

Mucha, Hans-Joachim; Sofyan, Hizir

**Working Paper**

## Cluster analysis

SFB 373 Discussion Paper, No. 2000,49

**Provided in Cooperation with:**

Collaborative Research Center 373: Quantification and Simulation of Economic Processes,  
Humboldt University Berlin

*Suggested Citation:* Mucha, Hans-Joachim; Sofyan, Hizir (2000) : Cluster analysis, SFB 373 Discussion Paper, No. 2000,49, Humboldt University of Berlin, Interdisciplinary Research Project 373: Quantification and Simulation of Economic Processes, Berlin,  
<https://nbn-resolving.de/urn:nbn:de:kobv:11-10047756>

This Version is available at:

<https://hdl.handle.net/10419/62242>

**Standard-Nutzungsbedingungen:**

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

**Terms of use:**

*Documents in EconStor may be saved and copied for your personal and scholarly purposes.*

*You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.*

*If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.*

# Cluster Analysis

*Hans-Joachim Mucha and Hizir Sofyan*

As an explorative technique, cluster analysis provides a description or a reduction in the dimension of the data. It classifies a set of observations into two or more mutually exclusive **unknown** groups based on combinations of many variables. Its aim is to construct groups in such a way that the profiles of objects in the same groups are relatively homogenous whereas the profiles of objects in different groups are relatively heterogeneous.

Clustering is distinct from classification techniques, like discriminant analysis or classification tree algorithms. Here no **a priori** information about classes is required, i.e., neither the number of clusters nor the rules of assignment into clusters are known. They have to be discovered exclusively from the given data set without any reference to a training set. Cluster analysis allows many choices about the nature of the algorithm for combining groups.

## 1 Introduction

In general, cluster analysis could be divided into hierarchical clustering techniques and nonhierarchical clustering techniques. Examples of hierarchical techniques are single linkage, complete linkage, average linkage, median, Ward. Nonhierarchical techniques include K-means, adaptive K-means, K-medoids, fuzzy clustering. To determine which algorithms are good is dependent on the type of data available and the particular purpose of analysis. Therefore, it is better to run more than one algorithm and then analyze and compare the results carefully. In more objective way, the stability of clusters can be investigated in simulation studies (Mucha 1992).

## 1.1 Distance Measures

```
d = distance(x{, metric})  
  computes the distance between p-dimensional data points de-  
  pending on a specified metric  
  
d = lpdist(x, q, p)  
  computes the so-called  $L_p$ -distances between the rows of a data  
  matrix. In the case  $p = 1$  (absolute metric) or  $p = 2$  (euclidean  
  metric) one should favour the function distance
```

The distances between points play an important role in clustering. There are several distance measures available by the XploRe command `distance`. Moreover, additional distance measures can be computed by using the XploRe matrix language.

For a distance between two  $p$ -dimensional observations  $x = (x_1, x_2, \dots, x_p)^T$  and  $y = (y_1, y_2, \dots, y_p)^T$ , we consider the Euclidean metric defined as

$$d(x, y) = \left[ \sum_{i=1}^p (x_i - y_i)^2 \right]^{\frac{1}{2}} \quad (1)$$

In matrix notation, this is written as the following:

$$d(x, y) = \sqrt{(x - y)^T (x - y)} \quad (2)$$

The statistical distance between these two observations is the following

$$d(x, y) = \sqrt{(x - y)^T A (x - y)} \quad (3)$$

where  $A = S^{-1}$  is the inverse of  $S$ , the matrix of sample variances and covariances. It is often called Mahalanobis distance.

In XploRe, we have some of distances, those are Euclidean, diagonal, Mahalanobis. The distance measure or metric should be chosen with care. The Euclidean metric should not be used where different attributes have widely varying average values and standard deviations, since large numbers in one attribute will prevail over smaller numbers in another. With the diagonal and Mahalanobis metrics, the input data are transformed before use. Choosing the

diagonal metric results in transformation of the data set to one in which all attributes have equal variance. Choosing the Mahalanobis metric results in transformation of the data set to one in which all attributes have zero mean and unit variance. Correlations between variables are taken into account.

Here is the example:

```
x = #(1,4)~#(1,5)
distance (x, "l1")
distance (x, "l2")
distance (x, "maximum")
```

 clust01.xpl

The results of this code program are

```
Contents of distance
[1,]    0    7
[2,]    7    0
Contents of distance
[1,]    0    5
[2,]    5    0
Contents of distance
[1,]    0    4
[2,]    4    0
```

That means that the distance between two observations with City-block distance is 7, with Euclidean distance is 5 and with maximum distance is 4.

Alternatively, a distance measure could be also computed by quantlet `lpdist`. This aim is to compute the so-called  $L_p$ -distances between the rows of a data matrix.


Here is the quantlet `lpdist` in XploRe,

```
d = lpdist(x, q, p)
```

where  $x$  is  $n \times m$  matrix,  $q$  is  $m \times 1$  matrix of nonnegative weights of columns, and  $p$  is scalar parameter ( $p > 0$ ) of the  $L_p$ -metric. In the case  $p=1$  (absolute metric) or  $p=2$  (euclidean metric).

To see an example, we start with loading the quantlib `xclust`, then, we generate eight pairs of data, determine the column weights, and apply euclidean metric

```
library ("xclust")
x = #(5, 2, -2, -3, -2, -2, 1, 1)~#(-3, -4, -1, 0, -2, 4, 2, 4)
q = #(1, 1)
lpdist (x, q, 2)
```

 clust02.xpl

The output of this code as follows,

```
Content of object d
[1,] 3.1623
[2,] 7.0821
[3,] 8.5440
...
...
[26,] 3.6056
[27,] 3
[28,] 2
```

This result is  $28 \times 1$  matrix of paired distances between the 28 row points, and it is also the input for hierarchical clustering which is presented in the following section.

## 1.2 Similarity of Objects

According to Härdle and Simar (1998), for measuring the similarity between objects, we can compare pairs of observations  $(x_i, x_j)$ ;  $x_i^T = (x_{i1}, \dots, x_{ip})$ ,  $x_j^T = (x_{j1}, \dots, x_{jp})$ ,  $x_{ik}, x_{jk} \in \{0, 1\}$ . Actually, we have four cases:

$$x_{ik} = x_{jk} = 1, x_{ik} = 0, x_{jk} = 1, x_{ik} = 1, x_{jk} = 0, x_{ik} = x_{jk} = 0. \quad (4)$$

We define

$$a_1 = \sum_{k=1}^p I(x_{ik} = x_{jk} = 1), a_2 = \sum_{k=1}^p I(x_{ik} = 0, x_{jk} = 1), \quad (5)$$

$$a_3 = \sum_{k=1}^p I(x_{ik} = 1, x_{jk} = 0), a_4 = \sum_{k=1}^p I(x_{ik} = x_{jk} = 0). \quad (6)$$

General measures are used in practice

$$T_{ij} = \frac{a_1 + \delta a_4}{a_1 + \delta a_4 + \lambda(a_2 + a_3)}, \quad (7)$$

where  $\delta$  and  $\lambda$  are weighting vectors. According to the weighting factors we have the following table.


Name	$\delta$	$\lambda$	<i>Definition</i> ( $T(x_i, x_j)$ )
Jaccard	0	1	$\frac{a_1}{a_1 + a_2 + a_3}$
Tanimoto	1	2	$\frac{a_1 + a_4}{a_1 + 2(a_2 + a_3) + a_4}$
Simple Matching (M)	1	1	$\frac{a_1 + a_4}{p}$

Table 1: The common similarity coefficient.

Note that each  $a_l, l = 1, \dots, 4$  depends on the pair  $(x_i, x_j)$ . In XploRe the similarity matrix  $T$  given above is transformed into a distance matrix  $D$  by  $D = 1^T - T$ .

The example of this problem as follows

```
x = #(1,0, 0)~#(1,0,1)
distance (x, "tanimoto")
```

 clust03.xpl

The result is the similarity object using Tanimoto coefficient.

```
Contents of distance
[1,]    0    1    0.5
[2,]    1    0    1
[3,]  0.5    1    0
```

## 2 Hierarchical Clustering

At any stage of the procedure, a hierarchical clustering technique performs either a merger of clusters or a division of a cluster at previous stage. It will conceptually give rise to a tree like structure of the clustering process. It is understood that the clusters of items formed at any stage are nonoverlapping or mutually exclusive.

Hierarchical clustering techniques proceed by either a series of successive mergers or a series of successive divisions.

The results of these methods can be displayed in a **dendrogram**. The dendrogram is the tree-like diagram that can depict the mergers or divisions which have been made at successive level. Below, in Figure 1, is the example of the dendrogram by using eight pairs of data above.

### 2.1 Agglomerative Hierarchical Methods

```
cagg = agglom (d, method, no{, opt })  
    performs a hierarchical cluster analysis
```

This method starts with each object describing a cluster, and then combines them into more inclusive clusters until only one cluster remains. Härdle and Simar (1998) considered the algorithm of agglomerative hierarchical method as follows,

1. Construct the finest partition
2. Compute the distance matrix  $D$
3. **DO**
  - Find the clusters with the closest distance
  - Put those two clusters into one cluster
  - Compute the distances between the new groups and the remaining groups by (8) to obtain a reduced distance matrix  $D$
4. **UNTIL** all clusters are agglomerated into one group.

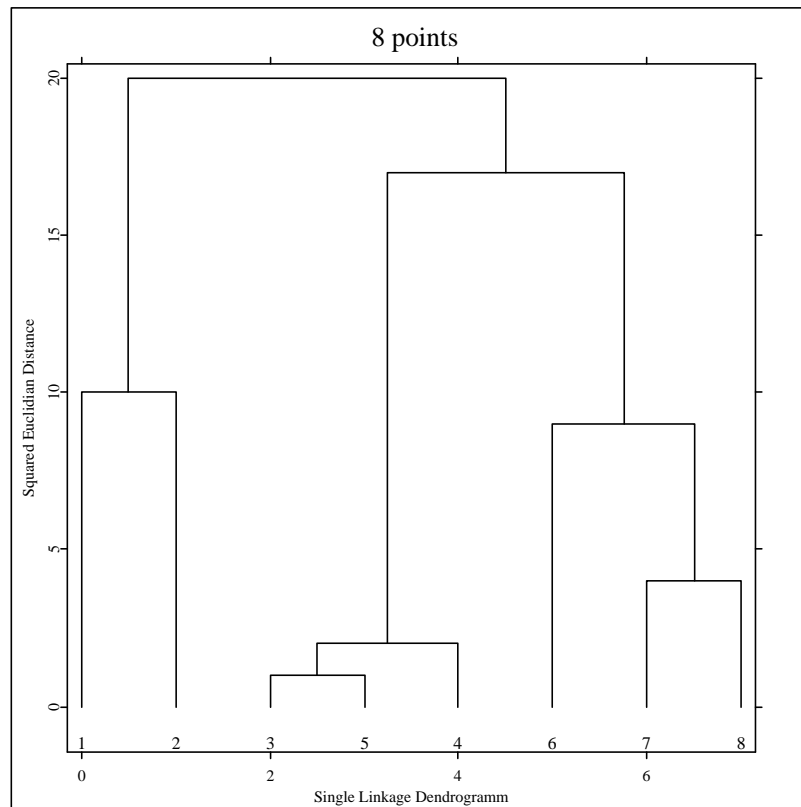


Figure 1: An example of a dendrogram using eight pairs of data.



If two objects or groups  $P$  and  $Q$  are to be united one obtains the distance to another group (object)  $R$  by the following distance function

$$d(R, P + Q) = \delta_1 d(R, P) + \delta_2 d(R, Q) + \delta_3 d(P, Q) + \delta_4 |d(R, P) - d(R, Q)| \quad (8)$$

The  $\delta_j$ 's are weighting factors that lead to different agglomerative algorithms as described in Table 2. Here  $n_P = \sum_{i=1}^n I(x_i \in P)$  is the number of objects in group  $P$ . The values of  $n_Q$  and  $n_R$  are defined analogously. The flexible method requires a parameter  $\beta$  that is specified by the user. Below is the

Name	$\delta_1$	$\delta_2$	$\delta_3$	$\delta_4$
Single linkage	1/2	1/2	0	-1/2
Complete linkage	1/2	1/2	0	1/2
Simple Average linkage	1/2	1/2	0	0
Average linkage	$\frac{n_P}{n_P + n_Q}$	$\frac{n_Q}{n_P + n_Q}$	0	0
Centroid	$\frac{n_P}{n_P + n_Q}$	$\frac{n_Q}{n_P + n_Q}$	$-\frac{n_P n_Q}{(n_P + n_Q)^2}$	0
Median	1/2	1/2	-1/4	0
Ward	$\frac{n_R + n_P}{n_R + n_P + n_Q}$	$\frac{n_R + n_Q}{n_R + n_P + n_Q}$	$\frac{-n_R}{n_R + n_P + n_Q}$	0
Flexible Method	$\frac{1-\beta}{2}$	$\frac{1-\beta}{2}$	$\beta$	0

Table 2: Computation of group distances available in XploRe.

quantlet `agglom`, which is implemented in XploRe to perform hierarchical cluster analysis.

```
cagg = agglom (d, method, no{, opt})
```

where  $d$  is a  $n \times 1$  vector or  $l \times l$  matrix of distances, `method` is the string that specify one of the following `agglom` methods: “WARD”, “SINGLE”, “COMPLETE”, “MEAN\_LINK”, “MEDIAN\_LINK”, “AVERAGE”, “CENTROID”, and “LANCE” (flexible method), `no` is a scalar that shows the number of clusters and `opt` is an optional argument for flexible methods, with the default value  $-0.15$ .

The output of this quantlet `agglom` are: `cagg.p` is a vector with partition numbers  $(1, 2, \dots)$ , `cagg.t` is a matrix with the dendrogram for the number of clusters (`no`), `cagg.g` is a matrix with the dendrogram for all  $l$  clusters, `cagg.pd` is a matrix with partition numbers  $(1, 2, \dots)$ , and `cagg.d` is a vector matrix with distances between the cluster centers.

### 2.1.1 Single Linkage Method

The single linkage method is also called nearest neighbor method or minimum distance method. This method is defined by

$$d(R, P + Q) = \min(d(R, P), d(R, Q)) \quad (9)$$

The process is continuous from the weak clustering to the strong clustering. This method is invariant to monotone transformations of the input data. Therefore the algorithm can be used with similarity and dissimilarity measures. The effect of the algorithm that it tends to merge clusters is sometimes undesirable because it prevents the detection of not well separated clusters. On the other hands, the criteria maybe useful to detect outliers in the data set.

For example, we describe the single linkage method for the eight data points displayed in Figure 1

First we prepare the data,

```
x=#(5,2,-2,-3,-2,-2,1,1)~#(-3,-4,-1,0,-2,4,2,4)
                                ; creates 8 pairs of data
n=rows(x)                      ; rows of data
xs=string("%1.0f", 1:n)        ; adds labels
setsize(500, 500)
dd1=createdisplay(1,1)
setmaskp(x, 0, 0, 0)
setmaskt(x, string("%1.0f", 1:rows(x)), 0, 0, 16)
setmaskl(x, 1~2~7~8~6~0~7~3~5~0~3~4, 0, 1, 1)
```

```

show(dd1, 1, 1, x) ; shows data
setgopt(dd1, 1, 1,"xlab","first coord.",
        "ylab","second coord.")
setgopt(dd1, 1, 1,"title","8 points","xoff",7|7,"yoff",7|7)

```

then we calculate the Euclidean distance and apply the single linkage method,

```

d=distance(x, "euclid") ; Euclidean distance
d.*d ; squared distance matrix
t=agglom(d.*d, "SINGLE", 5) ; here single linkage method
g=tree(t.g, 0, "CENTER")
g=g.points
l = 5.*(1:rows(g)/5) + (0:4)' - 4
setmaskl (g, l, 0, 1, 1)
setmaskp (g, 0, 0, 0, 0)


```

finally we show the plot of the raw data and the dendrogram

```

tg=paf(t.g[,2], t.g[,2]~=0)
numbers=(0:(rows(x)-1))
numbers=numbers~((-1)*matrix(rows(x)))
setmaskp(numbers, 0, 0, 0)
setmaskt(numbers, string("%.0f", tg), 0, 0, 14)
dd2=createdisplay(1,1)
show (dd2, 1, 1, g, numbers)
setgopt(dd2, 1, 1, "xlab","Single Linkage Dendrogramm",
        "ylab","Squared Euclidian Distance")
setgopt(dd2,1,1,"title","8 points","xoff",7|7,"yoff",7|7)

```

 clust04.xpl

Plot of eight pairs of data is shown in Figure 2

The plot of the dendrogram with single linkage method is shown in Figure 1. If we decide to cut the tree at the level 10 then we find three clusters: {1,2} and {3,4,5} and {6,7,8}.

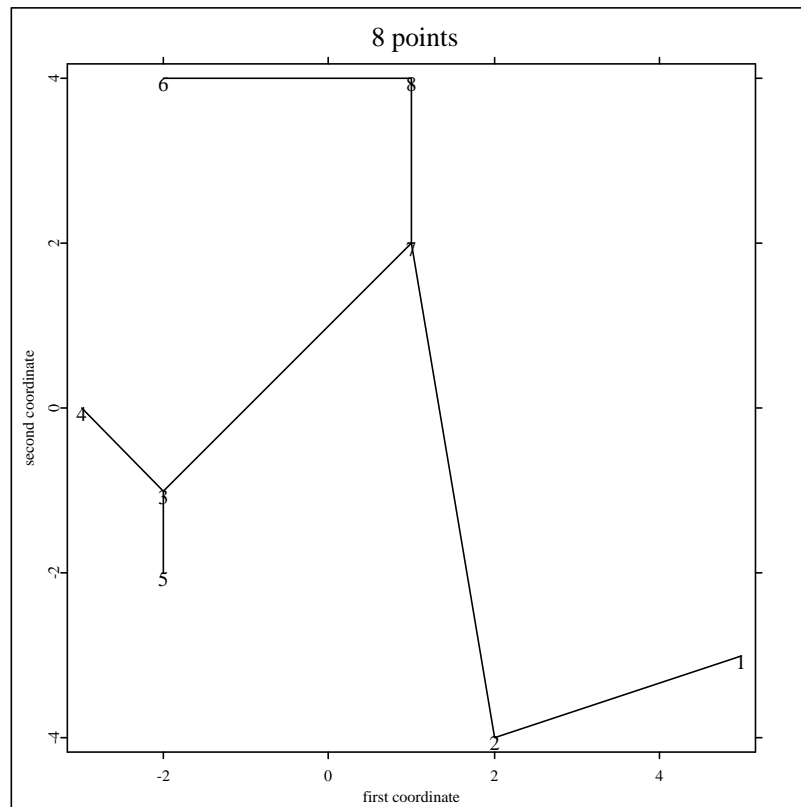


Figure 2: Plot of eight pairs of data.

### 2.1.2 Complete Linkage Method

The Complete linkage method is also called farthest neighbor or maximum distance method. This method is defined by


$$d(R, P + Q) = \max(d(R, P), d(R, Q)) \quad (10)$$

If we change SINGLE into COMPLETE in the example above

```
...  
t=agglom(d.*d, "SINGLE", 5)      ; here single linkage method  
...
```

then we get

```
...  
t=agglom(d.*d, "COMPLETE", 5)   ; here complete linkage method  
...
```

 clust05.xpl

The dendrogram is shown in Figure 3. If we decide to cut the tree at the level 10 then we find three clusters: {1, 2}, {3, 4, 5} and {6, 7, 8}. This method proceeds exactly as the single linkage method except that at the crucial step of revising the distance matrix, the maximum instead of the minimum distance is used to look for the new item.

Both of these two methods are

- relatively sensitive to outliers,
- invariant under monotone transformation of proximity,
- and dependent on the metric.

The single linkage method tends to maximize connectivity in a closeness sense, whereas the maximization method typically leads to more clustering, with smaller, tighter, and more compact clusters.

### 2.1.3 Average Linkage Method

The average linkage method is the hierarchical method that avoids the extremes of either large clusters or tight compact clusters. This method appears as a compromise between the nearest and the farthest neighbor methods.

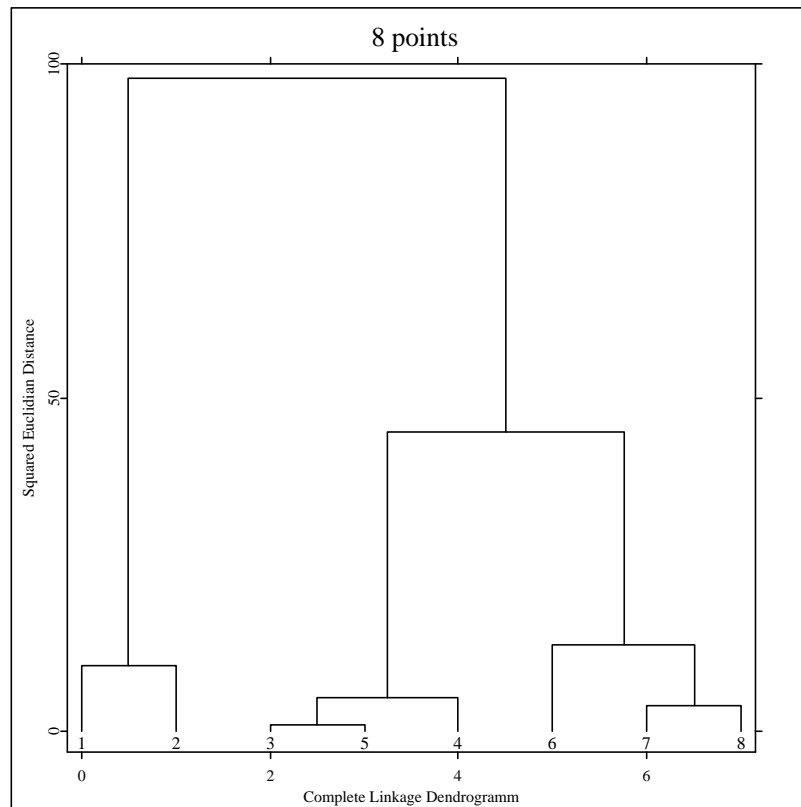



Figure 3: Plot of a dendrogram with complete linkage method.

The simple average linkage (mean linkage) method takes both elements of the new cluster into account:

$$d(R, P + Q) = 1/2 (d(R, P) + d(R, Q)) \quad (11)$$

After the new distances are computed the matrix is reduced by one element of the new cluster. The algorithm loops back to find the next minimum value and continues until all objects are united into one cluster. However, this method is not invariant under monotone transformation of the distance.


If we change SINGLE into AVERAGE in the example above then we get as follows,

```
...
t=agglom(d.*d, "AVERAGE", 5)    ; here average linkage method
...
 clust06.xpl
```

The dendrogram is shown in Figure 4. If we decide to cut the tree at the level 10 then we find three clusters: {1, 2}, {3, 4, 5} and {6, 7, 8}.

#### 2.1.4 Centroid Method

Everitt (1993) explained that with the centroid method, groups once formed are represented by their mean values for each variables (mean vector), and inter-group distance is defined in terms of distance between two such mean vectors. The use of the mean strictly implies that the variables are on an interval scale.

Figure 5 is plot of a dendrogram using centroid linkage method based on the eight pairs of data with the quantlet  clust07.xpl.

#### 2.1.5 Median Method

If the sizes of two groups to be merged are very different, then the centroid of the new group will be very close to that of the larger group and may remain within that group. This is the disadvantage of the centroid method. For that reason, Gower (1967) suggested an alternative strategy, called **median** method

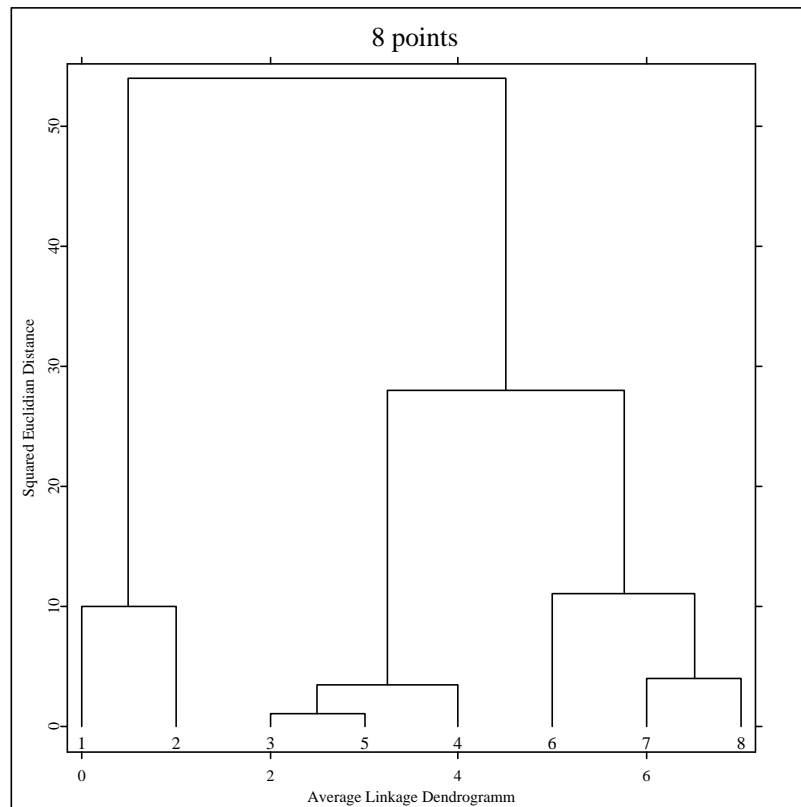


Figure 4: Plot of a dendrogram with average linkage method.



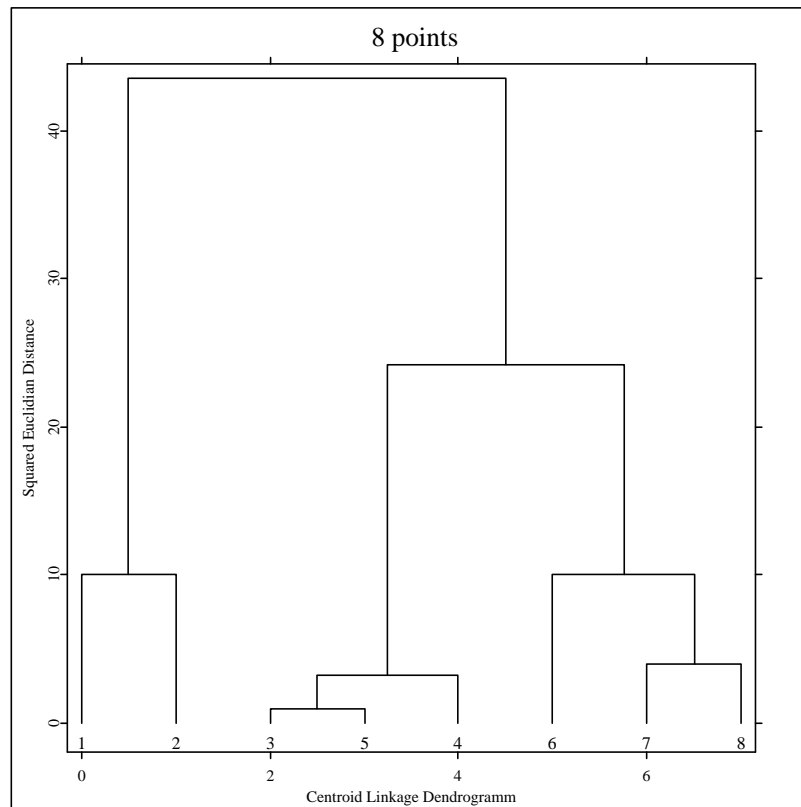


Figure 5: Plot of a dendrogram with centroid linkage method.

because this method could be made suitable for both similarity and distance measures.

Plot of a dendrogram using median method based on the eight pairs of data is as in Figure 6 with the quantlet `clust08.xpl`.

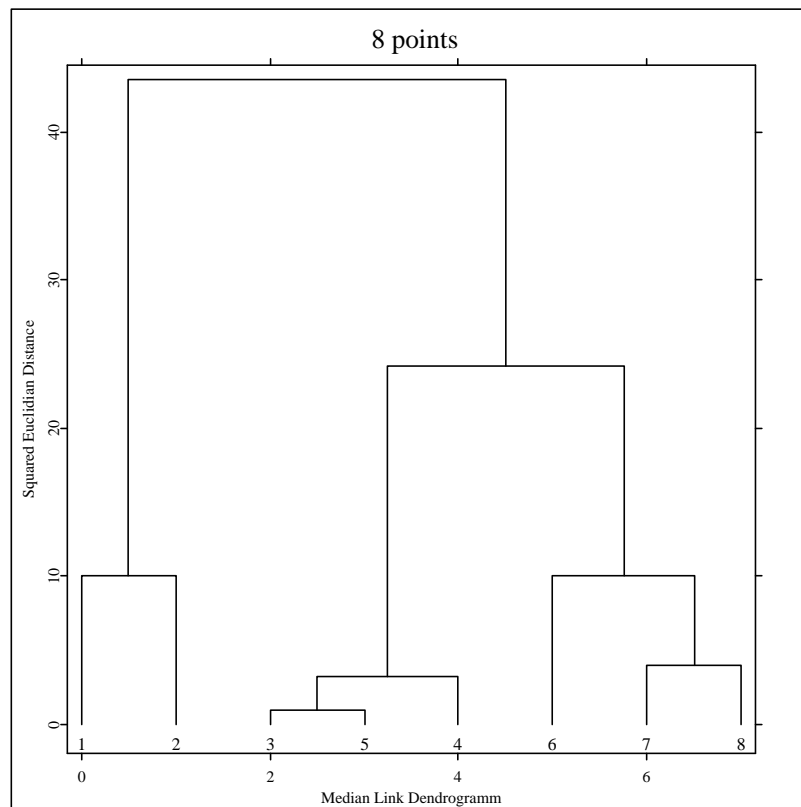


Figure 6: Plot of a dendrogram with median method.

### 2.1.6 Ward Method

```

cw = wardcont(x, k, 1)
    performs Ward's hierarchical cluster analysis of the rows as well as
    of the columns of a contingency table including the multivariate
    graphic using the correspondence analysis; makes available the
    factorial coordinates of the row points and column points (scores)

```

Ward (1963) proposed a clustering procedure seeking to form the partitions  $P_k, P_{k-1}, \dots, P_1$  in a manner that minimizes the loss associated with each grouping and to quantifies that loss in readily interpretable form. Information loss is defined by Ward in terms of an error sum-of-squares (ESS) criterion. ESS is defined as the following

$$ESS = \sum_{k=1}^K \sum_{x_i \in C_k} \sum_{j=1}^p (x_{ij} - \bar{x}_{kj})^2 \quad (12)$$

with the cluster mean  $\bar{x}_{kj} = \frac{1}{n_k} \sum_{x_i \in C_k} x_{ij}$ , where  $x_{ij}$  denotes the value for the  $i$ -th individual in the  $j$ -cluster,  $k$  is the total number of clusters at each stage, and  $n_j$  is the number of individuals in the  $j$ -th cluster.

The corresponding quantlet in XploRe as below

```

t = agglom (d, "WARD", 2)

```

The main difference between this method and the linkage methods consists in the unification procedure. This method does not put together groups with smallest distance, but it joins groups that do not increase too much a given measure of heterogeneity. The aim of the Ward method is to unify groups such that the variation inside these groups is not increased too drastically. This results groups in clusters that are as homogeneous as possible.

The following quantlet gives an example of how to show the dendrogram with the WARD method in XploRe.

In this example we use bank2 dataset taken from Flury and Riedwyl (1988). This dataset consists of 200 measurements on Swiss bank notes. One half of these bank notes are genuine, the other half are forged bank notes. The variables that use in this data set as follows:  $X_1$  = length of the bill,  $X_2$  = height of the bill (left),  $X_3$  = height of the bill (right),  $X_4$  = distance of the


inner frame to the lower border,  $X_5$  = distance of the inner frame to the upper border,  $X_6$  = length of the diagonal of the central picture.

After starting, we compute the euclidean distance between banknotes:

```
proc()=main()
  x=read("bank2")
  i=0                                ; compute the euclidean distance
  d=0.*matrix(rows(x),rows(x))
  while (i.<cols(x))
    i = i+1
    d = d+(x[,i] - x[,i]')^2
  endo
  d = sqrt(d)
```

Next, we use the WARD method and show the dendrogram

```
t = agglom (d, "WARD", 2) ; use WARD method
g = tree (t.g, 0)         ; to cluster the data
g=g.points
l = 5.*(1:rows(g)/5) + (0:4)' - 4
setmaskl (g, l, 0, 1, 1)
setmaskp (g, 0, 0, 0)
d = createdisplay (1,1)
show (d, 1, 1, g)        ; show the dendrogram
endp
;
main()
```

 clust09.xpl

The result gives the partition of the data into 2 clusters and dendrogram is plotted in Figure 7. With Ward method, we see that only one observation, namely 70-th observation, belongs to the false cluster. The rest of observations belong to the same groups.

```
[ 1,]      1
[ 2,]      1
...
...
[ 68,]     1
```

```

[ 69,]      1
[ 70,]      2
[ 71,]      1
[ 72,]      1
....
....
[ 99,]      1
[100,]      1
[101,]      2
[102,]      2
...
...
[199,]      2
[200,]      2

```

The other quantlet that we use is `wardcont`. The aim of this quantlet is to perform Ward's hierarchical cluster analysis of the rows as well as of the columns of a contingency table. It includes the multivariate graphic using the correspondence analysis. It makes available the factorial coordinates of the row points and column points (scores).

The syntax of this quantlet is as follows.

```
cw = wardcont (x, k, l)
```

where  $x$  is an  $n \times p$  matrix of  $n$  row points to be clustered (the elements must be  $> 0$ , with positive marginal sums),  $k$  is scalar the maximum number of clusters of rows, and  $l$  is scalar the maximum number of clusters of columns.

For an example, we use `bird` dataset taken from Mucha (1992). This dataset consists of 412 area (each area = 1 quadrat km) and 102 kinds of bird. The area is divided into 12 groups and the kinds of birds are divided into 9 groups.

After loading the quantlib `xclust`, we apply the `wardcont` method:

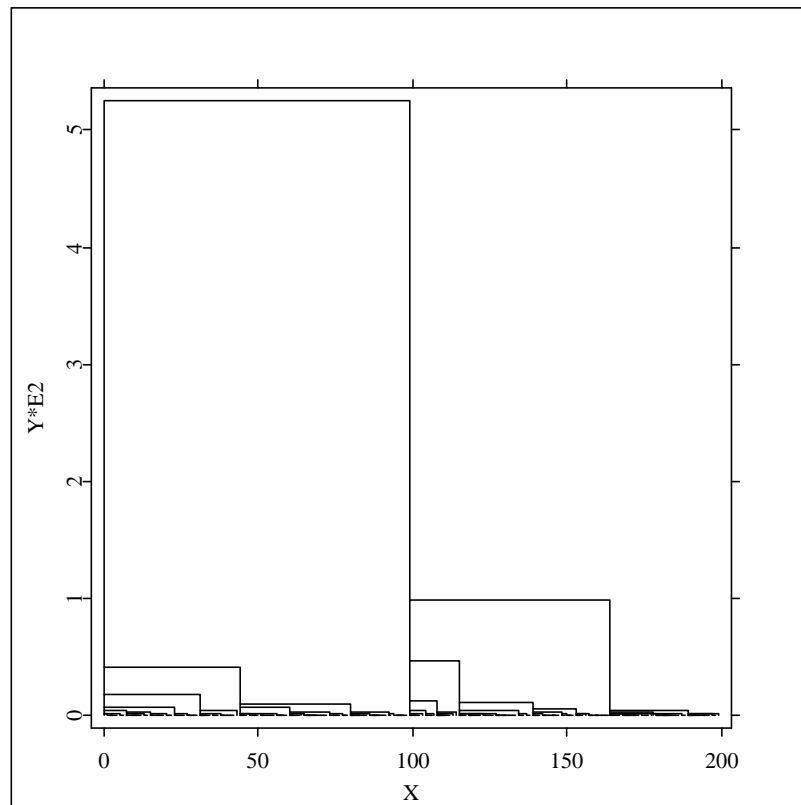



Figure 7: Dendrogram for 200 Swiss banknotes data .

```
library("xclust")
x=read("bird.dat")
cw = wardcont(x, 3, 3)
```

 clust10.xpl

Figures 8, 9, and 10 visualize the matrix correspondence analysis scores of the rows points, the columns points, and both rows and columns.

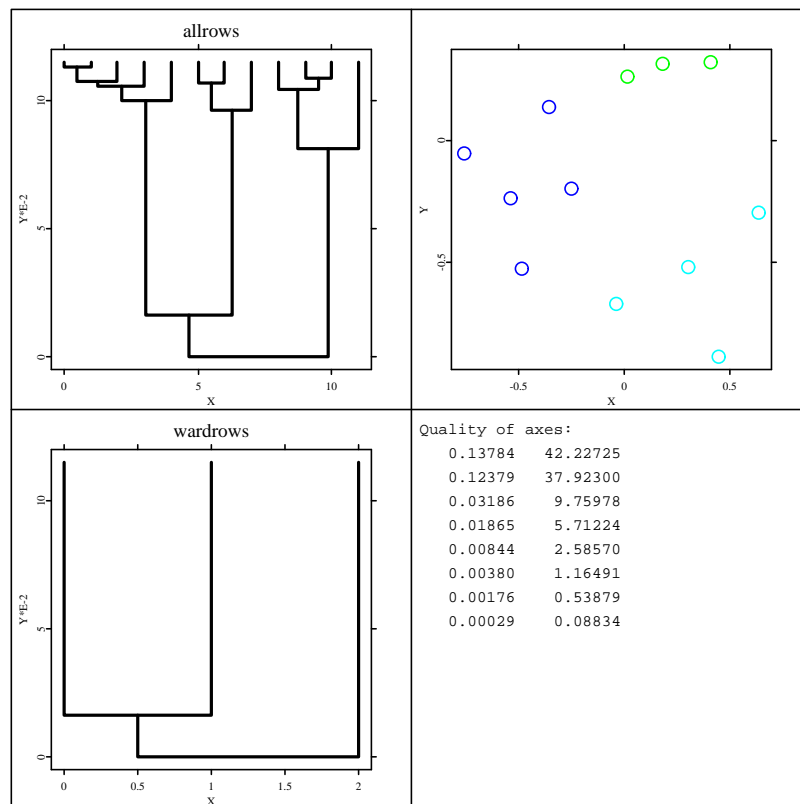


Figure 8: Correspondence analysis scores of the row points

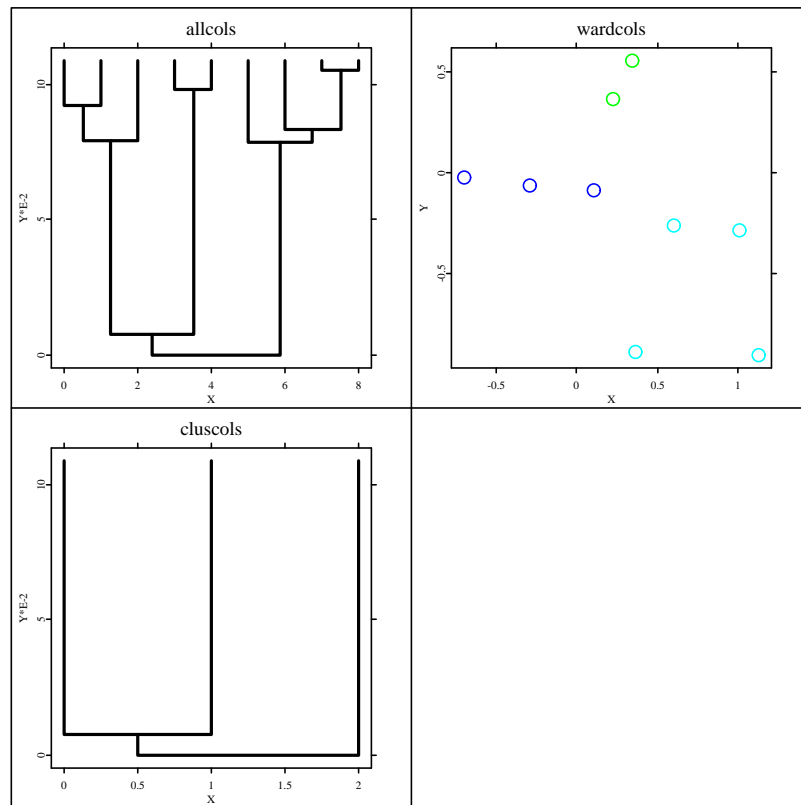


Figure 9: Correspondence analysis scores of the column points.



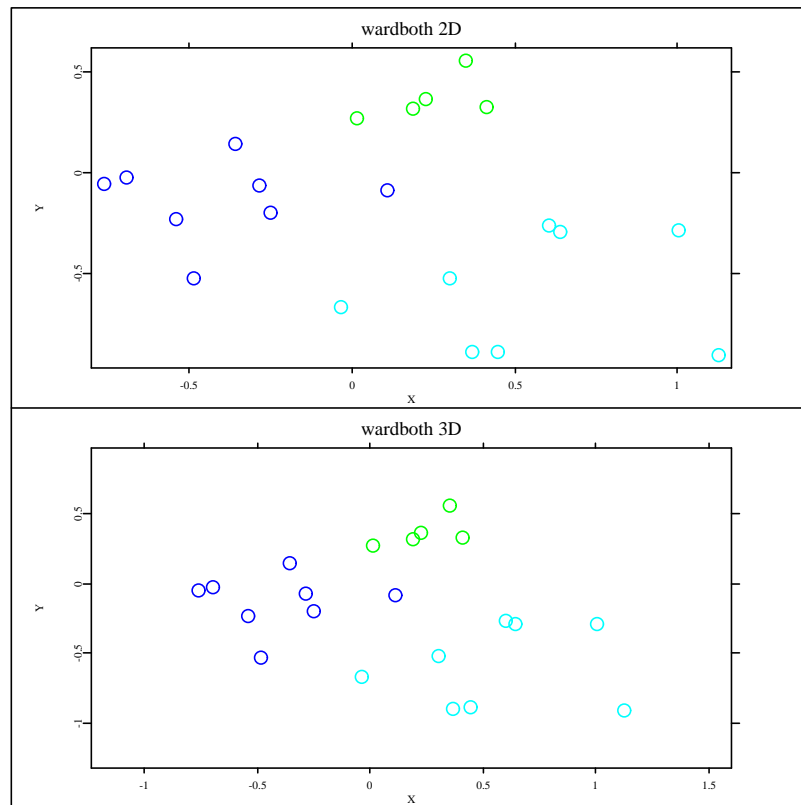


Figure 10: Correspondence analysis scores of both rows and columns.

## 2.2 Divisive Hierarchical Methods

```
cd = divisive(x , k, w, m, sv)
      performs an adaptive divisive K-means cluster analysis with ap-
      propriate (adaptive) multivariate graphic using principal compo-
      nents
```

The divisive hierarchical methods proceed in the opposite way of the agglomerative hierarchical method. In this method, an initial single group of objects is divided into two groups such that the objects in one subgroup are far from the objects in the other. We can divide this method into two types: **monothetic**, which divides the data on the basis of the possession of a single specified attribute, and **polythetic**, where divisions are based on the values taken by several attributes.

The `divisive` quantlet in XploRe performs an adaptive divisive K-means cluster analysis with an appropriate (adaptive) multivariate graphic using principal components:

```
cd = divisive (x, k, w, m, sv)
```

where  $\mathbf{x}$  is an  $n \times p$  matrix of  $n$  row points to be clustered,  $k$  is the number of clusters,  $\mathbf{w}$  is a  $p \times 1$  matrix of weights of column points,  $\mathbf{m}$  is a  $n \times 1$  matrix of weights (masses) of row points, and  $\mathbf{sv}$  is scalar seed value for random numbers.

The outputs of this quantlet `divisive` are: `cd.p` is the partition of  $n$  points of  $\mathbf{x}$  into  $k$  clusters, `cd.n` is the number of observations of clusters, `cd.a` is the matrix of final (pooled) adaptive weights of the variables.

We illustrate the usage of quantlet `divisive` in the following example.

After loading the quantlib `xclust`, we generate random data with 4 clusters,


```
randomize(0)
x  = normal(30, 5)
x1 = x - #(2,1,3,0,0)'
x2 = x + #(1,1,3,1,0.5)'
x3 = x + #(0,0,1,5,1)'
x4 = x - #(0,2,1,3,0)'
x  = x1|x2|x3|x4
```

Then, we compute column variances and row weights

```
w = 1./var(x)
m = matrix(rows(x))
```

Next, we apply divisive methods and compare the results between estimated and true partition,

```
cd = divisive (x, 4, w, m, 1111)
conting (cd.p, ceil((1:120)/30))
```

 clust11.xpl

The result is the following:

Contents of h				
[1,]	0	30	0	0
[2,]	0	0	30	0
[3,]	30	0	0	0
[4,]	0	0	0	30

The output is the crosstable of 120 observations that divide into four clusters. Each cluster consists of 30 observations and corresponds to the given class without any error.

### 3 Nonhierarchical Clustering

Nonhierarchical clustering possesses as a monotonically increasing ranking of strengths as clusters themselves progressively become members of larger clusters. These clustering methods do not possess tree-like structures and new clusters are formed in successive clustering either by merging or splitting clusters.

One of the nonhierarchical clustering methods is the partitioning method. Consider a given number of clusters, for example  $g$ , as the objective and the partition of the object to obtain the required  $g$  clusters. In contrast to the hierarchical method, this partitioning technique permits objects to change group membership through the cluster formation process. The partitioning method usually begins with an initial solution, after which reallocation occurs according to some optimality criterion.

Partitioning method constructs  $k$  clusters from the data as follows:

- Each clusters consists of at least one object  $n$  and each object  $k$  must be belong to one clusters. This condition implies that  $k \leq n$ .
- The different clusters cannot have the same object, and the construct  $k$  groups up to the full data set.

Note:  $k$  is determined by the user so it is better to run algorithms more times to select  $k$  that perform best characteristics. It is also possible to generate the value of  $k$  automatically and then choose the best one  $k$  under certain criteria.

### 3.1 K-means Method

```
ckm = kmeans(x, b, it{, w, m})  
    performs cluster analysis, i.e. computes a partition of  $n$  row points  
    into  $K$  clusters.  
  
ck = kmcont(x, k, t)  
    performs a K-means cluster analysis of the rows of a con-  
    tingency table, including the multivariate graphic using the  
    correspondence analysis, it makes the factorial coordinates  
    (scores)available.
```

This method is developed by Queen (1967). He suggests the name K-means for describing his algorithm that assigns each item to the cluster having the nearest centroid (mean). This process consists of three steps

1. Partition the items into  $K$  initial clusters
2. Proceed through the list of items, assigning an item to the cluster whose centroid (mean) is nearest. Recalculate the centroid for the cluster receiving in the new item and for the cluster losing the item.
3. Repeat step 2 until no more assignments take place

It is better to determine  $K$  initial centroids (seed points) first, before proceeding to step 2.

This method tries to minimize the sum of the within-cluster variances.

$$V_K = \sum_{k=1}^K \sum_{i=1}^n \delta_{ik} m_i d^2(x_i - \bar{x}_k) \quad (13)$$

The indicator function  $\delta_{ik}$  equals 1 if the observation  $x_i$  comes from cluster  $k$ , or 0 otherwise. Furthermore, the element  $\bar{x}_{kj}$  of the vector  $\bar{x}_k$  is the mean value of the variable  $j$  in the cluster  $k$ :

$$\bar{x}_{kj} = \frac{1}{n_k} \sum_{i=1}^I \delta_{ik} m_i x_{ij} \quad (14)$$

We denote the mass of the cluster  $k$  with  $n_k$ , which is equal to the sum of the masses of all observations belonging to the cluster  $k$ .

The above criterion of the K-means method can be derived straightforwardly by using the Maximum Likelihood approach assuming that the populations are independently and normally distributed.

Below is the usage of the quantlet `kmeans` in `XploRe`. It computes a partition of  $n$  row points into  $K$  clusters.

```
ckm = kmeans (x, b, it{, w, m})
```

where  $x$  is  $n \times p$  matrix,  $b$  is  $n \times 1$  matrix, giving the initial partition (for example random generated numbers of clusters  $1, 2, \dots, K$ ),  $it$  is number of iterations,  $w$  is  $p \times 1$  matrix with the weights of column points, and  $m$  is  $n \times 1$  matrix of weights (masses) of row points.


The output of this quantlet `kmeans` consists of `cm.g` which is a matrix containing the final partition, `cm.c` is a matrix of means (centroids) of the  $K$  clusters, `cm.v` is a matrix of within cluster variances divided by the weight (mass) of clusters, and `cm.s` is a matrix of the weight (mass) of clusters.

In the following example, we generate random data with 3 clusters

```
randomize(0)
x = normal(100, 4) ; generate random normal data
x1 = x - #(2,1,3,0)'
x2 = x + #(1,1,3,1)'
x3 = x + #(0,0,1,5)'
x = x1|x2|x3
b = ceil(uniform(rows(x)).*3) ; generate a random partition
```

furthermore, we apply K-means clustering to the data and show the initial partition and the final partition

```
{g, c, v, s} = kmeans(x, b, 100)
b~g
```

 clust12.xpl

The results of the initial and the final partition of the data in 3 clusters are as follows.

```
Contents of object _tmp
[ 1,]      1      2
[ 2,]      3      2
[ 3,]      1      2
...
[297,]     1      1
[298,]     2      1
[299,]     1      1
[300,]     2      1
```

### 3.2 Adaptive K-means Method

```
ca = adaptive(x, k, w, m, t)
    performs an adaptive K-means cluster analysis with appropriate
    (adaptive) multivariate graphic using the principal components
```

In order to increase the stability in cluster analysis, specific weights or adaptive weights in the distance formula could be applied rather than ordinary weight  $q_{jj} = \frac{1}{s_j^2}$  or  $q_{jj} = 1$ .

For example, the simple adaptive weights

$$q_{jj} = \frac{1}{\bar{s}_j^2} \quad (15)$$

can be used in the squared weighted Euclidean distance, where  $\bar{s}_j$  is the pooled

standard deviation of the variable  $j$ :

$$\bar{s}_j^2 = \frac{1}{M} \sum_{k=1}^K \sum_{i=1}^n \delta_{ik} m_i (x_{ij} - \bar{x}_{kj})^2 \quad (16)$$

The indicator  $\delta_{ik}$  is defined in the usual way. For simplicity, use  $M = \sum_{i=1}^K m_i$ ,  $i = 1, 2, \dots, n$ , i.e.  $M$  becomes independent from the number of clusters  $K$ .


The “true” pooled standard deviations cannot be computed in cluster analysis in advance because the cluster analysis structure is usually unknown. Otherwise, it is known that the pooled standard deviations concerning a random partition are nearly equal to the total standard deviations. Therefore, starting with the weights  $q_{jj} = 1/s_j^2$  and a random initial partition  $P^0(n, K)$  the K-means method computes a (local) optimum partition  $P^1(n, K)$  of  $I$  observations into  $K$  clusters.

Below is the quantlet `adaptive` to performs an adaptive K-means cluster analysis with appropriate (adaptive) multivariate graphic using the principal components

```
ca = adaptive(x, k, w, m, t)
```

Following is the example of adaptive clustering in XploRe

```
randomize(0)
x = normal(200, 5) ; generate random data with 3 clusters
x1 = x - #(2,1,3,0,0)'
x2 = x + #(1,1,3,1,0.5)'
x3 = x + #(0,0,1,5,1)'
x = x1|x2|x3
w = 1./var(x) ; compute column variances
m = matrix(rows(x)) ; generate true partition
t = matrix(200)|matrix(200).+1|matrix(200).+2
ca = adaptive (x, 3, w, m, t) ; apply adaptive clustering
```

 clust13.xpl

The result is shown below, it gives a partition `ca.b` of the row points into 3 clusters which minimizes the sum of within cluster variances according to the column weights (1/pooled within cluster variances).

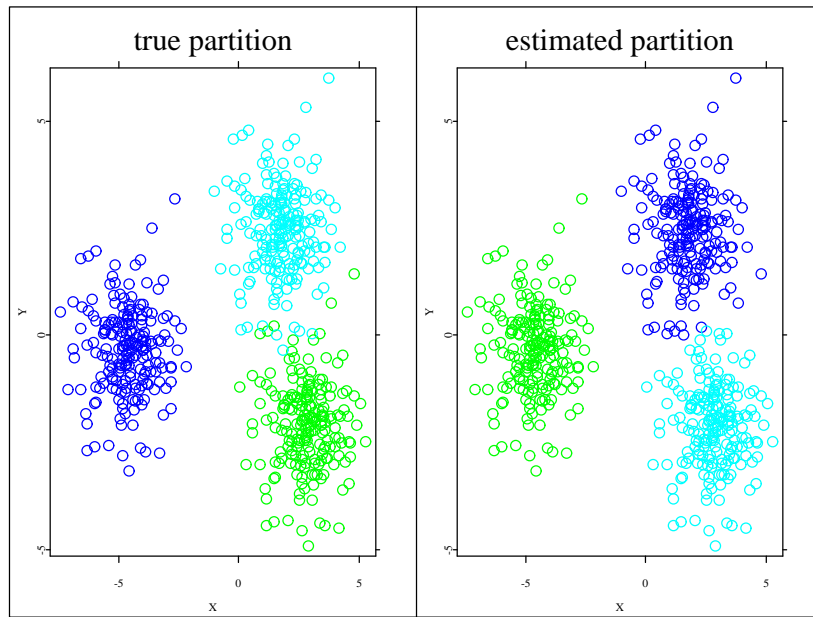


Figure 11: Start and final partition with adaptive clustering.



### 3.3 Hard C-means Method

```
v = xchcme(x, c, m, e)
    performs a hard C-means cluster analysis
```

Fuzzy sets were introduced by Zadeh (1965). It offers a new way to isolate and identify functional relationship - qualitative and quantitative, which also called the pattern recognition.

In general, fuzzy models can be constructed in two ways:

- a fuzzy expert systems, using human knowledge, in a manners similar to the design of knowledge-based fuzzy controllers. Here we use fuzzy theory to improve intelligent systems.
- using numerical data and suitable identification technique

We concentrate only on the identification techniques. One of this techniques is fuzzy clustering method. With a sufficiently informative identification data set, this method does not require any prior knowledge on the partitioning of the domains. Moreover, the use of membership values provides more flexibility and makes the clustering results locally interpretable and often corresponds well with the local behaviour of the identified process.

The idea of **fuzzy clustering** came from Ruspini (1969)'s hard C-means. He introduces the fuzzification of hard C-means to accommodate the intergrades for situations where the groups are not well-separated with hybrid points between groups as:

$$J_m(U, P : X) = \sum_{k=1}^n \sum_{i=1}^c u_{ik} d^2(x_k, v_i), \quad (17)$$

where  $X = (x_1, x_2, \dots, x_n)$  is n data sample vectors,  $U$  is a partition of  $X$  in c part,  $P = (v_1, v_2, \dots, v_c)$  are cluster centers in  $R^p$ ,  $d^2(x_k, v_i)$  is an inner product induced norm on  $R^p$ , and  $u_{ik}$  is referred to as the grade of membership of  $x_k$  to the cluster  $i$ , in this case the member of  $u_{ik}$  is 0 or 1.

The syntax of this algorithm in XploRe is the following:

```
hcm=xchcme(x, c, m, e)
```

The inputs are the following;  $x$  is a  $n \times p$  matrix of  $n$  row points to be clustered,  $c$  is scalar the number of clusters,  $m$  is an exponent weight factor ( $m = 1$ ),  $e$  is termination tolerance, and  $u$  is a  $n \times p$  matrix of initialize uniform distribution.


For an example, we use butterfly data set taken from Bezdek (1981). This data set consists of  $15 \times 2$  matrix. It is called “butterfly” because it scatters like butterfly.

After loading the quantlib `xclust`, we load the data set

```
library("xclust")
z=read("butterfly.dat")
x=z[,2:3]
c=2
m=1
e=0.001
```

and apply hard C-means clustering

```
hcm=xchcme(x,c,m,e)
hcm.clus
d=createdisplay(1,1)
setmaskp(x,hcm.clus,hcm.clus+2,8)
show(d,1,1,x)
title="Hard-c-means for Butterfly Data"
setgopt(d,1,1,"title", title)
```

 clust14.xpl

the result is here

```
Contents of hcm.clus
[ 1,]      2
[ 2,]      2
...
[ 7,]      2
[ 8,]      2
[ 9,]      1
...
[14,]      1
[15,]      1
```

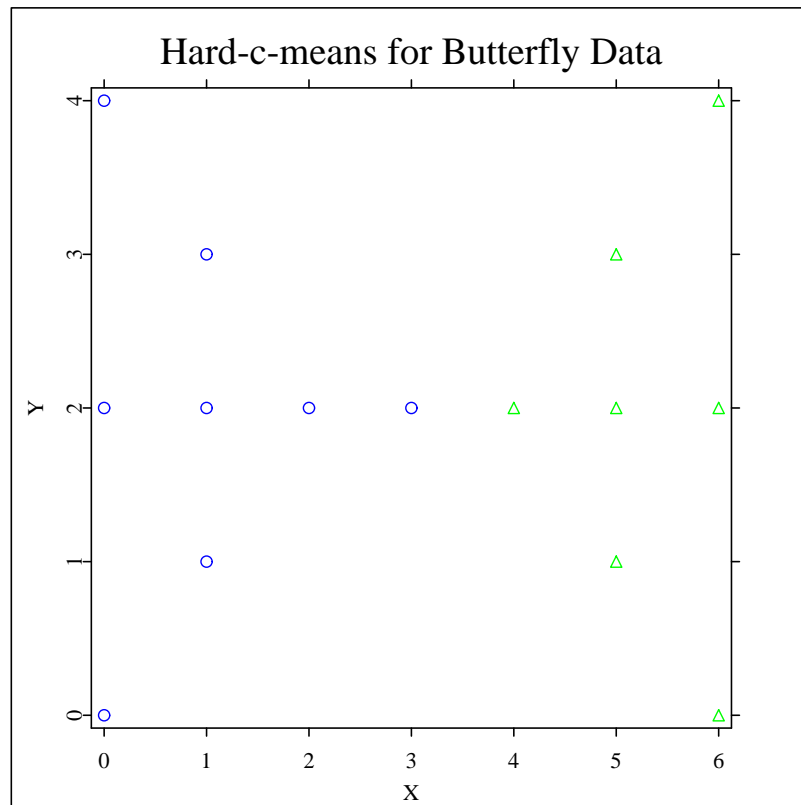


Figure 12: Hard C-means for butterfly data.

From the Figure 12, we can see that the data separate into two clusters. Although the observation number 8 namely (3, 2) exactly in the middle, but this observation must be belong to the first cluster or the second cluster. It can not be constructed another cluster. For this example we see that this observation belong to the first cluster.

### 3.4 Fuzzy C-means Method

```
v = xcfcme(x, c, m, e, alpha)
    Performs a fuzzy C-means cluster analysis
```

One approach to fuzzy clustering, probably the best and most commonly used, is the fuzzy C-means Bezdek (1981). Before Bezdek, Dunn (1973) had developed the fuzzy C-means Algorithm. The idea of Dunn's algorithm is to extend the classical within groups sum of squared error objective function to a fuzzy version by minimizing this objective function. Bezdek generalized this fuzzy objective function by introducing the weighting exponent  $m$ ,  $1 \leq m < \infty$ ;

$$J_m(U, P : X) = \sum_{k=1}^n \sum_{i=1}^c (u_{ik})^m d^2(x_k, v_i), \quad (18)$$

where  $U$  is a partition of  $X$  in  $c$  part,  $P = v = (v_1, v_2, \dots, v_c)$  are the cluster centers in  $R^p$ , and  $A$  is any  $(p \times p)$  symmetric positive definite matrix defined as the following :

$$d^2(x_k, v_i) = \|x_k - v_i\|_A = \sqrt{(x_k - v_i)^T A (x_k - v_i)}, \quad (19)$$

where  $d^2(x_k, v_i)$  is an inner product induced norm on  $R^p$ ,  $u_{ik}$  is referred to as the grade of membership of  $x_k$  to the cluster  $i$ . This grade of membership satisfies the following constraints:

$$\begin{aligned} 0 &\leq u_{ik} \leq 1, \text{ for } 1 \leq i \leq c, 1 \leq k \leq n, \\ 0 &< \sum_{k=1}^n u_{ik} < n, \text{ for } 1 \leq i \leq c, \\ \sum_{i=1}^c u_{ik} &= 1, \text{ for } 1 \leq k \leq n. \end{aligned}$$

The fuzzy C-means (FCM) uses an iterative optimization of the objective function, based on the weighted similarity measure between  $x_k$  and the cluster center  $v_i$ .

Steps of the fuzzy C-means algorithm, according to Hellendorn and Driankov (1997) are the following:

1. Given a data set  $X = \{x_1, x_2, \dots, x_n\}$ , select the number of clusters  $2 \leq c < N$ , the maximum number of iterations  $T$ , the distance norm  $\|\bullet\|_A$ , the fuzziness parameter  $m$ , and the termination condition  $\varepsilon > 0$ .
2. Give an initial value  $U_0 \in M_{fcn}$ .
3. For  $t = 1, 2, \dots, T$

- (a) Calculate the  $c$  cluster centers  $\{v_{i,t}\}, i = 1, \dots, c$

$$v_{i,t} = \frac{\sum_{k=1}^n u_{ik,t-1}^m x_k}{\sum_{k=1}^n u_{ik,t-1}^m} \quad (20)$$

- (b) Update the membership matrix. Check the occurrence of singularities ( $d_{ik,t} = \|x_k - v_{i,t}\|_A = 0$ ). Let  $\Upsilon = \{1, \dots, c\}$ ,  $\Upsilon_{k,t} = \{i \in \Upsilon | d_{ik,t} = 0\}$ , and  $\Upsilon_{k,t} = \Upsilon \setminus \Upsilon_{k,t}$ . Then calculate the following

$$u_{ik,t} = \sum_{j=1}^c \left( \frac{d_{ik,t}}{d_{jk,t}} \right)^{\frac{2}{m-1}}, \text{ if } \Upsilon_{k,t} = 0 \quad (21)$$

Choose  $a_{ik,t} = 1/\#(\Upsilon_{k,t}), \forall i \in \Upsilon; \#(\cdot)$  denotes the ordinal number.

4. If  $E_t = \|U_{t-1} - U_t\| \leq \varepsilon$  then stop otherwise return to step 3.

This procedure converges to a local minimum or a saddle point of  $J_m$ . The FCM algorithm computes the partition matrix  $U$  and the clusters' prototypes in order to derive the fuzzy models from these matrices.

In pseudocode, we can say

```

Initialize membership (U)
iter = 0
Repeat {Picard iteration}
    iter = iter+1
    Calculate cluster center (C)

```

```

    Calculate distance of data to centroid  $||X-C||$ 
     $U'=U$ 
    Update membership  $U$ 
    Until  $||U-U'|| \leq \text{tol\_crit}$  .or.  $\text{iter} = \text{Max\_iter}$ 

```

The syntax of this algorithm in XploRe is

```
fcm=xcfcme(x,c,m,e,alpha)
```


The inputs are the following;  $x$  is a  $n \times p$  matrix of  $n$  row points to be clustered,  $c$  is the number of clusters,  $m$  is an exponent weight factor ( $m > 1$ ),  $e$  is termination tolerance, and  $u$  is  $n \times p$  matrix of initialized uniform distribution.

Below is an example. We use the same data as quantlet `xcfcme`. And also we do exactly the same procedure, except the part of applying fuzzy C-means clustering.

```

library("xclust")
z=read("butterfly.dat")
x=z[,2:3]
c=2
m=1.25
e=0.001
alpha=0.9
fcm=xcfcme(x,c,m,e,alpha) ; apply fuzzy c-means clustering
fcm.clus
d=createdisplay(1,1)
setmaskp(x,fcm.clus,fcm.clus+3,8)
show(d,1,1,x)
title="Fuzzy-c-means for Butterfly Data"
setgopt(d,1,1,"title", title)

```

 clust15.xpl

The result is here

```

Contents of fcm.clus
[ 1,]      1
[ 2,]      1
...

```

```

[ 8,]      3
[ 9,]      2
...
[14,]      2
[15,]      2

```

This result can be shown in Figure 13.

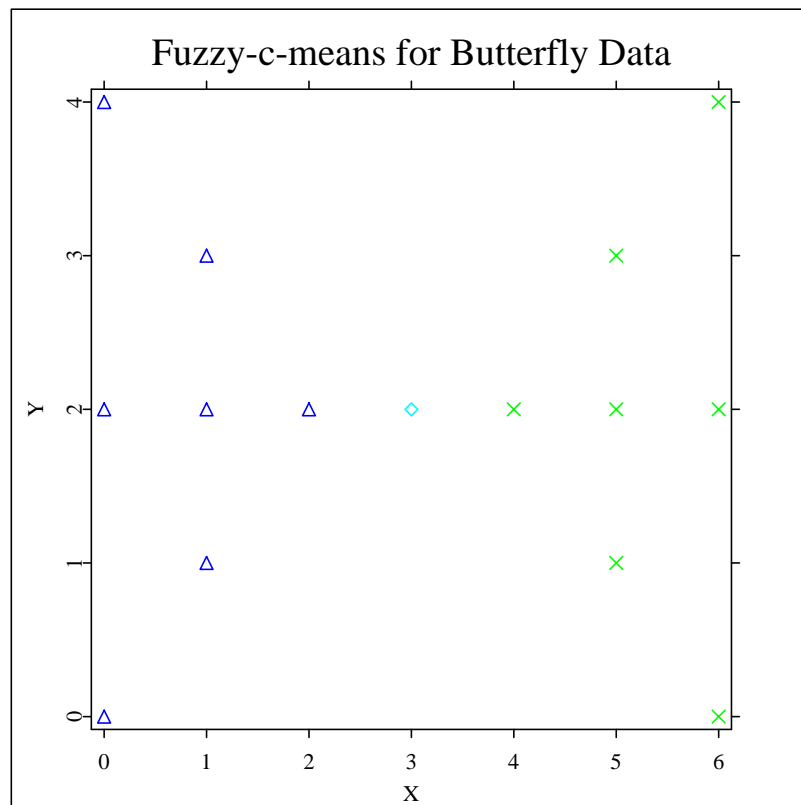


Figure 13: Fuzzy C-means for butterfly data

By using  $m = 1.25$  and  $\alpha = 0.9$ , we can see that, not all observations

belong to the first cluster or the second cluster. But the 8-th observation form a new cluster. Because this observation has the same distance to the center of both previous cluster.

For another example, we use `bank2` that has also explained by Ward method.

After loading the quantlib `xclust`, we load the `bank2` dataset. We do exactly the same with the previous example with the quantlet `clust16.xpl`

The result that we have as follows

```
[ 1,]      1
[ 2,]      1
...
...
[ 98,]     1
[ 99,]     1
[100,]     1
[101,]     2
[102,]     2
[103,]     1
[104,]     2
...
[199,]     2
[200,]     2
```

If we compare to the Ward method depicted by Figure 14, we have not exactly the same cluster. By fuzzy C-mean, the 103-rd observation belongs to the first cluster and by Ward method, the 70-th observation belongs to the second cluster. To make it easy to see how the data to be clustered, below we present the variables;  $X_4$  that is the distance of the inner frame to the lower border, vs  $X_6$  that is the length of the diagonal of the central picture. Using the contingency table, we can conclude that both of these methods constructed the same clusters.

Now, we want to compare both of these methods for three clusters depicted by Figure 15 with the quantlet `clust17.xpl`. Using the contingency table, we see that there are 16 observations in the second cluster of Ward method but these observations are belong to the third cluster of fuzzy C-means.



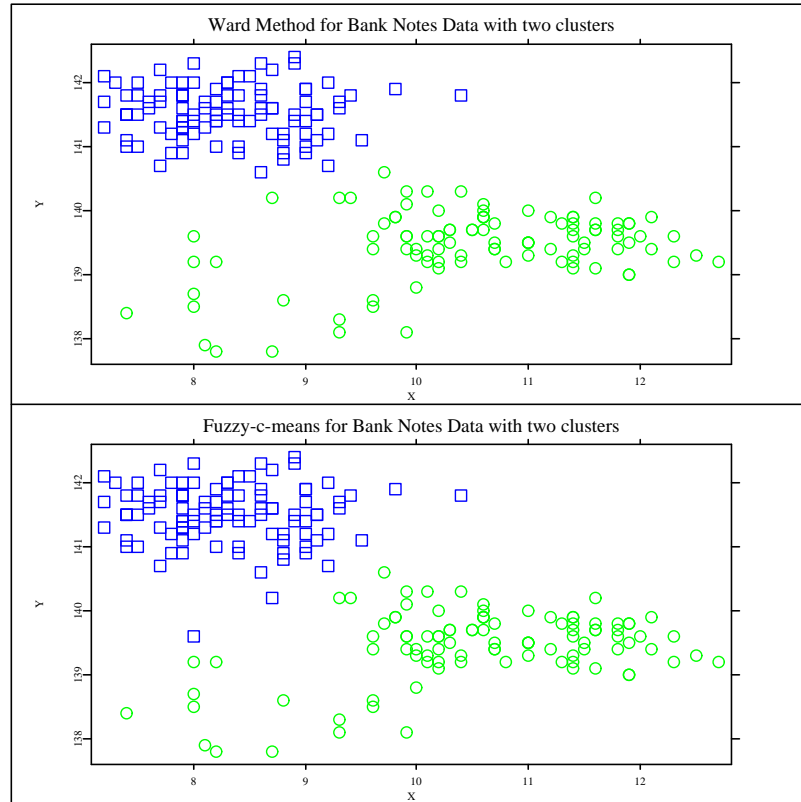


Figure 14: Ward method vs fuzzy C-means for Swiss banknotes data (X4 vs X6) with two clusters

Ward / Fuzzy	Cluster 1	Cluster 2	
Cluster 1	1	99	100
Cluster 2	99	1	100
	100	100	200

Table 3: Contingency table between Ward method vs fuzzy C-means method with two clusters.

Ward / Fuzzy	Cluster 1	Cluster 2	Cluster 3	
Cluster 1	100	0	0	100
Cluster 2	0	48	16	64
Cluster 3	0	0	36	36
	100	48	52	200

Table 4: Contingency table between Ward method vs fuzzy C-means method with three clusters.

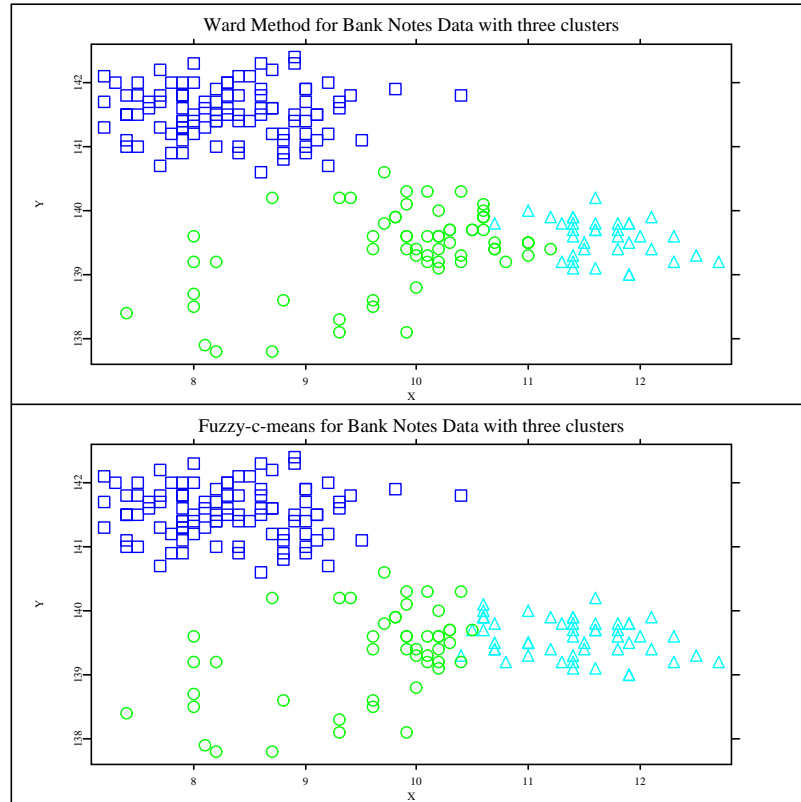


Figure 15: Ward method vs fuzzy C-means for Swiss banknotes data (X4 vs X6) with three clusters

## References

- Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York.
- Bezdek, J. C. and Pal, S. K. (1992). *Fuzzy Models for Pattern Recognition*, IEEE Press, New York.
- Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters, *Journal of Cybernetics* **3**: 32–57.
- Flury, B. and Riedwyl, H. (1988). *Multivariate Statistics, A Practical Approach*, Cambridge University Press.
- Everitt, B. S. (1993). *Cluster Analysis*, Edward Arnold, London.
- Gower, J. C. (1967). A comparison of some methods of cluster analysis, *Biometrics* **23**: 623–628.
- Härdle, W., Klinker, S., and Turlach, B.A. (1995). *XploRe: An Interactive Statistical Computing Environment*, Springer Verlag, New York.
- Härdle, W. and Simar, L. (2000). Applied Multivariate Statistical Analysis, *Lecture Notes*, electronic book available at <http://ise.wiwi.hu-berlin.de>, Humboldt Universität zu Berlin.
- Hellendorn, H. and Driankov, D. (1998). Fuzzy: Model Identification, *Springer Verlag*, Heidelberg.
- Johnson, R. A., and Wichern D. W. (1992). *Applied Multivariate Statistical Analysis*, Prentice-Hall.
- MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, **1**: 281–297.
- Mucha, H. J. (1992). Clusteranalyse mit Microcomputern, *Akademie Verlag*, Berlin.
- Mucha, H. J. (1995). Clustering in an Interactive Way, *Discussion Paper 9513*, Institut für Statistik und Ökonometrie, Humboldt-Universität zu Berlin.

- Mucha, H. J. (1996). CLUSCORR: Cluster Analysis and Multivariate Graphics under MS-EXCEL, *Report No. 10*, WIAS Institut, Berlin.
- Ruspini, E. H. (1969). A New Approach to Clustering, *Information Control* **15**: 22–32.
- Ward, J. H. (1963). Hierarchical grouping to optimize an objective function, *Journal of Amer. Statist. Assoc.* **58**: 236–244.
- Zadeh, L. A. (1965). Fuzzy Sets, *Information Control* **8**: 338–353.