Applied Statistics and Data Analysis
Lab 7: Unsupervised methods

Luca Grassetti and Paolo Vidoni
Department of Economics and Statistics, University of Udine

September, 2019

1 Example: US arrest data

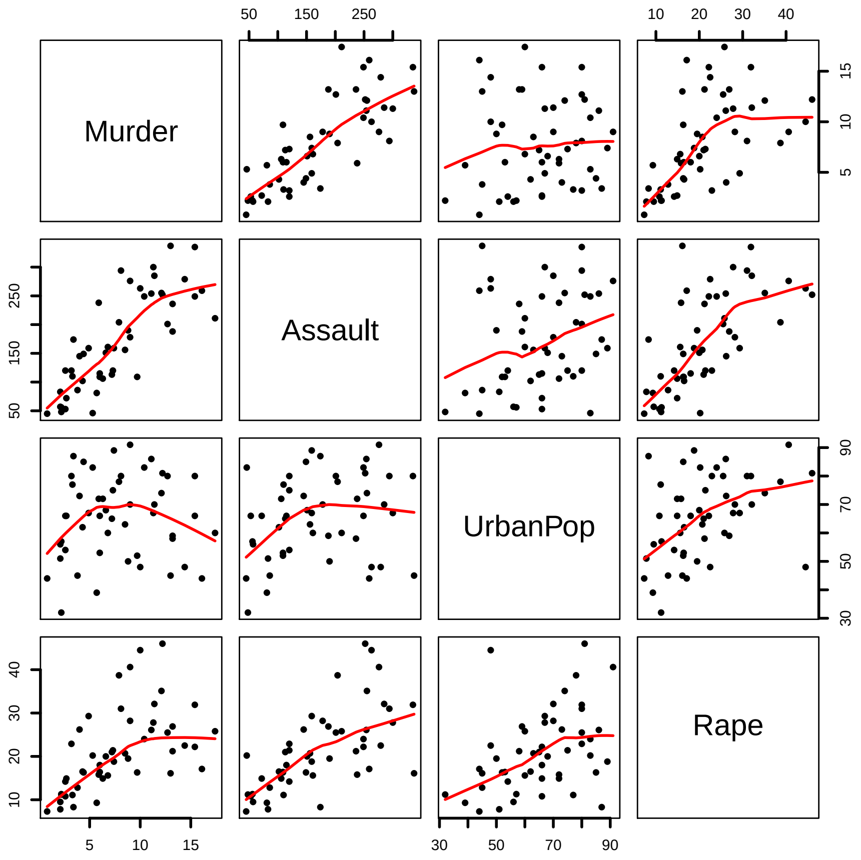
The `USArrests` data set, available in the R system libraries, contains data concerning violent crimes in each of the 50 US states in 1973. The variables are:

- `Murder`, the murder arrests per $100,000$ residents;
- `Assault`, the assault arrests per $100,000$ residents;
- `Rape`, the rape arrests per $100,000$ residents;
- `UrbanPop`, the percent of the population living in urban areas.

The scatterplot matrix for the four numerical variables is obtained using the function `pairs`. It gives a global representation, in a single graphical device, of the scatterplots involving each pair of variables. With the option `panel = panel.smooth`, a smooth curve is drawn through the data points.

```
str(USArrests)

'data.frame': 50 obs. of  4 variables:
 $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
 $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
 $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
 $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...

summary(USArrests)
```

```
     Murder          Assault         UrbanPop          Rape      
 Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30  
 1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07  
 Median : 7.250   Median :159.0   Median :66.00   Median :20.10  
 Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23  
 3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18  
 Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00  

pairs(USArrests, panel = panel.smooth, pch=16, lwd=2)
```



We may readily conclude that the four variables are related and, with the aim of summarizing the information given by the data frame `USArrests` and understanding the relationships among the relevant phenomena, we apply a Principal Component Analysis (PCA) to the observed data. To this end, we use the function `princomp`, which performs a principal components analysis on the given numeric data matrix and returns the results as an object of class `princomp`. Its first argument is a `formula` object with no response variable, referring only to numeric variables, or a numeric matrix (data frame). Indeed, the argument `cor` corresponds to a logical value (the default value is `FALSE`) indicating whether the calculation should use the correlation matrix (so that standardized variables are in fact taken into account) or the covariance matrix. It is recommendable to standardized the variables before applying the method, since the original variables may have different scales. Then, performing PCA will lead to extremely large loadings for variables with high variance and to dependence of a principal component on the variable with high variance.

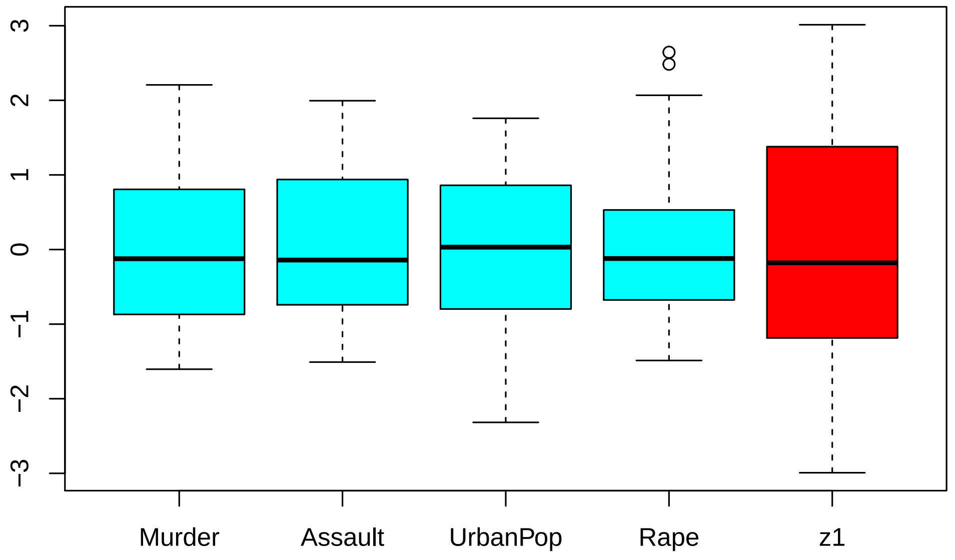
The function `princomp` returns a list containing the following main components:

- `sdev`, the standard deviations of the principal components (the square root of the uncorrected sample variances of the corresponding scores);
- `loadings`, the matrix of variable loadings (where the columns contain the eigenvectors);
- `scores`, the scores of the supplied data on the principal components, given as the columns of the output matrix (if the default option `scores = TRUE` is maintained and a numeric matrix, or a covariance matrix, is supplied as argument).

Notice that the signs of the columns of the loadings and of the scores are arbitrary, and so they may differ between different programs for PCA, and even between different versions of R. The function `princomp` uses the spectral decomposition approach on the covariance or on the correlation matrix, in order to obtain the principal components. The PCA may be performed in R using also the function `prcomp`, which instead uses a singular value decomposition procedure on the data matrix.

In the following analysis, we consider standardized variables, since they vary by orders of magnitude and then a suitable scaling is appropriate. Indeed, we specify the signs in order to improve interpretation and understanding of the results. Firstly, the boxplots of the four (standardized) variables are represented together with the boxplot of the scores given from the first principal component. The graphical representation confirms that, using the first principal component, some global information has been extracted from the original data set.

```r
obj <- princomp(USArrests, cor=TRUE)
z1 <- -obj$scores[,1] # the sign of the scores is modified
boxplot(cbind(scale(USArrests), z1), col=c(rep(5,4),2))
```

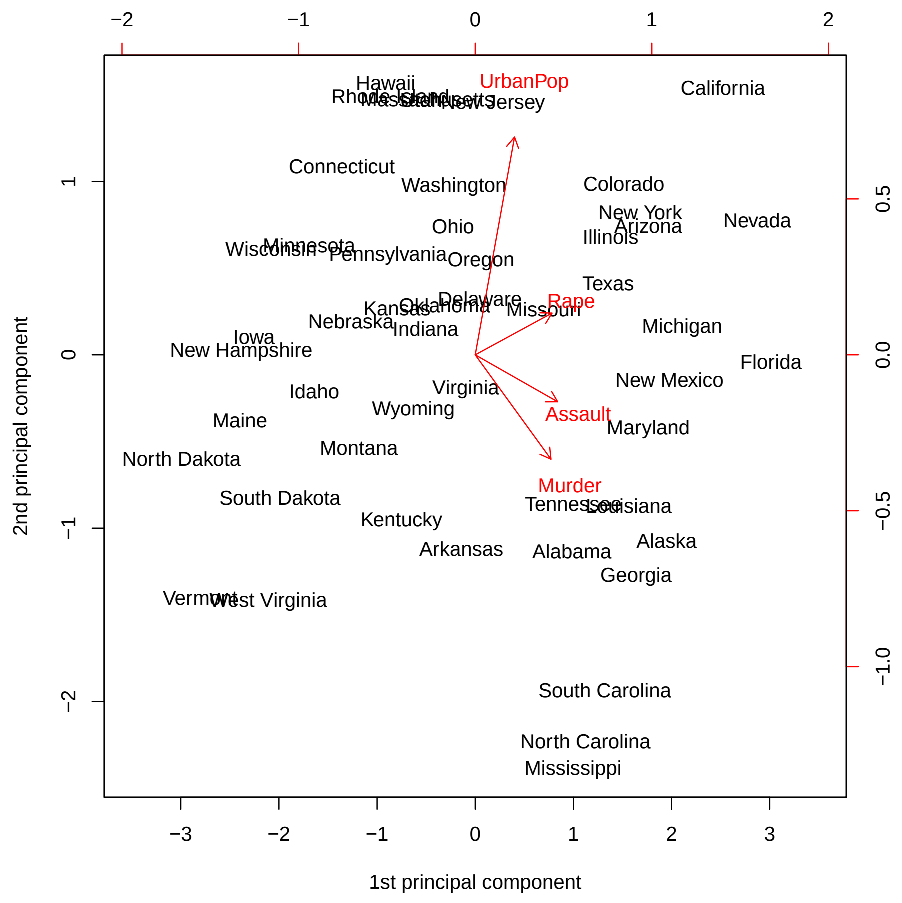


The first two principal component loading vectors are given below. Since the first loading vector places approximately equal weight on `Murder`, `Assault`, and `Rape`, with much less weight on `UrbanPop`, the first principal component roughly corresponds to a measure of overall rates of serious crimes. On the other hand, the second loading vector places most of its weight on `UrbanPop` and much less weight on the other three variables, so that the second principal component roughly describes to the level of urbanization of the state.

```
phi1<--obj$loadings[,1] # the sign of the loadings is modified
phi1 # 1st principal component loadings

   Murder   Assault  UrbanPop      Rape 
0.5358995 0.5831836 0.2781909 0.5434321 

phi2<--obj$loadings[,2] # the sign of the loadings is modified
phi2 # 2nd principal component loadings

    Murder    Assault   UrbanPop       Rape 
-0.4181809 -0.1879856  0.8728062  0.1673186 
```

The results of the principal component analysis can be represented by means of a biplot, which is a graphical summary showing both the scores and the loadings associated to the first two principal components. We may consider the function `biplot`, which produces a biplot from the output of function `princomp`. The first argument is an object of class `princomp` and the second is a vector of length 2 specifying the components to plot (the default value is `choices = 1:2`, giving the first two principal components). The option `scale = 0` ensures that the arrows are scaled to suitably represent the loadings.

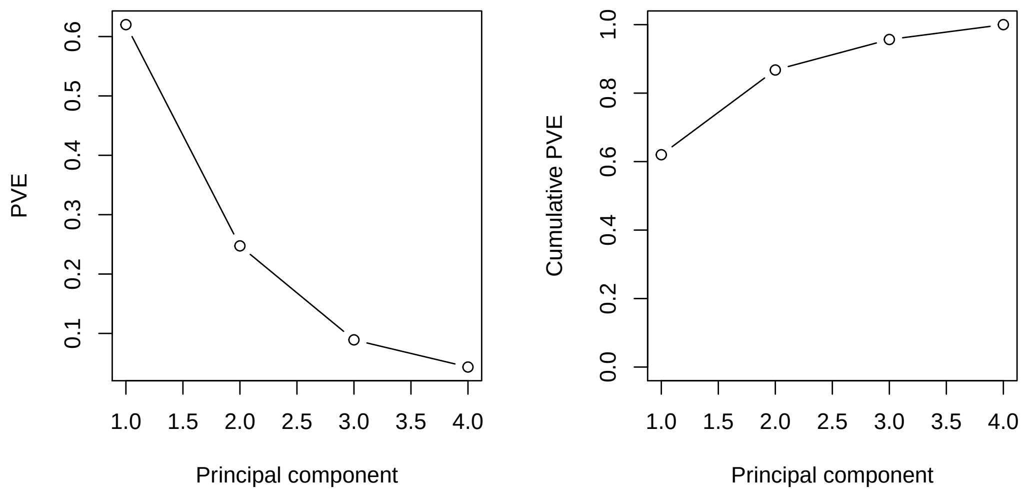
```
obj$loadings<--obj$loadings # the sign of the loadings is modified
obj$scores<--obj$scores # the sign of the scores is modified
biplot(obj, xlab="1st principal component", ylab="2nd principal component",
       xlim=c(-3.5,3.5), col=c(1,2), scale=0)
```



The state names (in black) identify the scores for the first two principal components (left and down axes), while the arrows and the variable names (in red) indicate the first two loadings associated to the four variables (top and right axes). Then, we may state that the crime-related variables are correlated with each other (for example, states with high murder rates tend to have high assault and rape rates), since these variables are located close to each other. On the other hand, the `UrbanPop` variable is far from the other three and this indicates that it is less correlated with the crime-related variables. Furthermore, states with large positive scores on the first component, such as California, Nevada and Florida, have high crime rates, while states like North Dakota, with negative scores on the first component, have low crime rates. California has a high score on the second component too, indicating a high level of urbanization. Finally, states with values for the two components close to zero, such as Indiana, have approximately average levels of both crime and urbanization.

In order to evaluate the proportion of variance explained by the principal components, we may consider the screeplot (left panel) and the plot of the cumulative proportion of variance explained (right panel). The proportion of the variance explained by a single component is given by the ratio between the (estimated) variance of the component and the total variance, which in this case corresponds to 4, since the four variables are standardized. Here, we use function `cumsum` for computing a vector whose elements are the cumulative sums of the elements of the argument.

```
par(mfrow=c(1,2), pty="s")
plot(obj$sdev^2/4, xlab="Principal component", ylab="PVE", type='b')
plot(cumsum(obj$sdev^2)/4, xlab="Principal component", ylab="Cumulative PVE",
     ylim=c(0,1), type='b')
```



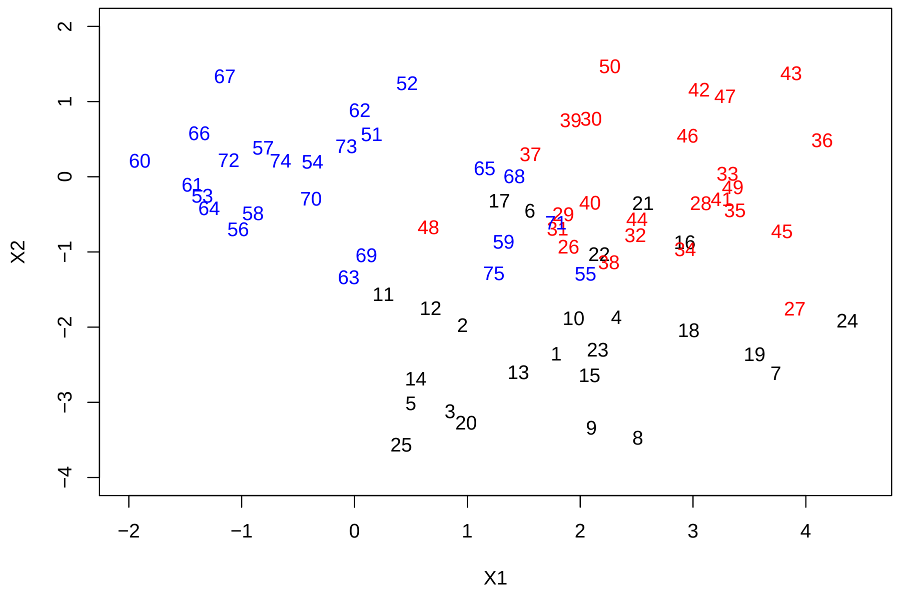
```
par(mfrow=c(1,1))
```

The first component explains more than 60% of the total variance, and the second one about 25% and then they explain almost 87% of the total variance in data. Indeed, the screeplot displays an elbow after the second principal component, which corresponds to the point where the proportion of variance explained drops off. Thus, we may conclude that the first two components provide a very useful summary of the original data set.

2 Example: three clusters simulated data

We consider a simulated data set with 75 observations of two variables, classified in three different clusters having dimension 25. The data are obtained as simulated values from a standard Gaussian distribution and then their coordinates are suitably modified in order to specify the three clusters. The observed bivariate data are represented in the following scatterplot where the clusters are described with different colors (black, red and blue). At first, we use the function `plot` with the option `type='n'` in order to represent the plotting region without the points. Secondly, we consider the function `text` for adding to the existing plot the strings corresponding to the labels of the observations in three different colors. The first two arguments define the numeric vectors of coordinates where the text labels should be written and the argument `labels` is a character vector or an expression specifying the text to be written (the default value corresponds to the labels associated to the point coordinates).

```
set.seed(25)
x<-matrix(rnorm(75*2), ncol=2)
x[1:25,1]<-x[1:25,1]+2
x[1:25,2]<-x[1:25,2]-2
x[26:50,1]<-x[26:50,1]+3
plot(x[1:25,1], x[1:25,2], xlim=c(-2,4.5), ylim=c(-4,2), type='n', xlab='X1',
     ylab='X2')
text(x[1:25,1], x[1:25,2])
text(x[26:50,1], x[26:50,2],labels = seq(26,50,1),col='red')
text(x[51:75,1], x[51:75,2],labels = seq(51,75,1),col='blue')
```



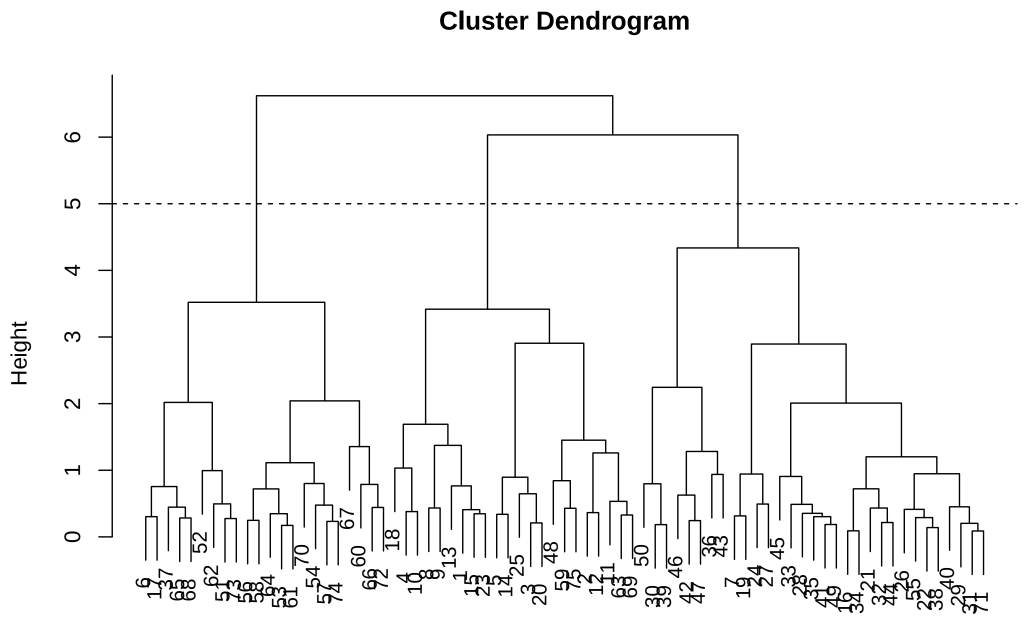
The class labels are now treated as unknown and, using a suitable clustering algorithm, we aim at discovering the true classification from the observed, simulated data. Hierarchical (agglomerative) cluster analysis is performed in R using the function `hclust`. The main arguments are:

- `d`, a distance matrix or a dissimilarity matrix computed using function `dist`;
- `method`, the linkage (agglomeration) criterion to be used (the default method is `"complete"`, but other options can be considered, such as `"single"`, `"average"` and `"centroid"`.

This function performs a hierarchical agglomerative cluster analysis using a set of dissimilarities for the objects to be clustered. In the first step, each object is assigned to its own cluster and then the algorithm proceeds iteratively, joining at each stage the two most similar clusters. The procedure continues until a single cluster is obtained. The output is an object of class `hclust`, which is a list giving the elements for representing the tree produced by the clustering process. Furthermore, function `dist` computes and returns the distance matrix obtained by using a specified distance measure for calculating the distances between the rows of a data matrix. The required arguments are a numeric matrix (data frame) and the distance measure to be considered (the default option is `method="euclidean"`, but other options can be defined, namely `"maximum"`, `"manhattan"`, `"binary"` and `"minkowski"`; the optional argument `p` specifies the power of the Minkowski distance.

Here, hierarchical clustering, with complete linkage and Euclidean distance, is performed. The result is considered as argument of function `plot`, in order to obtain the cluster dendrogram; the `hang` argument can be used to set the fraction of the plot height by which labels should hang below the rest of the plot (with a negative value the labels will hang down from 0). The leaves of the dendrogram represent the 75 observations and, when moving up the tree, similar leaves are fused into branches. The height of the fusion is measured on the y-axis and then observations that fuse at the very bottom of the tree are quite similar. With the function `abline`, the dendrogram is cut at the height of 5 and three distinct clusters are specified. Alternative cut points may give different clustering results.

```r
hc.complete<-hclust(dist(x), method="complete")
plot(hc.complete, xlab="", sub="", cex=.9)
abline(5,0,lty=2)
```

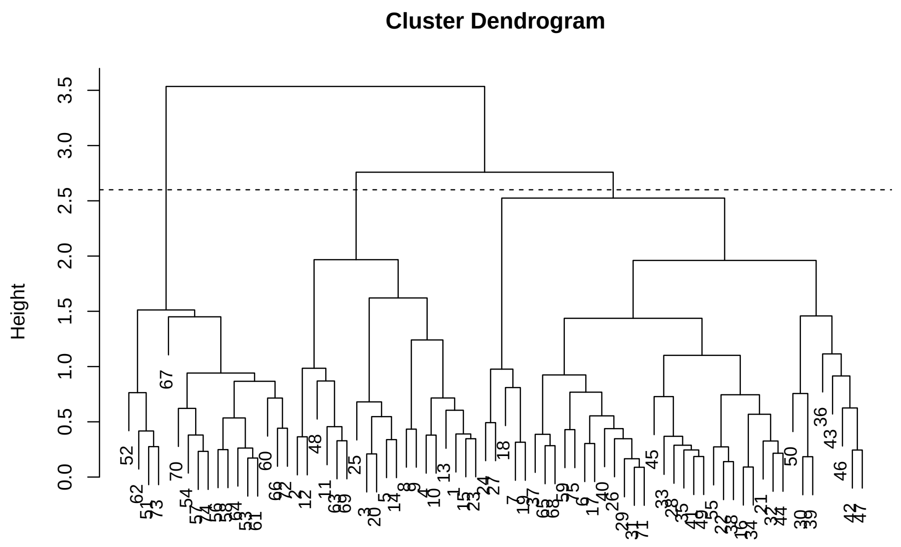


Finally, using function `cutree`, it is possible to define the cluster membership of the observations by cutting the tree into several groups, either by specifying the desired number of groups or by giving the cut height. This function requires as argument an object of class `hclust` and at least one argument between `k` (an integer with the desired number of groups) and `h` (the height where the tree should be cut). Here, we consider three groups and we obtain a vector with the label group memberships of each observation.

```
cutree(hc.complete, 3)

 [1] 1 1 1 1 1 2 3 1 1 1 1 1 1 1 1 3 2 1 3 1 3 3 1 3 1 3 3 3 3 3 3 3 3
[34] 3 3 3 2 3 3 3 3 3 3 3 3 3 3 1 3 3 2 2 2 2 3 2 2 2 1 2 2 2 1 2 2 2
[67] 2 2 1 2 3 2 2 2 1
```

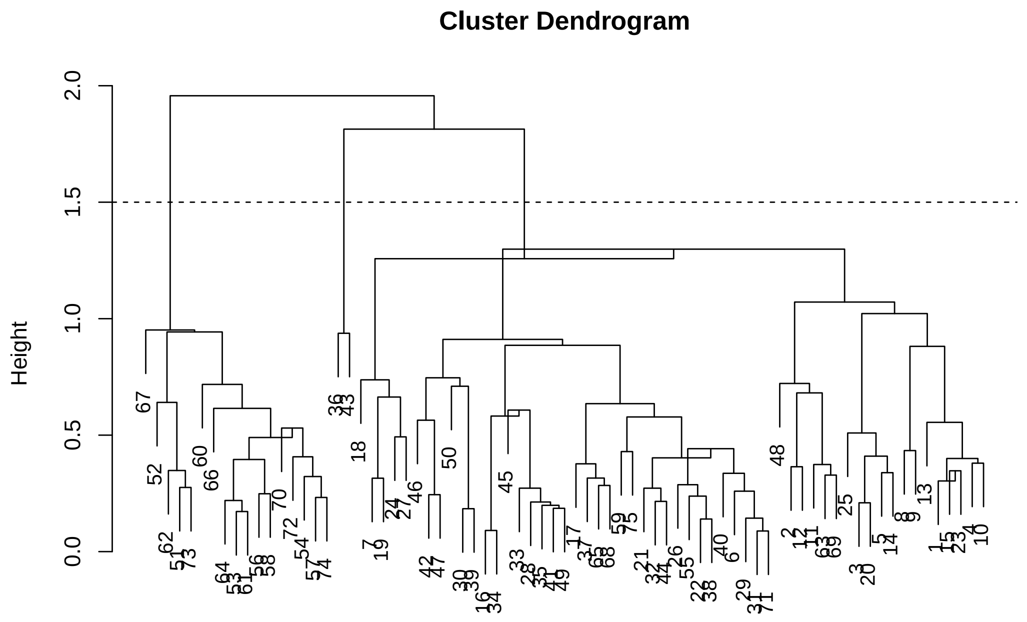
The following results are obtained with a hierarchical agglomerative cluster analysis using the Euclidean distance and the average and the centroid linkage. In order to highlight the three clusters, the dendrogram is cut at the height of 2.6 and at the height of 1.5, respectively

```
hc.average<-hclust(dist(x), method="average")
plot(hc.average, xlab="", sub="", cex=.9)
abline(2.6,0,lty=2)
```



```
cutree(hc.average, 3)

 [1] 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 2 2 2 2 1 2 2 1 2 1 2 2 2 2 2 2 2 2
[34] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 3 3 3 3 2 3 3 3 2 3 3 3 1 3 2 3
[67] 3 2 1 3 2 3 3 3 2
```

```
hc.centroid<-hclust(dist(x), method="centroid")
plot(hc.centroid, xlab="", sub="", cex=.9)
abline(1.5,0,lty=2)
```



```
cutree(hc.centroid, 3)

 [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 1 1 1 1
[34] 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 3 3 3 3 1 3 3 3 1 3 3 3 1 3 1 3
[67] 3 1 1 3 1 3 3 3 1
```

An alternative class of clustering algorithms corresponds to the partitioning methods, which aim at classifying the observations into $K > 0$ distinct, non-overlapping clusters. These methods require to fix in advance the number K of clusters and they optimize the allocation of units to clusters according to a suitable objective function. The most commonly used method of this class is K-means clustering, where the aim is to minimize the within cluster variation. This method is most appropriate for continuous variables, suitably scaled, and the Euclidean distance is commonly considered.

K-means clustering is performed in R using the function `kmeans`. The main arguments are:

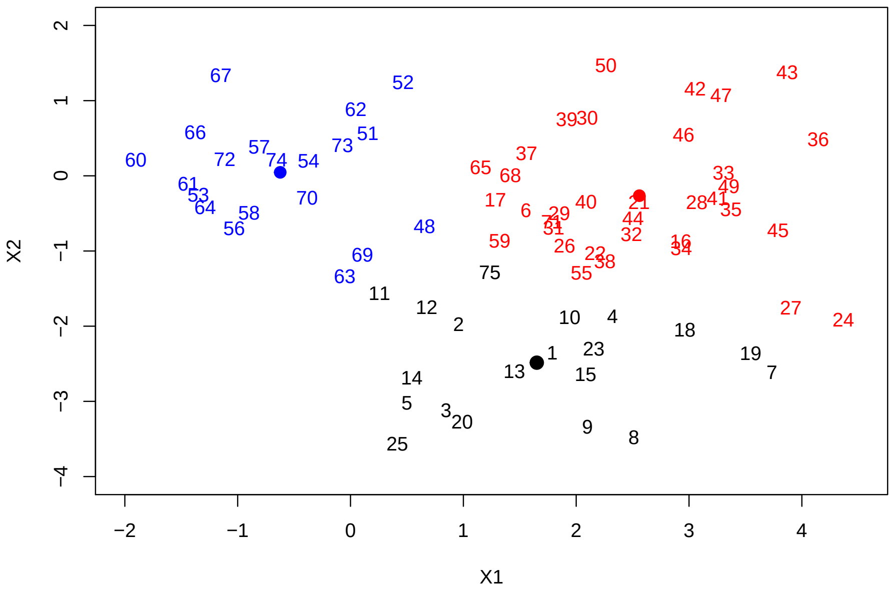
- `x`, a numeric matrix of data (differently from hierarchical clustering, the entire data matrix is required, not just the matrix of dissimilarities);
- `centers`, which is either the number of clusters `k` or a set of initial (distinct) cluster centres (in the first case, a random set of distinct rows in `x` is chosen as the initial centres);
- `nstart`, which specifies how many random sets should be chosen (if `centers` is a number).

It is important to run the algorithm multiple times from different random initial configurations (that is, to use the option `nstart`) and then to select the best solution, namely, that one for which the objective function is smallest. The output given by function `kmeans` is an object of class `kmeans`, which is a list giving the elements for representing the results produced by the clustering process. In particular, the element `cluster` is a vector of integers, from `1` to `k`, indicating the clusters where the points are allocated and `centers` is a matrix with the coordinates of the cluster centers.

This function now is applied to the simulated data set of bivariate observations, by considering 3 groups and `nstart=20`, so that 20 initial random cluster assignments are considered and the best solution is selected. The resulting partition is given below, and it is represented in the following scatterplot, with indication of the three clusters and of the corresponding centroids in different colors.

```
set.seed(5)
km3<-kmeans(x, 3, nstart =20)
km3

K-means clustering with 3 clusters of sizes 20, 35, 20

Cluster means:
        [,1]        [,2]
1  1.6507639 -2.48521165
2  2.5596938 -0.26425851
3 -0.6229167  0.04619656

Clustering vector:
 [1] 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 2 1 2 1 2 2 2 2 2 2 2 2
[34] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 3 3 3 3 2 3 3 3 2 3 3 3 3 3 2 3
[67] 3 2 3 3 2 3 3 3 1

Within cluster sum of squares by cluster:
[1] 28.72538 50.11093 19.39997
 (between_SS / total_SS =  68.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

```
plot(x[1:25,1], x[1:25,2], xlim=c(-2,4.5), ylim=c(-4,2), type='n',
     xlab='X1', ylab='X2')
text(x[which(km3$cluster==1),1], x[which(km3$cluster==1),2],
     labels = which(km3$cluster==1))
text(x[which(km3$cluster==2),1], x[which(km3$cluster==2),2],
     labels = which(km3$cluster==2),col='red')
text(x[which(km3$cluster==3),1], x[which(km3$cluster==3),2],
     labels = which(km3$cluster==3),col='blue')
points(km3$centers[1,1], km3$centers[1,2], pch=19, cex=1.5)
points(km3$centers[2,1], km3$centers[2,2], pch=19, col='red', cex=1.3)
points(km3$centers[3,1], km3$centers[3,2], pch=19, col='blue', cex=1.3)
```



A further partitioning method, which is a robust variant of the K-means, is the K-medoids clustering procedure. In this framework, as for K-means, the number K of clusters is fixed in advance but, as centers of these clusters, suitable datapoints are taken into account. The optimization algorithm attempts to minimize the distance between points labeled to be in a cluster and the point designated as the center of that cluster. Thus, in contrast to K-means, this method chooses datapoints as centers, and work with an arbitrary dissimilarity between points, rather than only the Euclidean distance.

There are several K-medoids variants and the most common is the Partitioning Around Medoids (PAM) algorithm, which is performed in R using the function `pam` of the library `cluster`. The main arguments are:

- `x`, a numeric data matrix (data frame) or a dissimilarity matrix;
- `k`, a positive integer giving the number of clusters;

- `metric`, a character string specifying the metric to be used for calculating dissimilarities between observations, with options `"euclidean"`, which is the default, and `"manhattan"` (if `x` is already a dissimilarity matrix, then this argument will be ignored).

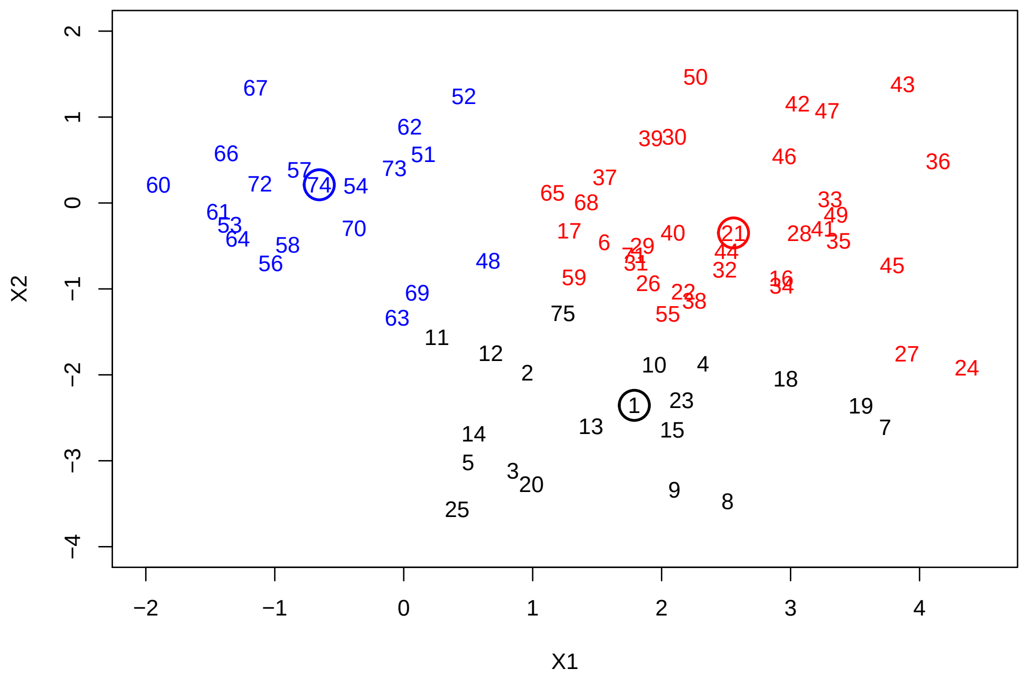
The output is an object of class `pam`, which is a list giving the elements for representing the results produced by the clustering process.

The PAM algorithm is applied to the simulated data set of bivariate observations, by considering 3 groups. In this case, the three clusters correspond exactly to those ones obtained using the K-means procedure. The resulting partition is given below, and it is represented in the following scatterplot, with indication of the three clusters in different colors. Furthermore, the three observations chosen by the algorithm as the medoids (namely, as the observations which better represent the clustering structure of the data) are highlighted using circles.

```
library(cluster)
me3<-pam(x, 3)
me3

Medoids:
     ID
[1,]  1  1.7881664 -2.3550041
[2,] 21  2.5574109 -0.3484911
[3,] 74 -0.6549297  0.2120810
Clustering vector:
 [1] 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 2 1 2 1 2 2 2 2 2 2 2 2
[34] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 3 3 3 3 2 3 3 3 2 3 3 3 3 3 2 3
[67] 3 2 3 3 2 3 3 3 1
Objective function:
   build     swap
1.093210 1.031341

Available components:
 [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
 [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

```
plot(x[1:25,1], x[1:25,2], xlim=c(-2,4.5), ylim=c(-4,2),type='n',
     xlab='X1', ylab='X2')
text(x[which(me3$cluster==1),1], x[which(me3$cluster==1),2],
     labels = which(me3$cluster==1))
text(x[which(me3$cluster==2),1], x[which(me3$cluster==2),2],
     labels = which(me3$cluster==2), col='red')
text(x[which(me3$cluster==3),1], x[which(me3$cluster==3),2],
     labels = which(me3$cluster==3), col='blue')
```

```
points(x[1,1], x[1,2], pch=21, cex=3, lwd=2)
points(x[21,1], x[21,2], pch=21, cex=3, col='red', lwd=2)
points(x[74,1], x[74,2], pch=21, cex=3, col='blue', lwd=2)
```



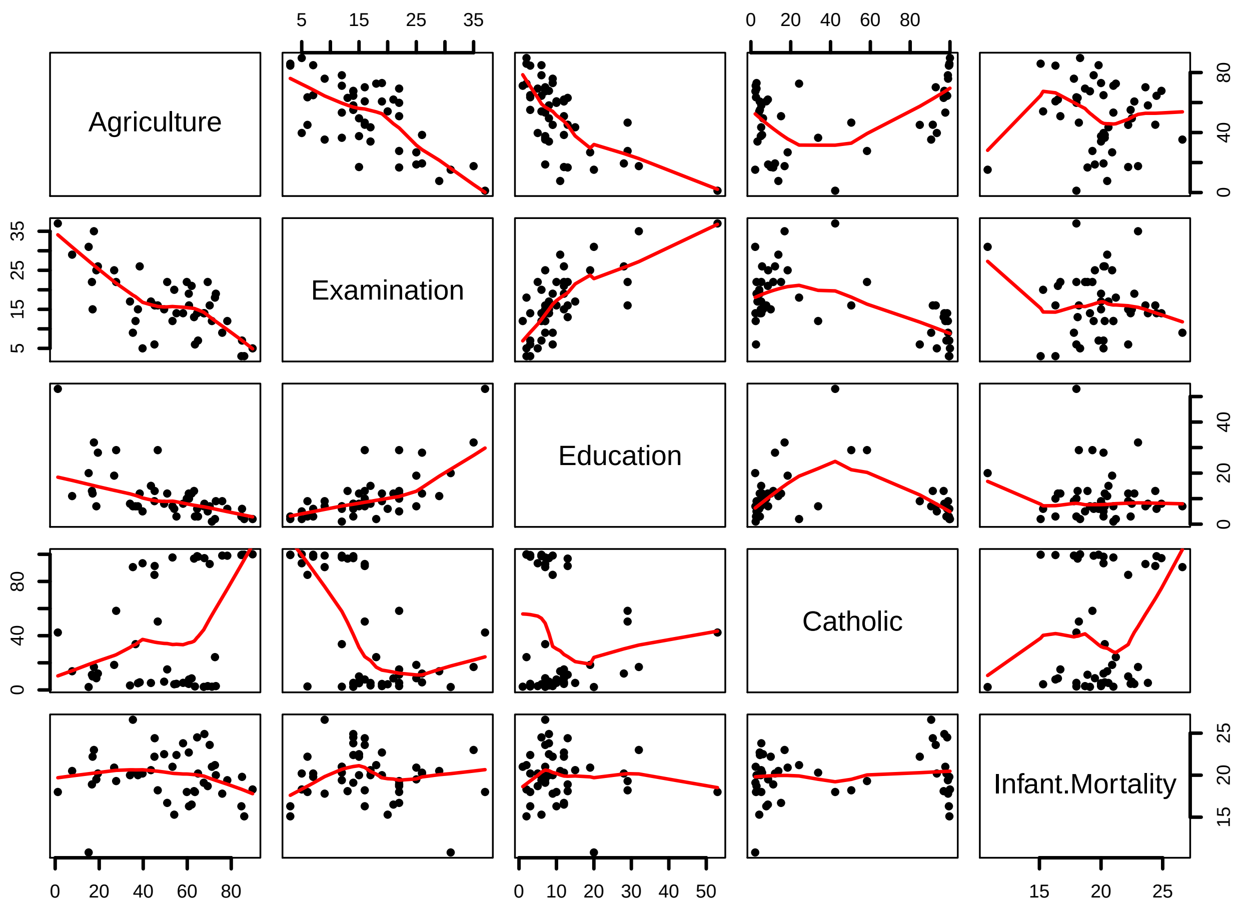
3 Example: Swiss socioeconomic indicators

We consider the data set `swiss`, available in the R system libraries, which contains data on a standardized fertility measure and on some socio-economic indicators for each of 47 French-speaking provinces of Switzerland at about 1888. The data frame contains 47 observations on the following 6 variables:

- `Fertility`, a common standardized fertility measure;
- `Agriculture`, the percentage of males involved in agriculture as occupation;
- `Examination`, the percentage of draftees receiving highest mark on army examination;
- `Education`, the percentage of education beyond primary school for draftees;
- `Catholic`, the percentage of Catholics as opposed to Protestants;
- `Infant.Mortality`, the percentage of live births who live less than 1 year.

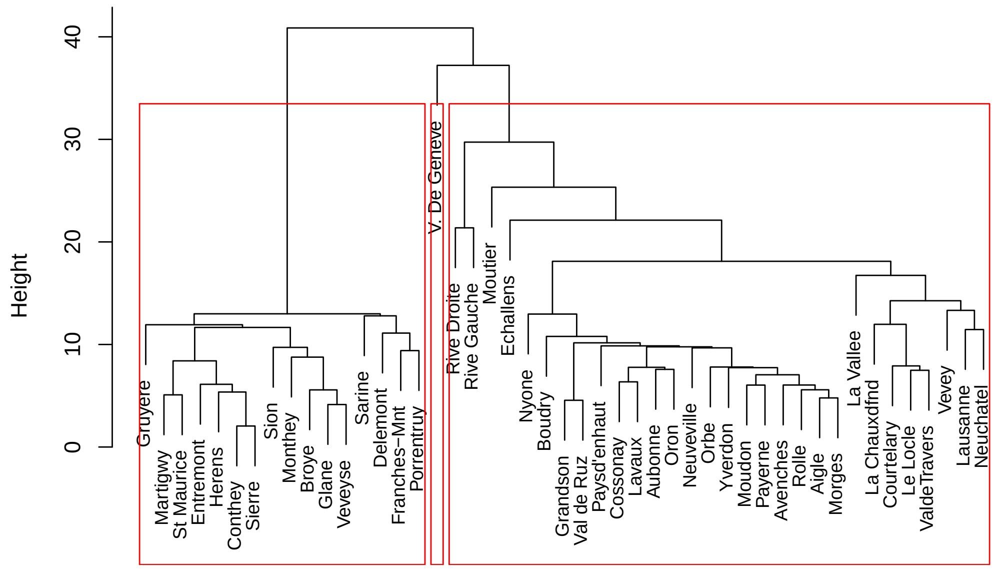
In the subsequent analysis we do not consider the variable `Fertility` and then we define the data matrix `swiss.x`, where only the remaining five variables are taken into account. The scatterplot matrix for these variables is obtained using the function `pairs`, with the option `panel = panel.smooth`, assuring that a smooth curve is drawn through the data points.

```
data(swiss)
str(swiss)

'data.frame':	47 obs. of  6 variables:
 $ Fertility       : num  80.2 83.1 92.5 85.8 76.9 76.1 83.8 92.4 82.4 82.9 ...
 $ Agriculture     : num  17 45.1 39.7 36.5 43.5 35.3 70.2 67.8 53.3 45.2 ...
 $ Examination     : int  15 6 5 12 17 9 16 14 12 16 ...
 $ Education       : int  12 9 5 7 15 7 7 8 7 13 ...
 $ Catholic        : num  9.96 84.84 93.4 33.77 5.16 ...
 $ Infant.Mortality: num  22.2 22.2 20.2 20.3 20.6 26.6 23.6 24.9 21 24.4 ...

swiss.x <- as.matrix(swiss[, -1]) # new data matrix without Fertility
pairs(swiss.x, panel = panel.smooth, pch=16, lwd=2)
```



An agglomerative hierarchical clustering, with single linkage, is performed. Since data are percentages, hereafter we use the Euclidean distance as a reasonable measure of dissimilarity between pairs of observations. The output is considered as argument of function `plot`, in order to obtain the cluster dendrogram. Furthermore, using function `rect.hclust`, we draw rectangles around the branches of the dendrogram, highlighting the corresponding clusters. The main arguments are `tree`, which is an object of class `hclust`, a scalar which gives either the required number of clusters `k` or the cutting height `h` for specifying the groups and `border`, which is a vector with border colors for the rectangles. By setting `k=3`, we obtain two main groups, with a single point group, well separated from them.

```
h <- hclust(dist(swiss.x), method = "single")
plot(h,cex=.8, xlab=' ',sub=' ', main=' ')
rect.hclust(h, k=3, border='red')
```



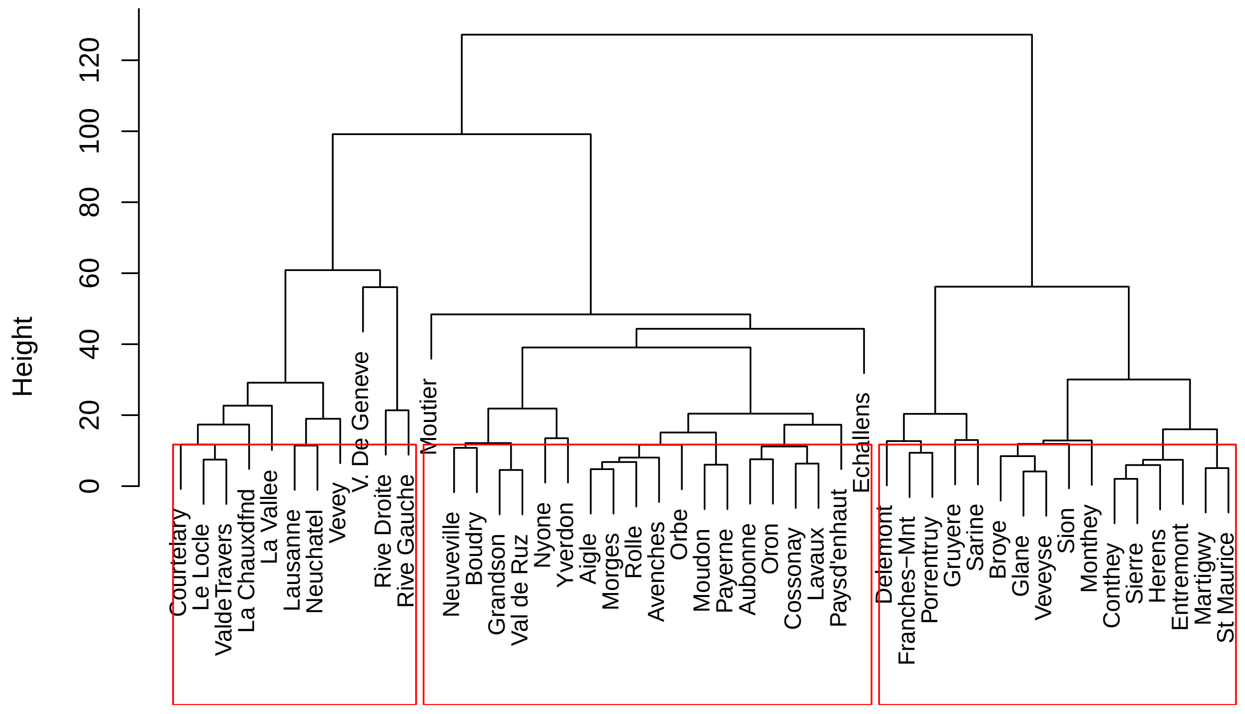
On the other hand, divisive hierarchical clustering may be performed in R using function `diana` of the library `cluster`. The main arguments are:

- `x`, a numeric data matrix (data frame) or a dissimilarity matrix;
- `metric`, a character string specifying the metric to be used for calculating dissimilarities between observations, with options `"euclidean"`, which is the default, and `"manhattan"` (if `x` is already a dissimilarity matrix, then this argument will be ignored).

The diana-algorithm constructs a hierarchy of clusterings, starting with one large cluster containing all the observations. At each stage, the cluster with the largest diameter is selected and then divided in two new clusters. The procedure continues until each group contains only a single observation. The output of function `diana` is an object of class `diana`, which is a list giving the elements for representing the results produced by the clustering process.

Divisive clustering, applied to the data matrix `swiss.x`, gives the three clusters highlighted in the following dendrogram using function `rect.hclust`. The dendrogram is drawn using the function `pltree` of the library `cluster`.

```
library(cluster)
h1<-diana(swiss.x)
pltree(h1, cex=.8,xlab=' ',sub=' ', main=' ')
rect.hclust(h1, k=3, border='red')
```



The K-means clustering, with $K = 3$, is applied to the data set `swiss.x` on Swiss provinces. In this first analysis, we consider, as starting point for the K-means algorithm, the means of the three clusters identified by the output of the hierarchical clustering procedure, with euclidean distance and average linkage, saved in the object `h2`. More precisely, we define three clusters using the command `cutree(h2,3)` and, with function `tapply`, we obtain the 3×5 matrix `initial`, where the rows give the mean values of the five variables computed in each group. The column names are defined as the names of the variables.

```
h2 <- hclust(dist(swiss.x), method = "average")
initial <- tapply(swiss.x, list(rep(cutree(h2,3), ncol(swiss.x)),
                    col(swiss.x)), mean)
dimnames(initial) <- list(NULL, dimnames(swiss.x)[[2]])
initial

     Agriculture Examination Education Catholic Infant.Mortality
[1,]    44.38333    19.56667    11.900 11.76733         19.56333
[2,]    65.51875     9.43750     6.625 96.15000         20.77500
[3,]     1.20000    37.00000    53.000 42.34000         18.00000
```

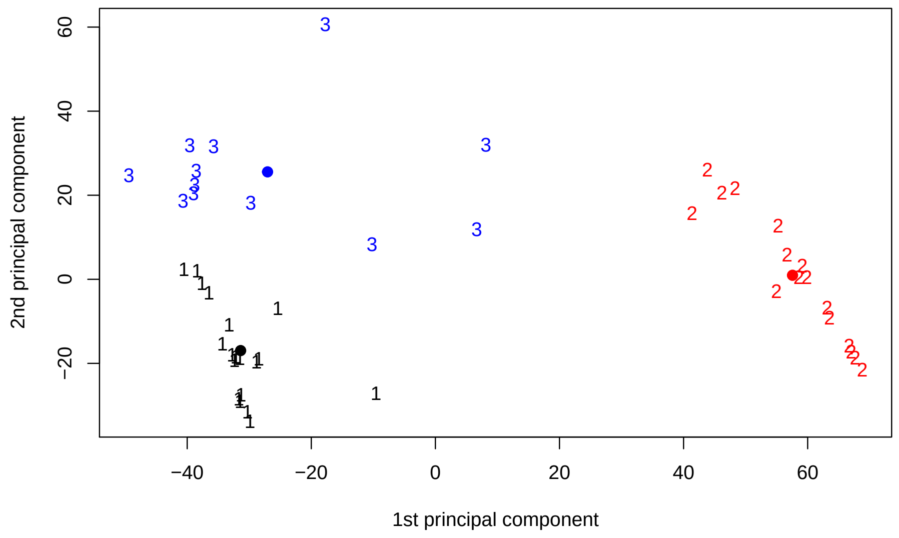
The result of the K-means clustering are saved in the object `h3` given below.

```
h3<-kmeans(swiss.x,centers=initial)
h3

K-means clustering with 3 clusters of sizes 19, 16, 12

Cluster means:
  Agriculture Examination Education Catholic Infant.Mortality
1    56.92632    17.31579  7.894737   6.1700         19.63684
2    65.51875     9.43750  6.625000  96.1500         20.77500
3    20.92500    24.58333 21.666667  23.1775         19.31667

Clustering vector:
  Courtelary     Delemont Franches-Mnt      Moutier   Neuveville
           3            2            2            3            1
  Porrentruy        Broye        Glane      Gruyere       Sarine
           2            2            2            2            2
     Veveyse        Aigle      Aubonne     Avenches     Cossonay
           2            1            1            1            1
   Echallens     Grandson     Lausanne    La Vallee       Lavaux
           1            1            3            3            1
      Morges       Moudon        Nyone         Orbe         Oron
           1            1            1            1            1
     Payerne Paysd'enhaut        Rolle        Vevey      Yverdon
           1            1            1            3            1
     Conthey    Entremont       Herens     Martigwy      Monthey
           2            2            2            2            2
  St Maurice       Sierre         Sion       Boudry La Chauxdfnd
           2            2            2            1            3
    Le Locle    Neuchatel   Val de Ruz ValdeTravers V. De Geneve
```

```
           3            3            1            3            3 
 Rive Droite  Rive Gauche 
           3            3 

Within cluster sum of squares by cluster:
[1] 3851.992 5516.926 8056.096
 (between_SS / total_SS =  84.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"    
[5] "tot.withinss" "betweenss"    "size"         "iter"        
[9] "ifault"      
```

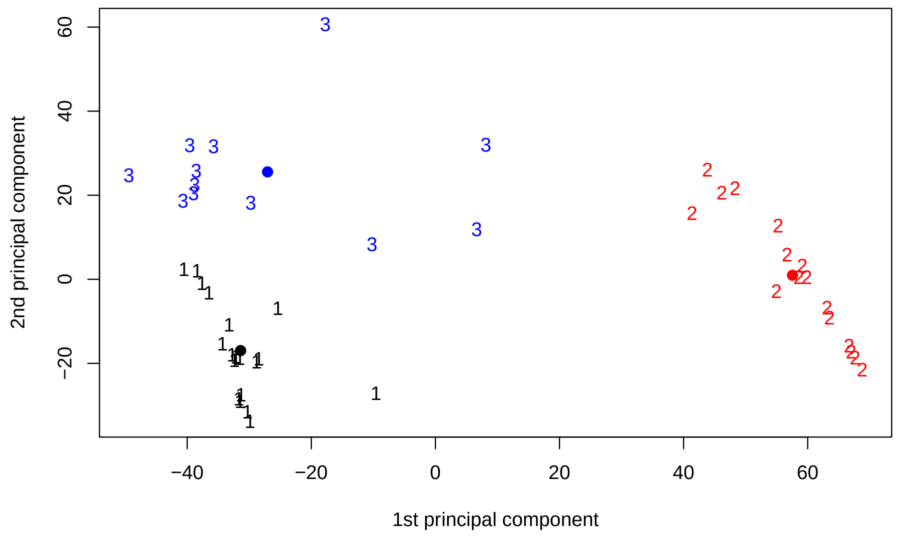
Since there are more than two variables, in order to plot the clusters of observations, with the corresponding centroids, we may consider a summary representation involving only the first two principal components. We performs a principal components analysis on the given data matrix **swiss.x** using function **prcomp**. Then, with the **predict** function, we obtain the observed values of the five principal components (which may be interpreted as new variables) with regard to the 47 provinces of Switzerland and to the three cluster centroids. By considering the first two principal components, which describes most of the variability of the original data set, we may represent the cluster in a two-dimensional plot using different colors.

```r
swiss.pca <- prcomp(swiss.x)
swiss.px <- predict(swiss.pca)
swiss.centers <- predict(swiss.pca, h3$centers)
plot(swiss.px[, 1:2], type = "n", xlab = "1st principal component",
     ylab = "2nd principal component")
text(swiss.px[which(h3$cluster==1), 1:2],
     labels = h3$cluster[which(h3$cluster==1)])
text(swiss.px[which(h3$cluster==2), 1:2],
     labels = h3$cluster[which(h3$cluster==2)], col='red')
text(swiss.px[which(h3$cluster==3), 1:2],
     labels = h3$cluster[which(h3$cluster==3)], col='blue')
points(swiss.centers[1,1], swiss.centers[1,2], pch=19, lwd=2)
points(swiss.centers[2,1], swiss.centers[2,2], pch=19, lwd=2, col='red')
points(swiss.centers[3,1], swiss.centers[3,2], pch=19, lwd=2, col='blue')
```



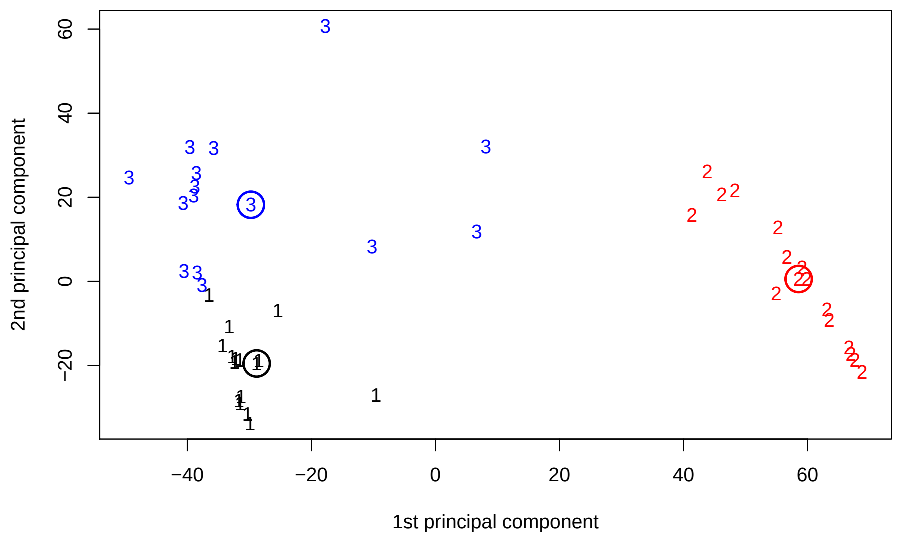
The same result is obtained by considering the K-means clustering, with $K = 3$, where a multiple initial random cluster assignment is performed by setting the option `nstart=20`. The results of this second clustering analysis using K-means is represented below and the first two principal components are considered for obtaining the graphical representation of the clusters and the associated centroids.

```
set.seed(5)
h4<-kmeans(swiss.x,3,nstart=20)
swiss.centers <- predict(swiss.pca, h4$centers)
plot(swiss.px[, 1:2], type = "n", xlab = "1st principal component",
     ylab = "2nd principal component")
text(swiss.px[which(h4$cluster==1), 1:2],
     labels = h4$cluster[which(h4$cluster==1)])
text(swiss.px[which(h4$cluster==2), 1:2],
     labels = h4$cluster[which(h4$cluster==2)]+1,col='blue')
text(swiss.px[which(h4$cluster==3), 1:2],
     labels = h4$cluster[which(h4$cluster==3)]-1, col='red')
points(swiss.centers[1,1], swiss.centers[1,2], pch=19, lwd=2)
points(swiss.centers[2,1], swiss.centers[2,2], pch=19, lwd=2, col='blue')
points(swiss.centers[3,1], swiss.centers[3,2], pch=19, lwd=2, col='red')
```



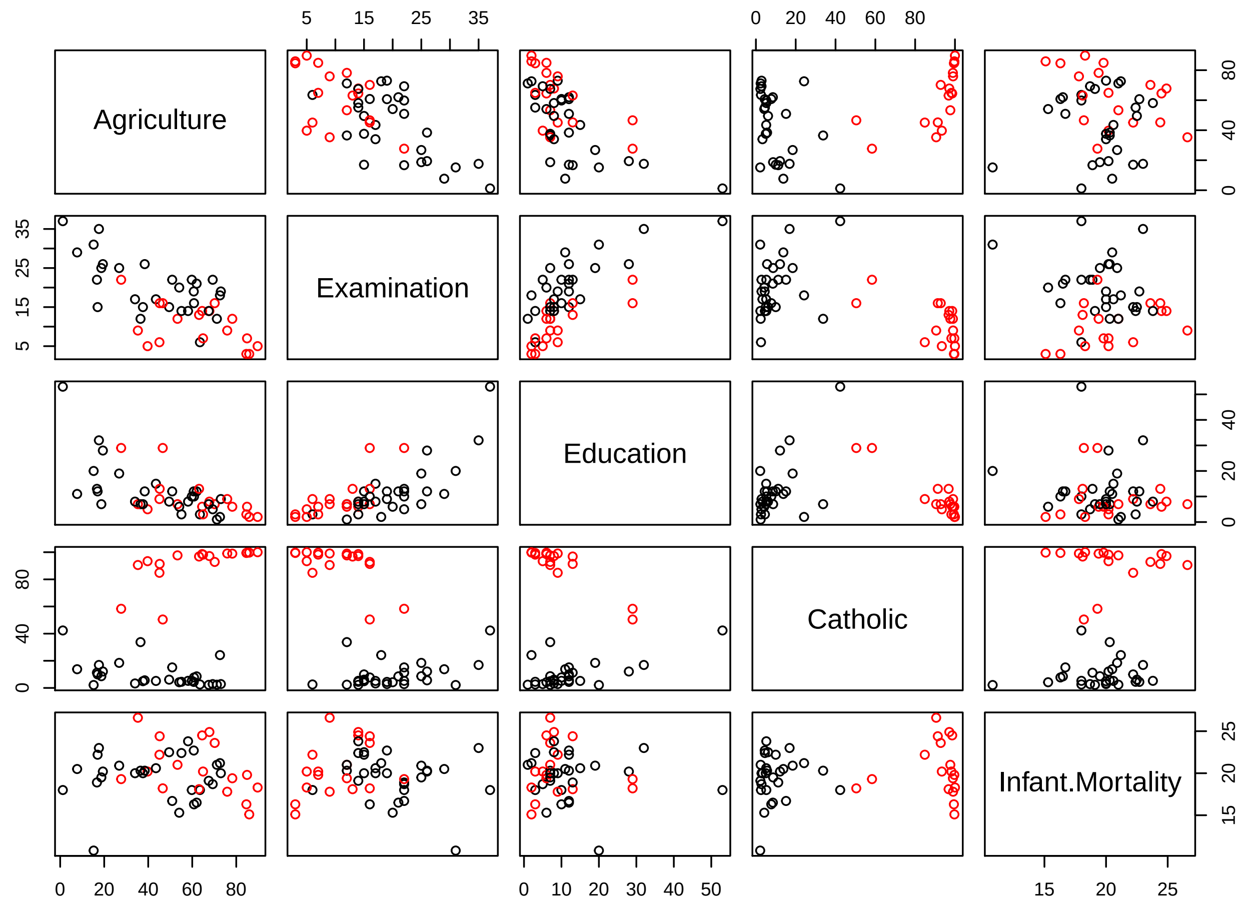
Finally, the K-medoids methods can also be applied using the PAM algorithm with $K = 3$ clusters. The clustering results are given below and the plot with the three different clusters and the corresponding medoids (observations used as cluster centers) is obtained using the first two principal components. The conclusions are similar to those obtained using the K-means procedure.

```
h5 <- pam(swiss.x,3)
h5

Medoids:
      ID Agriculture Examination Education Catholic Infant.Mortality
Vevey 29        26.8          25        19    18.46             20.9
Glane  8        67.8          14         8    97.16             24.9
Rolle 28        60.8          16        10     7.72             16.3
Clustering vector:
  Courtelary     Delemont Franches-Mnt      Moutier   Neuveville
           1            2            2            1            3
  Porrentruy        Broye        Glane      Gruyere       Sarine
           2            2            2            2            2
     Veveyse        Aigle      Aubonne     Avenches     Cossonay
           2            3            3            3            3
   Echallens     Grandson     Lausanne    La Vallee       Lavaux
           3            1            1            1            3
      Morges       Moudon        Nyone         Orbe         Oron
           3            3            3            3            3
     Payerne Paysd'enhaut        Rolle        Vevey      Yverdon
           3            3            3            1            3
     Conthey    Entremont       Herens     Martigwy      Monthey
           2            2            2            2            2
  St Maurice       Sierre         Sion       Boudry La Chauxdfnd
           2            2            2            1            1
    Le Locle    Neuchatel  Val de Ruz ValdeTravers V. De Geneve
           1            1            1            1            1
 Rive Droite  Rive Gauche
           1            1
Objective function:
   build     swap
18.86566 17.18954

Available components:
 [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
 [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"

swiss.medoids <- predict(swiss.pca, h5$medoids)
plot(swiss.px[, 1:2], type = "n", xlab = "1st principal component",
     ylab = "2nd principal component")
text(swiss.px[which(h5$cluster==3), 1:2],
     labels = h5$cluster[which(h5$cluster==3)]-2)
text(swiss.px[which(h5$cluster==2), 1:2],
     labels = h5$cluster[which(h5$cluster==2)], col='red')
text(swiss.px[which(h5$cluster==1), 1:2],
     labels = h5$cluster[which(h5$cluster==1)]+2, col='blue')
points(swiss.medoids[3,1], swiss.medoids[3,2], pch=21, cex=3, lwd=2)
points(swiss.medoids[2,1], swiss.medoids[2,2], pch=21, cex=3, col='red', lwd=2)
points(swiss.medoids[1,1], swiss.medoids[1,2], pch=21, cex=3, col='blue', lwd=2)
```



It is quite immediate to conclude that the obtained clusters tend to isolate the counties with a prevalence of Catholics, splitting the remaining counties in two groups. This meas that the counties with Catholic majority displays a peculiar behavior. The following scatterplot matrix, where the counties with Catholic majority are in red, confirms this fact by showing the relationships among the five interest variables.

```
swiss.x <- as.data.frame(swiss.x)
pairs(swiss.x, col = 1 + (swiss.x$Catholic > 50))
```


Agriculture
Examination
Education
Catholic
Infant.Mortality