Bachelor's Thesis presented to obtain the Bachelor Degree

Cluster Analysis and CART implemented in XploRe

Submitted to:

Prof. Dr. Wolfgang Härdle
Humboldt-Universität zu Berlin
Faculty of Economics and Business Administration
Department of Statistics and Econometrics

By :

Hizir Sofyan
Lynarstr. 5 App. 65206
13353 Berlin

April 2, 2001

Declaration of Authorship

I hereby confirm that I have authored this thesis independently and without use of other than the indicated resources.

All passages, which are literally or in general manner taken out of publications or other sources, are marked as such.

Hizir Sofyan

Berlin, April 2nd 2001

Acknowledgements

I would like to gratefully acknowledge the help of Prof. Dr. Wolfgang Härdle for the enthusiastic supervision, orientation offered and penetrating criticism.

I am grateful to my colleagues for their collaboration in writing this thesis, Dipl. Math. Hans-Joachim Mucha, Dr. Sigbert Klinke, Dr. Jussi Klemelä.

I thank to all my friends in the institute of statistics and econometrics who have assisted me in the course of this study. Among them Dr. Marlene Müller, Dr. Axel Welwartz, Dr. Zdenek Hlavka, and Dipl. Math. Torsten Kleinow have been particularly helpful and generous with their time and expertise.

For the financial support I would like to thank the Deutscher Akademischer Austauschdienst (DAAD) and Sonderforschungsbereich 373 (SFB 373).

Finally, this thesis is dedicated to my family for their understanding, endless patience, and relentless words of encouragement .

Contents

1 Introduction **9**

2 Cluster Analysis **11**
2.1 Introduction 11
2.1.1 Distance Measures 12
2.1.2 Similarity of Objects 14
2.2 Hierarchical Clustering 15
2.2.1 Agglomerative Hierarchical Methods 16
2.2.2 Divisive Hierarchical Methods 30
2.3 Nonhierarchical Clustering 33
2.3.1 K-means Method 33
2.3.2 Adaptive K-means Method 35
2.3.3 Hard C-means Method 37
2.3.4 Fuzzy C-means Method 39

3 Classification and Regression Trees **49**
3.1 Introduction 49
3.2 Steps in CART 50
3.2.1 Growing the Tree 50
3.2.2 Pruning the Tree 51
3.2.3 Selecting the Final Tree 53
3.2.4 Plotting the result of CART 54
3.3 Examples 55
3.3.1 Simulated Example 55
3.3.2 Boston Housing Data 57
3.3.3 Density Estimation 68

Bibliography **75**

List of Figures

2.1 An example of a dendrogram using eight pairs of data. 16
2.2 Plot of eight pairs of data. 20
2.3 Plot of a dendrogram with complete linkage method. 21
2.4 Plot of a dendrogram with average linkage method. 23
2.5 Plot of a dendrogram with centroid linkage method. 24
2.6 Plot of a dendrogram with median method. 25
2.7 Dendrogram for 200 Swiss banknotes data 28
2.8 Correspondence analysis scores of the row points 29
2.9 Correspondence analysis scores of the column points. 30
2.10 Correspondence analysis scores of both rows and columns. . . 31
2.11 Start and final partition with adaptive clustering. 37
2.12 Hard C-means for Butterfly Data. 40
2.13 Fuzzy C-means for Butterfly Data 43
2.14 Ward method vs fuzzy C-means for Swiss banknotes data (X4 vs X6) with two clusters 45
2.15 Ward Method vs fuzzy C-means for Swiss banknotes data (X4 vs X6) with three clusters 47

3.1 Plot of 100 simulated data from function $f(x_1, x_2)$. The datapoints in the upper left (marked with filled crosses) are in the area of $f(x_1, x_2) = 100$, the datapoints in the upper right (marked with triangles) are in the area of $f(x_1, x_2) = 120$ and the datapoints in the lower part (marked with circles) are in the area of $f(x_1, x_2) = 0$. 56
3.2 Initial regression tree for 100 simulated data from function $f(x_1, x_2)$ (left). The total number of leaves (41) is shown at the right. 58
3.3 Final regression tree for 100 simulated data from function $f(x_1, x_2)$ after pruning. The final tree consists of three leaves which separate the x_1, x_2-plane into three parts. 59

3.4 Initial regression tree for Boston Housing Data. The total number of leaves (29) is shown at the right. 61
3.5 Sub-Tree consisisting of 10 leaves for 20% sample of the Boston Housing Data 62
3.6 Cross Validation for 20% sample of Boston Housing Data. . 64
3.7 Final tree for 20% sample of Boston Housing Data 65
3.8 Final tree for 20% sample of Boston Housing Data with numbers of observations 66
3.9 Final tree for 20% sample of Boston Housing Data with mean values 67
3.10 The upper left plot gives the cuts in the bottom-top plane, the lower left plot the cuts in the bottom-diagonal plane and the lower right plot the cuts in the top-diagonal plane. The CART tree is shown in the upper right window. 71
3.11 The upper left plot gives the cuts in the bottom-top plane, the lower left plot the cuts in the bottom-diagonal plane and the lower right plot the cuts in the top-diagonal plane. The CART tree is shown in the upper right window. 73

1

Introduction

With the ever increasing forms of data collection and the availability of methods to save them electronically, the use of multivariate data analysis techniques has remained very significant. One of these methods is classification. The word 'classification' associates with investigating a group of objects so that the data can be validly summarized into a number of fewer classes, with relatively similar objects in the same class.

Below is the explanation of two methods, which are often used, namely cluster analysis and classification and regression trees (CART) analysis.

Cluster analysis provides a description 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 homogeneous whereas the profiles of objects in different groups are relatively heterogeneous.

CART analysis is a binary splitting method which partitions a data set into discrete sub-groups based on the value of a user-defined classification variable. It employs statistical regression techniques (e.g. stepwise multiple regression) to construct dichotomous decision tree. CART is an efficient method because it can deal with incomplete data, multiple features (float, enumerated sets) both in input features and predicted features, and the trees it produces often contain rules which are humanly readable.

This thesis is supposed to study the implementation of cluster analysis and CART analysis in XploRe (an interactive statistical software).

In the course of writing this thesis, I have worked jointly with Dipl. Math. Hans-Joachim Mucha for the chapter on cluster analysis chapter and Dr. Jussi Klemelä and Dr. Sigbert Klinke for the chapter on CART

analysis. This work contains examples and applications. For this purpose we used the bird data (the data about the different type of birds found in Berlin), Swiss banknote data, and the butterfly data as the samples for cluster analysis. We applied simulated data and the Boston housing data as samples for CART analysis.

This thesis is composed of three main sections prefaced by an introduction. Cluster analysis is presented in the second chapter. This is followed, in chapter three, by a description of Classification on Regression Tree (CART) analysis.

2

Cluster Analysis

2.1 Introduction

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 heteregoneous.

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.

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, etc. Nonhierarchical techniques include K-means, adaptive K-means, K-medoids, fuzzy clustering, etc. 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).

2.1.1 Distance Measures

`d = distance(x{, metric})`
computes the distance between p-dimensional data points depending 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 $x = (x_1, x_2, ..., x_p)^T$ and $y = (y_1, y_2, ..., y_p)^T$, we consider the Euclidean metric defined as

$$d(x, y) = \left[\sum_{i=1}^{p} (x_i - y_i)^2 \right]^{\frac{1}{2}} \tag{2.1}$$

In matrix notation, this is written as the following:

$$d(x, y) = \sqrt{(x - y)^T (x - y)} \tag{2.2}$$

The statistical distance between these two observations is the following

$$d(x, y) = \sqrt{(x - y)^T A (x - y)} \tag{2.3}$$

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 28×1 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.

2.1.2 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); $x_i^T = (x_{i1}, \ldots, x_{ip}), x_j^T = (x_{j1}, \ldots, x_{jp})$, $x_{ik}, x_{jk} \in \{0, 1\}$. Actually, we have four cases:

$$x_{ik} = x_{jk} = 1, x_{ik} = 0, x_{jk} = 1, x_{ik} = 1, x_{jk} = 0, x_{ik} = x_{jk} = 0. \tag{2.4}$$

We define

$$a_1 = \sum_{k=1}^{p} I(x_{ik} = x_{jk} = 1), a_2 = \sum_{k=1}^{p} I(x_{ik} = 0, x_{jk} = 1), \tag{2.5}$$

$$a_3 = \sum_{k=1}^{p} I(x_{ik} = 1, x_{jk} = 0), a_4 = \sum_{k=1}^{p} I(x_{ik} = x_{jk} = 0). \tag{2.6}$$

General measures are used in practice

$$T_{ij} = \frac{a_1 + \delta a_4}{a_1 + \delta a_4 + \lambda(a_2 + a_3)}, \tag{2.7}$$

where δ and λ are weighting vectors. According to the weighting factors we have the following table.

Note that each $a_l, l = 1, \ldots, 4$ 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

Name	δ	λ	$Definition(T(x_i, x_j))$
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 2.1: The common similarity coefficient.

```
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
```

2.2 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 2.1, is the example of the dendrogram by using eight pairs of data above.

clust04.xpl

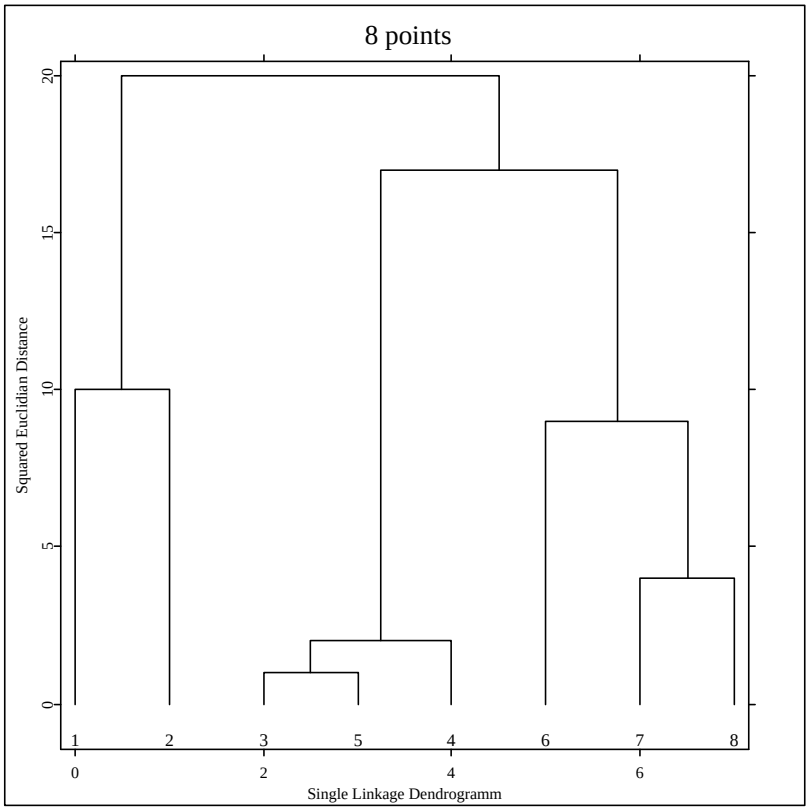


Figure 2.1: An example of a dendrogram using eight pairs of data.

2.2.1 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 and the remaining groups by (2.8) to obtain a reduced distance matrix D
4. **UNTIL** all clusters are agglomerated into one group.

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

(2.8)
$$d(R, P+Q) = \delta_1 d(R,P) + \delta_2 d(R,Q) + \delta_3 d(P,Q) + \delta_4 |(d(R,P) - d(R,Q)|$$

The δ_j's are weighting factors that lead to different agglomerative algorithms as described in Table 2.2. 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 β that is specified by the user. Below is the 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 $(1, 2, \ldots)$, `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 $(1, 2, \ldots)$, and `cagg.d` is a vector matrix with distances between the cluster centers.

Single Linkage Method

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

$$d(R, P+Q) = min\,(d(R,P), d(R,Q)) \tag{2.9}$$

Name	δ_1	δ_2	δ_3	δ_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	$\frac{1-\beta}{2}$	$\frac{1-\beta}{2}$	β	0

Table 2.2: Computation of group distances available in XploRe.

The process is continous 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 undesireable 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 2.1

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("%.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 lingkage 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)
```

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 2.2

The plot of the dendrogram with single linkage method is shown in Figure 2.1. If we decide to cut the tree at the level 10 then we find three clusters: $\{1, 2\}$ and $\{3, 4, 5\}$ and $\{6, 7, 8\}$.

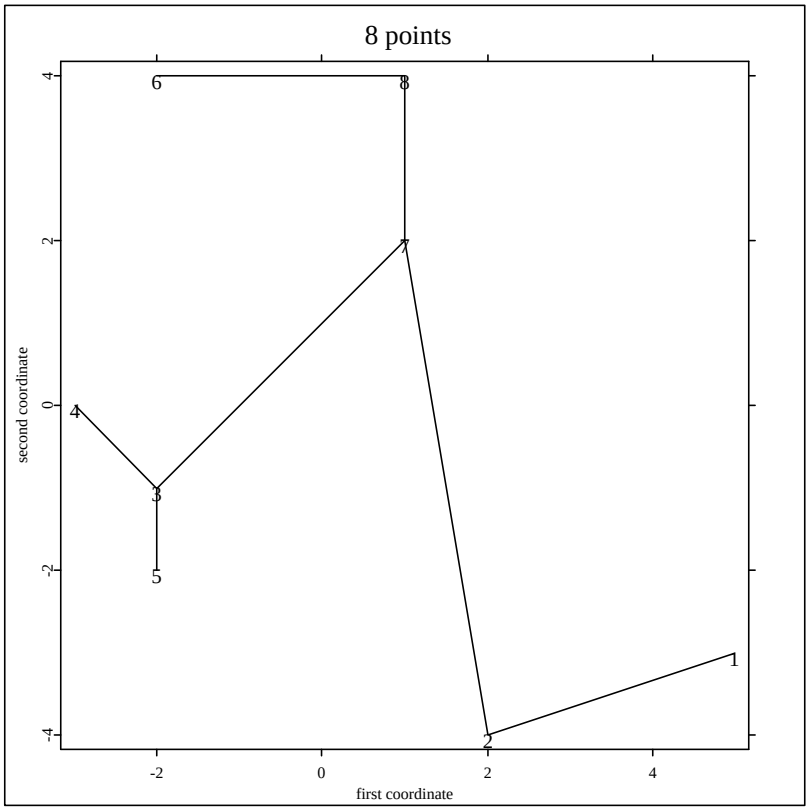


Figure 2.2: Plot of eight pairs of data.

Complete Linkage Method

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

$$d(R, P+Q) = max\,(d(R,P), d(R,Q)) \tag{2.10}$$

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 2.3. 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

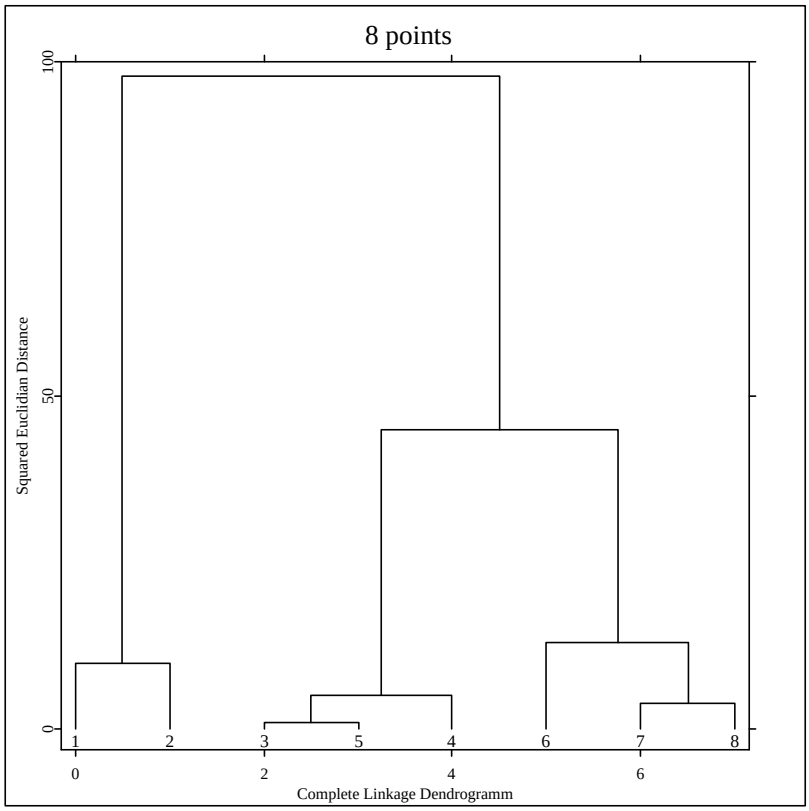


Figure 2.3: Plot of a dendrogram with complete linkage method.

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.

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.

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

$$d(R, P + Q) = 1/2\,(d(R, P) + d(R, Q)) \tag{2.11}$$

After the new distances are computed the matrix is reduced by one element of the new cluster. The algorithms 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 2.4. If we decide to cut the tree at the level 10 then we find three clusters: $\{1, 2\}, \{3, 4, 5\}$ and $\{6, 7, 8\}$.

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

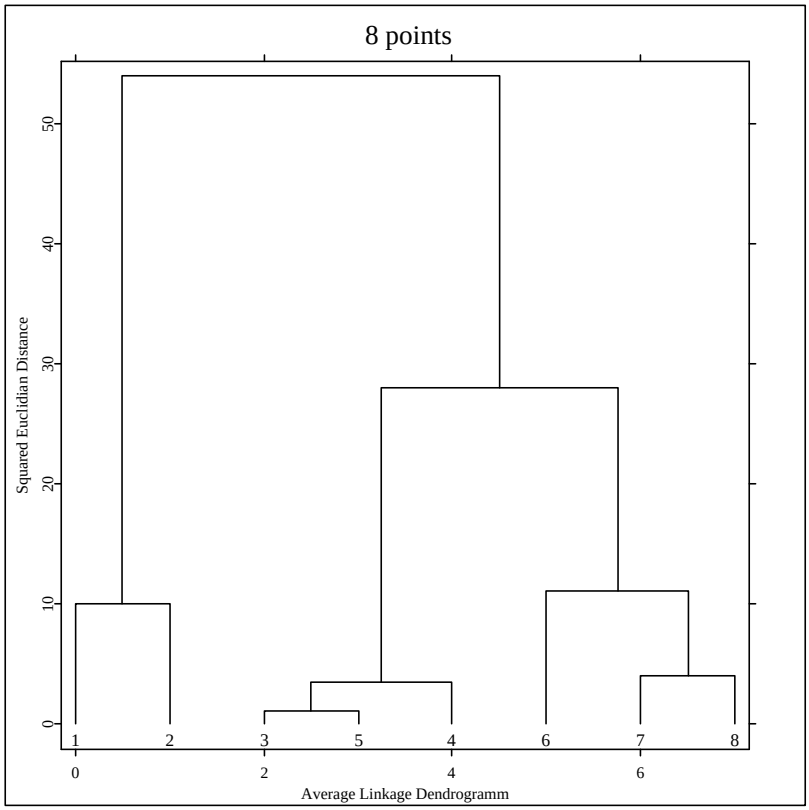


Figure 2.4: Plot of a dendrogram with average linkage method.

vectors. The use of the mean strictly implies that the variables are on an interval scale.

Figure 2.5 is plot of a dendrogram using centroid linkage method based on the eight pairs of data with the quantlet `clust07.xpl`.

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 stategy, called **median** method because this method could be made suitable for both similarity and

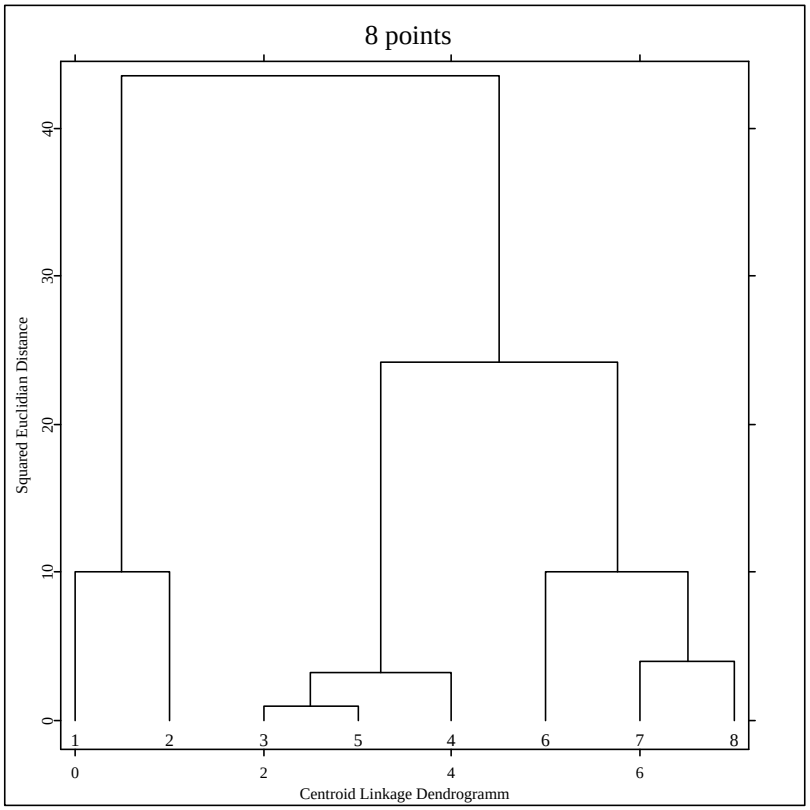


Figure 2.5: Plot of a dendrogram with centroid linkage method.

distance measures.

Plot of a dendrogram using median method based on the eight pairs of data is as in Figure 2.6 with the quantlet `clust08.xpl`.

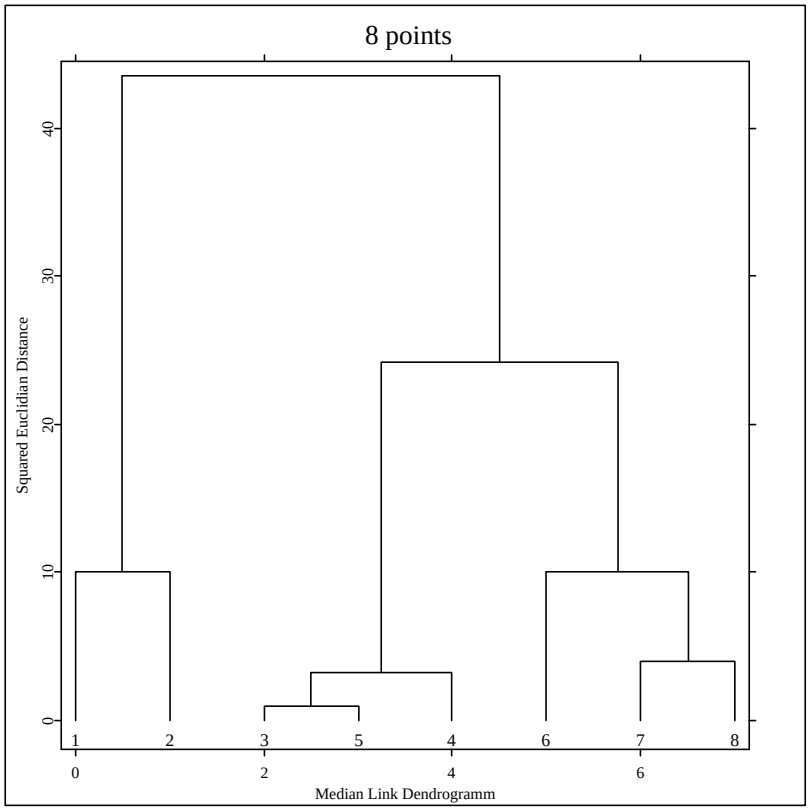


Figure 2.6: Plot of a dendrogram with median method.

Ward Method

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

$P_k, P_{k-1}, \ldots, P_1$ 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

$$ESS = \sum_{k=1}^{K} \sum_{x_i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2 \tag{2.12}$$

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 heterogenity. 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 2.7. 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

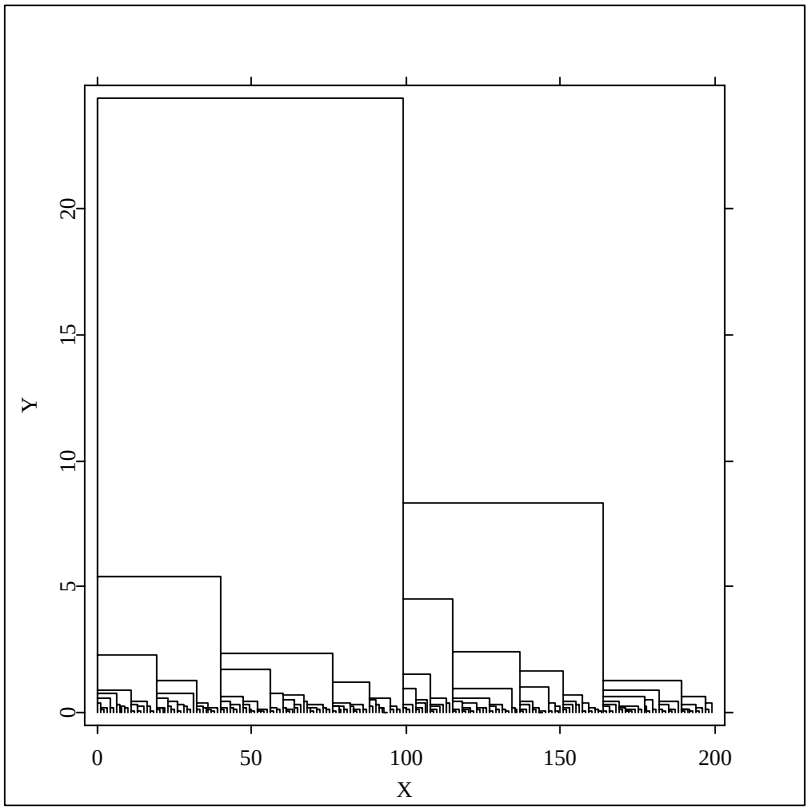


Figure 2.7: Dendrogram for 200 Swiss banknotes data .

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:

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

clust10.xpl

Figures 2.8, 2.9, and 2.10 visualize the matrix correspondence analysis scores of the rows points, the columns points, and both rows and columns.

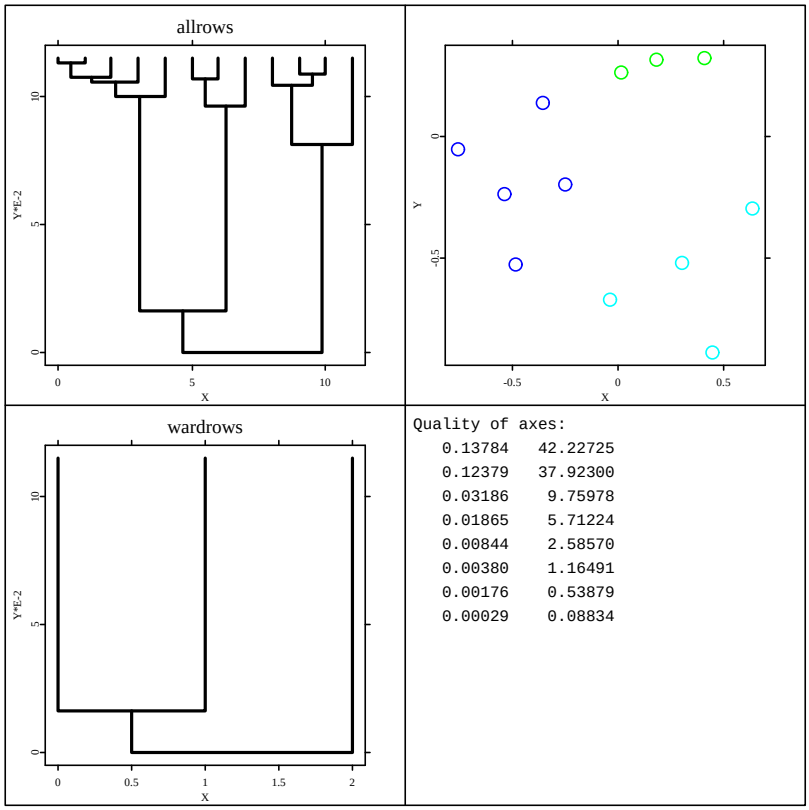


Figure 2.8: Correspondence analysis scores of the row points

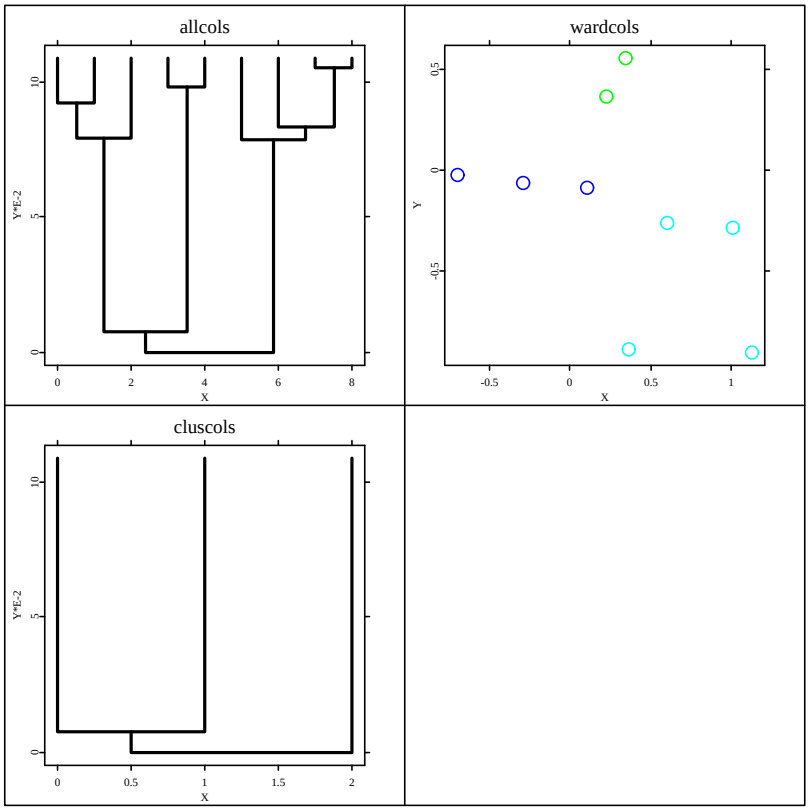


Figure 2.9: Correspondence analysis scores of the column points.

2.2.2 Divisive Hierarchical Methods

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

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

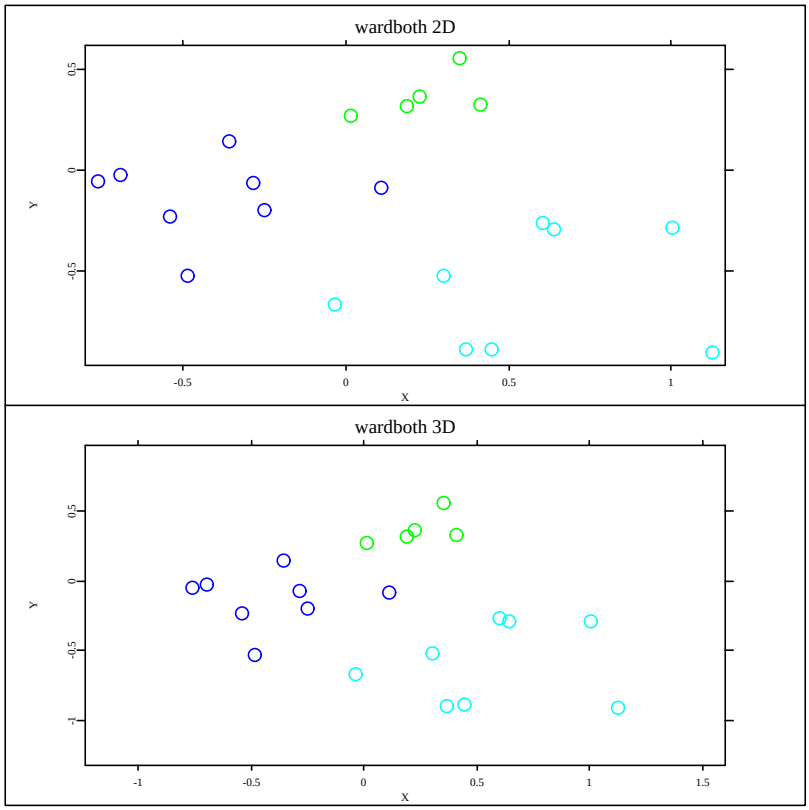


Figure 2.10: Correspondence analysis scores of both rows and columns.

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 `x` is an $n \times p$ matrix of n row points to be clustered, `k` is the number of clusters, `w` is a $p \times 1$ matrix of weights of column points, `m` is a $n \times 1$ matrix

of weights (masses) of row points, and **sv** is scalar seed value for random numbers.

The outputs of this quantlet `divisive` are: `cd.p` is the partition of n points of `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.

2.3 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.

2.3.1 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 contingency 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.

$$V_K = \sum_{k=1}^{K} \sum_{i=1}^{n} \delta_{ik} m_i d^2 \left(x_i - \overline{x}_k\right) \tag{2.13}$$

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

$$x_{kj} = \frac{1}{n_k} \sum_{i=1}^{I} \delta_{ik} m_i x_{ij} \tag{2.14}$$

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 $1, 2, ..., K$), `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
```

2.3.2 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

$$q_{jj} = \frac{1}{\overline{s}_j^2} \tag{2.15}$$

can be used in the squared weighted Euclidean distance, where $\overline{s}_j$ is the pooled standard deviation of the variable j:

$$\overline{s}_j^2 = \frac{1}{M} \sum_{k=1}^{K} \sum_{i=1}^{n} \delta_{ik} m_i \left(x_{ij} - \overline{x}_{kj}\right)^2 \tag{2.16}$$

The indicator δ_{ik} is defined in the usual way. For simplicity, use $M = \sum_{i=1}^{K} m_i$, $i = 1, 2, ..., n$, 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).

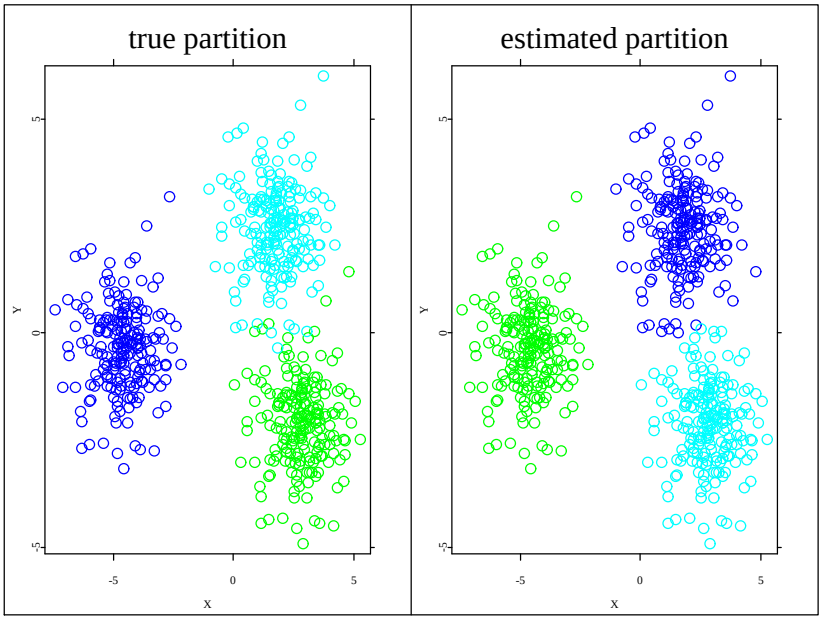


Figure 2.11: Start and final partition with adaptive clustering.

2.3.3 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 tech-

niques 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:

$$J_m(U, P : X) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} d^2(x_k, v_i), \tag{2.17}$$

where $X = (x_1, x_2, ..., x_n)$ is n data sample vectors, U is a partition of X in c part, $P = (v_1, v_2, ..., v_c)$ 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 15×2 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
```

¿From the Figure 2.12, 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.

2.3.4 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, $1 \leq m < \infty$;

$$J_m(U, P : X) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m d^2(x_k, v_i), \tag{2.18}$$

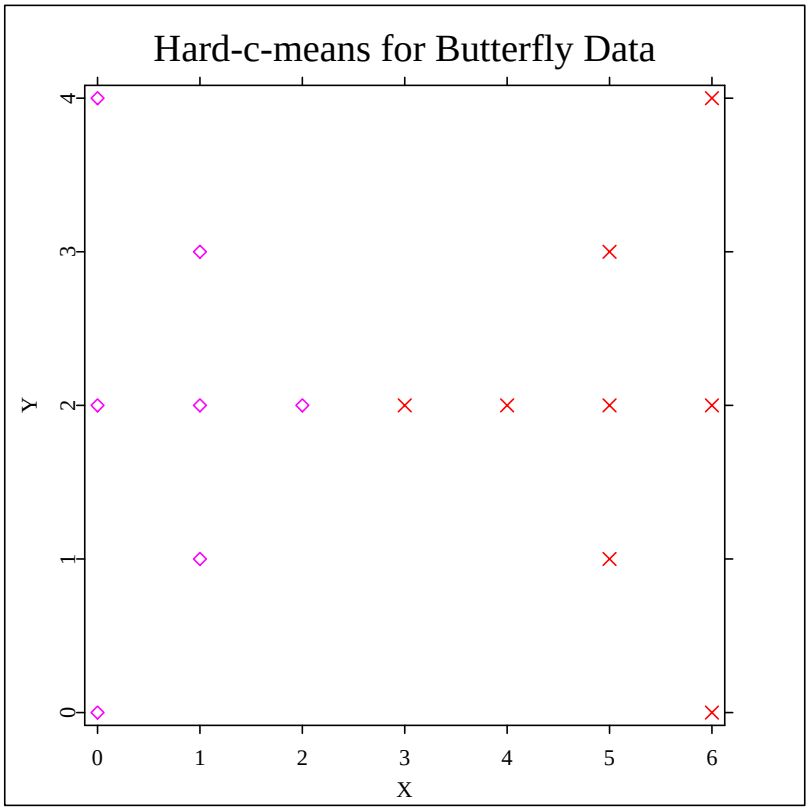


Figure 2.12: Hard C-means for Butterfly Data.

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

$$d^2(x_k, v_i) = \|x_k - v_i\|_A = \sqrt{(x_k - v_i)^T A (x_k - v_i)}, \tag{2.19}$$

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:

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

$\sum_{i=1}^{c} u_{ik} = 1, \text{for } 1 \leq k \leq n.$

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 $X = \{x_1, x_2, ..., x_n\}$, 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 $t = 1, 2, ..., T$

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

   $$v_{i,t} = \frac{\sum_{k=1}^{n} u_{ik,t-1}^{m} x_k}{\sum_{k=1}^{n} u_{ik,t-1}^{m}} \tag{2.20}$$

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

   $$u_{ik,t} = \sum_{j=1}^{c} \left(\frac{d_{ik,t}}{d_{jk,t}} \right)^{\frac{2}{m-1}}, \text{ if } \Upsilon_{\mathrm{k,t}} = 0 \tag{2.21}$$

   Choose $a_{ik,t} = 1/\#(\Upsilon_{k,t}), \forall i \in \Upsilon; \#(.)$ 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'|| <= tol_crit .or. iter = Max_iter
```

The syntax of this algorithm in XploRe is

```
fcm=xcfcme(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 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 2.13.

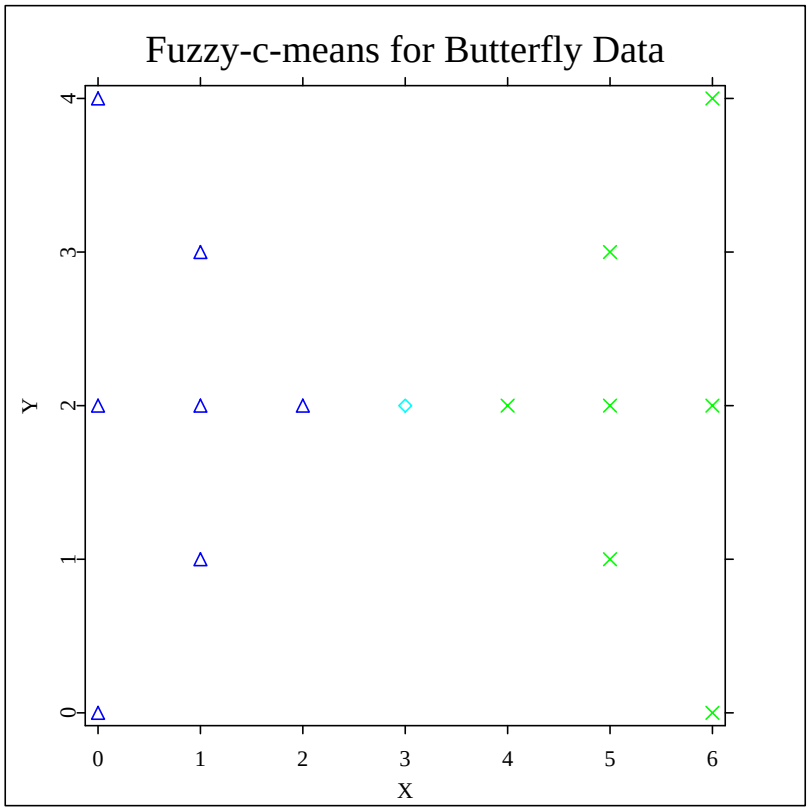


Figure 2.13: Fuzzy C-means for Butterfly Data

By using `m` = `1.25` and α = `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 2.14, 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 2.15 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.

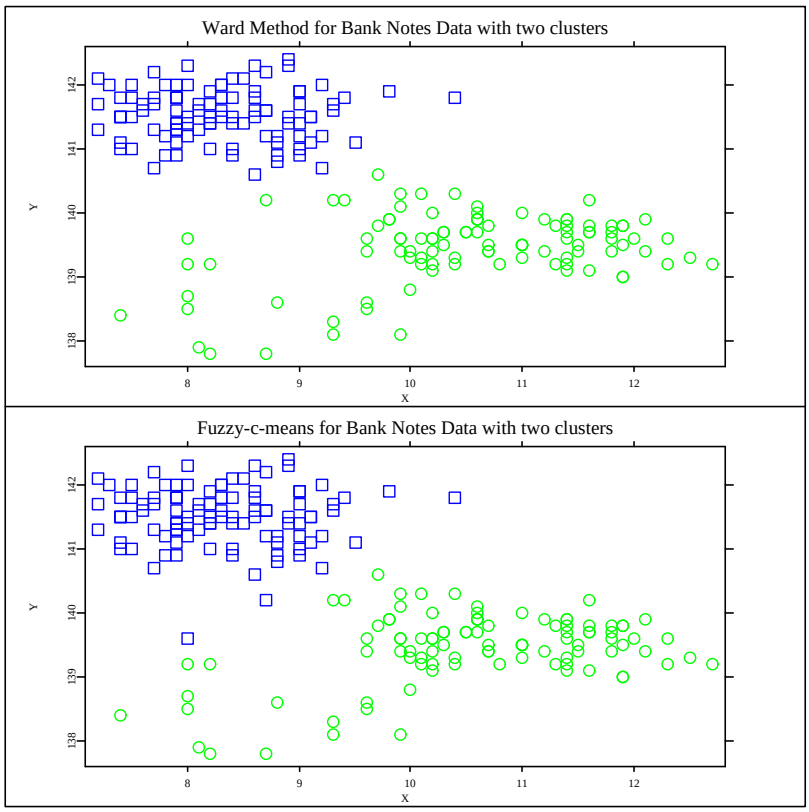


Figure 2.14: Ward method vs fuzzy C-means for Swiss banknotes data (X4 vs X6) with two clusters

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

Table 2.3: 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 2.4: Contingency table between Ward method vs fuzzy C-means Method with three clusters.

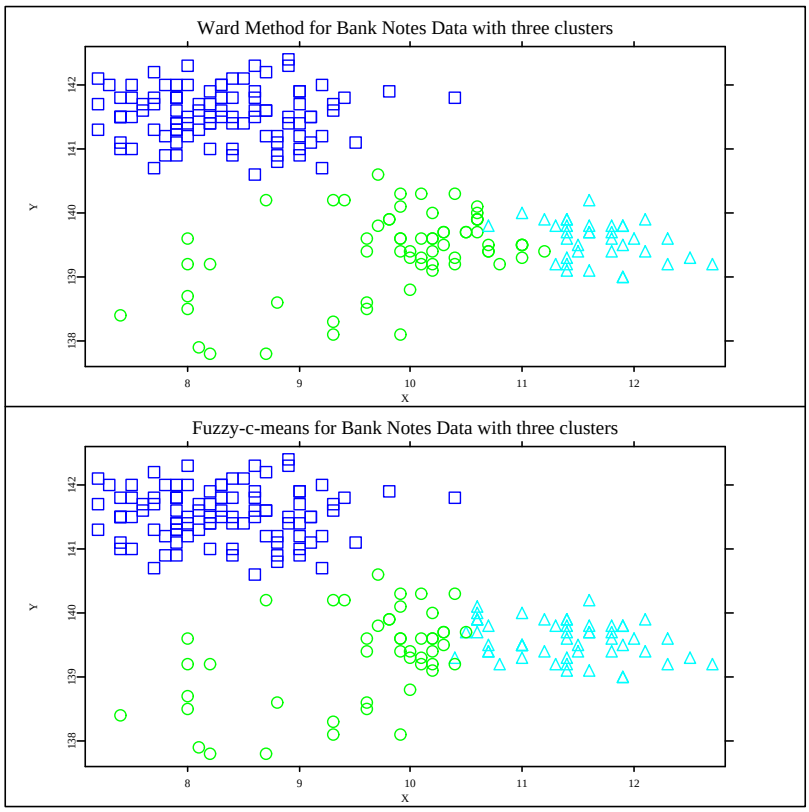


Figure 2.15: Ward Method vs fuzzy C-means for Swiss banknotes data (X4 vs X6) with three clusters

3

Classification and Regression Trees

3.1 Introduction

Regression trees are regression function estimators that are constant in rectangles. The rectangles need not have equal size, as in the case of the (standard) histogram estimators. Regression trees have the special property that they are representable as binary trees. This makes the graphical presentation of estimates possible, even in the case of many regression variables. Indeed, the regression tree is especially useful in the multidimensional cases. Furthermore, the regression tree has the advantage that it works also when the regression variables are a mixture of categorical variables and continuous variables. The response variable is assumed to be a continuous variable. Regression tree is well suited for the random design as well as for the fixed design. The theory of regression trees was developed by Breiman et al. (1984).

CART methodology consists of three parts. First, we grow a regression tree which overfits the data. Secondly we prune from the overfitting tree a sequence of subtrees and lastly we try to select from the sequence of subtrees a subtree which estimates the true regression function as best as possible.

3.2 Steps in CART

3.2.1 Growing the Tree

```
cs = cartsplit(x, y, type{, opt})
```
grows the tree

```
opt = cartsplitopt(s1{, s2{, s3,...}})
```
sets the parameters for growing the tree

Growing the tree proceeds sequentially. As a first step we take the regression estimator to be just a constant over the sample space. The constant in question is the mean value of the response variable. Thus, when the observed values of the response variable are $Y_1, \ldots Y_n$, the regression estimator is given by

$$\hat{f}(x) = \left(\frac{1}{n} \sum_{i=1}^{n} Y_i \right) I_R(x)$$

where R is the sample space and I_R is the indicator function of R. We assume that the sample space R, that is, the space of the values of the regression variables, is a rectangle.

Secondly the sample space is divided into two parts. Some regression variable X_j is chosen, and if X_j is a continuous random variable, then some real number a is chosen, and we define

$$R_1 = \{x \in R : x_j \leq a\}, \;\; R_2 = \{x \in R : x_j > a\}.$$

If X_j is categorical random variable with values $A_1, \ldots, A_q$, then some subset $I \subset \{A_1, \ldots, A_q\}$ is chosen, and we define

$$R_1 = \{x \in R : x_j \in I\}, \;\; R_2 = \{x \in R : x_j \in \{A_1, \ldots, A_q\} \backslash I\}.$$

The regression estimator in the second step is

$$\hat{f}(x) = \left(\frac{1}{|I_1|} \sum_{I_1} Y_i \right) I_{R_1}(x) + \left(\frac{1}{|I_2|} \sum_{I_2} Y_i \right) I_{R_2}(x)$$

where $I_1 = \{i : X_i \in R_1\}$ and $|I_1|$ is the number of elements in I_1.

The splitting of R to R_1 and R_2 is chosen in such a way that the sum of squared residuals of the estimator $\hat{f}$ is minimized. The sum of squared residuals is defined as

$$\sum_{i=1}^{n} \left(Y_i - \hat{f}(X_i) \right)^2 .$$

Now we proceed to split R_1 and R_2 separately. Splitting is continued in this way until the number of observations in every rectangle is small or the sum of squared residuals is small. The rectangle R corresponds to the root node of the binary tree. The rectangle R_1 is the left child node and the rectangle R_2 is the right child node. The end result is a binary tree.

3.2.2 Pruning the Tree

`subcs = prune(cs, alfa)`
: prunes a subtree from the given tree, given the complexity parameter α

`subcs = prunetot(cs, lnum)`
: prunes a subtree from a tree, given desired number of leaves

`subcs = maketr(cs, node)`
: forms a subtree, cutting at a given node

`resu = prunecv(tr, alfaseq, x, y, type)`
: for a given tree, this quantlet calculates the MSE of squared residuals for the subtrees of a given tree

`seq = pruneseq(cs)`
: for a given tree, this quantlet gives the sequence of α values and numbers of leaves of the pruned subtrees

One might think that the optimal way of choosing a regression tree is to stop growing the tree before it gets too large. For example, one could stop growing when the sum of the mean squared residuals of the regression estimator does not decrease substantially anymore. However, it might happen that the decrease in the sum of the mean squared residuals is momentarily slow, but some further splits result again in considerable decrease in this sum.

Let us denote

$$R(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}\Big(Y_i - \hat{f}(X_i)\Big)^2$$

and for $0 \leq \alpha < \infty$,

$$R_\alpha(\hat{f}) = R(\hat{f}) + \alpha \text{Leaves}(\hat{f})$$

where $\text{Leaves}(\hat{f})$ is the number of the leaves of the regression tree $\hat{f}$, which could be viewed as the number of rectangulars on which $\hat{f}$ is constant. The criterion $R_\alpha(\hat{f})$ takes into account not only the sum of the squared residuals but it also penalizes with respect to the complexity of the tree, measured by number of leaves in the tree. The number α is like smoothing parameter in kernel estimation.

Let $\hat{f}$ be a large, overfitting tree and let $\hat{f}_\alpha$ be a subtree of $\hat{f}$ for which $R()$ is minimal. Note that $\hat{f}_0$ is $\hat{f}$, and when α is sufficiently large, $\hat{f}_\alpha$ is the constant estimator, that is, it consists of the root node only. When α increases from 0 to infinity, there are only finite number of values of α at which $\hat{f}_\alpha$ is different. Let us call those values $0 = \alpha_1 < \cdots < \alpha_k$. The number k is less or equal to the number of leaves in $\hat{f}$. For $\alpha_k \leq \alpha < \alpha_{k+1}$, $\hat{f}_\alpha$ is the smallest subtree of $\hat{f}$ minimizing $R_\alpha()$. Now the sequence $\hat{f}_{\alpha_1}, \ldots, \hat{f}_{\alpha_k}$ forms a decreasing sequence of trees, $\hat{f}_{\alpha_1}$ is the original tree $\hat{f}$, and $\hat{f}_{\alpha_k}$ consists of the root node only.

3.2.3 Selecting the Final Tree

```
cross = cartcv(x, y, type, opt, wv)
```
cross-validation is done by this quantlet, which calculates the sequence of α values, number of leaves of the corresponding pruned subtrees, estimates of the expected values of the mean squared residuals and their standard errors

```
ln = leafnum(cs, node)
```
gives the number of leaves for a given tree

```
res = ssr(cs, node)
```
calculates the sum of squared residuals

```
enn = pred (tr, x, type)
```
calculates the prediction of the regression tree for a certain point x

```
mssr = prederr(tr, x, y, type)
```
calculates the sum of prediction errors for given tree and number of x and y values

Now we have to choose the best tree from the sequence $\hat{f}_{\alpha_1}, \ldots, \hat{f}_{\alpha_k}$. In other words, we have to choose the smoothing parameter α. We will try to estimate the expectation of the mean of squared residuals $R(\hat{f}_{\alpha_i})$, and then choose the regression estimate for which this estimate is minimal. This can be done by way of cross validation.

For example, in the ten fold cross validation we take 90% of the sample, grow the tree using this part of the sample, prune a sequence of subtrees and calculate the mean of squared residuals for every subtree in the sequence using the rest 10% of the sample as a test set. This is repeated 10 times, every time using different part of the sample as an estimation set and as a test set.

There is a problem that because we have used every time different data to grow and prune, we get every time different α-sequences. The approach proposed by Breiman, Friedman, Olshen, and Stone (1984, Section 8. 5. 2, page 234) is to first grow and prune using all of the data, which gives us the sequence $\{\alpha_i\}$, then form a new sequence $\beta_i = \sqrt{\alpha_i \alpha_{i+1}}$. The number β_i is the geometric mean of α_i and α_{i+1}. When pruning trees grown with 90% of

the sample, we choose subtrees which minimize $R_{\beta_i}()$.

Finally, the estimate for the expectation of $R(\hat{f}_{\alpha_i})$ is the mean of $R(\hat{f}^v_{\beta_i})$. Mean is over 10 cross validation estimates $\hat{f}^v_{\beta_i}$, $v = 1, \ldots, 10$. In practice, the estimates for the expectation of $R()$ do not have clear minimum, and it is reasonable to choose the smallest tree such that the estimate for the expectation of $R()$ is reasonably close to the minimum.

3.2.4 Plotting the result of CART

`plotcarttree(carttree {, outval})`
plots the CART tree

`dispcarttree(ctdisp, xn, yn, carttree {, outval})`
plots the CART tree in an user given display `ctdisp`

`{tree, treelabel} = grcarttree(carttree,{, outval})`
generates two graphical objects which contain the plot of the CART tree and the labels in the CART tree

`plotcart2(x, tree, {, xname})`
plots the cuts of CART in a two-dimensional projection of the dataset `x`

`dispcart2(cartdisp, xn, yn, x, carttree, ix, iy, depth, ssr)`

plots the cuts of CART in a twodimensional projection of the dataset `x` in an user given display `cartdisp`

`cut = grcart2(x, carttree, ix, iy, depth, ssr)`
generate the cuts which contains a two-dimensional projection of the data `x`

For visualizing the results are two methods provided:

1. plotting the tree via `plotcarttree`, `dispcarttree` or `grcarttree`
2. `plotcart2`, `dispcart2` or `grcart2` show how CART tesselates a two-dimensional projection of the data

As an output of `cartsplit` we receive the CART tree (`carttree`). The first method plots the CART tree. Depending on the value given in `outval`

we get as labels at the nodes in the tree the splitting rules (default), the numbers of observations in a node (`"nelem"`), the mean value of the observations in a node (`"mean"`) or the sum of squared residuals (`"ssr"`) in a node.

To get an overview how CART tesselates the space we can plot the tesselations in two dimensional projections of the data. The quantlet `plotcart2` allows the user to interactively change the projections. Also interactively we can choose if we want to see all cuts to a specified level (see `depth`) or all cuts where the sum of squared residuals (see `ssr`) is above a specified limit.

Note: if you view the cuts at the projection then be aware that cuts could appear in the tree which ar not visible. Thus the plot has to be interpreted with care ! Additionally the plot can only be applied to projections of continuous variables.

3.3 Examples

3.3.1 Simulated Example

Let us generate the observations

$$Y_i = f(X_i) + \epsilon_i, \;\; i = 1, \ldots, 200$$

where

$$f(x_1, x_2) = 100\ I(0 \leq x_1 \leq 0.5, 0.5 < x_2 \leq 1) + 120\ I(0.5 < x_1 \leq 1, 0.5 < x_2 \leq 1),$$

X_i are independently uniformly distributed on $[0,1]$x$[0,1]$, and ϵ_i are independently standardly normally distributed.

The Figure 3.1 shows simulated from the function f. The quantlet for generating the observations is

```
proc(y)=tuto(seed,n)
  randomize(seed)
  xdat=uniform(n,2)
  index=(xdat[,2]<=0.5)+(xdat[,2]>0.5).*(xdat[,1]<=0.5)*2
  layout=3*(index==1)+4.*(index==0)+5.*(index==2)
  ydat=100.*(index==2)+120.*(index==0)
  y=list(xdat,ydat,layout)
endp

library("xclust")
```

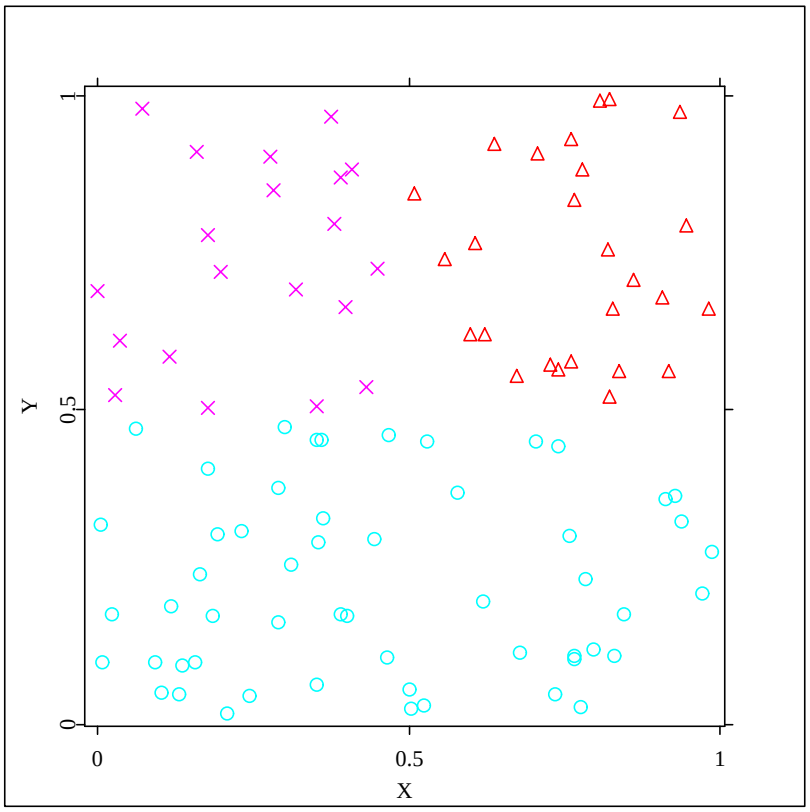


Figure 3.1: Plot of 100 simulated data from function $f(x_1, x_2)$. The datapoints in the upper left (marked with filled crosses) are in the area of $f(x_1, x_2) = 100$, the datapoints in the upper right (marked with triangles) are in the area of $f(x_1, x_2) = 120$ and the datapoints in the lower part (marked with circles) are in the area of $f(x_1, x_2) = 0$.

```
d=createdisplay(1,1)
data=tuto(1,100)
x=data.xdat
setmaskp(x, data.layout, data.layout, 8)
show(d,1,1,x)
```

cart01.xpl

Let us grow such a tree that the number of observations in a leaf nodes is less or equal to 5 (`mincut`), the deviation in a leaf node is larger or equal 0 (`mindev`) and cut will be only done if the number of the resulting nodes is larger as 1 (`minsize`). The type of variable is continuous.

```
library("xclust")
data=tuto(1,100)
type=#(1,1)
opt=cartsplitopt("minsize",1,"mindev",0,"mincut",5)
tr=cartsplit(data.xdat,data.ydat,type,opt)
totleaves=leafnum(tr,1)
totleaves
plotcarttree(tr)
```

cart02.xpl

The Figure 3.2 shows the regression tree `tr` with 41 leaves. From this figure, we prefer to choose the tree with 3 leaves because it is easier to see that in general it has three groups.

Let us choose the tree with 3 leaves with the following command.

```
trfin=prunetot(tr,3)
plotcarttree(trfin)
```

cart03.xpl

The Figure 3.3 shows final tree for simulated data.

3.3.2 Boston Housing Data

The Boston housing data was collected by Harrison and Rubinfeld (1978). The following variables are in the data:

1) crime rate

2) percent of land zoned for large lots

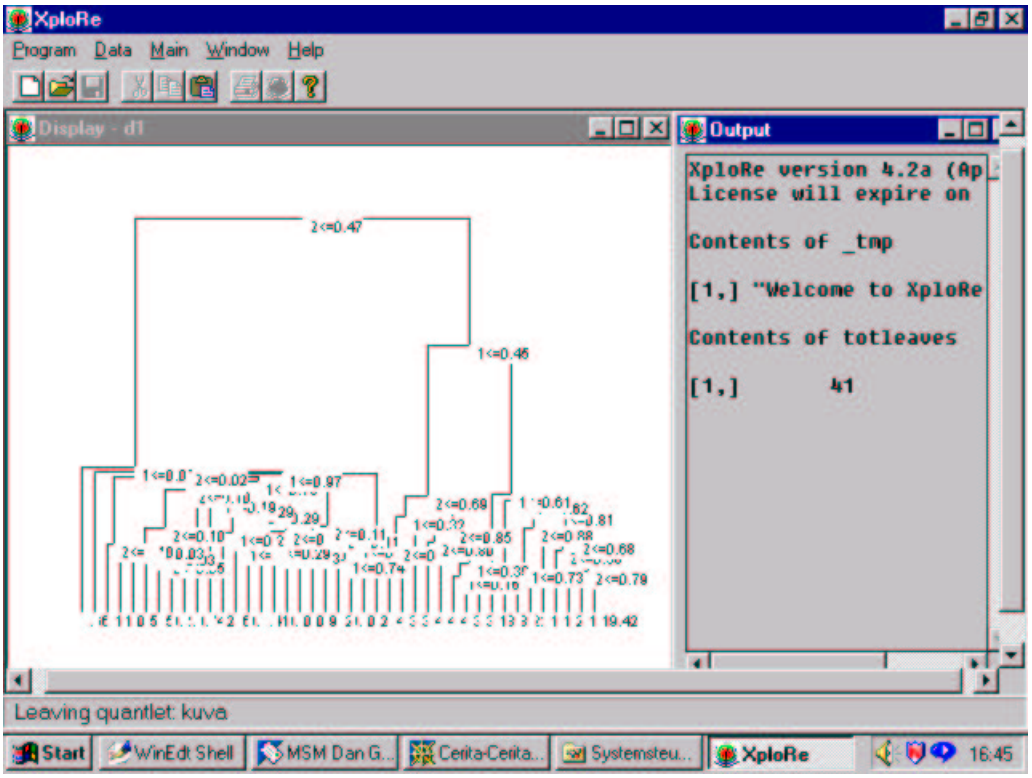


Figure 3.2: Initial regression tree for 100 simulated data from function $f(x_1, x_2)$ (left). The total number of leaves (41) is shown at the right.

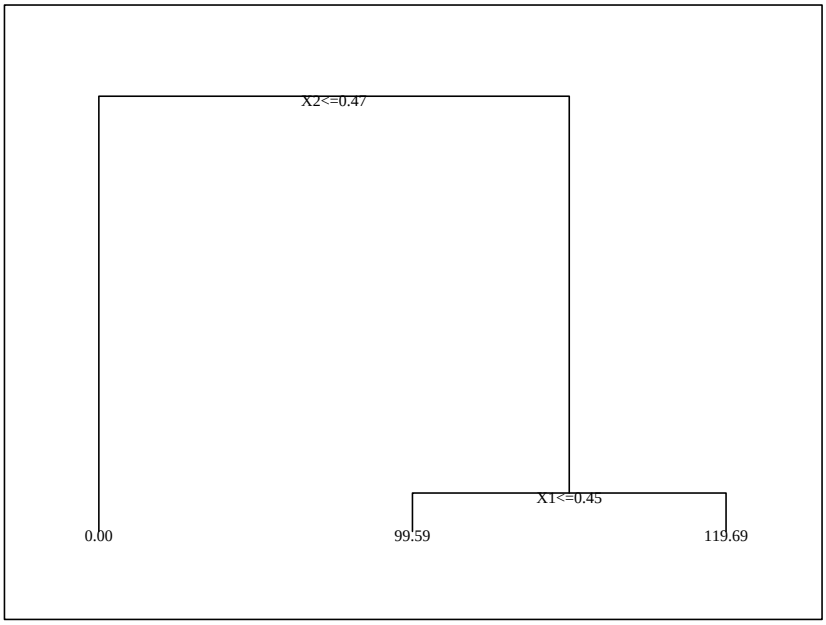


Figure 3.3: Final regression tree for 100 simulated data from function $f(x_1, x_2)$ after pruning. The final tree consists of three leaves which separate the x_1, x_2-plane into three parts.

3) percent non retail business

4) Charles river indicator, 1 if on Charles river, 0 otherwise

5) nitrogen oxide concentration

6) average number of rooms

7) percent built before 1980

8) weighted distance to employment centers

9) accessibility to radial highways

10) tax rate

11) pupil–teacher ratio

12) percent black

13) percent lower status

14) median value of owner–occupied homes in thousands of dollars.

The variable 14 is the response variable. The variables $1-13$ are predictor variables. The 4th and 9th are categorical variables, the other are continuous. There are 506 observations. Let us generate such a tree that the number of observations in leaf nodes is less or equal to 8.

```
library("xclust")
randomize(10)
boston=read("bostonh")
boston=paf(boston,uniform(rows(boston))<0.20)
yvar=boston[,14]
xvar=boston[,1:13]
type=matrix(13)
type[4]=0
type[9]=0
opt=cartsplitopt("minsize",1,"mindev",0,"mincut",8)
tr=cartsplit(xvar,yvar,type,opt)
totleaves=leafnum(tr,1)
totleaves
plotcarttree(tr)
```

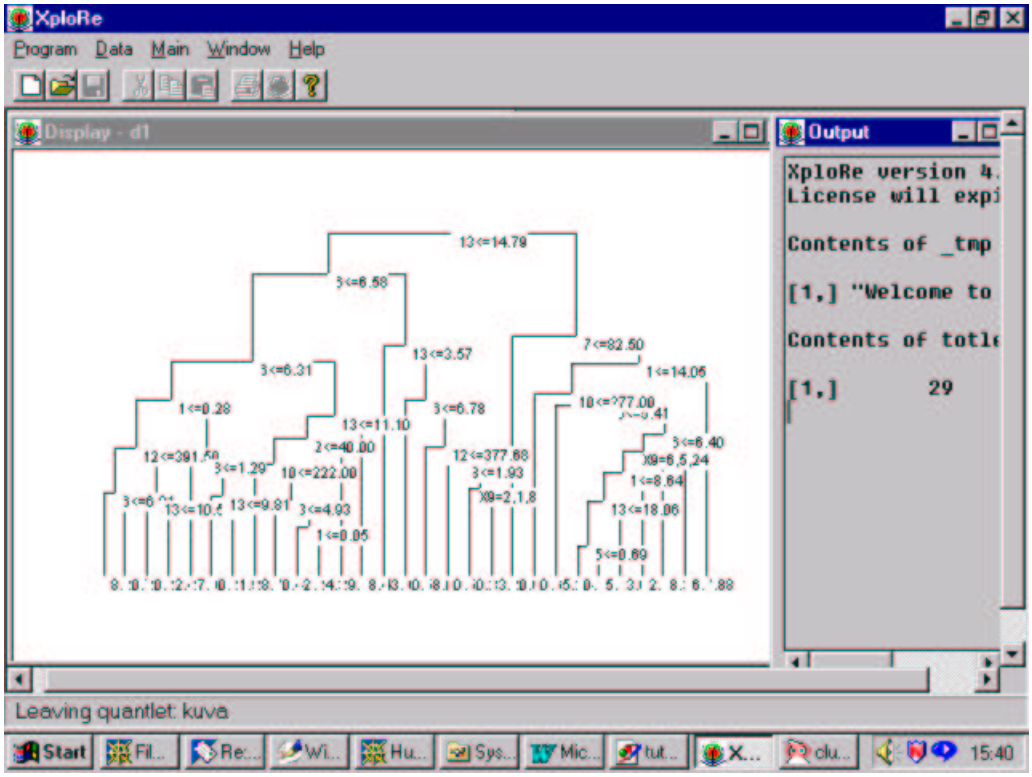


Figure 3.4: Initial regression tree for Boston Housing Data. The total number of leaves (29) is shown at the right.

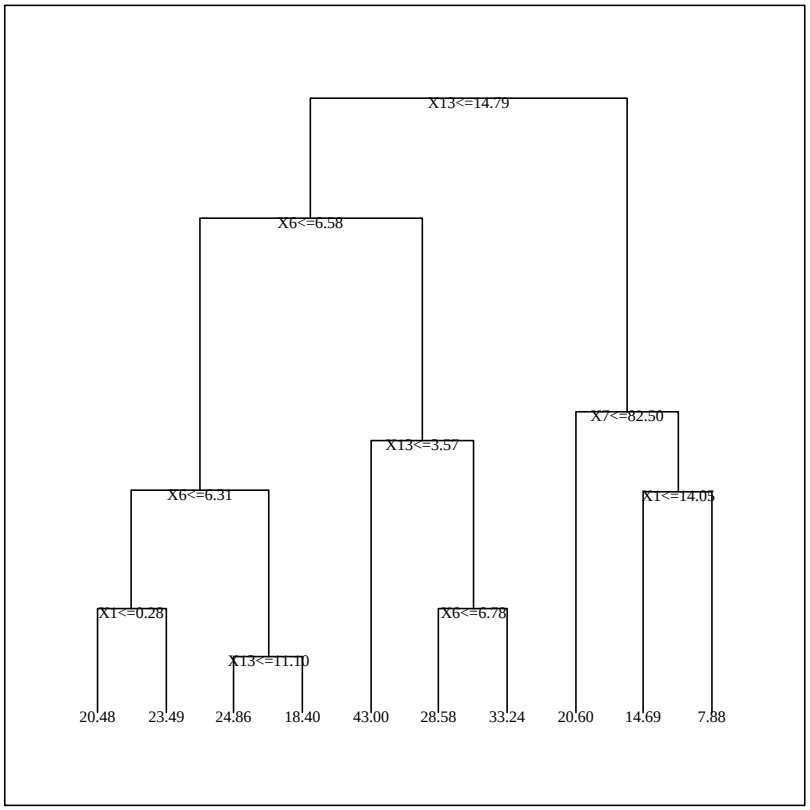


Figure 3.5: Sub-Tree consisisting of 10 leaves for 20% sample of the Boston Housing Data

cart04.xpl

We can observe that the tree `tr` with 29 leaves is large.

It is not so easy to read Figure 3.4. We can look at the optimal subtree consisting of 10 leaves by using these commands:

```
prtr=prunetot(tr,10)
plotcarttree(prtr)
```

cart05.xpl

The Figure 3.5 shows pruning tree for Boston Housing Data. Let us try to choose the optimal number of leaves with 10 fold cross validation.

```
cval=cartcv(xvar,yvar,type,opt,10)
```

```
res=cval.lnumber~cval.alfa~cval.cv~cval.cvstd
res=sort(res,1)
res=res[1:12,]
title=" no   alfa    cv   cvstd"
restxt=title|string("%3.0f %6.2f %6.2f %6.2f",
res[,1], res[,2], res[,3], res[,4])

dd=createdisplay(2,2)
show(dd, 1, 1, cval.lnumber~cval.alfa)
setgopt(dd, 1, 1, "title","number obs. vs alpha")
show(dd, 1, 2, cval.lnumber~cval.cv)
setgopt(dd, 1, 2, "title","number obs. vs cv")
show(dd, 2, 1, cval.lnumber~cval.cvstd)
setgopt(dd, 2, 1, "title","number obs. vs cvstd")
show(dd, 2, 2, restxt)
```

cart06.xpl

We get the result shown in Figure 3.6.

The first column gives the numbers of leaves in the sequence of pruned subtrees and the second column gives the sequence α_i, The estimates for the expectation of the mean of squared residuals, $ER(\hat{f}_{\alpha_i})$, are in the third column of the above matrix. The fourth column gives the estimates of the standard error of the corresponding estimators.

We can see that there is a clear minimum for the estimates for the expectation of the mean of squared residuals.

Therefore, it seems reasonable to choose as final estimate the tree with 7 leaves. Let us choose $\alpha = 0.9$ and form the corresponding tree.

```
fin=prune(tr,0.9)
plotcarttree(fin)
```

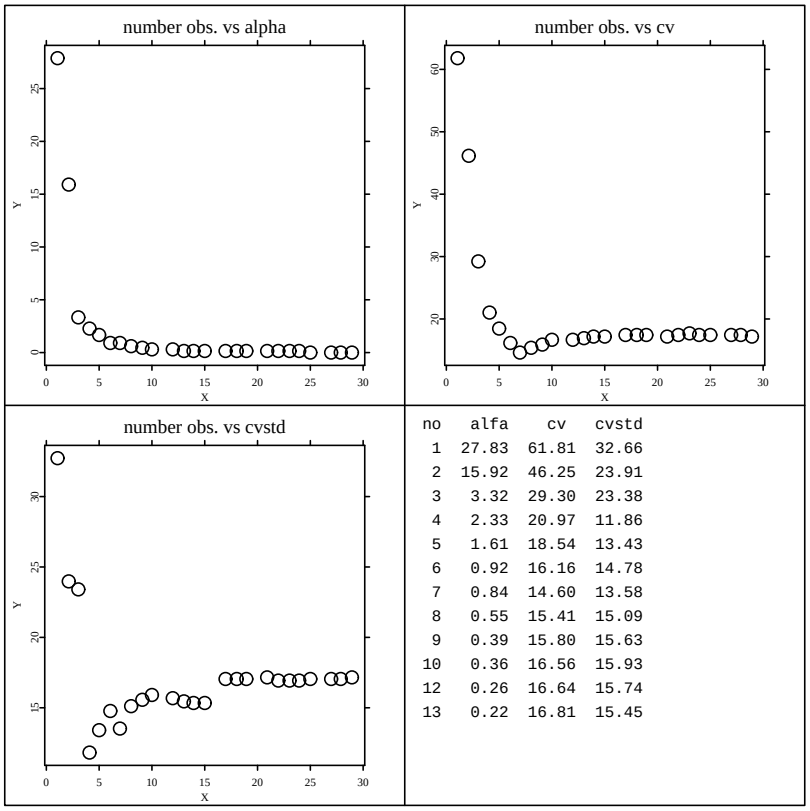
cart07.xpl

The final estimate is in the Figure 3.7. Let us look at the numbers of observations and the mean values in each node with command

```
plotcarttree(fin,"nelem")
plotcarttree(fin,"mean")
```

cart08.xpl

The result is displayed in the Figure 3.8 and Figure 3.9 respectively.



no	alfa	cv	cvstd
1	27.83	61.81	32.66
2	15.92	46.25	23.91
3	3.32	29.30	23.38
4	2.33	20.97	11.86
5	1.61	18.54	13.43
6	0.92	16.16	14.78
7	0.84	14.60	13.58
8	0.55	15.41	15.09
9	0.39	15.80	15.63
10	0.36	16.56	15.93
12	0.26	16.64	15.74
13	0.22	16.81	15.45

Figure 3.6: Cross Validation for 20% sample of Boston Housing Data.

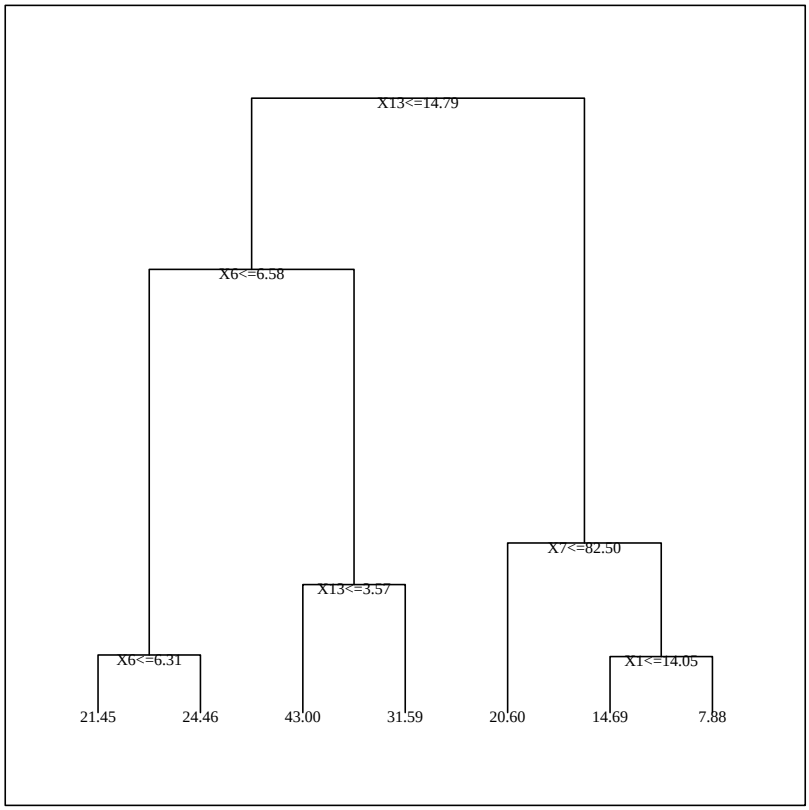


Figure 3.7: Final tree for 20% sample of Boston Housing Data

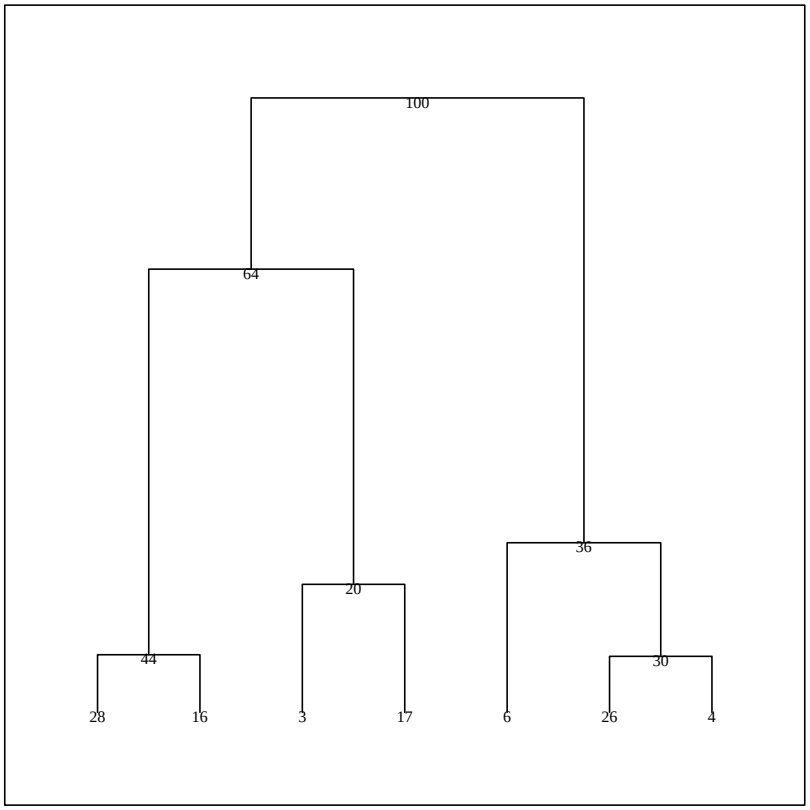


Figure 3.8: Final tree for 20% sample of Boston Housing Data with numbers of observations

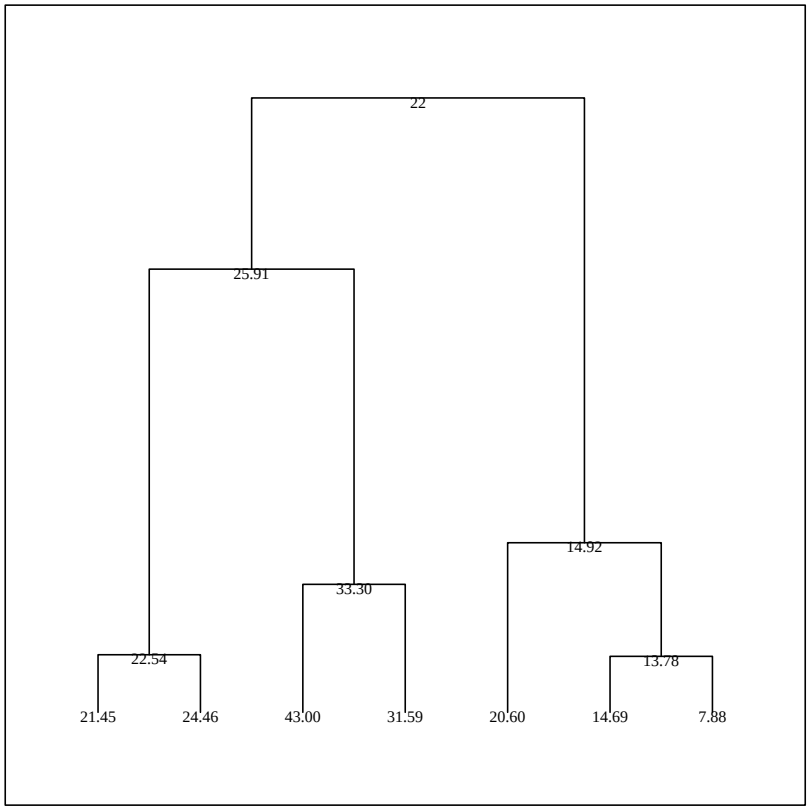


Figure 3.9: Final tree for 20% sample of Boston Housing Data with mean values

3.3.3 Density Estimation

```
regdat = dentoreg(dendat, binlkm)
```
transforms density data to regression data using variance stabilizing transform

Instead of writing separate procedures for the estimation of density functions, we will transform density data to the regression data and use regression tree to estimate density functions.

The basic idea is to divide the sample space into bins, calculate the number of observations in every bin, and consider these frequencies as a dependent regression variable. The independent regression variables are the midpoints of the bins. To be more precise, after we have calculated the frequencies of the bins Z_i, we will transform these to

$$Y_i = \sqrt{Z_i + 3/8}.$$

This was suggested by Anscombe (1948) and Donoho, Johnstone, Kerkyacharian, and Picard (1995, page 327).

Use the procedure first to make a histogram estimator for the density. This estimator will have a large number of equal size bins and so it will not be a good density estimator, but we will then combine some of these bins together in an optimal way using CART. The new regresson data will have dimension equal to the number of bins to the power of the number of variables. Given moment computing capability, probably 9 is the maximum number of variables for this method.

As an example we will analyze data which consists of 200 measurements on Swiss bank notes. These data are taken from Flury and Riedwyl (1988). One half of these bank notes are genuine, the other half are forged bank notes. The following variables are in the data.

1) length of the note (width)

2) height of the note (left)

3) height of the note (right)

4) distance of the inner frame to the lower border (bottom)

5) distance of the inner frame to the upper border (top)

6) length of the diagonal of the central picture (diagonal)

The macro `dentoreg` transforms density data to regression data. Let us choose 9 bins for every coordinate axes because we for the last 3 variables in the data.

```
; load library xclust and plot
library("xclust")
library("plot")
; set random seed
randomize(1)
; read swiss banknote data
dendat=read("bank2")
; select the last three variables
dendat=dendat[,4:6]
; choose 9 bins in each dimension
binlkm=9
; compute density estimate
regdat=dentoreg(dendat,binlkm)
; compute CART and tree
type=matrix(cols(dendat))
opt=cartsplitopt("minsize",50,"mindev",0,"mincut",1)
tr=cartsplit(regdat.ind,regdat.dep,type,opt)
; color datapoints after node the fall in
g=cartregr(tr, dendat, "node")
{gcat,gind}=groupcol(g, rows(g))
; compute cuts up level 2 for (X1,X2)
xdat=regdat.ind
gr12=grcart2(xdat, tr, 1, 2, 10, 0)
xdat12=dendat[,1|2]
setmaskp(xdat12, gind)
; compute cuts up level 2 for (X1,X3)
gr13=grcart2(xdat, tr, 1, 3, 10, 0)
xdat13=dendat[,1|3]
setmaskp(xdat13, gind)
; compute cuts up level 2 for (X2,X3)
gr23=grcart2(xdat, tr, 2, 3, 10, 0)
xdat23=dendat[,2|3]
setmaskp(xdat23, gind)
; compute tree and its labels
{tree, treelabel}=grcarttree(tr)
```

```
; show all projections and the tree in a display
setsize(640, 480)
d=createdisplay(2,2)
show(d, 1,1, xdat12, gr12)
setgopt(d,1,1, "xlabel", "top (X1)", "ylabel", "bottom (X2)")
show(d, 2,1, xdat13, gr13)
setgopt(d,2,1, "xlabel", "top (X1)", "ylabel", "diagonal (X3)")
show(d, 2,2, xdat23, gr23)
setgopt(d,2,2, "xlabel", "bottom (X2)", "ylabel", "diagonal (X3)")
axesoff()
show(d, 1,2, tree, treelabel)
axeson()
```

cart09.xpl

The result is shown in Figure 3.10. The upper left plot gives the cuts in the bottom-top plane, the lower left plot the cuts in the bottom-diagonal plane and the lower right plot the cuts in the top-diagonal plane. The CART tree is shown in the upper right window.

All splits are done in the bottom-diagonal plane. The lower right plot shows that CART algorithm just cuts from the main bulk of the data. Note the different colors in the left plots which shows that we have some cuts which are not visible in the top-bottom or top-diagonal projection.

Since we have choosen to stop splitting if the number of the observations is less than 75 (see the parameters `cartsplitopt` in `cart09`) we may choose a smaller number.

In `cart10` we have choosen a smaller number (20), do not color of the datapoints and omit the tree labels. The main result is here again that the CART algorithm cuts away the tails of the distribution and generates at least 4 different group of nodes.

```
; load library xclust and plot
library("xclust")
library("plot")
; set random seed
randomize(1)
; read swiss banknote data
dendat=read("bank2")
; select the last three variables
dendat=dendat[,4:6]
; choose 9 bins in each dimension
```

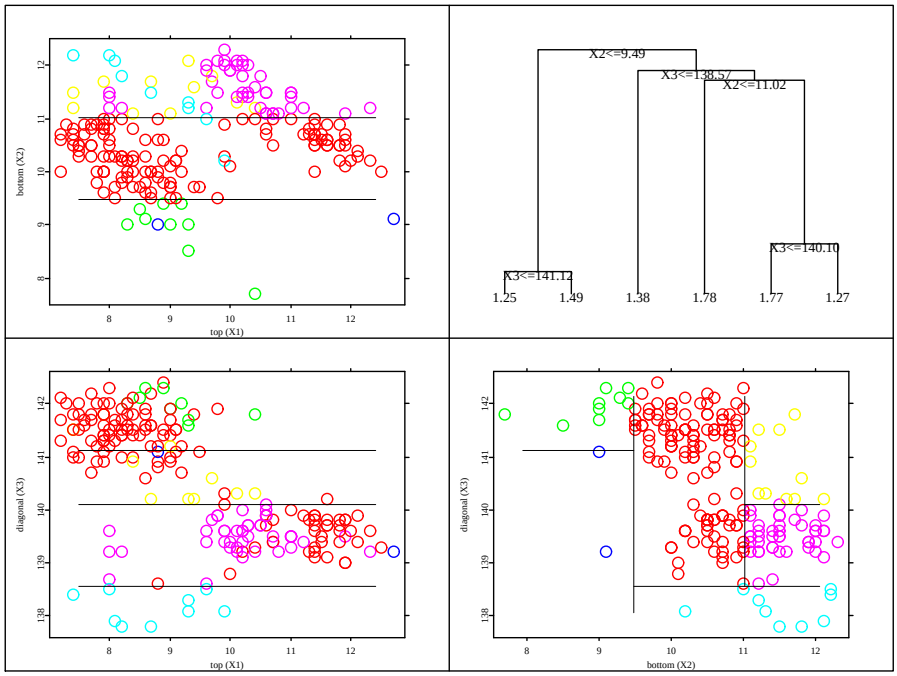


Figure 3.10: The upper left plot gives the cuts in the bottom-top plane, the lower left plot the cuts in the bottom-diagonal plane and the lower right plot the cuts in the top-diagonal plane. The CART tree is shown in the upper right window.

```
binlkm=9
; compute density estimate
regdat=dentoreg(dendat,binlkm)
; compute CART and tree
type=matrix(cols(dendat))
opt=cartsplitopt("minsize",20,"mindev",0,"mincut",1)
tr=cartsplit(regdat.ind,regdat.dep,type,opt)
; compute cuts up level 2 for (X1,X2)
xdat=regdat.ind
gr12=grcart2(xdat, tr, 1, 2, 10, 0)
xdat12=dendat[,1|2]
; compute cuts up level 2 for (X1,X3)
gr13=grcart2(xdat, tr, 1, 3, 10, 0)
xdat13=dendat[,1|3]
; compute cuts up level 2 for (X2,X3)
gr23=grcart2(xdat, tr, 2, 3, 10, 0)
xdat23=dendat[,2|3]
; compute tree and its labels
{tree, treelabel}=grcarttree(tr)
; show all projections and the tree in a display
setsize(640, 480)
d=createdisplay(2,2)
show(d, 1,1, xdat12, gr12)
setgopt(d,1,1, "xlabel", "top (X1)", "ylabel", "bottom (X2)")
show(d, 2,1, xdat13, gr13)
setgopt(d,2,1, "xlabel", "top (X1)", "ylabel", "diagonal (X3)")
show(d, 2,2, xdat23, gr23)
setgopt(d,2,2, "xlabel", "bottom (X2)", "ylabel", "diagonal (X3)")
show(d, 1,2, tree)
setgopt(d,1,2, "xlabel", " ", "ylabel", "log10(1+SSR)")
```

cart10.xpl

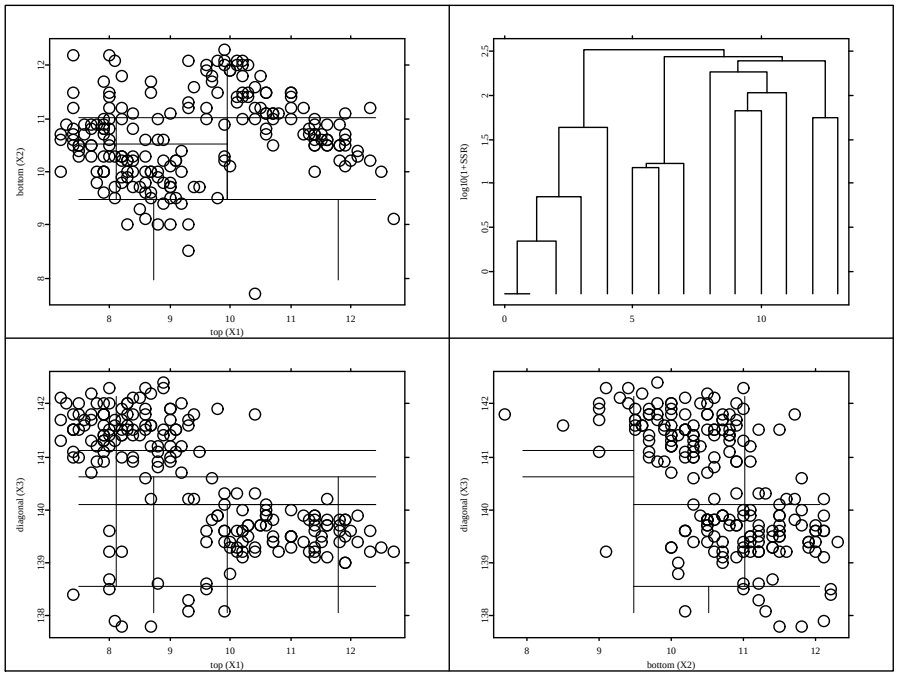


Figure 3.11: The upper left plot gives the cuts in the bottom-top plane, the lower left plot the cuts in the bottom-diagonal plane and the lower right plot the cuts in the top-diagonal plane. The CART tree is shown in the upper right window.

Bibliography

Anscombe, F. (1948). The transformation of Poisson, binomial and negative-binomial data. *Biometrika* **35**: 246–254.

Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algoritmss*, Plenum Press, New York.

Bezdek, J. C. and Pal, S. K. (1992). *Fuzzy Models for Pattern Recognition*, IEEE Press, New York.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. J. (1984). *Classification and Regression Trees.*, Chapman and Hall, New York.

Donoho, D. L., Johnstone, I. M., Kerkyacharian, G., and Picard, D. (1995). Wavelet shrinkage: asymptotia? (with discussion) *J. Roy. Statist. Soc. B* **57**: 301–369.

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., Klinke, S., and Turlach, B.A.(1995). *XploRe: An Interactive Statistical Computing Environment*, Springer Verlag, New York.

Härdle, W. and Simar, L. (1998). Applied Multivariate Statistical Analysis, *Lecture Notes*, Humboldt Universität zu Berlin.

Harrison, D. and Rubinfeld, D. L. (1978). Hedonic prices and the demand for clean air. *J. Envir. Econ. and Management* **5**: 81–102.

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.