Exploratory Data Analysis and Clustering: Part II

Thierry Denoeux

8/15/2022

Multidimensional data visualization

Principal Component Analysis

We will illustrate PCA with the `Country` dataset:

```
data<-read.csv(file="/Users/Thierry/Documents/R/Data/Economics/country-data.csv",
               header=TRUE, sep=",")
head(data)
```

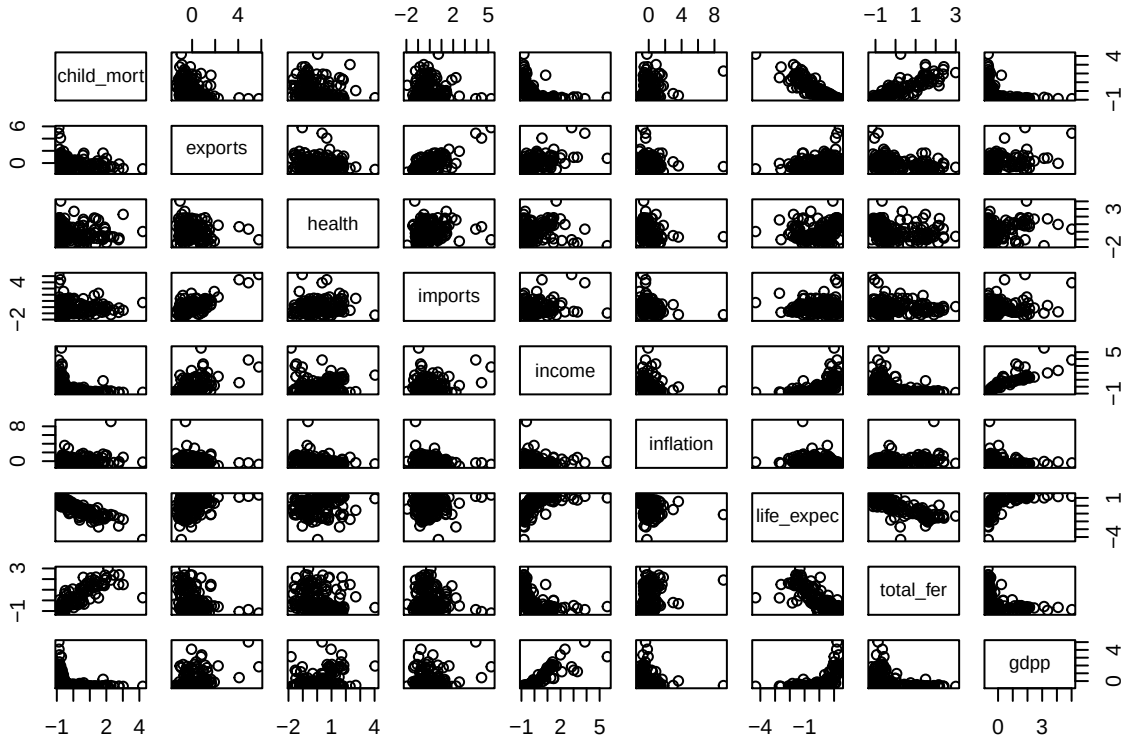
```
##               country child_mort exports health imports income inflation
## 1         Afghanistan       90.2    10.0   7.58    44.9   1610      9.44
## 2             Albania       16.6    28.0   6.55    48.6   9930      4.49
## 3             Algeria       27.3    38.4   4.17    31.4  12900     16.10
## 4              Angola      119.0    62.3   2.85    42.9   5900     22.40
## 5 Antigua and Barbuda       10.3    45.5   6.03    58.9  19100      1.44
## 6           Argentina       14.5    18.9   8.10    16.0  18700     20.90
##   life_expec total_fer  gdpp
## 1       56.2      5.82   553
## 2       76.3      1.65  4090
## 3       76.5      2.89  4460
## 4       60.1      6.16  3530
## 5       76.8      2.13 12200
## 6       75.8      2.37 10300
```

As the variables are heterogeneous, we scale the data:

```
x<-scale(data[,2:10])
row.names(x)<-data[,1]
x<-as.data.frame(x)
```

The matrix plot contains quite a lot of information and is difficult to interpret:

```
plot(x)
```



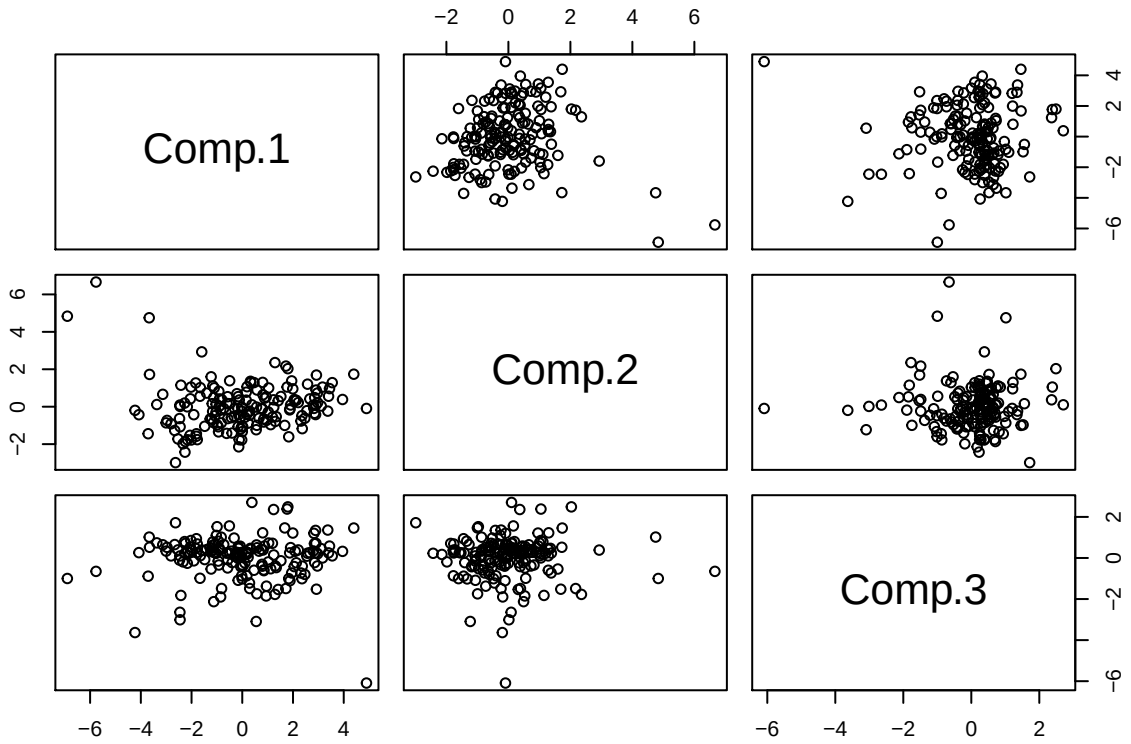
PCA is implement in function `princomp`:

```
pca<-princomp(x)
summary(pca)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3    Comp.4     Comp.5
## Standard deviation     2.0275335 1.2397930 1.0785986 0.9943982 0.81034760
## Proportion of Variance 0.4595174 0.1718163 0.1300426 0.1105316 0.07340211
## Cumulative Proportion  0.4595174 0.6313337 0.7613762 0.8719079 0.94530998
##                           Comp.6    Comp.7      Comp.8      Comp.9
## Standard deviation     0.47142583 0.3357968 0.296287904 0.257826602
## Proportion of Variance 0.02484235 0.0126043 0.009812817 0.007430556
## Cumulative Proportion  0.97015232 0.9827566 0.992569444 1.000000000
```

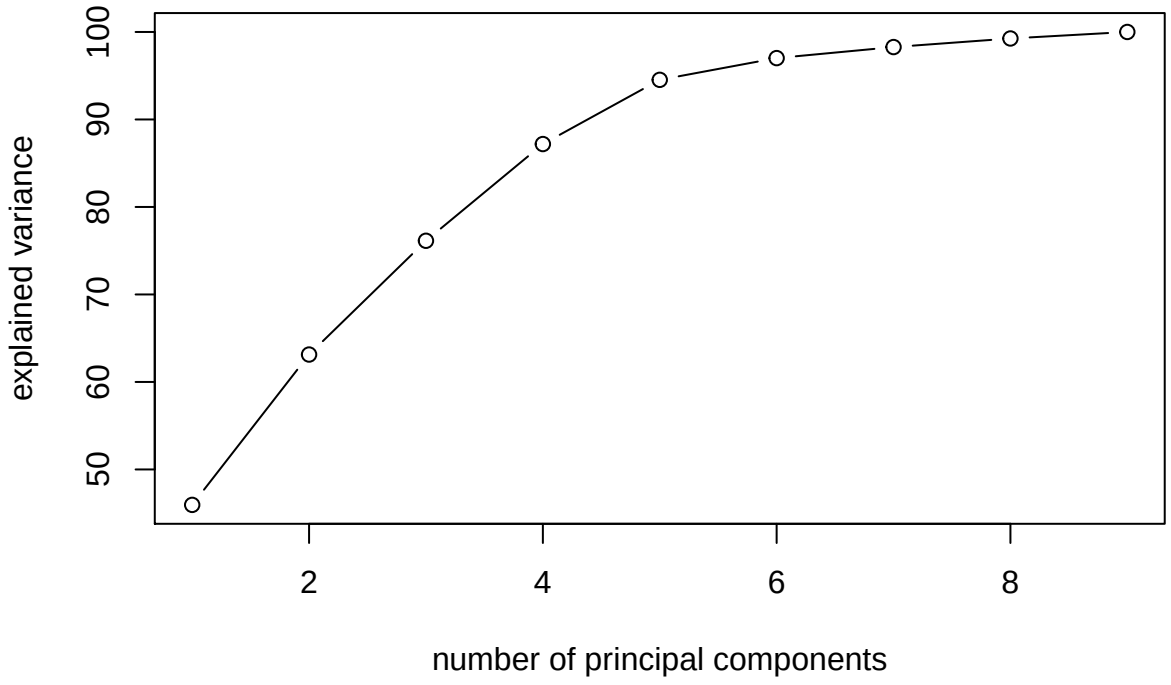
The principal component score are contained in matrix `pca$scores`. Let us draw a matrix plot of the first three principal components:

```
z<-pca$scores
pairs(z[,1:3])
```



To determine the number of principle components, let us plot the proportion of explained variance vs. the number of principal components. The standard deviations $\sqrt{\lambda_j}$ of the principle components are contained in vector `pca$sdec`:

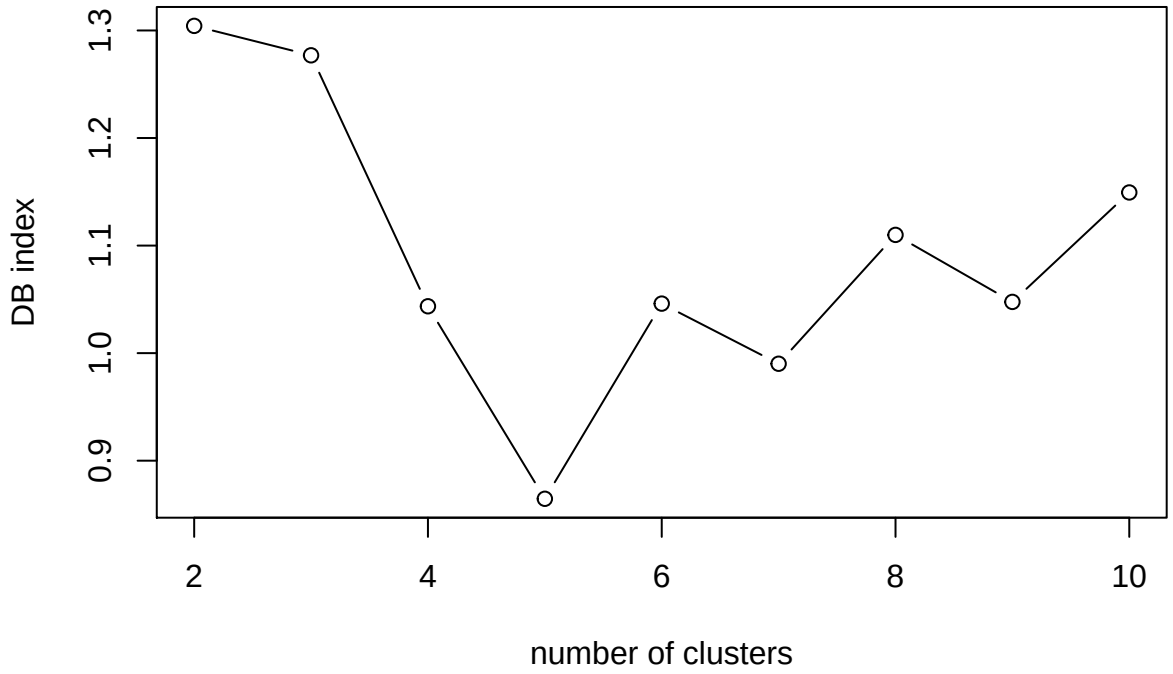
```
plot(cumsum(pca$sdev^2)/sum(pca$sdev^2)*100,xlab="number of principal components",
     ylab="explained variance",type="b")
```



The curve does not show a "knee" that could help us to select a "natural" number of dimensions. We will choose $q = 2$ for easy visualization.

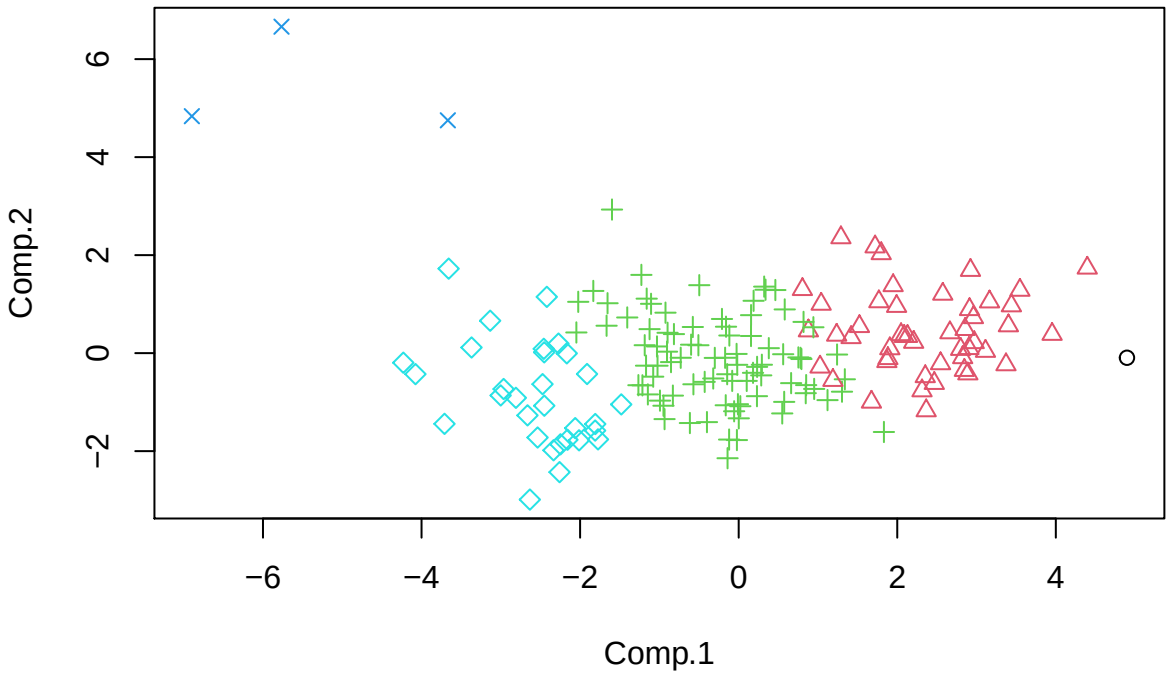
Let us use PCA to visualize the partition obtained by the HCM algorithm. We start by determining the number of clusters using the Davis-Bouldin index:

```
library(clusterCrit)
set.seed(220815)
C<- 2:10
N<-length(C)
DB<-rep(0,N)
for(i in 1:N){
  km<-kmeans(x,centers=C[i],nstart=50)
  DB[i]<-intCriteria(as.matrix(x), km$cluster, crit="Davies_Bouldin")
}
plot(C,DB,type="b",xlab="number of clusters",ylab="DB index")
```



The best number of clusters seems to be $c = 5$. Let us recompute and display the corresponding partition in the first plane spanned by the first two principal components:

```
km<-kmeans(x,centers=5,nstart=100)
plot(z[,1:2],col=km$cluster,pch=km$cluster)
```



There are two "small" classes with 1 and 3 observations, which can be considered as outliers:

```
table(km$cluster)
```

```
##
##  1  2  3  4  5
##  1 47 86  3 30
```

```
print(row.names(x)[which(km$cluster==1)])
```

```
## [1] "Nigeria"
```

```
print(row.names(x)[which(km$cluster==4)])
```

```
## [1] "Luxembourg" "Malta"      "Singapore"
```

Multidimensional scaling

MDS is implemented in package `smacof`. To apply MDS to the `country`data, we first compute a dissimilarity matrux, which, here, will actually be a distance matrix because we already have attributes. Let us use, for instance, the Manhattan distance:

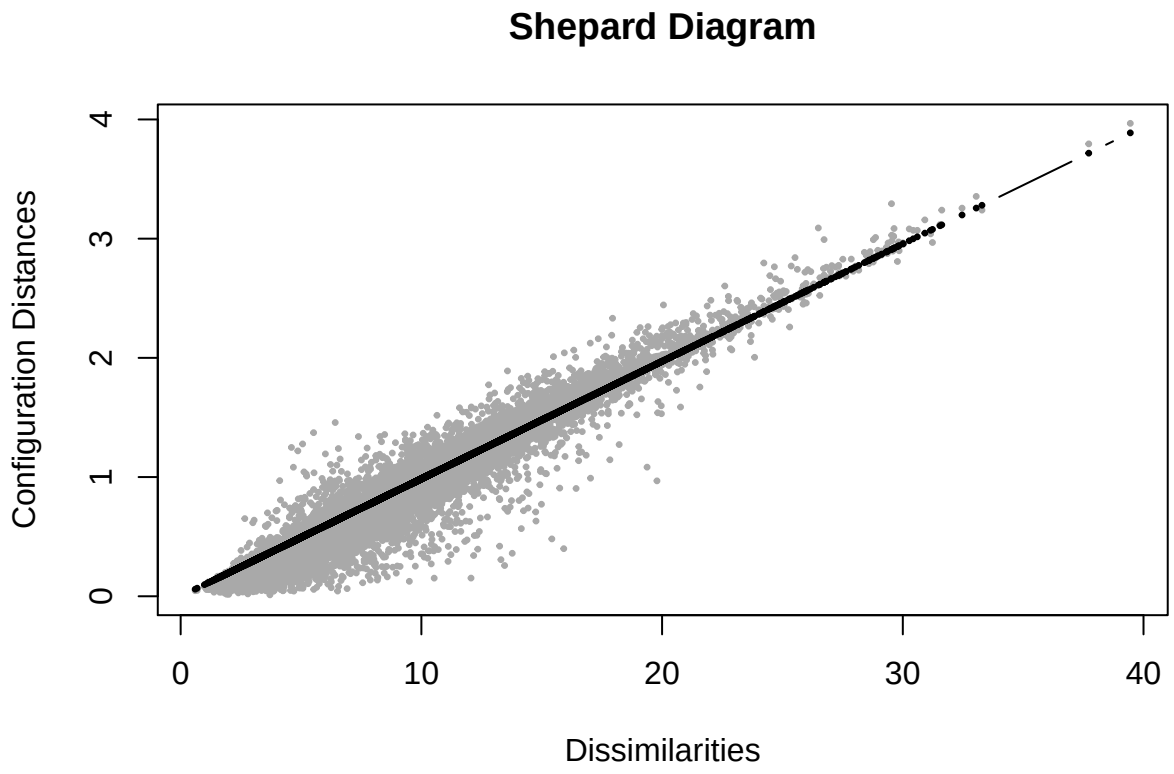
```
D<-dist(x,method="manhattan")
```

Next, we use function `mds`of package `smacof` to compute a 2D configuration using ratio MDS:

```
library(smacof)
D<-dist(x,method="manhattan")
fit<-mds(D,ndim=2,type="ratio")
```

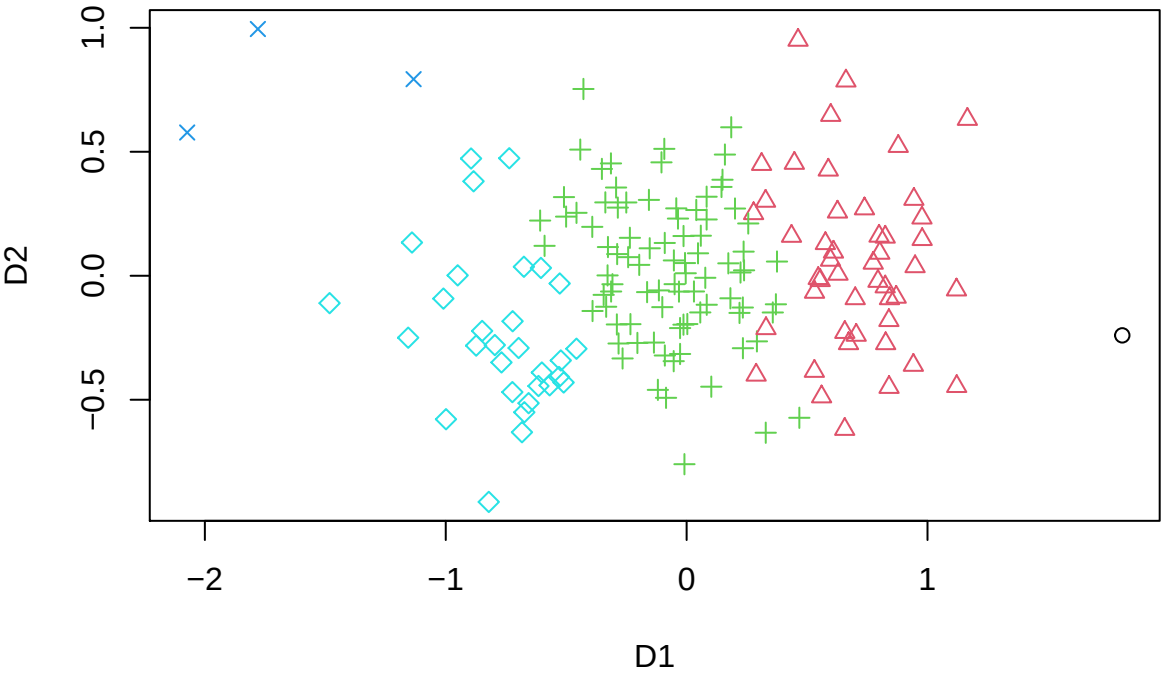
The goodness of fit of the model can be visually assessed using the Shepard diagram, which displays the configuration distances vs. the dissimilarities:

```
plot(fit,"Shepard")
```



We can now plot the previously found partition in the configuration space:

```
plot(fit$conf,col=km$cluster,pch=km$cluster)
```



Dissimilarity measures

Gower distance

To illustrate the use of the Gower distance, let us consider the datset `College` is package `ISLR`:

```
library(ISLR)
data(College)
```

This dataset contains statistics for 777 US Colleges from the 1995 issue of US News and World Report. It contains one binary variable and 17 numerical variables. To simplify the dataset and have a greater variery of data types, we will consider only five variables: a binary variable indicating whether the college is public or private, the acceptance rate defined as the ratio of number of applications accepted to the number of applications received, the logarithm of the number of new students enrolled, the out-of-state tuition fee, and an ordinal variable `prestige` with three levels indicating whether the percentage new students from top 10% of high-school classes is below 20%, between 20% and 50%, or above 50%:

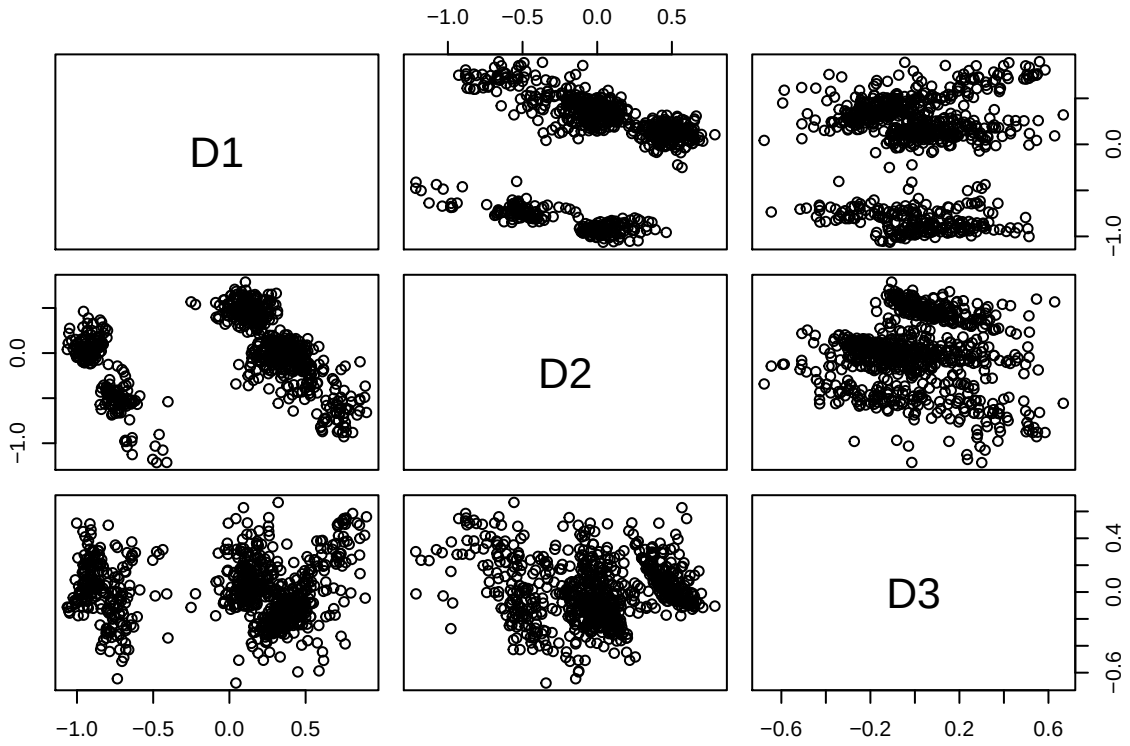
```
accept_rate<-College$Accept/College$Apps
tuition<-College$Outstate
enroll<-log(College$Enroll)
private<-College$Private
prestige<-cut(College$Top10perc,
           breaks = c(0,20,50,100),
           labels = c("low", "medium","high"),include.lowest=TRUE)
prestige<-as.ordered(prestige)
x<-data.frame(accept_rate,tuition,enroll,private,prestige)
```

To compute the Gower distance matrix, we use function `daisy` in package `cluster`:

```
library(cluster)
D<-daisy(x)
```

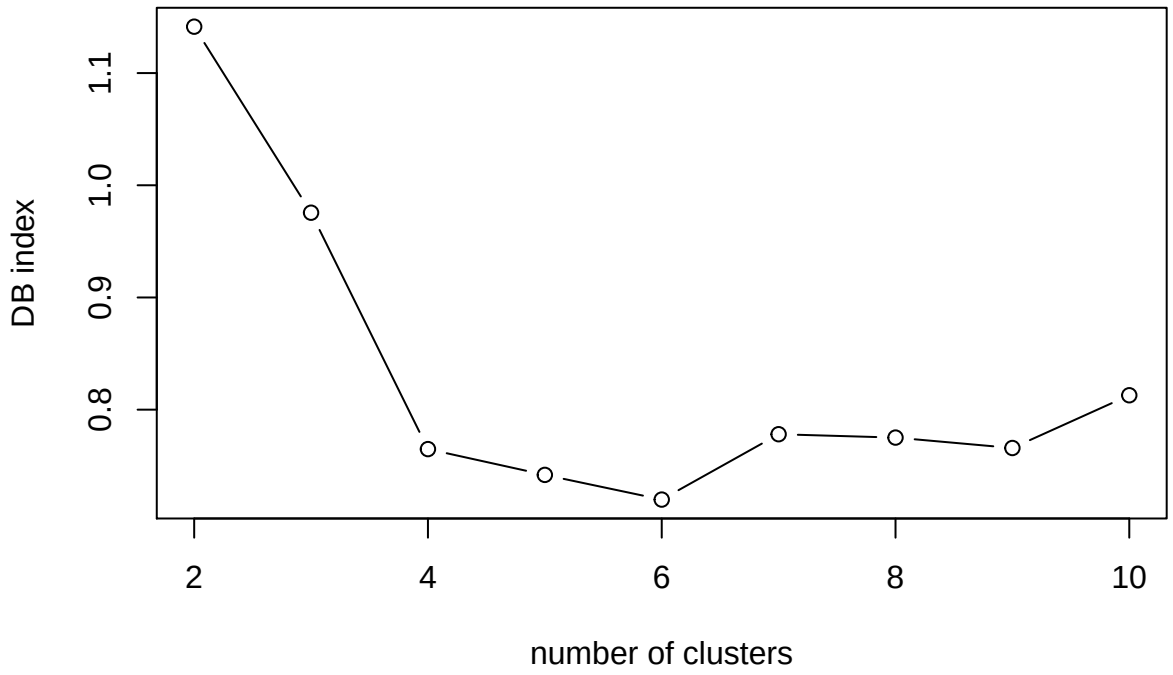
We can then compute a 3D configuration using MDS, and visualize the data:

```
fit<-mds(D,ndim=3,type="ordinal")
pairs(fit$conf)
```



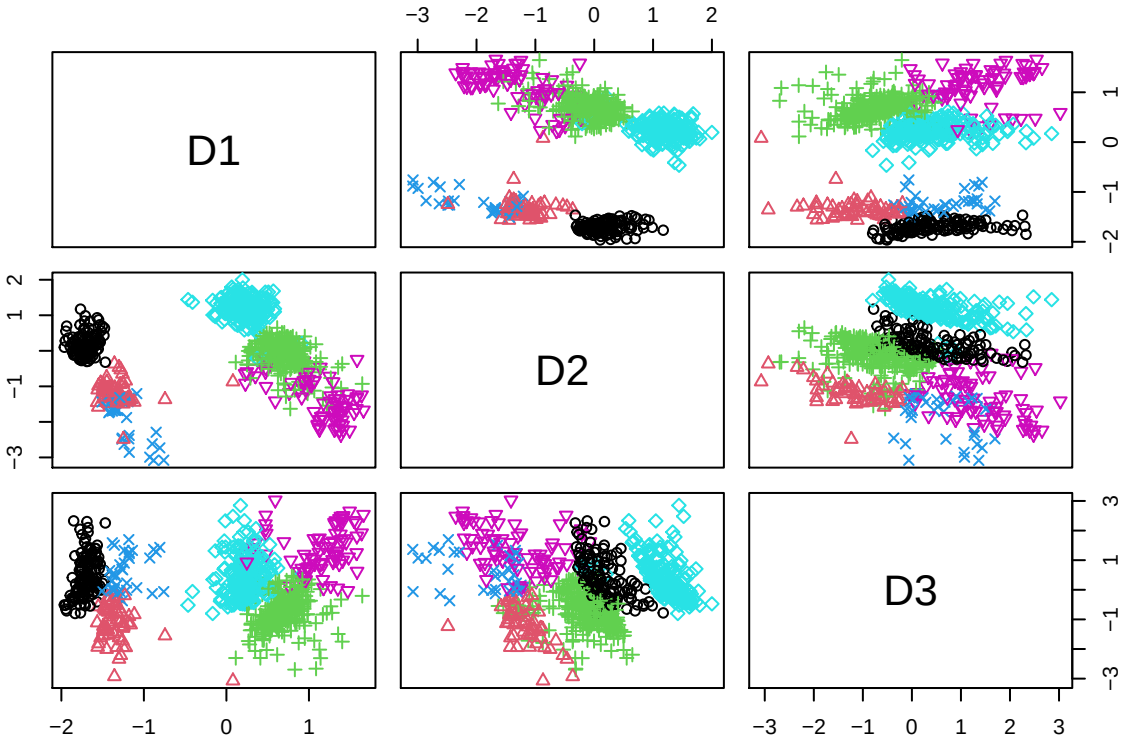
Let us now standardize the data, and compute the Davies-Bouldin index for partitions from 2 to 10 clusters computed by the HCM algorithm:

```
z<-scale(fit$conf)
C<- 2:10
N<-length(C)
DB<-rep(0,N)
for(i in 1:N){
  km<-kmeans(z,centers=C[i],nstart=100)
  DB[i]<-intCriteria(as.matrix(z), km$cluster, crit="Davies_Bouldin")
}
plot(C,DB,type="b",xlab="number of clusters",ylab="DB index")
```



This analysis suggests to select $c = 6$ clusters. Let us run again the HCM algorithm with 6 clusters and visualize the obtained partition:

```
km<-kmeans(z,centers=6,nstart=100)
pairs(z,col=km$cluster,pch=km$cluster)
```



Dynamic time warping

To illustrate the use of the DTW dissimilarity, we consider the `Synthetic Control Chart Time Series` dataset. It consists of 600 time series of length 60 belonging to six classes: "Normal", "Cyclic", "Increasing trend", "Decreasing trend", "Upward shift", and "Downward shift".

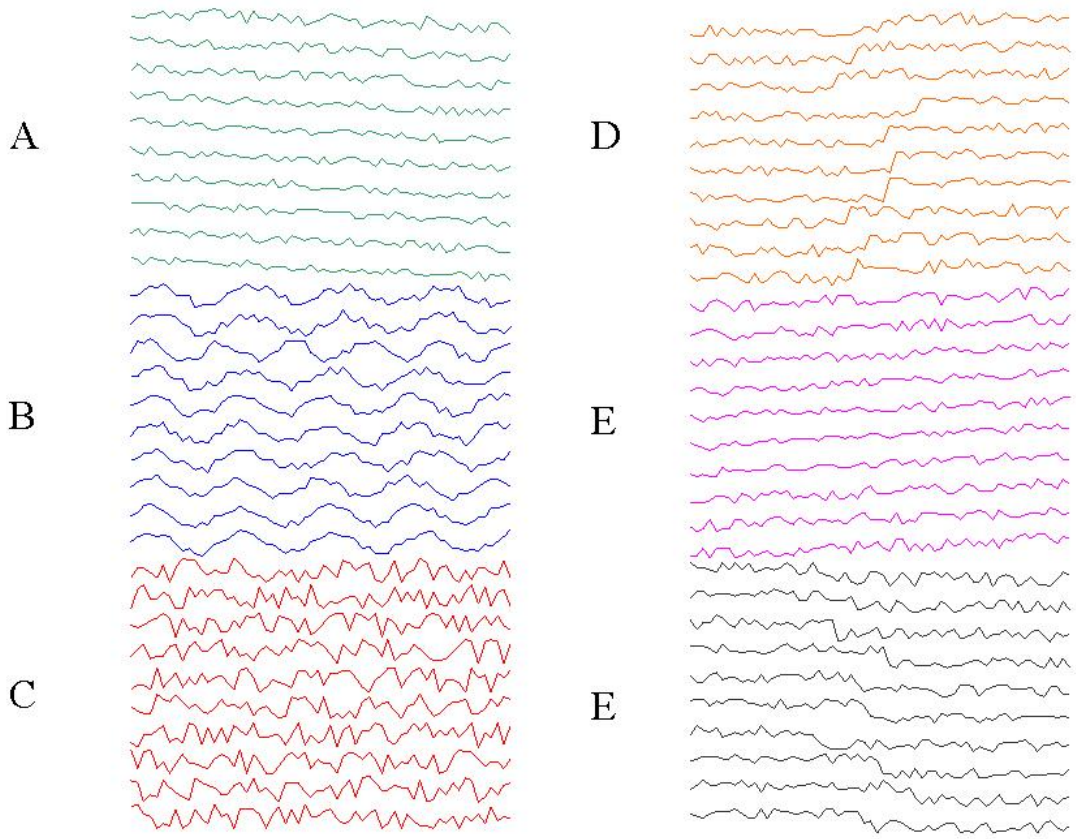


Figure 1: Examples of series from the six classes

We start by reading the data and making a vector containing the true classes for future comparison:

```
data<-read.table(file="/Users/Thierry/Documents/R/Data/Economics/synthetic_control.data")
ytrue<-c(rep(1,100),rep(2,100),rep(3,100),rep(4,100),rep(5,100),rep(6,100))
```

We then compute the matrix of DTW using function `dtwDist` in package `dtw`:

```
library(dtw)
```

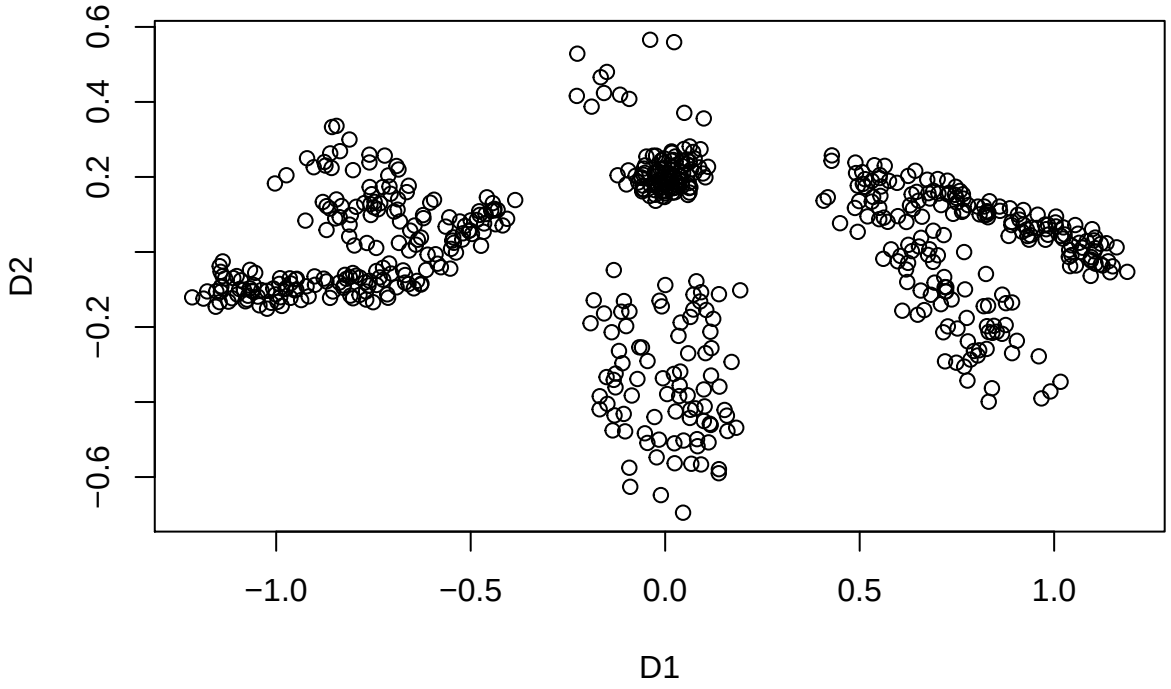
```
## Warning: package 'dtw' was built under R version 4.0.2
## Loading required package: proxy
## Warning: package 'proxy' was built under R version 4.0.2
##
## Attaching package: 'proxy'
## The following objects are masked from 'package:stats':
```

```
##
##     as.dist, dist

## The following object is masked from 'package:base':
##
##     as.matrix

## Loaded dtw v1.22-3. See ?dtw for help, citation("dtw") for use in publication.
```

```
D<-dtwDist(data)
```

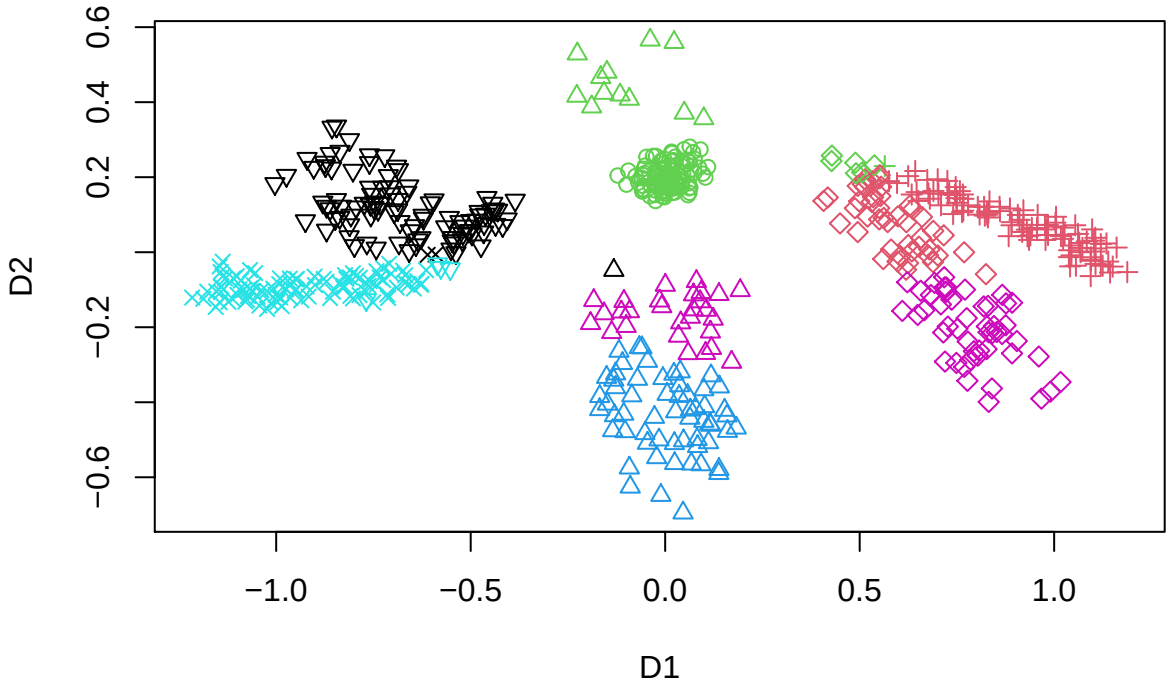
We compute a 2D configuration with ordinal MDS:

```
fit<-mds(D,ndim=2,type="ordinal")
plot(fit$conf)
```



The 6 classes are clearly visible in the 2D configuration. However, some classes are not spherical and it might not be easy to find them with the HCM algorithm. Let us try:

```
z<-scale(fit$conf)
km<-kmeans(z,centers=6,nstart=100)
plot(fit$conf,col=km$cluster,pch=ytrue)
```

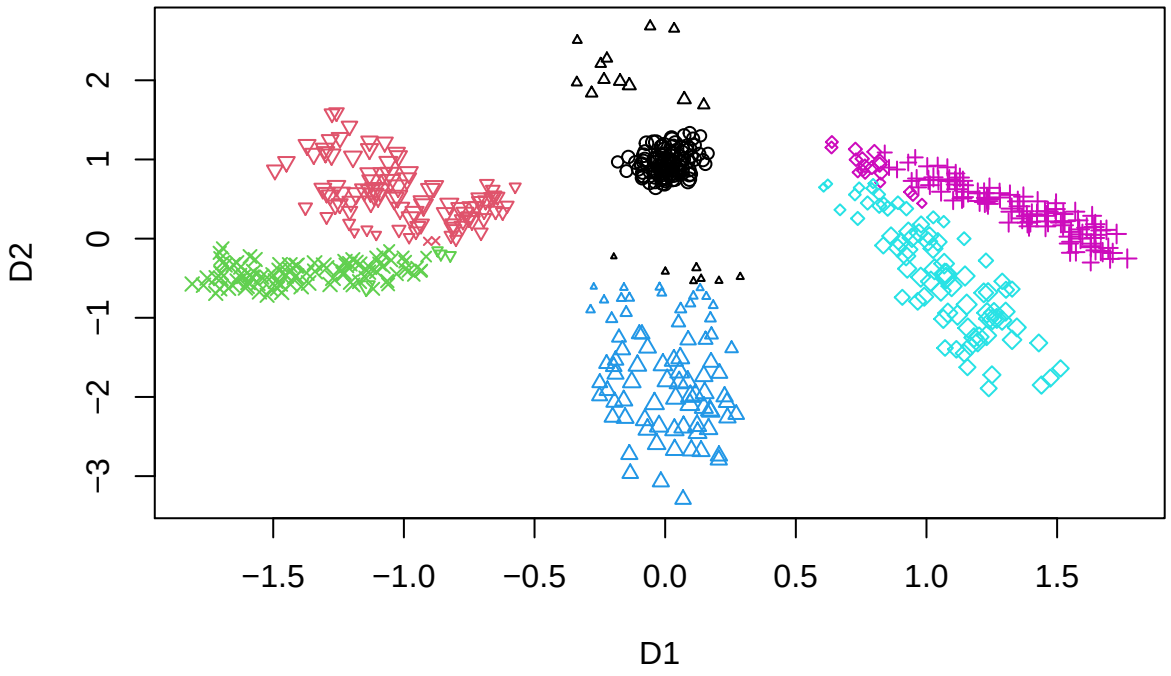


This is not so bad but as expected, the HCM algorithm does not easily capture nonspherical clusters. The Gustafson-Kessel algorithm is a variant of FCM that allows for ellipsoidal cluster shape. Let us try it here:

```
library(fclust)
```

```
## Registered S3 method overwritten by 'fclust':
##   method       from
##   print.fclust e1071
```

```
gk<-FKM.gk (z, k=6)
plot(z,col=gk$clus[,1],pch=ytrue,cex=gk$clus[,2])
```



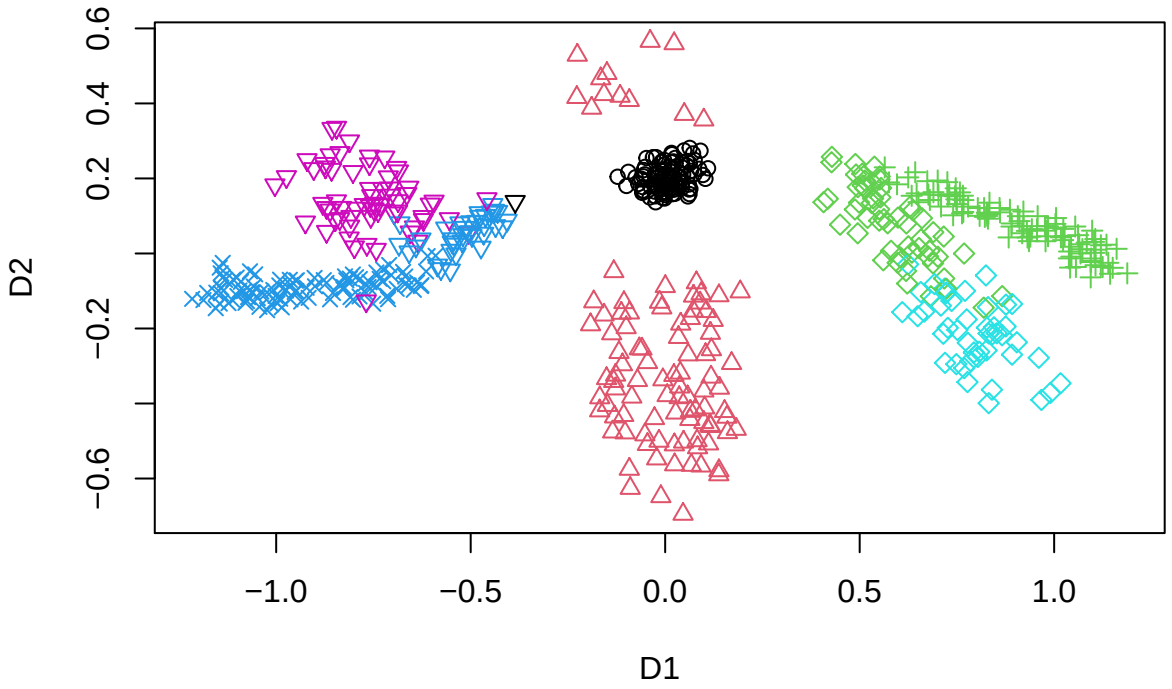
It is not perfect, but much better than HCM.

Algorithms for clustering dissimilarity data

PAM algorithm

The PAM algorithm is implemented in function `pam` of package `cluster`. let us apply it ot the `country` dataset. Let us apply this algorithm to the synthetic control chart data:

```
library(cluster)
clus<-pam(as.dist(D),k=6)
plot(fit$conf,pch=ytrue,col=clus$clustering)
```



The result is quite good, better than that of HCM.