.4 versicolor
## 61          5.0         2.0          3.5         1.0 versicolor
## 62          5.9         3.0          4.2         1.5 versicolor
## 63          6.0         2.2          4.0         1.0 versicolor
## 64          6.1         2.9          4.7         1.4 versicolor
## 65          5.6         2.9          3.6         1.3 versicolor
## 66          6.7         3.1          4.4         1.4 versicolor
## 67          5.6         3.0          4.5         1.5 versicolor
## 68          5.8         2.7          4.1         1.0 versicolor
## 69          6.2         2.2          4.5         1.5 versicolor
## 70          5.6         2.5          3.9         1.1 versicolor
## 71          5.9         3.2          4.8         1.8 versicolor
## 72          6.1         2.8          4.0         1.3 versicolor
## 73          6.3         2.5          4.9         1.5 versicolor
## 74          6.1         2.8          4.7         1.2 versicolor
## 75          6.4         2.9          4.3         1.3 versicolor
## 76          6.6         3.0          4.4         1.4 versicolor
## 77          6.8         2.8          4.8         1.4 versicolor
## 78          6.7         3.0          5.0         1.7 versicolor
## 79          6.0         2.9          4.5         1.5 versicolor
## 80          5.7         2.6          3.5         1.0 versicolor
## 81          5.5         2.4          3.8         1.1 versicolor
## 82          5.5         2.4          3.7         1.0 versicolor
## 83          5.8         2.7          3.9         1.2 versicolor
## 84          6.0         2.7          5.1         1.6 versicolor
## 85          5.4         3.0          4.5         1.5 versicolor
## 86          6.0         3.4          4.5         1.6 versicolor
## 87          6.7         3.1          4.7         1.5 versicolor
## 88          6.3         2.3          4.4         1.3 versicolor
## 89          5.6         3.0          4.1         1.3 versicolor
## 90          5.5         2.5          4.0         1.3 versicolor
## 91          5.5         2.6          4.4         1.2 versicolor
## 92          6.1         3.0          4.6         1.4 versicolor
## 93          5.8         2.6          4.0         1.2 versicolor
## 94          5.0         2.3          3.3         1.0 versicolor
## 95          5.6         2.7          4.2         1.3 versicolor
## 96          5.7         3.0          4.2         1.2 versicolor
## 97          5.7         2.9          4.2         1.3 versicolor
## 98          6.2         2.9          4.3         1.3 versicolor
## 99          5.1         2.5          3.0         1.1 versicolor
## 100         5.7         2.8          4.1         1.3 versicolor
## 101         6.3         3.3          6.0         2.5  virginica
## 102         5.8         2.7          5.1         1.9  virginica
## 103         7.1         3.0          5.9         2.1  virginica
## 104         6.3         2.9          5.6         1.8  virginica
```

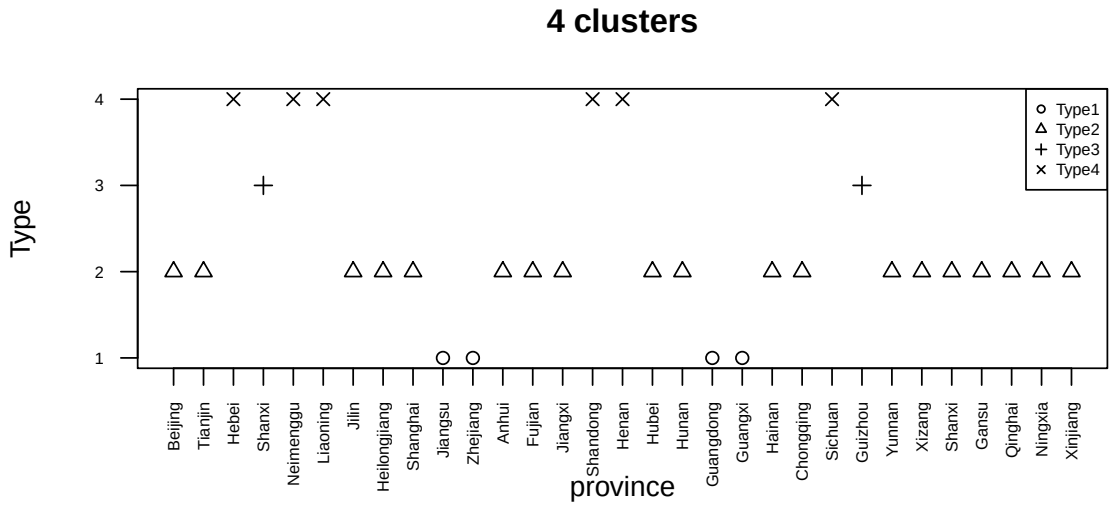
```
## 105          6.5         3.0          5.8         2.2  virginica
## 106          7.6         3.0          6.6         2.1  virginica
## 107          4.9         2.5          4.5         1.7  virginica
## 108          7.3         2.9          6.3         1.8  virginica
## 109          6.7         2.5          5.8         1.8  virginica
## 110          7.2         3.6          6.1         2.5  virginica
## 111          6.5         3.2          5.1         2.0  virginica
## 112          6.4         2.7          5.3         1.9  virginica
## 113          6.8         3.0          5.5         2.1  virginica
## 114          5.7         2.5          5.0         2.0  virginica
## 115          5.8         2.8          5.1         2.4  virginica
## 116          6.4         3.2          5.3         2.3  virginica
## 117          6.5         3.0          5.5         1.8  virginica
## 118          7.7         3.8          6.7         2.2  virginica
## 119          7.7         2.6          6.9         2.3  virginica
## 120          6.0         2.2          5.0         1.5  virginica
## 121          6.9         3.2          5.7         2.3  virginica
## 122          5.6         2.8          4.9         2.0  virginica
## 123          7.7         2.8          6.7         2.0  virginica
## 124          6.3         2.7          4.9         1.8  virginica
## 125          6.7         3.3          5.7         2.1  virginica
## 126          7.2         3.2          6.0         1.8  virginica
## 127          6.2         2.8          4.8         1.8  virginica
## 128          6.1         3.0          4.9         1.8  virginica
## 129          6.4         2.8          5.6         2.1  virginica
## 130          7.2         3.0          5.8         1.6  virginica
## 131          7.4         2.8          6.1         1.9  virginica
## 132          7.9         3.8          6.4         2.0  virginica
## 133          6.4         2.8          5.6         2.2  virginica
## 134          6.3         2.8          5.1         1.5  virginica
## 135          6.1         2.6          5.6         1.4  virginica
## 136          7.7         3.0          6.1         2.3  virginica
## 137          6.3         3.4          5.6         2.4  virginica
## 138          6.4         3.1          5.5         1.8  virginica
## 139          6.0         3.0          4.8         1.8  virginica
## 140          6.9         3.1          5.4         2.1  virginica
## 141          6.7         3.1          5.6         2.4  virginica
## 142          6.9         3.1          5.1         2.3  virginica
## 143          5.8         2.7          5.1         1.9  virginica
## 144          6.8         3.2          5.9         2.3  virginica
## 145          6.7         3.3          5.7         2.5  virginica
## 146          6.7         3.0          5.2         2.3  virginica
## 147          6.3         2.5          5.0         1.9  virginica
## 148          6.5         3.0          5.2         2.0  virginica
## 149          6.2         3.4          5.4         2.3  virginica
## 150          5.9         3.0          5.1         1.8  virginica

newiris$Species <- NULL
kc <- kmeans(newiris, 3)
names(kc)

## [1] "cluster"      "centers"      "totss"        "withinss"
```

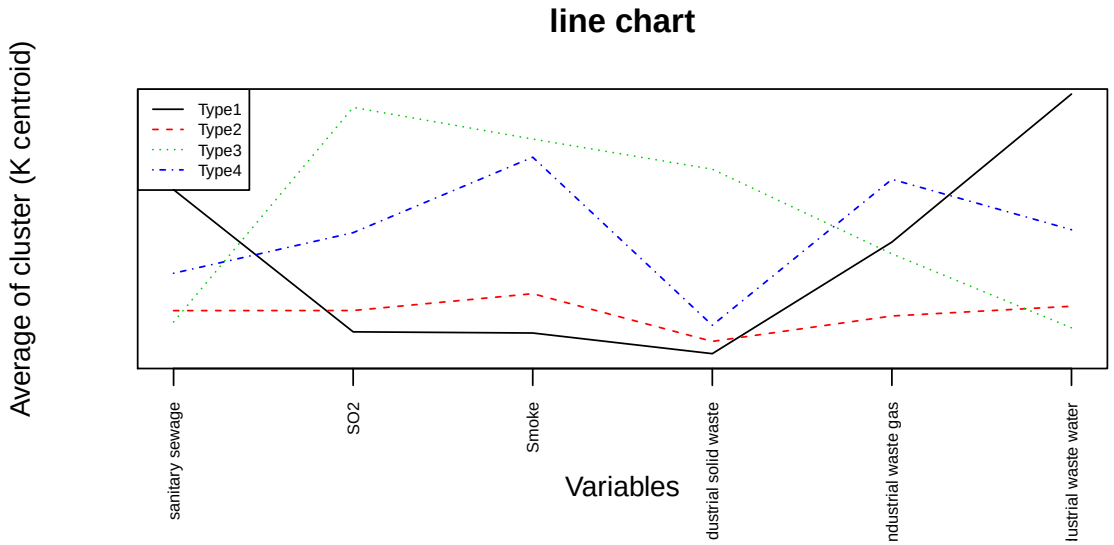
```
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"

kc$cluster

##   [1] 2 1 1 1 2 2 2 2 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 1 1 2 2 2 1 1 2 2 2 1
##  [36] 2 2 2 1 2 2 1 1 2 2 1 2 1 2 2 3 3 3 3 3 3 3 1 3 3 1 3 3 3 3 3 3 3 3
##  [71] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3 3 3 3
## [106] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3

table(iris$Species, kc$cluster)

##
##                1  2  3
##   setosa      17 33  0
##   versicolor   4  0 46
##   virginica    0  0 50
```

K-means clustering can handle larger datasets than hierarchical cluster approaches. Additionally, observations are not permanently committed to a cluster. They are moved when doing so improves the overall solution. However, the use of means implies that all variables must be continuous and the approach can be severely affected by outliers. They also perform poorly in the presence of non-convex (e.g., U-shaped) clusters.

```
PoData<-read.table(file="PollutionData1.txt",header=TRUE)
CluData<-PoData[,2:7]
set.seed(12345)
CluR<-kmeans(x=CluData,centers=4,nstart=4,iter.max=10)
CluR$size

## [1]  4 19  2  6

CluR$centers

##         x1       x2       x3       x4       x5       x6
## 1 53.39250  8.33500  7.97000  1.42250 36.78750 83.69250
## 2 15.06895 15.09263 20.43263  5.31000 13.37316 16.45105
## 3 11.48000 79.47000 69.43000 59.88000 33.07000  9.62000
## 4 26.91000 39.77167 63.68333 10.42833 56.67667 40.70000

opar<-par(no.readonly = TRUE)
par(mfrow=c(2,1))
#Graph_1
PoData$CluR<-CluR$cluster
plot(PoData$CluR,pch=PoData$CluR,ylab="Type",xlab="province",main="4 clusters",axes=FALSE)
par(las=2)
axis(1,at=1:31,labels=PoData$province,cex.axis=0.6)
axis(2,at=1:4,labels=1:4,cex.axis=0.6)
box()
legend("topright",c("Type1","Type2","Type3","Type4"),pch=1:4,cex=0.6)
#Graph_2
plot(CluR$centers[1,],type="l",ylim=c(0,82),xlab="Variables",
```

```
ylab="Average of cluster (K centroid)",main="line chart",axes=FALSE)
axis(1,at=1:6,labels=c("sanitary sewage","SO2","Smoke","Industrial solid waste",
"industrial waste gas","industrial waste water"),cex.axis=0.6)
box()
lines(1:6,CluR$centers[2,],lty=2,col=2)
lines(1:6,CluR$centers[3,],lty=3,col=3)
lines(1:6,CluR$centers[4,],lty=4,col=4)
legend("topleft",c("Type1","Type2","Type3","Type4"),lty=1:4,col=1:4,cex=0.6)
```




```
#And Graph_2
```

# 3 Discriminant Analysis

Discriminant analysis is used to distinguish distinct sets of observations and allocate new observations to previously defined groups. This method is commonly used in biological species classification, in medical classification of tumors, in facial recognition technologies, and in the credit card and insurance industries for determining risk.

- To train (create) a classifier
- To predict the classes of new data

Why do we need another method, when we have logistic regression? There are several reasons:

- When the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. Linear discriminant analysis does not suffer from this problem.
- If $n$ is small and the distribution of the predictors $X$ is approximately normal in each of the classes, the linear discriminant model is again more stable than the logistic regression model.
- Linear discriminant analysis is popular when we have more than two response classes.

## 3.1 Distance

### Euclidean distance

$$D_E^2(X, G_i) = (x - \mu)^T (x - \mu)$$

### Mahalanobis distance

The Mahalanobis distance of an observation $x = (x_1, x_2, x_3, \ldots, x_N)^T$ from a set of observations with mean $\mu = (\mu_1, \mu_2, \mu_3, \ldots, \mu_N)^T$ and covariance matrix $V$ is defined as

$$D_M^2(X, G_i) = (x - \mu)^T V^{-1} (x - \mu)$$

where $G_i$ is the population.

We can rewrite the above Equation as

$$D_M^2(X, G_i) = \sqrt{\frac{(x_1 - \mu_1)^2}{S_1^2} + \frac{(x_2 - \mu_2)^2}{S_2^2} + \ldots + \frac{(x_p - \mu_p)^2}{S_p^2}}$$

where $p$ is the number of discriminant variables.

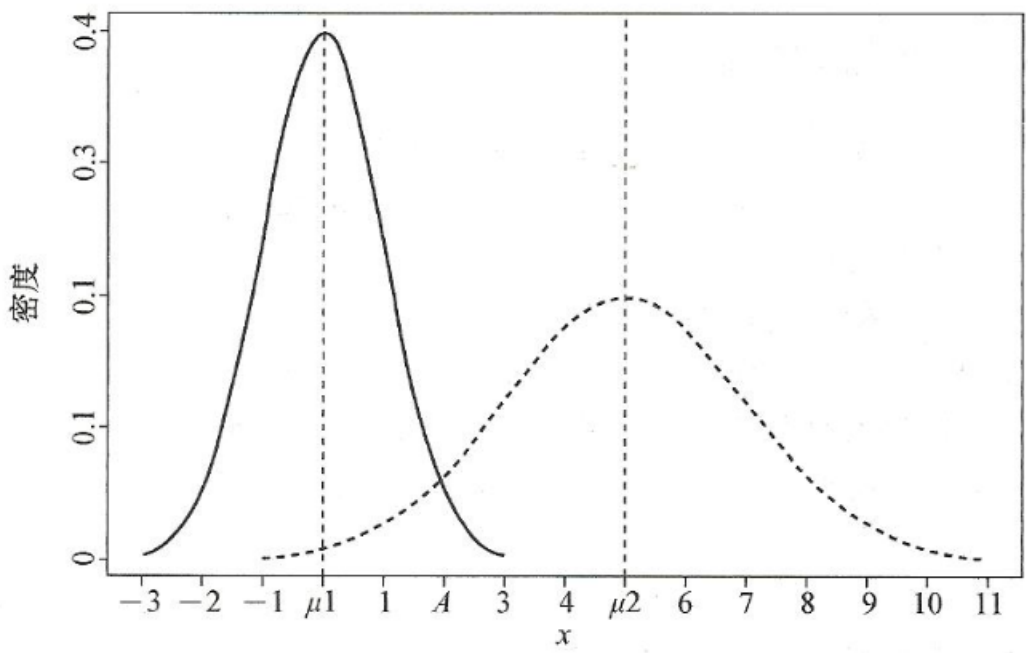



Figure 2: Two normal distribution with different parameters, $N(0,1)$ and $N(5,4)$

If $D_M^2(X,G_1) < D_M^2(X,G_2)$, Then $X \in G_1$; If $D_M^2(X,G_1) > D_M^2(X,G_2)$, Then $X \in G_2$.

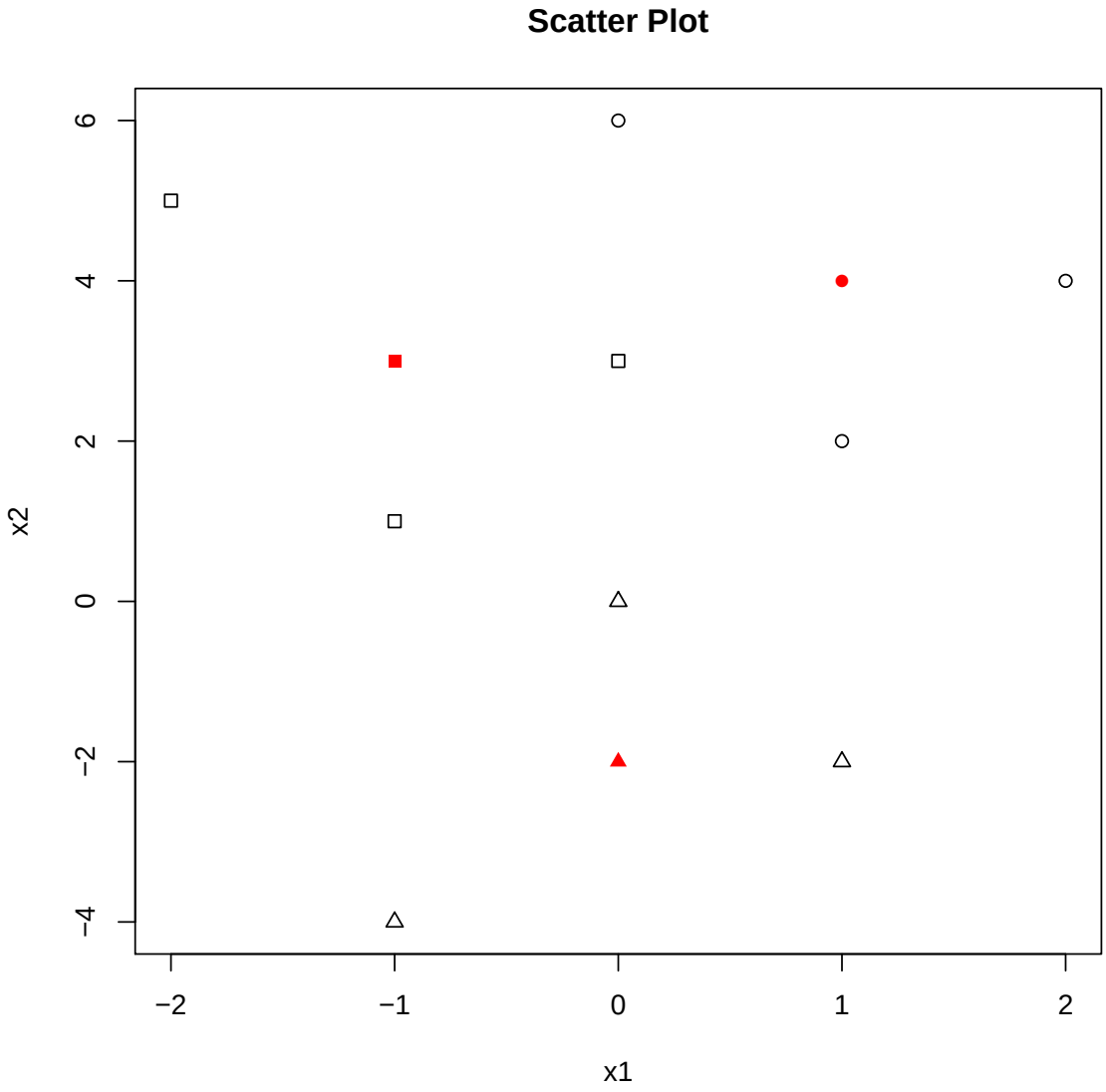
## An Example

| | $x_1^{(1)}$, $x_2^{(1)}$, $y$ | | $x_1^{(2)}$, $x_2^{(2)}$, $y$ | | $x_1^{(3)}$, $x_2^{(3)}$, $y$ |
|---|---|---|---|---|---|
| $\boldsymbol{X}_1^{(1)}$ | −2, 5, 1 | $\boldsymbol{X}_1^{(2)}$ | 0, 6, 2 | $\boldsymbol{X}_1^{(3)}$ | 1, −2, 3 |
| $\boldsymbol{X}_2^{(1)}$ | 0, 3, 1 | $\boldsymbol{X}_2^{(2)}$ | 2, 4, 2 | $\boldsymbol{X}_2^{(3)}$ | 0, 0, 3 |
| $\overline{\boldsymbol{X}}_3^{(1)}$ | −1, 1, 1 | $\overline{\boldsymbol{X}}_3^{(2)}$ | 1, 2, 2 | $\overline{\boldsymbol{X}}_3^{(3)}$ | −1, −4, 3 |
| $\overline{\boldsymbol{X}}^{(1)}$ | −1, 3 | $\overline{\boldsymbol{X}}^{(2)}$ | 1, 4 | $\overline{\boldsymbol{X}}^{(3)}$ | 0, −2 |

```
Data<-c(-2,5,1,0,3,1,-1,1,1,0,6,2,2,4,2,1,2,2,1,-2,3,0,0,3,-1,-4,3)
Data<-matrix(Data,nrow=9,ncol=3,byrow=TRUE)
Data
```

```
##       [,1] [,2] [,3]
##  [1,]   -2    5    1
##  [2,]    0    3    1
##  [3,]   -1    1    1
##  [4,]    0    6    2
##  [5,]    2    4    2
```

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

T1<-subset(Data[,1:2],Data[,3]==1)
T2<-subset(Data[,1:2],Data[,3]==2)
T3<-subset(Data[,1:2],Data[,3]==3)
CenterT1<-colMeans(T1)
CenterT2<-colMeans(T2)
CenterT3<-colMeans(T3)
Center<-rbind(CenterT1,CenterT2,CenterT3)
plot(Data[,1:2],pch=Data[,3]-1,xlab="x1",ylab="x2",main="Scatter Plot")
points(Center[,1],Center[,2],pch=c(15,16,17),col=2)
```



```r
#Calculate SSCP
S1<-(3-1)*cov(T1)
S2<-(3-1)*cov(T2)
S3<-(3-1)*cov(T3)
(S<-(S1+S2+S3)/(9-3))
```

```
##            [,1]       [,2]
## [1,]  1.0000000 -0.3333333
## [2,] -0.3333333  4.0000000
```

```r
#Calculate Mahalanobis distance
(mahalanobis(c(-2,5),center=CenterT1,cov=S))
```

```
## [1] 1.714286
```

```r
(mahalanobis(c(-2,5),center=CenterT2,cov=S))
```

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

```r
(mahalanobis(c(-2,5),center=CenterT3,cov=S))
```

```
## [1] 14.31429
```

## 3.2 Discriminant function

Discriminant function can be defined as

$$W(X) = D_M^2(X, G_1) - D_M^2(X, G_2) \tag{4}$$

The pooled within-groups covariance is defined assigned

$$V = \frac{1}{n_1 + n_2 - 2}(S_1 + S_2)$$

where $i$ is the sample size of $i$ class, and $S_i$ is sum of square and cross-product (SSCP) defined as

$$S_i = \sum_{j=1}^{n_i}(X_j^{(i)} - \bar{X}^{(i)})(X_j^{(i)} - \bar{X}^{(i)})^T, i = 1, 2$$

The equation (4) can be rewritten as

$$W(X) = (X - \bar{X})^T V^{-1}(\bar{X}^{(1)} - \bar{X}^{(2)})$$

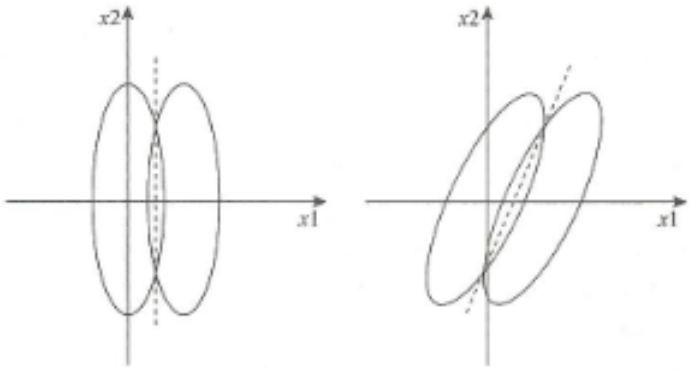
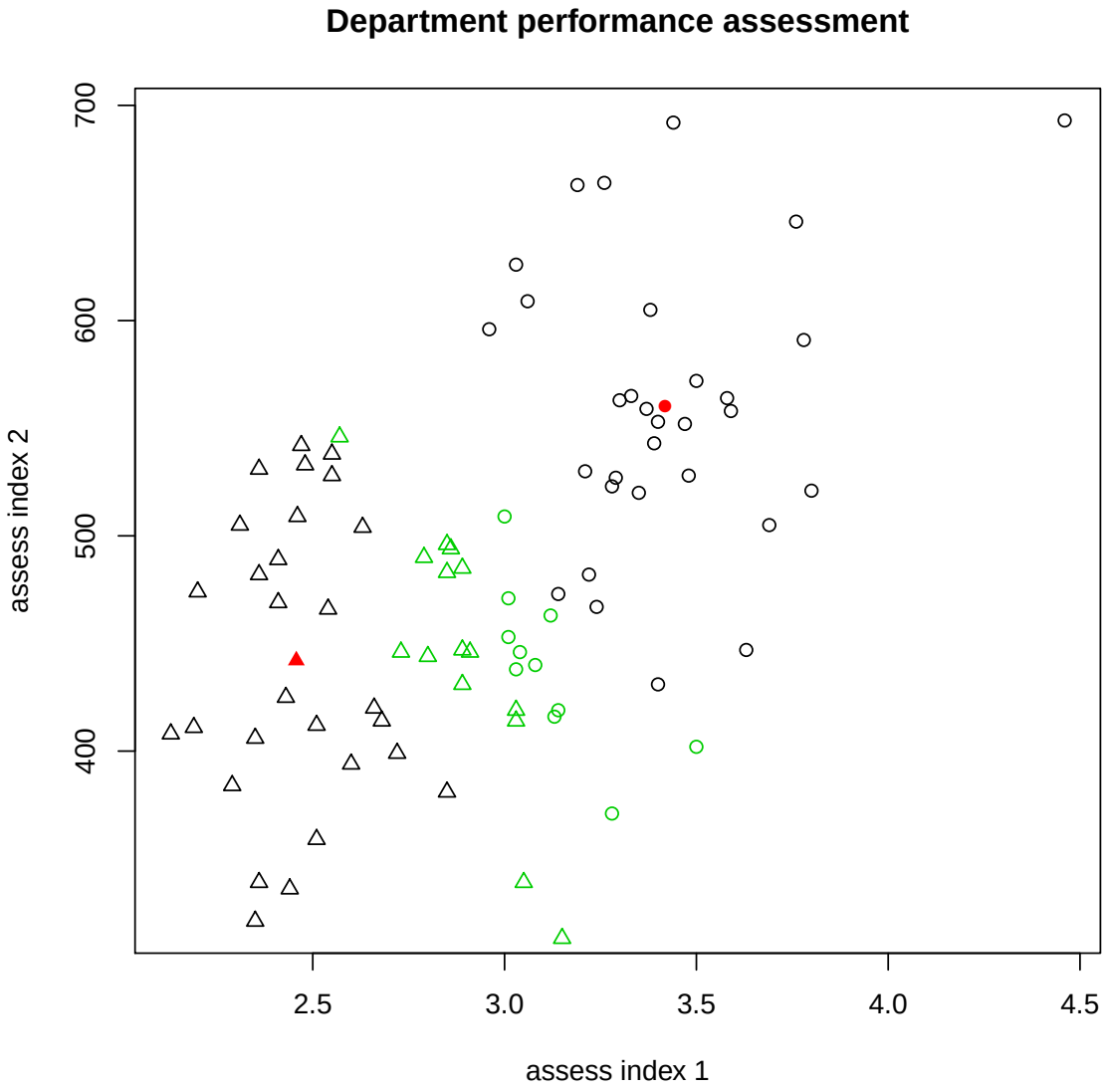



Figure 3: $D_M^2(X, G_1) = D_M^2(X, G_2)$

## An Example

```
EvaData<-read.table(file="EvaData.txt",header=TRUE)
head(EvaData)

##     X1  X2 Y
## 1 2.96 596 1
## 2 3.14 473 1
## 3 3.22 482 1
## 4 3.29 527 1
## 5 3.69 505 1
## 6 4.46 693 1

D1.Data<-subset(EvaData,EvaData$Y == 1 | EvaData$Y == 2 )
D2.Data<-subset(EvaData,EvaData$Y == 3)
plot(D1.Data[,1:2],pch=D1.Data$Y,xlab="assess index 1",
ylab="assess index 2",main="Department performance assessment")
T1<-subset(D1.Data[,1:2],D1.Data$Y==1)
T2<-subset(D1.Data[,1:2],D1.Data$Y==2)
CenterT1<-colMeans(T1)
CenterT2<-colMeans(T2)
Center<-rbind(CenterT1,CenterT2)
points(Center[,1],Center[,2],pch=c(16,17),col=2)
S1<-(length(T1$X1)-1)*cov(T1)
S2<-(length(T2$X1)-1)*cov(T2)
S<-(S1+S2)/(length(T1$X1)+length(T2$X1)-2)
for(i in 1:length(D2.Data$X1)){
    R1<-mahalanobis(D2.Data[i,1:2],center=CenterT1,cov=S)
    R2<-mahalanobis(D2.Data[i,1:2],center=CenterT2,cov=S)
    ifelse(R1<R2,D2.Data[i,3]<-1,D2.Data[i,3]<-2)
}
points(D2.Data[,1],D2.Data[,2],pch=D2.Data[,3],col=3)
```




### 3.3 Fisher discriminant analysis

In this case, you know the identity of each individual (unlike cluster analysis) and you want to know how the explanatory variables contribute to the correct classification of individuals. The method works by uncovering relationships among the groups' covariance matrices to discriminate between groups. With $k$ groups you will need $k-1$ discriminators. The functions you will need for discriminant analysis are available in the MASS library.

- Main idea: find projection to a line subject to samples from different classes are well separated.

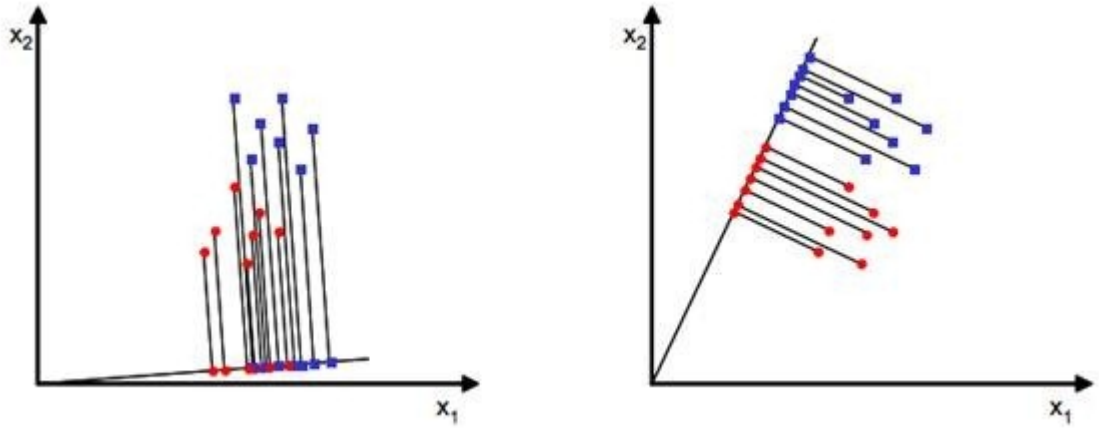



Figure 4: schematic diagram

Suppose we have 2 classes and d-dimensional samples $x_1, \ldots, x_n$ where $n_1$ samples come from the first class, and $n_2$ samples come from the second class. Consider projection on a line and Let the line direction be given by unit vector v. Scalar $V^t X_i$ is the distance of projection of $X_i$ from the origin. Thus it $V^t X_i$ is the projection of $X_i$ into a one dimensional subspace.

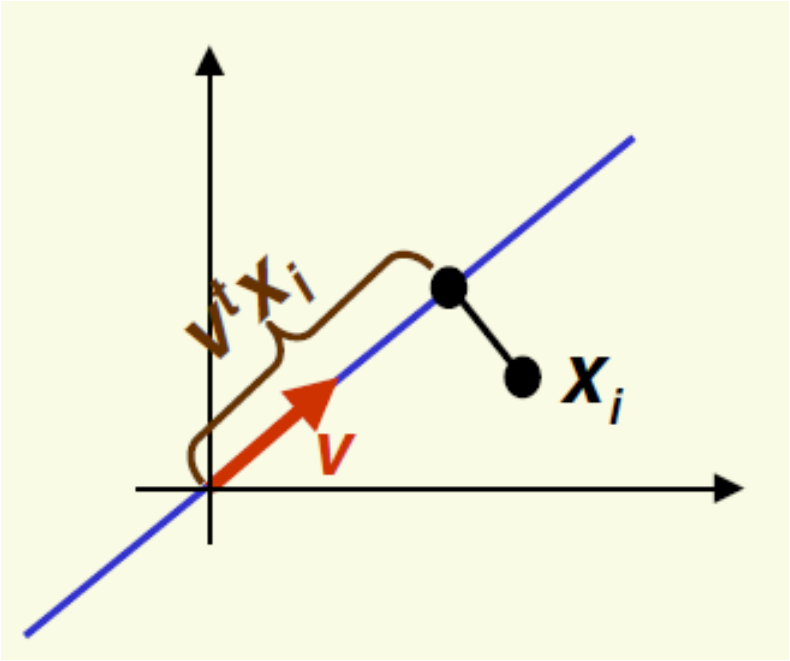



Figure 5: schematic diagram

Let $\tilde{\mu}_1$ and $\tilde{\mu}_2$ be the means of projections of classes 1 and 2, where

$$\tilde{\mu}_1 = \frac{1}{n_1} \sum_{X_i \in Class1}^{n_1} V^t X_i = V^t (\frac{1}{n_1} \sum_{X_i \in class1}^{n_1} X_i) = V^t \mu_1$$

The larger $|\tilde{\mu}_1 - \tilde{\mu}_2|$, the better is the expected separation.

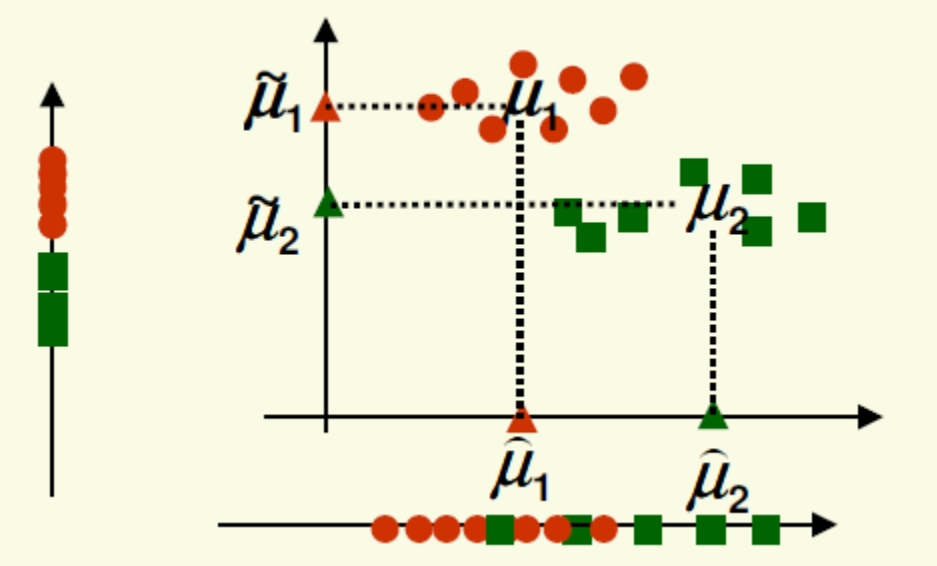



Figure 6: schematic diagram

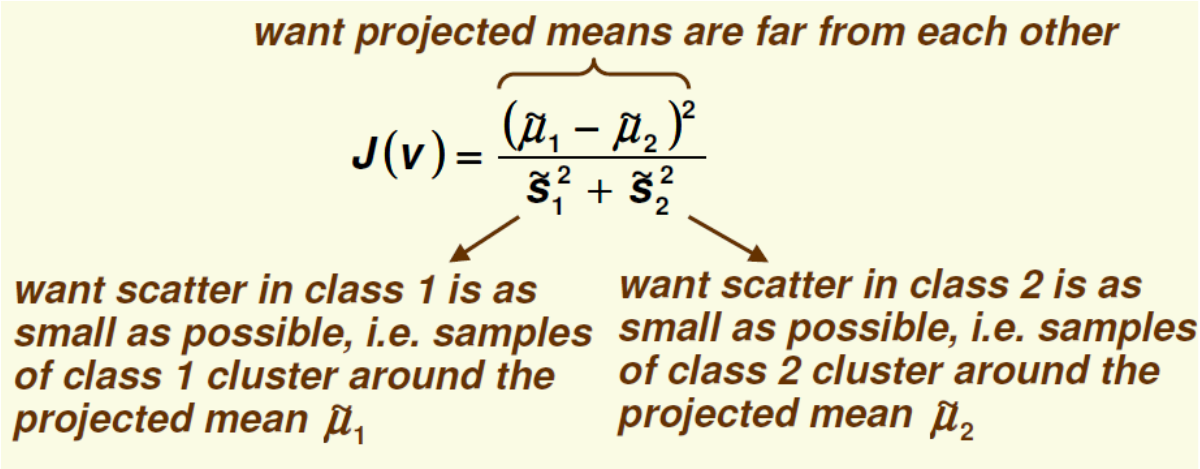
The vertical axes is a better line than the horizontal axes to project to for class separability. The problem with $|\tilde{\mu}_1 - \tilde{\mu}_2|$ is that it does not consider the variance of the classes. Let $y_i = V^t X_i$ , i.e. $y_i$ s are the projected samples. Scatter for projected samples of class 1 is

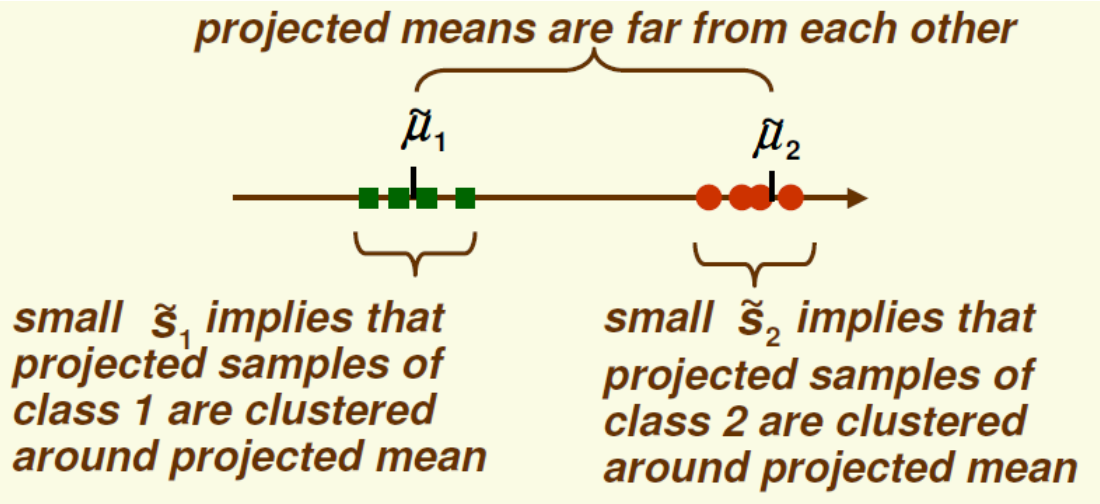
$$\tilde{s}_1 = \sum_{y_i \in Class1}^{n_1} (y_i - \tilde{\mu}_1)^2$$

Scatter for projected samples of class 2 is

$$\tilde{s}_2 = \sum_{y_i \in Class2}^{n_1} (y_i - \tilde{\mu}_2)^2$$

Thus Fisher linear discriminant is to project on line in the direction $V$ which maximizes.





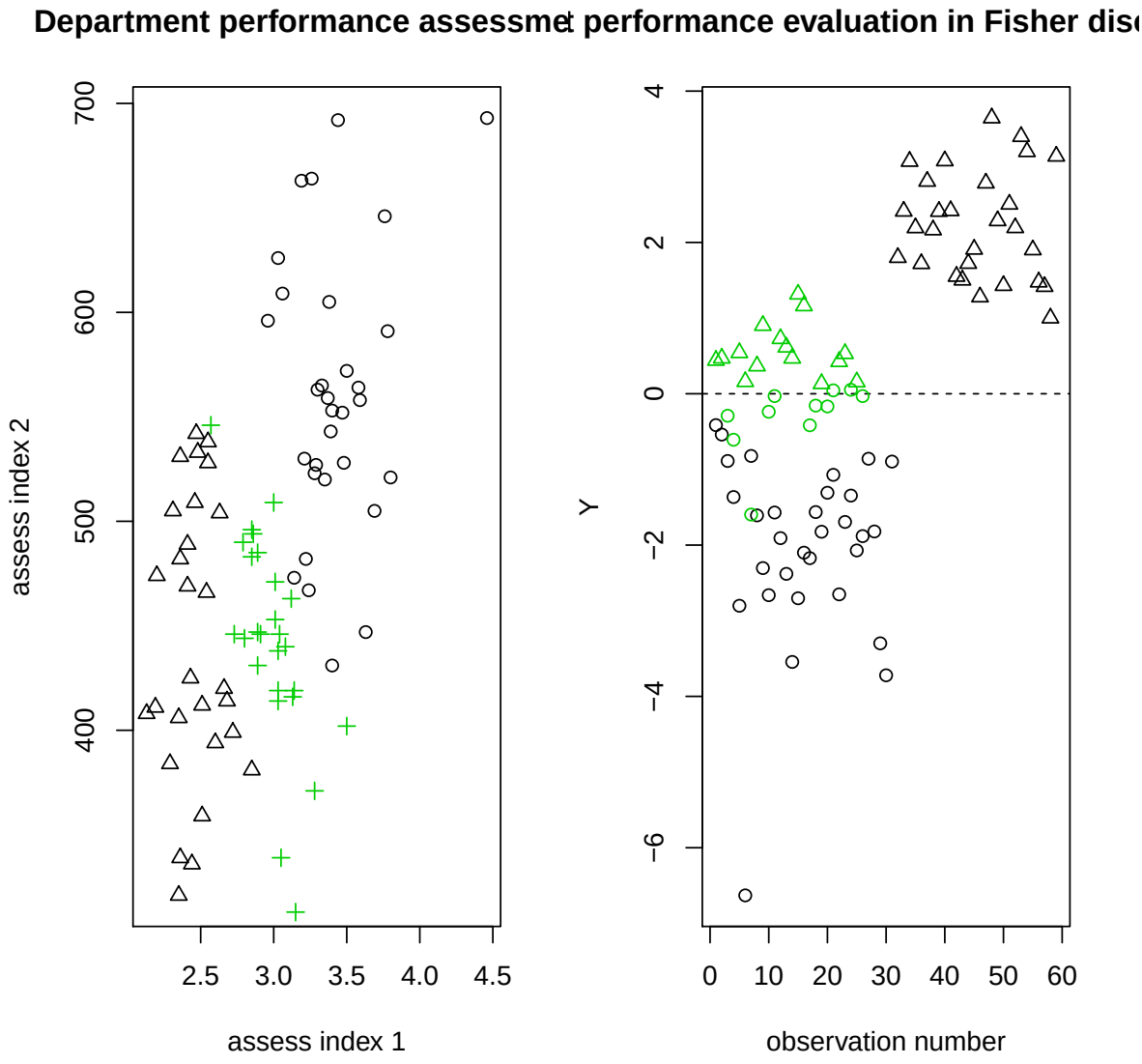

If we find v which makes $J(V)$ large, we are guaranteed that the classes are well separated.

### An Example

install.packages(MASS)

```
library("MASS")
EvaData<-read.table(file="EvaData.txt",header=TRUE)
D1.Data<-subset(EvaData,EvaData$Y == 1 | EvaData$Y == 2 )
D2.Data<-subset(EvaData,EvaData$Y == 3)
opar <- par(no.readonly = TRUE)
par(mfrow=c(1,2))
plot(D1.Data[,1:2],pch=D1.Data$Y,xlab="assess index 1",
ylab="assess index 2",main="Department performance assessment")
points(D2.Data[,1],D2.Data[,2],pch=D2.Data[,3],col=3)
(Result<-lda(Y~.,data=D1.Data))

## Call:
## lda(Y ~ ., data = D1.Data)
##
## Prior probabilities of groups:
##         1         2
## 0.5254237 0.4745763
##
## Group means:
##         X1       X2
## 1 3.418710 560.2581
## 2 2.457143 442.1071
##
## Coefficients of linear discriminants:
##             LD1
## X1 -3.845521581
## X2 -0.004610379

Y<-predict(Result,D1.Data)
```

```r
YN<-predict(Result,D2.Data)
length(which(D1.Data$Y==1))

## [1] 31

length(which(Y$class==1))

## [1] 31

plot(Y$x,pch=as.integer(as.vector(Y$class)),xlab="observation number",
ylab="Y",main="Department performance evaluation in Fisher discriminant space")
abline(h=0,lty=2)
points(1:26,YN$x,pch=as.integer(as.vector(YN$class)),col=3)
```



```r
par(opar)
```