Lecture notes for
STATG019
Selected Topics in Statistics: Cluster Analysis

Dr Christian Hennig
Department of Statistical Science, UCL

2015-2016

Preliminaries

- This book does NOT contain a complete set of notes for this course.
- These notes will be supplemented by handouts of examples of data analysis that will be given out and discussed in the lectures. The computer output in these examples has been obtained using the R software.
- If you require further explanation of a method or a proof that is not given or wish to study further examples, then do refer to a textbook on the subject. At the end, some references to texts are provided for optional background reading.
- You are strongly advised to try the exercises set each week – that is the way to learn the material!

Cluster analysis books and general overviews

- Aggarwal, C. C. and Reddy, C. K. (2014), Data Clustering: Algorithms and Applications, CRC Press.

  This is a cluster analysis handbook from a machine learning rather than a statistics perspective. It is astonishing how different this is from Hennig et al.'s Handbook of Cluster Analysis below. It has more on dimension reduction, big data issues and specific applications, but less on the statistical background of standard techniques used in the statistics community. Not really a good reference for the content of this course, rather an interesting addition.

- Everitt, B. S., Landau, S., Leese, M. and Stahl, D. (2011), Cluster Analysis (5th ed.), Wiley.

  Fairly comprehensive coverage of the most important approaches and concepts in cluster analysis. Make sure you pick the 5th edition or newer; this is much better than some older editions of this book.

- Hennig, C., Meila, M., Murtagh, F., and Rocci, R. (2015), Handbook of Cluster Analysis, Taylor & Francis.

  Chapters on various approaches were written by many experts in the field. Due to the large number of authors and writing styles, the book is not a systematic introduction but

rather covers many approaches and new developments. Chapter 1 gives an overview and Chapter 31 discusses general cluster analysis strategy.

- Jain, A. K. (2010), Data clustering: 50 years beyond K-means, Pattern Recognition Letters 31, 651-666.

  Interesting overview article by one of the most experienced researchers in the field.

- Kaufman, L. and Rousseeuw, P. (1990), Finding Groups in Data, Wiley.

  Authoritative reference for the techniques implemented in R's "cluster" package, but somewhat scarce regarding other approaches.

Further (somewhat outdated) books on cluster analysis are for example Gordon (1999); Jain and Dubes (1988).

Chapter 1

Introduction

Informally speaking, clustering means finding groups in data. Aristotle's classification of living things was one of the first known clusterings, a hierarchical clustering. The knowledge of biology has grown, yet the schematic organisation of all known species remains in the form of a (hierarchical) clustering. Doctors defining categories of tumours by their properties, astronomers grouping galaxies by their shapes, companies observing that users of their products group according to behaviour, archaeologists defining cultural periods from features of found artifacts, programs that label the pixels of an image by the object they belong to, other programs that segment a video stream into scenes, recommender systems that group products into categories, all are performing clustering. One can even say that classification and categorisation are very basic tasks for the development of human language and conceptual thinking.

There are also quite general tasks for which clustering is applied in many subject areas:

- exploratory data analysis looking for "interesting patterns" without prescribing any specific interpretation, potentially creating new research questions and hypotheses,
- information reduction and structuring of sets of entities from any subject area for simplification, more effective communication, or more effective access/action such as complexity reduction for further data analysis,
- investigating the correspondence of a clustering in specific data with other groupings or characteristics, either hypothesised or derived from other data.

Cluster analysis is a very lively research area. Cluster analysis research is done in several subject areas, most notably in statistics, machine learning and discrete mathematics. Inspired by specific application areas, researchers in social sciences, psychology, biology and management science have contributed to cluster analysis research as well. This means that there is quite a variety of approaches and methods in cluster analysis and little unification and consensus regarding what is best to do when. Although new methods for cluster analysis are introduced all the time, I think that research on cluster validation and comparison of different approaches is urgently required and more useful for the end user than devising ever more new methods for clustering.

There is no unique definition of what a cluster is. A general intuition is that clusters should be homogeneous, i.e., objects in the same cluster should be close to each other, and should be

separated from each other, i.e., the distance between objects from different clusters should be large. Often different clusters are seen as corresponding to high density areas with density gaps between different clusters.

Sometimes, though, within-cluster homogeneity and between-cluster separation are conflicting because objects that are very far from each other may be connected by high density areas so that putting them together in the same cluster means that the cluster is not very homogeneous.

Another possible cluster definition is that objects in the same cluster are generated by the same homogeneous probability distribution, such as a normal distribution, and a different cluster may be generated by, for example, a normal distribution with different parameters. There is also work in which clusters are defined by regression or time series models, but this is beyond the scope of this course, see, e.g., Hennig et al. (2015).

In some literature, cluster analysis is referred to as "unsupervised classification" as opposed to "supervised classification". In unsupervised classification, it is not known to which cluster the observations belong and this is to be estimated, whereas techniques for supervised classification (such as Linear Discriminant Analysis and the k-nearest neighbour classifier) are about deriving classification rules for new observations based on a training set of observations with known classes.

1.1 Some example datasets

1.1.1 Old Faithful Geyser

This dataset from Azzalini and Bowman (1990) consists observations from the Old Faithful geyser in Yellowstone National Park, which is one of the best studied geysers. It is about the connection between the duration of an eruption (in minutes) and the following waiting time for the next eruption, also in minutes. The interest here lies in classifying the different eruptions for understanding the geyser's behaviour and for predicting waiting times (which nowadays is done by classifying eruptions roughly into "short" and "long" because analysing the data including cluster analysis reveals that discovering this class structure is pretty much the best one can do for predicting the waiting times). Durations and waiting times are given for 299 eruptions in the dataset.

The dataset is available in the R-package `MASS` and can be accessed as follows:

```
library(MASS)
data(geyser)
# You can also load geyser.dat from Moodle and read it with
# geyser <- read.table("geyser.dat",header=TRUE)

# Here's how it looks like:
str(geyser)
#'data.frame': 299 obs. of  2 variables:
# $ waiting : num  80 71 57 80 75 77 60 86 77 56 ...
# $ duration: num  4.02 2.15 4 4 4 ...
```

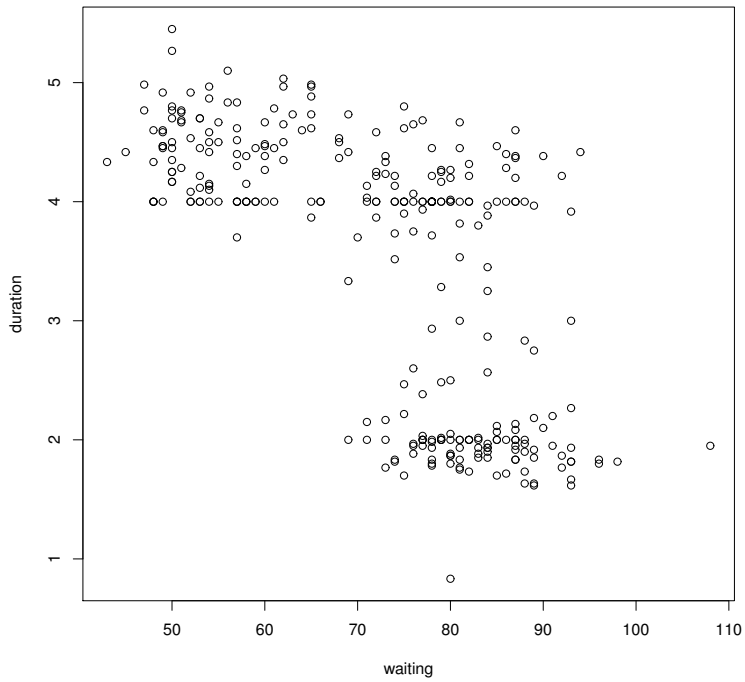

Figure 1.1: Old Faithful Geyser data

```
plot(geyser)
```

The plot is shown in Figure 1.1.

1.1.2 German Bundestag Election 2005

The Bundestag is the German parliament. The German Bundestag Election 2005 dataset is in the R-package `flexclust`. By pure coincidence, this dataset also has 299 observations, namely the German constituencies. It gives the election results (proportions between 0 and 1) in the constituencies for the five biggest parties, namely the social democratic SPD, the conservative Christian UNION, the green party GRUENE, the liberal party FDP and the left wing party LINKE. The interest here is in finding clusters of constituencies with similar voting behaviour. The dataset also contains some information about to which of the 16 German federal states the constituencies belong. This information is not used for clustering, but is rather of interest for interpreting clustering results. From this I produced another variable distinguishing between the federal states that before 1989 belonged to West Germany, the communist East Germany, or Berlin (which was divided between the two and had a special status). Here is some R-code that loads the data:

```
library(flexclust)
p05 <- bundestag(2005)
# You can also load bundestag.dat from Moodle and read it with
```

```
# bundestagdata <- read.table("bundestag.dat",header=TRUE)
# The variables in p05 are bundestag[,1:5];
# the variables state and ewb below are variables 6 and 7
# in bundestagdata.

str(p05)
# How the object looks like:
# num [1:299, 1:5] 0.391 0.362 0.363 0.376 0.415 ...
# - attr(*, "dimnames")=List of 2
#  ..$ : chr [1:299] "Flensburg - Schleswig" "Nordfriesland -
# Dithmarschen Nord" "Steinburg - Dithmarschen Sued"
# "Rendsburg-Eckernfoerde" ...
#  ..$ : chr [1:5] "SPD" "UNION" "GRUENE" "FDP" ...

# federal states:
state <- bundestag(2005, state=TRUE)

# ewb takes values "East", "West", "Berlin" as explained:
ewb <- rep("West",299)
ewb[state=="Berlin"] <- "Berlin"
ewb[state %in% c("Brandenburg","Mecklenburg-Vorpommern","Sachsen",
                 "Sachsen-Anhalt","Thueringen")] <- "East"

# Plotting
pairs(p05,xlim=c(0,0.6),ylim=c(0,0.6),cex=0.5)
# Note the xlim, ylim parameters. All values of all variables
# have the same meaning, so fixing the x and y value range for
# all variables simultaneously will show which parties are
# big and which are small.
# cex=0.5 makes plot symbols smaller.
```

The plot is shown in Figure 1.2.

1.1.3 Olive Oil

This data set from the `pdfCluster`-package of R represents eight chemical measurements on 572 different specimen of olive oil produced in various regions in Italy (northern Apulia, southern Apulia, Calabria, Sicily, inland Sardinia and coast Sardinia, eastern and western Liguria, Umbria). These regions can further be classified into three macro-areas: Centre-North, South, Sardinia. Like for the Bundestag data, these regions are not of direct interest for clustering, but rather for interpreting and evaluating the clustering results. The clustering aim is to give a chemical classification of the olive oils.

The help page of `oliveoil` in the `pdfCluster`-package mentions that the data are compositional and the values should ideally add up to 10000 for every olive oil. However, because of

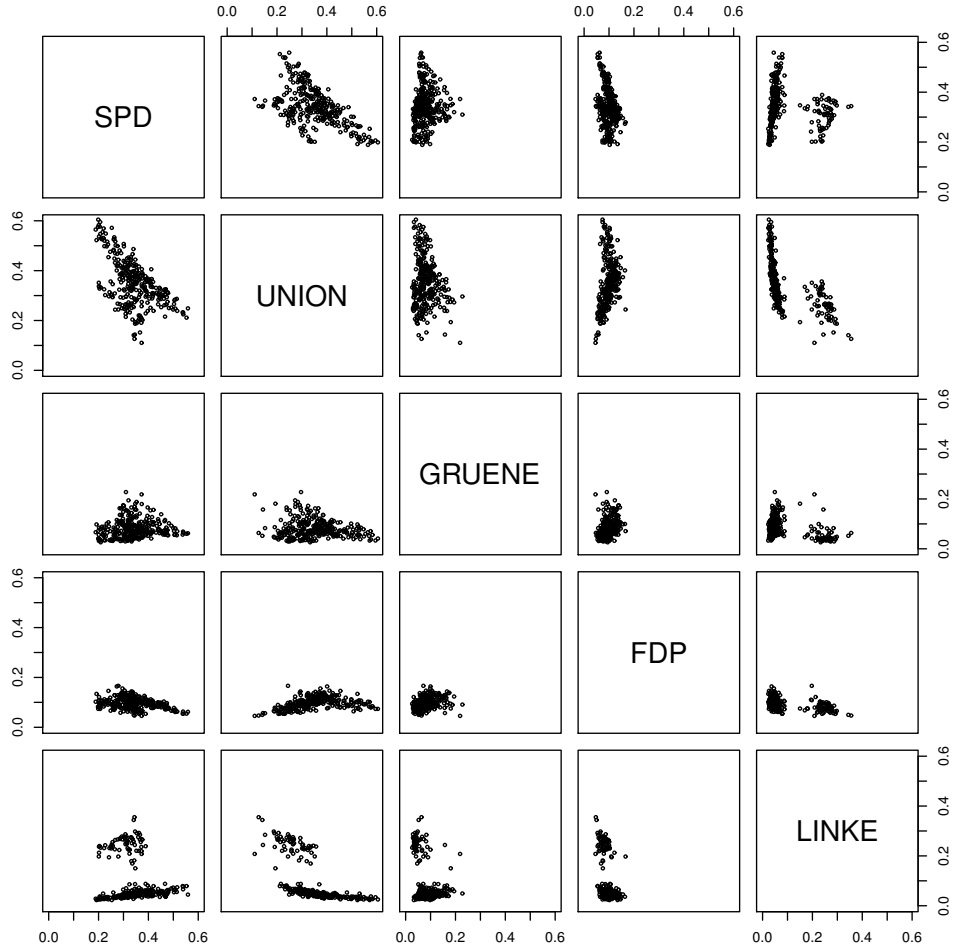

Figure 1.2: Bundestag data

measurement error this is not always the case. A transformation of the data is suggested there, but this doesn't seem to make much of a difference, so it is not applied here.

Actually, the regions and particularly the macro-areas are often used in the literature as the "true clusters" in order to evaluate the quality of clustering algorithms. On one hand such data with given "true clusters" are a good resource to compare different clustering algorithms. On the other hand, it is problematic to assume that the given classes, here the regions and macro-areas, are indeed the only "true clusters". If there is much chemical variation between olive oils within a region, there is nothing wrong with a clustering based on chemistry that differs from the regional information.

```
library(pdfCluster)
data(oliveoil)
# You can also load oliveoil.dat from Moodle and read it with
# oliveoil <- read.table("oliveoil.dat",header=TRUE)
```

```
# Look at object:
str(oliveoil)
# 'data.frame': 572 obs. of  10 variables:
#  $ macro.area : Factor w/ 3 levels "South","Sardinia",..: 1 1 1 1 1 1 1 1 1 1 ...
#  $ region     : Factor w/ 9 levels "Apulia.north",..: 1 1 1 1 1 1 1 1 1 1 ...
#  $ palmitic   : int  1075 1088 911 966 1051 911 922 1100 1082 1037 ...
#  $ palmitoleic: int  75 73 54 57 67 49 66 61 60 55 ...
#  $ stearic    : int  226 224 246 240 259 268 264 235 239 213 ...
#  $ oleic      : int  7823 7709 8113 7952 7771 7924 7990 7728 7745 7944 ...
#  $ linoleic   : int  672 781 549 619 672 678 618 734 709 633 ...
#  $ linolenic  : int  36 31 31 50 50 51 49 39 46 26 ...
#  $ arachidic  : int  60 61 63 78 80 70 56 64 83 52 ...
#  $ eicosenoic : int  29 29 29 35 46 44 29 35 33 30 ...

# The chemical variables:
olive <- oliveoil[,3:10]
# Plot:
pairs(olive,cex=0.3)
```

The plot is shown in Figure 1.3. Note that there are more sophisticated methods for visualisation of multidimensional datasets such as Principal Component Analysis, but they are not treated in this course.

1.1.4 Veronica Plant Species

This dataset is an example of a different data format that has to be plotted in different ways; the intuition about what "clustering" means here is different. The dataset is from Martinez-Ortega et al. (2004) and I put it on Moodle with the name `veronica.dat` (actually there is a version of it in the R-package `prabclus`, but this is ordered according to some clustering results, which is not how the data were collected in reality).

Veronica data has 207 objects, and these are individual Veronica plants. The aim of clustering these is to discover and delimit different species of such plants. The plants are characterised by 583 variables. These contain genetic information, which was obtained using AFLP-technology ("amplified fragment length polymorphism"). This detects the presence or absence of certain characteristics of DNA fragments (you may have a look at the Wikipedia page of amplified fragment length polymorphism for more detailed information). The variables can take the values 0 (absence) and 1 (presence) and refer to different fragments of the DNA. Species discovery and delimitation nowadays is done based on such genetic information.

```
veronica <- read.table("veronica.dat")
# This assumes that the dataset in in the directory in which R is run;
# otherwise you need to add the location where you have put the dataset.

str(veronica)
```

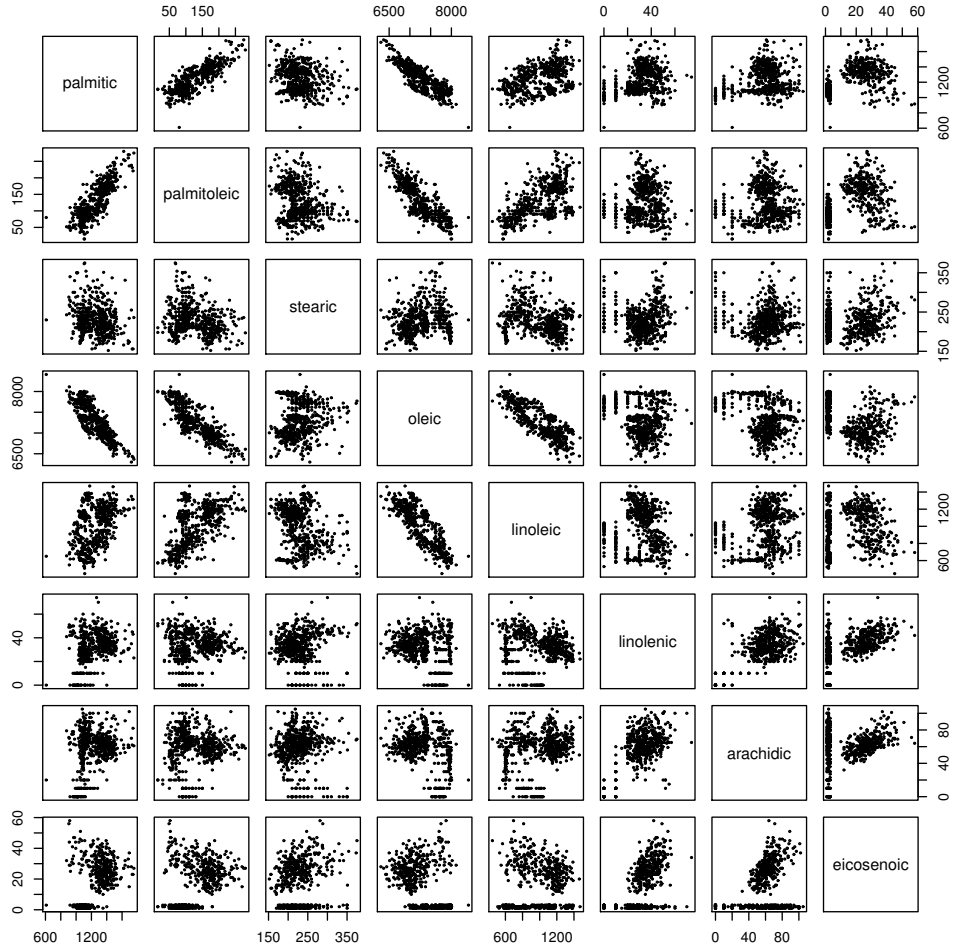

Figure 1.3: Olive data

```
#'data.frame': 207 obs. of  583 variables:
# $ V1  : int  0 0 0 1 0 0 0 0 0 1 ...
# $ V2  : int  0 0 0 0 0 1 0 0 0 0 ...
# $ V3  : int  1 1 0 1 0 0 0 0 0 0 ...
# $ V4  : int  0 0 0 0 0 0 0 0 0 0 ...
# (...)

# Some things can better be done on matrices:
veronicam <- as.matrix(veronica)

# A pairs plot isn't suitable for 0-1 data, and also not
# for 583 variables.
# A better plot is a heatmap, that plots value 1 in black
# and zero in white.
```

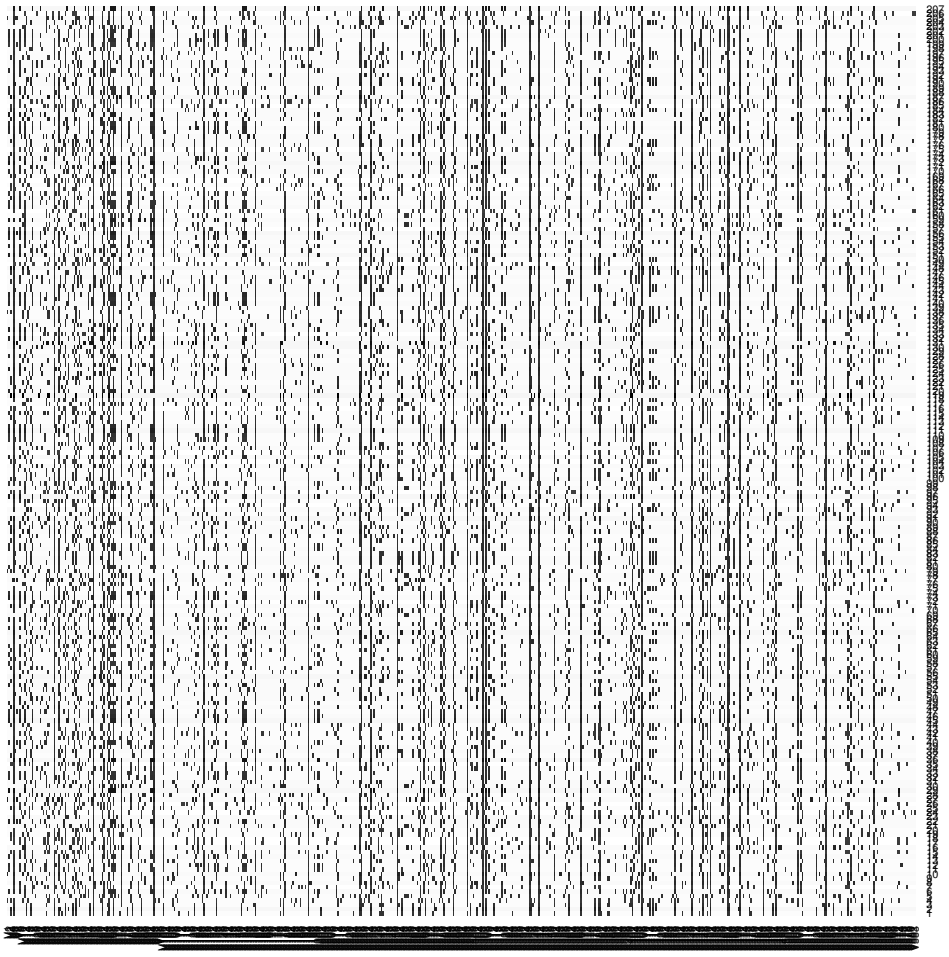

Figure 1.4: Veronica data. Rows are plants, columns are AFLP bands. Black is 1, white is 0.

```
heatmap(veronicam,Rowv=NA,Colv=NA,col=grey(seq(1,0,-0.01)))
# This is black for 1-entries
# rows are plants, columns are variables
```

The plot is shown in Figure 1.4.

1.1.5 Artificial Dataset 1

In order to demonstrate some issues, I also generated artificial datasets. The first dataset should be fairly easy to cluster. It is constructed to consist of three clusters. The first one comes from a bivariate Gaussian (normal) distribution, for the second and third I used so-called skew normal distributions as implemented in the R-package `sn`. The dataset is on Moodle and is called `clusterdata1.dat`.

```
# This is how I generated the dataset.
```

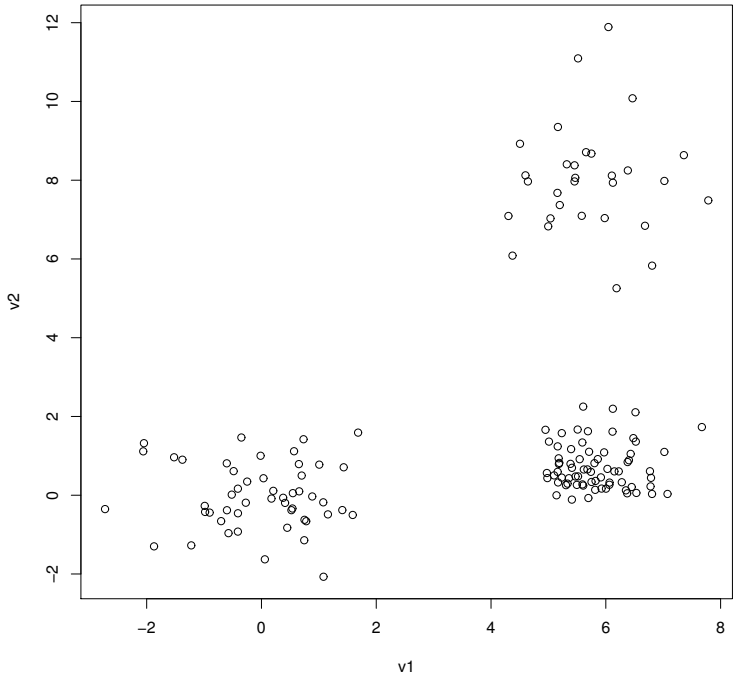

Figure 1.5: Artificial dataset 1.

```
# You actually don't need to do this.

library(sn)
set.seed(665544)
v1 <- c(rnorm(50,0,1), rsn(70,5,1,8), rnorm(30,6,1))
v2 <- c(rnorm(50,0,1), rsn(70,0,1,8), 8+rt(30,5))
clusterdata1 <- cbind(v1,v2)

# Here is how to load the dataset, again assuming that it is in
# R's search path.
clusterdata1 <- as.matrix(read.table("clusterdata1.dat"))
str(clusterdata1)
# num [1:150, 1:2] -0.982 0.378 0.547 1.433 0.75 ...
# - attr(*, "dimnames")=List of 2
#  ..$ : NULL
#  ..$ : chr [1:2] "v1" "v2"

plot(clusterdata1)
```

The plot is shown in Figure 1.5.

1.1.6 Artificial Dataset 2

The second artificial dataset illustrates a problem of some clustering methods, namely a situation in which different clusters have very different within-cluster variation. There are two small bivariate Gaussian clusters and one widespread bivariate uniform one. The dataset is on Moodle and is called `clusterdata2.dat`.

```
# This is how I generated the dataset.
# You actually don't need to do this.

set.seed(77665544)
x1 <- rnorm(20)
y1 <- rnorm(20)
x2 <- rnorm(20,mean=10)
y2 <- rnorm(20)
x3 <- runif(100,-20,30)
y3 <- runif(100,20,40)
clusterdata2 <- cbind(c(x1,x2,x3),c(y1,y2,y3))

# Here is how to load the dataset, again assuming that it's
# in R's search path.
clusterdata2 <- as.matrix(read.table("clusterdata2.dat"))
str(clusterdata2)
# num [1:140, 1:2] -1.694 0.494 -0.753 -0.16 -0.375 ...

plot(clusterdata2)
```

The plot is shown in Figure 1.6.

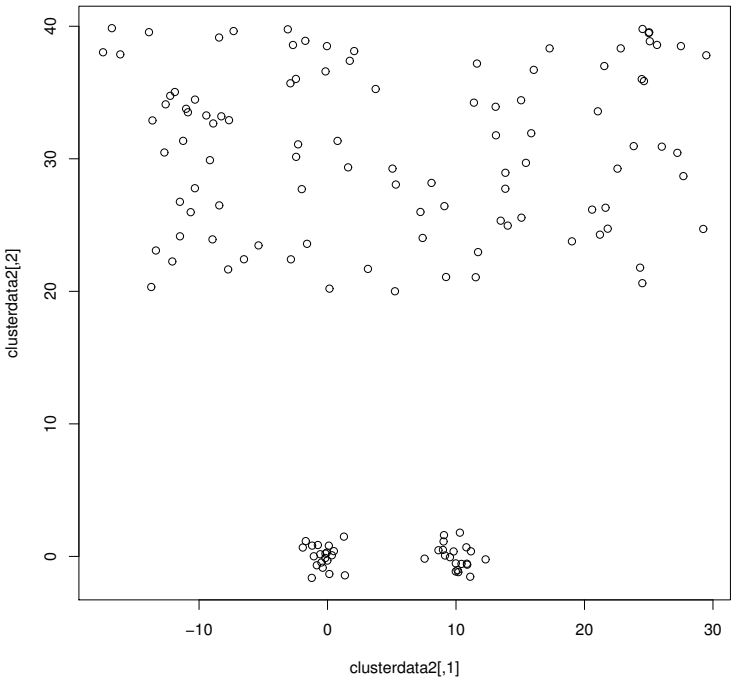

Figure 1.6: Artificial dataset 2.

Chapter 2

K-means, the most popular clustering method

K-means is the most widely used method for cluster analysis and also one of the oldest ones. It was first proposed by Steinhaus (1957). It is a method for multivariate data in $\mathbb{R}^p$ ($p \geq 1$), and is based on the least squares principle, just as least squares regression. The idea is that for finding K clusters in a dataset (K needs to be pre-specified) K "centroid points" are positioned in $\mathbb{R}^p$, and every observation is assigned to the closest centroid point, in such a way that the sum of all squared Euclidean distances of the observations to the centroids in minimised. In order to achieve this, the centroid points have to be the (multivariate) means of the observations assigned to them, i.e. the clusters of which they are the centroids, hence the name K-means. The idea is that because all observations are as close as possible to their respective centroid, the clusters are very homogeneous; all cluster members can be well represented by their centroid, which is of use in some applications such as data compression.

2.1 Basic definitions

Let $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\}$ contain n data points. The task of clustering is to group them into K disjoint subsets of $\mathcal{D}$, denoted by $C_1, C_2, \ldots, C_K$. $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$ is called a **clustering**.

Like many (but not all) clustering methods, K-means clustering produces a **partition** of $\mathcal{D}$, i.e., $C_1 \cup C_2 \cup \ldots \cup C_K = \mathcal{D}$ and $C_i \cap C_j = \emptyset$ for any two $C_i, C_j,\ i \neq j$ (every data point is in exactly one cluster).

For partitions, if a data point $\mathbf{x}_i$ belongs to cluster C_k, the **label** of $\mathbf{x}_i$ is k, or $c(i) = k$ for $i \in \mathbb{N}_n = \{1, \ldots, n\}$. This means that a partition is defined, equivalently to the set-based definition of a clustering above, by knowing the labels $c(1), \ldots, c(n)$ of all points.

Obviously not every such $\mathcal{C}$ qualifies as a "good" or "useful" clustering, and what is demanded of a "good" $\mathcal{C}$ will depend on the cluster analysis approach.

For the K-means method it is assumed that $\mathbf{x}_i \in \mathbb{R}^p$ for all $i \in \mathbb{N}_n$.

Definition 2.1 *For* $\mathbf{x} = (x_1, \ldots, x_p), \mathbf{y} = (y_1, \ldots, y_p) \in \mathbb{R}^p$, $d_{L2}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x}-\mathbf{y}\| = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2}$ *is called the* **Euclidean distance** *(or L2-distance) between* $\mathbf{x}$ *and* $\mathbf{y}$.

Definition 2.2 *The K-means clustering of $\mathcal{D}$ is defined by choosing $\hat{\mathbf{m}}_1{}^{Km}, \ldots, \hat{\mathbf{m}}_K{}^{Km}$ and $c^{Km}(1), \ldots, c^{Km}(K)$ in such a way that they minimise*

$$S(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{m}_{c(i)}\|^2.$$

Obviously, for given cluster centroids $\mathbf{m}_1, \ldots, \mathbf{m}_K$, $S(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K)$ is minimised by assigning every observation to the closest centroid:

$$c(i) = \arg\min_{j \in \{1,\ldots,K\}} \|\mathbf{x}_i - \mathbf{m}_j\|, \ i \in I\!N_n.$$

On the other hand, for given cluster labels $c(1), \ldots, c(n) \in I\!N_k$, $S(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K)$ is minimised by choosing, for $k \in I\!N_k$,

$$\hat{\mathbf{m}}_k = \frac{1}{n_k} \sum_{c(i)=k} \mathbf{x}_i,$$

where $n_k = |C_k| = |\{i : \ c(i) = k\}|$, i.e., the number of points in cluster k. This means that $\mathbf{m}_k^{Km}$ is the multivariate mean vector of the points in cluster k.
Here's why:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{m}_k} S(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K) &= \frac{\partial}{\partial \mathbf{m}_k} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{m}_{c(i)}\|^2 \\
&= \frac{\partial}{\partial \mathbf{m}_k} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} (x_{ij} - m_{c(i)j})^2 \right) \\
&= -2 \sum_{c(i)=k} \sum_{j=1}^{p} (x_{ij} - m_{kj}) = -2 \sum_{j=1}^{p} \Big(\sum_{c(i)=k} x_{ij} - n_k m_{kj} \Big).
\end{aligned}$$

Equating this to zero, $m_{kj} = \frac{1}{n_k} \sum_{c(i)=k} x_{ij}, \ j \in I\!N_p$ (the second derivative is positive, so this is a minimum).

2.2 Computation

The minimisation of $S(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K)$ is a hard discrete optimisation problem. In order to find the global minimum, in principle all possible partitions of $\mathcal{D}$ into K clusters would have to be tried out, which is computationally intractable for all except very small datasets. Some research indicates that the number of partitions to be tried can be somewhat reduced, but even using this, a guarantee to find the global minimum can only be given for quite small datasets so that in practice it is almost always necessary to apply algorithms that are only guaranteed to lead to local minima (although for some strongly clustered datasets with a suitable choice of K there is a chance to find the global optimum in this way).

A classical K-means algorithm (this was first developed by Lloyd in 1957; in some literature the term K-means is used exclusively for this algorithm) works as follows.

Step 0 Start with an initial set of centroids, i.e., $\mathbf{m}_1, \ldots, \mathbf{m}_K$. Then iterate:

Step 1 Assign every observation to the closest centroid:

$$c(i) = \underset{j \in \{1,\ldots,K\}}{\arg\min} \|\mathbf{x}_i - \mathbf{m}_j\|, \; i \in I\!N_n.$$

Step 2 For $k \in I\!N_K$, compute the means $\mathbf{m}_k = \frac{1}{n_k} \sum_{c(i)=k} \mathbf{x}_i$.

Stop when Step 1 and 2 do not change the clustering from the previous iteration.

This algorithm is guaranteed to converge, because every single step is guaranteed to decrease $S(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K)$; as shown above, Step 1 minimises S for fixed centroids and Step 2 minimises S for a fixed partition. There are only finitely many partitions, and therefore the algorithm will stop after a finite number of iterations. In practice, the algorithm usually converges very quickly.

The main issue with this algorithm is that it depends on the initial choice of centroids. Much software (including the `kmeans`-function in R) initialise the algorithm by taking K random data points as initial clusters. Actually the algorithm can also be initialised by assigning every point to a random cluster and find initial centroids by Step 2 before running Step 1. There is some research on how to find partitions that lead to good local minima of S in many situations. There are also some more sophisticated algorithms proposed in the literature, and R's `kmeans`-function by default runs an algorithm by Hartigan and Wong (1979), which also depends on initial centroids. In order to find a good minimum of S, the algorithm should be run many (e.g., 100) times with different random starting partitions, and finally the partition with the smallest value of s that was found by any of these runs is given out. In many datasets this will find the global minimum (depending on the size, dimension and strength of clustering of the dataset, the choice of K and the number of runs), although this cannot be guaranteed. It should be noted that, because of the random initialisation, running this procedure twice in the same dataset may give different results, although the probability of this to happen will go down with an increasing number of algorithm runs, i.e., different random initialisations.

2.3 Examples

In R, the `kmeans`-function computes K-means. As first example, consider the first artificial dataset from Figure 1.5. In R, the `kmeans`-function computes K-means.

```
library(fpc)
# This is needed for the use of clusym in plot later

set.seed(665544)
# This is for making the result of kmeans with random initialisation
# reproducible, although for this dataset I don't expect this to be
# unstable. I'd expect that usually you will arrive at the same solution
# with pretty much whatever random seed.
```

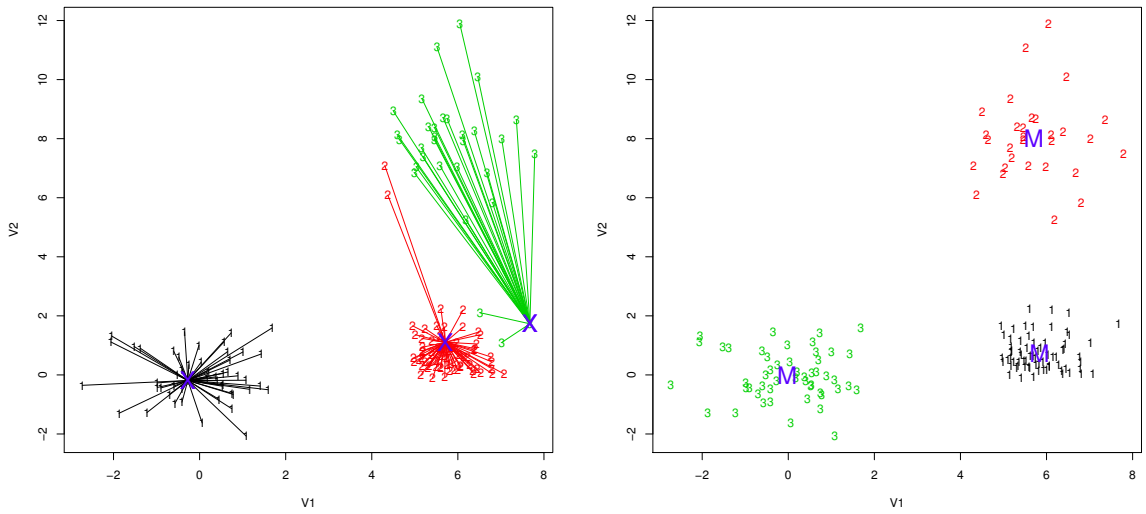

Figure 2.1: Artificial dataset 1. Left: data with three random cluster centroids ("X") and point assignments from Step 1 of the K-means algorithm. Right: 3-means clustering.

```
c1k3 <- kmeans(clusterdata1,centers=3,nstart=100)
# 3 is K, the number of clusters. nstart is the number of algorithm
# initialisations

plot(clusterdata1,col=c1k3$cluster,pch=clusym[c1k3$cluster])
# clusym is a character vector with numbers "1", "2", etc.

# Show the cluster means:
points(c1k3$centers,pch="M",cex=2,col=4)
```

The result is shown on the right side of Figure 2.1. This looks as it should be. The left side shows the initial step of the algorithm. Three data points ("X") are randomly drawn as centroids, and all points are assigned to the closest centroid. In the next Step 2, the mean of cluster 3 will be larger and the mean of cluster 2 will be smaller on the y-axis than their initial centroids, so that these will be closer to the final solution (actually, in this example, the final solution will be reached in only two iterations).

K-means applied to Old Faithful Geyser data:

```
set.seed(12345)
geyserk2 <- kmeans(geyser,2,nstart=100)
# 2 is K, the number of clusters. nstart is the number of algorithm
# initialisations

plot(geyser,col=geyserk2$cluster,pch=clusym[geyserk2$cluster])
```

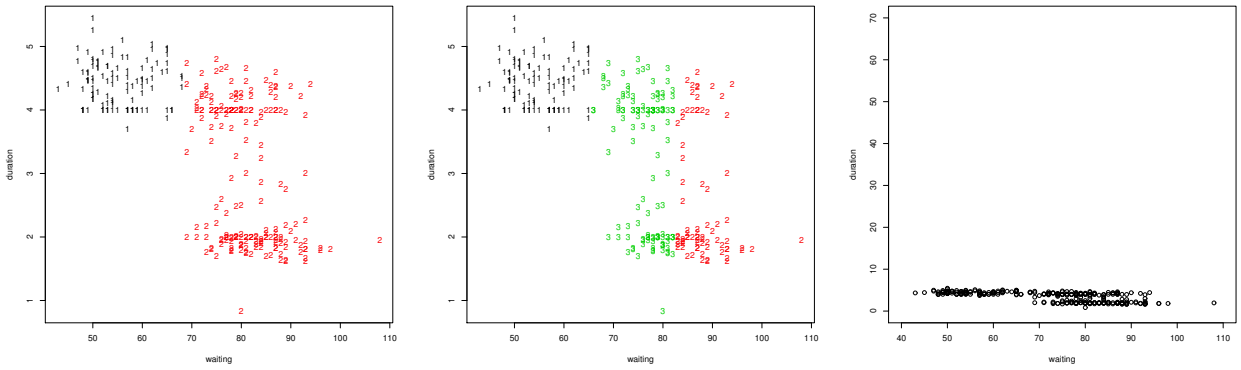

Figure 2.2: Old Faithful Geyser data, 2-means and 3-means clustering, and data with equally large plot ranges on x- and y-axis.

```
#...and for 3 clusters
geyserk3 <- kmeans(geyser,3,nstart=100)
```

```
plot(geyser,col=geyserk3$cluster,pch=clusym[geyserk3$cluster])
```

The results are shown in Figure 2.2. They don't look very intuitive. The reason for this is that the waiting variable has a much larger variation with a value range of about 60, whereas the duration variable only has a value range of a bit more than 4. This means that the Euclidean distance between most pairs of points and therefore the clustering, which is based on squared Euclidean distances, will be strongly dominated by the difference in values in the variable waiting, and the variable duration doesn't play much of a role. Note that `plot` in R adapts the value ranges on the x- and y-axis to the value ranges of the data, so that the standard plot gives a somewhat misleading impression of the Euclidean distances. The third picture in Figure 2.2 is the result of

```
plot(geyser,xlim=c(40,110),ylim=c(0,70))
```

with a value range of 70 on both axes, and this is how `kmeans` actually interprets the data.

If it is desired to give both variables the same chance to influence the clustering, it is advisable to scale the variables first, i.e., to divide them by their standard deviation so that both have variance 1.

```
sgeyser <- scale(geyser)
# This divides the two variables by their respective standard
# deviations and it also subtracts the mean, i.e., it centers them
# at zero, although this is irrelevant for clustering.

geysersk2 <- kmeans(sgeyser,2,nstart=100)
```

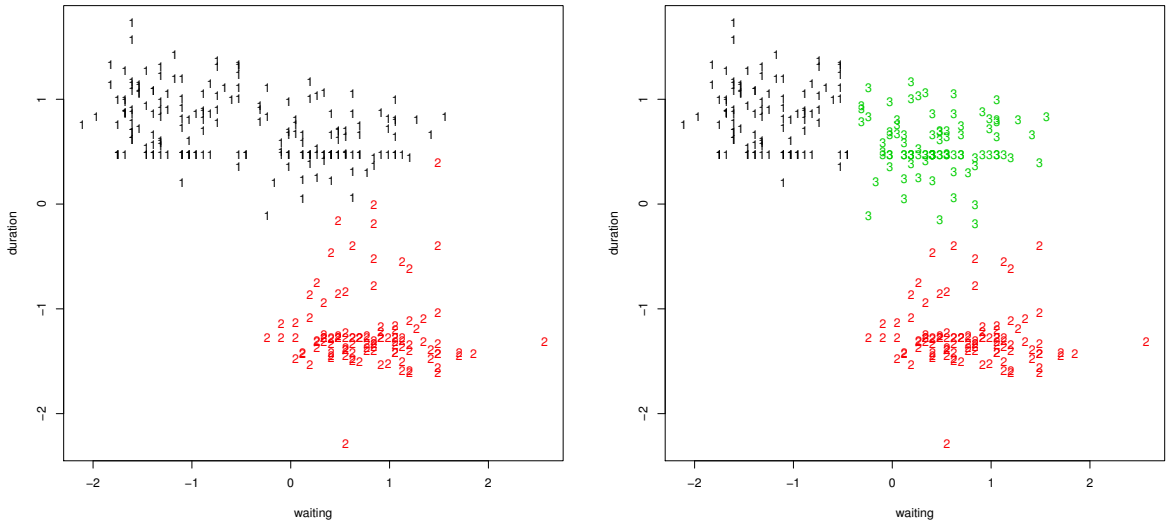

Figure 2.3: Scaled Old Faithful Geyser data, 2-means and 3-means clustering.

```
plot(sgeyser,col=geysersk2$cluster,pch=clusym[geysersk2$cluster])

geysersk3 <- kmeans(sgeyser,3,nstart=100)


plot(sgeyser,col=geysersk3$cluster,pch=clusym[geysersk3$cluster])
```

The results are shown in Figure 2.3, and look much more intuitive, although it may be seen as counter-intuitive that in the 2-means solution a point with a quite high duration appears in the low duration cluster 2. Section 2.5 will deal with estimating the number of clusters.

If however all variables have the same measurement units and comparing values across variables is meaningful, it is often better not to standardise, because in such a case a larger variance of a variable can be due to bigger differences between clusters along this variable, which is informative for clustering and should not be standardised "away".

An example for this are the German Bundestag data, in which data are proportions and a difference of 0.01 refers to the same number of voters for every party, despite the fact that the bigger parties SPD and UNION have overall larger proportions and larger variation.

There are five parties, so it may be reasonable to think that there could be five clusters corresponding to strongholds of the parties (although this is not a strong argument to rule out any other number of clusters).

```
set.seed(1234567)
bundestagk5 <- kmeans(p05,5,nstart=100)

pairs(p05,xlim=c(0,0.6),ylim=c(0,0.6),cex=0.7,col=bundestagk5$cluster,pch=clusym[bundesta
```

See Figure 2.4 for the results. Here are a few basic steps to learn something from these results. At first let's have a look at the cluster centroids, which are typical objects within the clusters and can tell us something about the characteristics of the clusters.

```
bundestagk5$centers
#         SPD     UNION     GRUENE        FDP      LINKE
# 1 0.4755049 0.2809211 0.07737812 0.07977076 0.05684986
# 2 0.3219290 0.4031411 0.08349274 0.11502019 0.04082300
# 3 0.3076717 0.2555014 0.05407416 0.07920767 0.24680508
# 4 0.3709695 0.3258766 0.11039143 0.10678831 0.05418565
# 5 0.2382177 0.5235427 0.06558486 0.09447968 0.03217685
```

As expected, the first cluster looks like SPD-strongholds, the fifth cluster looks like UNION-strongholds, and in the third cluster LINKE are for stronger than elsewhere (note that the order/numbering of clusters is arbitrary). Also, cluster 2 has the strongest FDP results and cluster 4 has the strongest GRUENE results, the latter two are much less pronounced than the other three parties in clusters 1, 3, 5. Cluster 2 also has UNION characteristically stronger than SPD, and cluster 4 the other way round. UNION/FDP are more right wing parties than SPD/GRUENE. One could also look at the distributions of all variables within all clusters, to get more detail on this.

Further one can have a look at how the clusters related to East/West/Berlin and state:

```
table(bundestagk5$cluster,ewb)
#    ewb
#     Berlin East West
#   1      0    0   44
#   2      0    0   82
#   3      6   53    4
#   4      6    0   66
#   5      0    0   38

table(bundestagk5$cluster,state)
#    state
#     Baden-Wuerttemberg Bayern Berlin Brandenburg Bremen Hamburg Hessen
#   1                  0      0      0           0      1       1      4
#   2                 27     12      0           0      0       0      6
#   3                  0      0      6          10      0       0      0
#   4                  7      1      6           0      1       5     11
#   5                  3     32      0           0      0       0      0
#    state
#     Mecklenburg-Vorpommern Niedersachsen Nordrhein-Westfalen Rheinland-Pfalz
#   1                      0            17                  20              0
#   2                      0             3                  21              9
#   3                      7             0                   0              0
```

```
#   4                    0            8                  21                6
#   5                    0            1                   2                0
#    state
#     Saarland Sachsen Sachsen-Anhalt Schleswig-Holstein Thueringen
#   1        0       0              0                   1          0
#   2        0       0              0                   4          0
#   3        4      17             10                   0          9
#   4        0       0              0                   6          0
#   5        0       0              0                   0          0
```

The first thing to see here is that cluster 3 is dominated by the eastern states, which are all in this cluster; actually the LINKE party grew out of the former Eastern German communist party SED, and still has some traditional support there. Cluster 3 also has half of Berlin (one could actually check at constituency level whether this is the eastern half), with the other half in cluster 4, together with most of German's second city Hamburg, among other constituencies from the west. This looks like a very urban cluster (there is a third city-state in Germany, which is Bremen). The further four constituencies in cluster 3 are the whole of Saarland, which is the home state of the 2005 leader of the LINKE party, Oskar Lafontaine.

Most western states are more heterogeneous, but cluster 5 is dominated by Bayern, and the states have quite distinct profiles which can be of interest to anyone interested in German politics, demography and geography.

2.4 Probabilistic background

K-means was originally introduced as a heuristic method, not motivated by statistical models. However, it can also be derived as a maximum likelihood (ML) estimator in a particular statistical model. This is very instructive about typical behaviour of K-means.

Recall that if we assume $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n$ be generated i.i.d. from $\mathcal{N}(\mathbf{a}, \Sigma)$ (p-variate normal/Gaussian distribution with mean vector $\mathbf{a}$ and covariance matrix Σ; let $\varphi_{\mathbf{a},\Sigma}$ be the density of this distribution), the mean vector $\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ is the ML estimator for $\mathbf{a}$.

For K-means, the K cluster centroid mean vectors are ML-estimators in a model in which the data are generated by K Gaussian distributions modelling the K clusters with different mean vectors. The covariance matrices need to be assumed to be equal and of the form $b\mathbf{I}_p$ with $b \geq 0$ a constant and $\mathbf{I}_p$ being the p-dimensional unit matrix. This means that data are not i.i.d.; independence is assumed (therefore there is a product in (2.1)) but not identity because there are different distributions within different clusters.

Theorem 2.3 *Let $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n$ be distributed according to the joint density*

$$f(\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n) = \prod_{i=1}^{n} \varphi(\mathbf{x}_1; \mathbf{a}_{\gamma(i)}, b\mathbf{I}_p), \tag{2.1}$$

with $\gamma : \mathbb{N}_n \mapsto \mathbb{N}_K$ denoting the true distributions for the n observations, and all parameters unknown. Then the K-means solutions $\mathbf{m}_1^{Km}, \ldots, \mathbf{m}_n^{Km}$ are ML-estimators for $\mathbf{a}_1, \ldots, \mathbf{a}_K$, and the cluster labels $c^{Km}(1), \ldots, c^{Km}(n)$ are ML-estimators of the true cluster labels $\gamma(1), \ldots, \gamma(n)$.

Proof:

$$\log f(\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n) = \sum_{i=1}^{n} \left(-\log(\sqrt{2\pi \det(b\mathbf{I}_p)}) - \frac{1}{2b}(\mathbf{x}_i - \mathbf{a}_{\gamma(i)})'(\mathbf{x}_i - \mathbf{a}_{\gamma(i)}) \right).$$

Maximising this does not depend on b and amounts to minimising

$$\sum_{i=1}^{n}(\mathbf{x}_i - \mathbf{a}_{\gamma(i)})'(\mathbf{x}_i - \mathbf{a}_{\gamma(i)}) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{a}_{\gamma(i)}\|^2$$

by choice of $\mathbf{a}_1, \ldots, \mathbf{a}_K, \gamma(1), \ldots, \gamma(n)$, which is the same problem as minimising $S(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K)$ and is therefore solved by K-means.

Remark 2.4 *What can be learnt from this is that K-means is implicitly based on Gaussian clusters with equal and spherical covariance matrices. Indeed K-means has difficulties correctly finding clusters if their covariance matrices differ strongly, are not spherical (i.e., variances are not the same in all directions of p-dimensional space) or nonlinear.*

Particularly if covariance matrices differ and/or are not spherical, assigning all observations to the closest mean is often not the best thing to do. This is illustrated by the artificial dataset 2.

```
c2 <- kmeans(clusterdata2,3,nstart=100)$cluster
plot(clusterdata2,col=c2,pch=clusym[c2])
```

The results are in Figure 2.5. This lumps the two low variance Gaussian clusters together but splits up the uniform cluster with larger within-cluster variation. There are applications in which the 3-means solution can still be useful (e.g., in data compression tasks where large within-cluster distances are not desired), but in most cases a clustering is preferred that rather corresponds to the intuitive patterns in the data.

In Chapter 6 on mixture models, methods will be introduced that work in such a situation.

Remark 2.5 *Another thing that follows from the implicit assumption of spherical covariance matrices is that K-means is not scale equivariant, i.e., if one variable is multiplied by a constant (e.g., by 100 for switching measurement units from m to cm) and other variables are not multiplied by the same constant, the K-means clustering will normally change. This was already explored in the Old Faithful Geyser data example above, in which scaling changed the clustering.*

Remark 2.6 *K-means is usually introduced without mentioning the probabilistic background, and in some literature it is even stated that (as opposed to the methods that will be introduced in Chapter 6) K-means is a method that can be applied universally because it doesn't require any model assumptions, because it was originally introduced and motivated by heuristic considerations not using a probabilistic model. This is an exemplary case of a misunderstanding about model assumptions that is very widespread and can be met in almost all areas of statistics.*

On one hand it does not make sense to state that K-means does not require any model assumptions because Theorem 2.3 establishes K-means as ML-estimator under a certain model assumption, and indeed this model assumption is very informative for understanding what K-means does and under what circumstances it is not appropriate.

On the other hand, it can make sense to apply K-means in situations in which the model assumptions are not fulfilled. If in a certain application the aim of clustering is to assign objects to classes in order to represent them by a centroid object (for example for showing a user representative images of a database of images that is to be clustered for helping the users to find images), for this aim it is useful to optimise $S(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K)$ regardless of whether the data follow the assumption of Theorem 2.3, because this generates a clustering in which, according to the squared Euclidean distance, all the objects are well represented by the centroids. Whether this corresponds to an intuitive clustering as in the example in Remark 2.4 is irrelevant in such an application. In some other applications, in which researchers are interested in finding clusters with some hidden meaning, which give rise to certain within-cluster distributions, however, certain violations of the model assumptions of Theorem 2.3 can be problematic. Even if the model assumptions of the Theorem are violated, though, K-means still will do a good job in some situations, particularly where clusters are approximately spherical, cover regions of approximately the same size in data space, are fairly well separated and K is well chosen.

As for all statistical methods, it is a subtle but important issue to what extent real data can be tolerated to deviate from the formal model assumptions without doing much harm to the results of the analysis. Not only K-means, but all model-based methods will work well in some but not all cases in which the assumptions are violated, and to distinguish between acceptable and unacceptable violations of the model assumptions is a major skill for a statistician, which only comes with conscious evaluation of experience (one source of such experience is to simulate datasets with model assumptions violated, and to run the methods and try to understand the results).

Remark 2.7 *K-means as ML-estimator in Theorem 2.3 is a rare case of an inconsistent ML-estimator. In Figure 2.6 two one-dimensional Gaussian distributions with variance 1 and the two means -1 and 1 are showed.*

For $n \to \infty$, assuming that equally many points are observed from both Gaussian, K-means with $K = 2$ will cut the real line in two and assign all negative points to the first cluster and all positive ones to the second one. This makes sense in terms of clustering. However, the two cluster means will not be estimated as the means of the two original Gaussians, but rather as the means of truncated distributions for negative observations (or respectively positive ones) only, because K-means cannot see that some of the positive points were actually generated by the Gaussian with the negative mean, albeit with low probability. The estimated 2 means will therefore have a larger distance from each other and from zero than the true means -1 and 1. Theory about the consistency of ML-estimators does not apply here, because the model is not i.i.d. and the cluster label parameter γ that assigns one label to every observation behaves in a nonstandard way.

The figure was produced as follows:

```
xpoints <- seq(-4,4,by=0.01)
norm1 <- dnorm(xpoints,mean=-1,sd=1)
```

```
norm2 <- dnorm(xpoints,mean=1,sd=1)

# Plotting densities:
plot(xpoints,norm1,type="l",ylab="density")
points(xpoints,norm2,type="l")

# red line, and means of distributions
lines(c(0,0),c(0,0.4),col=2)
points(1,0,pch="X",col=4)
points(-1,0,pch="X",col=4)

# Computations for truncated distributions.
# See https://en.wikipedia.org/wiki/Truncated_normal_distribution
# for formulae.

# Probability of being below zero for left Gaussian
prob1 <- pnorm(0,-1,1)
# Probability of being below zero for right Gaussian
prob2 <- pnorm(0,1,1)

# Expected value of Gaussian with mean -1, truncated at 0.
te1 <- -1-dnorm(1)/pnorm(1)
# Expected value of Gaussian with mean 1, truncated at 0.
te2 <- 1-dnorm(-1)/pnorm(-1)
# Expected value for left cluster combines the two according to
# their probabilities for their controbutions to the left cluster.
ecluster1 <- prob1*te1+prob2*te2
# Symmetric for right cluster.
ecluster2 <- -ecluster1

# Plot cluster means
points(ecluster1,0,pch="M",col=3)
points(ecluster2,0,pch="M",col=3)
```

2.5 Estimating the number of clusters

The classical K-means requires the researcher to specify K. In some applications such as the image database one, K can be specified by practical considerations. More often, however, K is not known and the researcher would like to estimate it from the data.

A straightforward idea for this is to compute the K-means clustering for various values of K. For fixed K, $S_K = S(\mathcal{C}, \hat{\mathbf{m}}_1^{Km}, \ldots, \hat{\mathbf{m}}_K^{Km})$ measures the quality of the K-means solution, the smaller S_K the better. Unfortunately, comparing values of S_K for different K cannot directly be used to estimate K, because S_K will always decrease with increasing K. The reason for

this is that if there are more than K distinct observations, from every K-means clustering a $K+1$-means clustering can be constructed by choosing a single observation $\mathbf{x}_i$, say, that was a member of a bigger cluster before, as a cluster on its own with $\mathbf{m}^*_{K+1} = \mathbf{x}_i$, so that $\|\mathbf{x}_i - \hat{\mathbf{m}}^*_{K+1}\| = 0$. Defining $\mathcal{C}^*$ by keeping all other cluster assignments the same as for K-means, and $\mathbf{m}^*_1 = \mathbf{m}^{Km}_1, \ldots, \mathbf{m}^*_K = \mathbf{m}^{Km}_K$, a clustering with $K+1$ clusters is defined, which achieves $S(\mathcal{C}^*, \mathbf{m}^*_1, \ldots, \mathbf{m}^*_K) < S_K$, and therefore $S_{K+1} < S_K$, because S_{K+1} minimises S among all clusterings with $K+1$ clusters.

A folklore approach is to plot S_K against K and to look for an "elbow", where S_K decreases strongly up to a certain value K_0 and decreases only a little bit for $K > K_0$. However, such a value cannot always clearly found and the plot may be ambiguous in many cases, see the left sides of Figure 2.8; on the left side of Figure 2.7 this works better.

There are many attempts in the literature to define an index based on investigating the expected behaviour of S_K for growing K that can be chosen to estimate the number of clusters, for an overview see Halkidi et al. (2015); an alternative index will be defined in Chapter 5.

A theoretically quite elegant and promising approach is by Sugar and James (2003) (although experience how well this works in practice is somewhat limited up to now). The theory in that paper is skipped but one of the results there is as follows, assuming a mixture model, which is the model of Theorem 2.3 with true number of Gaussians G conditional on the labels $\gamma(i),\ i \in I\!N_n$, but with $\gamma(i)$ being an i.i.d. random variables that takes values $1, \ldots, G$ with probabilities $\pi_1, \ldots, \pi_G$ (see Section 6). These are here assumed to be equal. Also a certain minimum distance between cluster centers is assumed. Then, approximately,

$$\left(\frac{S_K}{p}\right)^{-p/2} \approx \begin{cases} a\frac{K}{G} + q & \text{for } K \geq G \\ q & \text{for } K < G \end{cases} \tag{2.2}$$

for a constant $q \geq 0$ and another constant $0 < a < 1$. The approximation is for $p \to \infty$ but seems to be a pretty good guideline for choosing K even for small p according to Sugar and James (2003).

(2.2) leads to the recommendation to estimate G by

$$\hat{K} = \arg\max_K D(K),\ D(K) = S_K^{-p/2} - S_{K-1}^{-p/2}, \tag{2.3}$$

because this is expected according to 2.2 to be about zero for $K < G$, about $ap^{-p/2}$ for $K = G$ and about $\frac{ap^{-p/2}}{G}$ for $K > G$. G is therefore estimated at the biggest "jump" in $S_K^{-p/2}$, see Figure 2.7.

The assumptions for this are somewhat restrictive (for example, the theory assumes that all clusters asymptotically have the same size and p is large), and more practical experience is required to compare this approach with others appearing in Halkidi et al. (2015). Sugar and James (2003) have more general theory, but this is based on looking at S_K^{-Y} with Y dependent on the in practice unknown within-cluster distributions; $Y = p/2$ as above is often good but not always.

Example 2.8 *Gareth James, co-author of Sugar and James (2003), put some free R-code for computing* (2.3) *on his website. I put this on Moodle as* `sugarjames.R`*. For artificial dataset 1:*

```
source("sugarjames.R")
# You may have to add the path.

set.seed(12345)
# K=20 indicates the maximum number of clusters; rand=100 indicates
# 100 algorithm initiatlisations.
jg <- jump(clusterdata1,K=20,rand=100)
# [1] "The maximum jump occurred at  3 clusters with Y= 1"
```

This produces Figure 2.7 as result, in which it can be seen how $\frac{S_K}{p}$ *changes with growing* K*, and how* $D(K)$ *from* (2.3) *picks 3 as best* K*. For this dataset,* (2.3) *seems to work well.*
As another example, consider the geyser dataset.

```
set.seed(76543)
# Use the scaled version of the dataset!
jg <- jump(sgeyser,K=20,rand=100)
# [1] "The maximum jump occurred at  3 clusters with Y= 1"

# Unfortunately, the "jump"-function doesn't give out a clustering,
# so this has to be computed afterwards again.
geyser3 <- kmeans(sgeyser,3,nstart=100)

# Also, after the jump-function, plot needs to be told
# that there should only be one plot in a window:
par(mfrow=c(1,1))

plot(sgeyser,col=geyser3$cluster,pch=clusym[geyser3$cluster])
```

The result was already in Figure 2.3. $K = 3$ *here indeed looks more intuitive than* $K = 2$*.*
For the Bundestag-data:

```
set.seed(12345)
jg <- jump(p05,K=20,rand=100)
# [1] "The maximum jump occurred at  19 clusters with Y= 2.5"
```

The output is in Figure 2.8. Unfortunately interpreting 19 clusters is hard, and even though this may be a good clustering, in practice it may be more useful to have fewer clusters, and it may be of interest to look at clusterings at local maxima of $D(K)$*, which occur, for example, at* $K = 4$ *and* $K = 7$*.*
Generally, methods for estimating K *may be quite sensitive to model assumptions (the method may decide that more smaller clusters are better for fulfilling Gaussian and equal size assumptions, despite lower numbers of clusters being perfectly fine intuitively) and datasets may be ambiguous in this respect.*
The estimation of the number of clusters is generally an extremely difficult problem.

As an alternative, graphical inspection of the data can often be used to find a good K. Although this is subjective and difficult for large p, the problem of estimating the number of clusters is so difficult and no existing index is reliable in all situations, it is in most cases advisable to use graphical diagnostics to at least check whether $D(K)$ or any other index appears to do a good job.

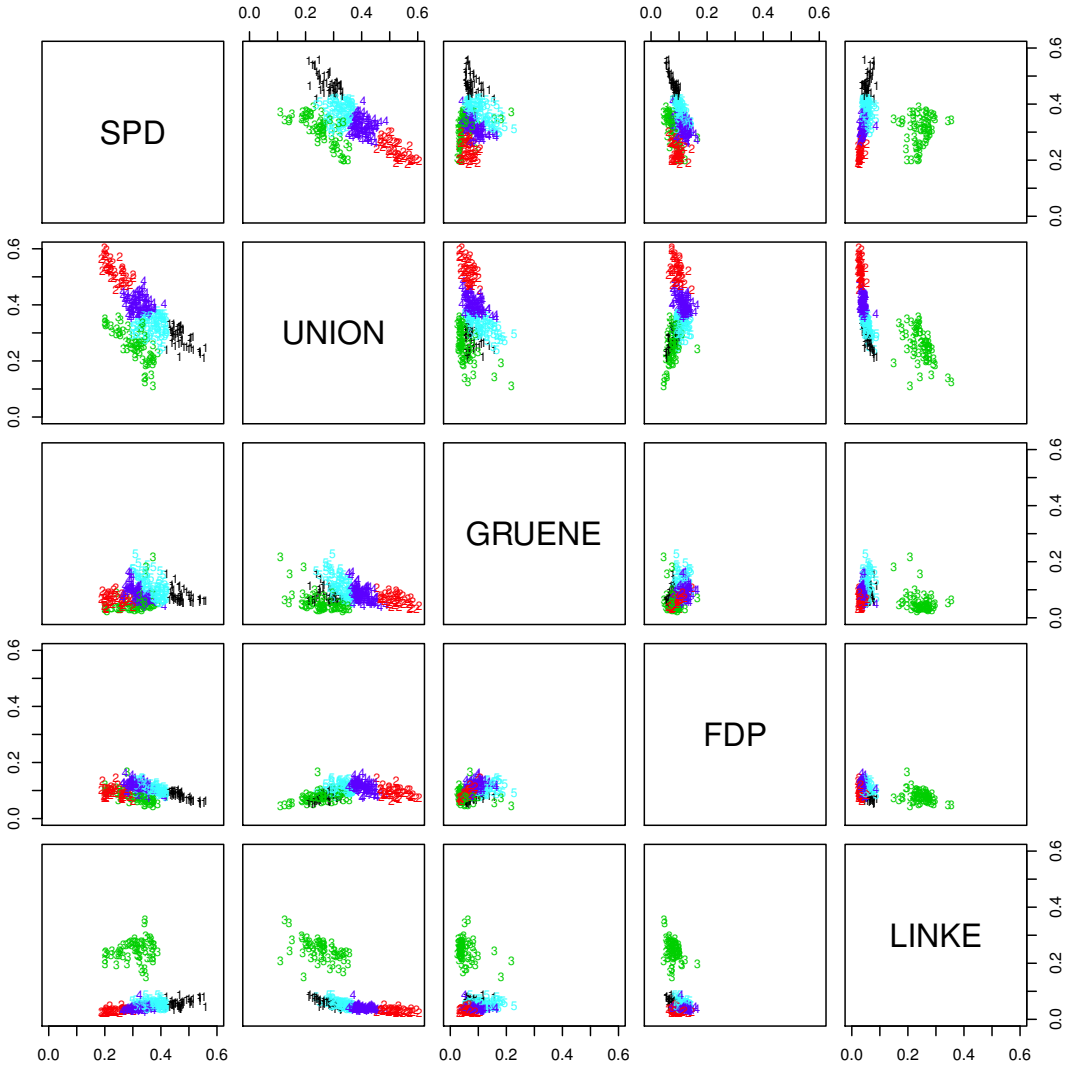

Figure 2.4: Bundestag data with 5-means clustering.

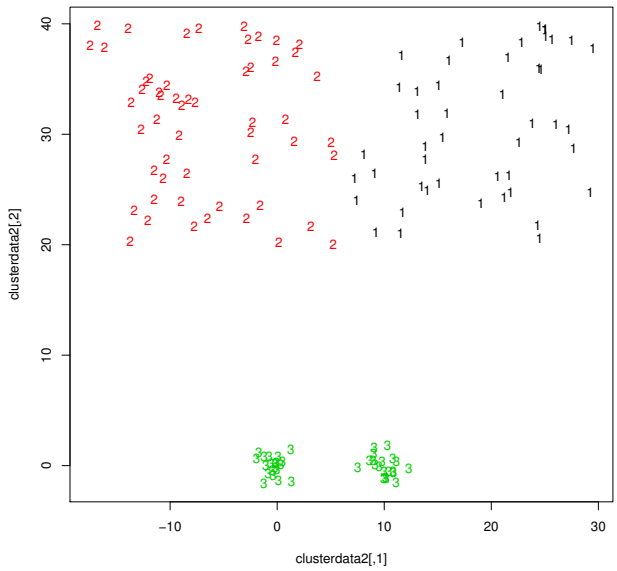

Figure 2.5: Artificial dataset 2 with 3-means clustering.

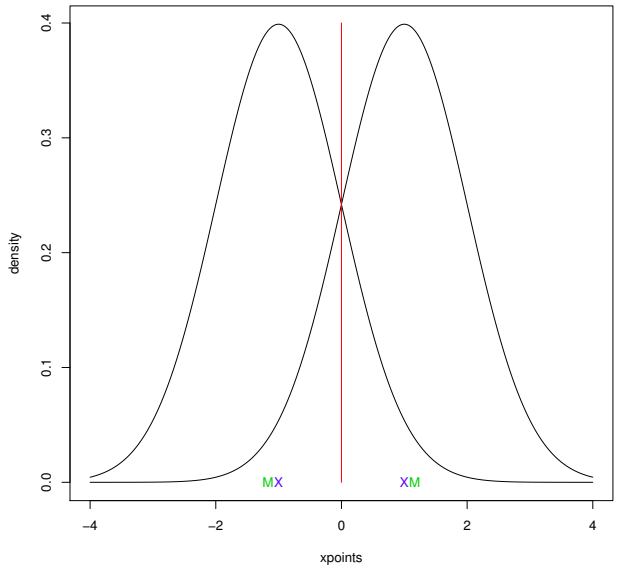

Figure 2.6: Densities for Gaussian distributions with means -1, 1, both with variance 1. The two means are indicated by blue "X". The red line indicates the border of the "asymptotic" 2-means clusters, i.e., if infinitely many points could be observed, 50% from each distribution. The means of the two clusters are then -1.17 and 1.17, indicated by "M".

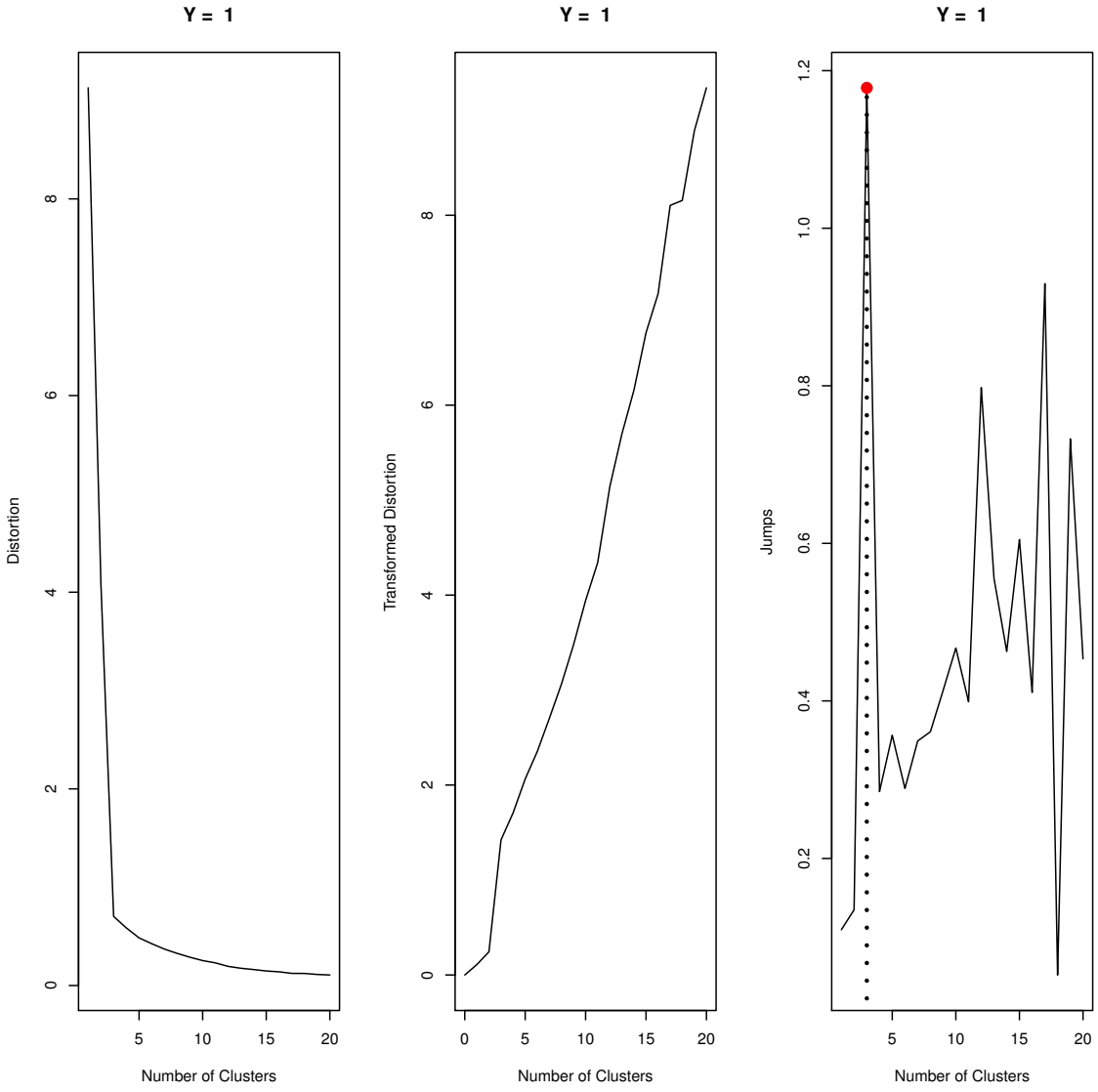

Figure 2.7: Artificial dataset 1. Number of clusters K plotted against left: "Distortion", i.e., $\frac{S_K}{p}$, middle: $\left(\frac{S_K}{p}\right)^{-Y}$ with $Y = \frac{p}{2} = 1$, right: $D(K)$ from (2.3).

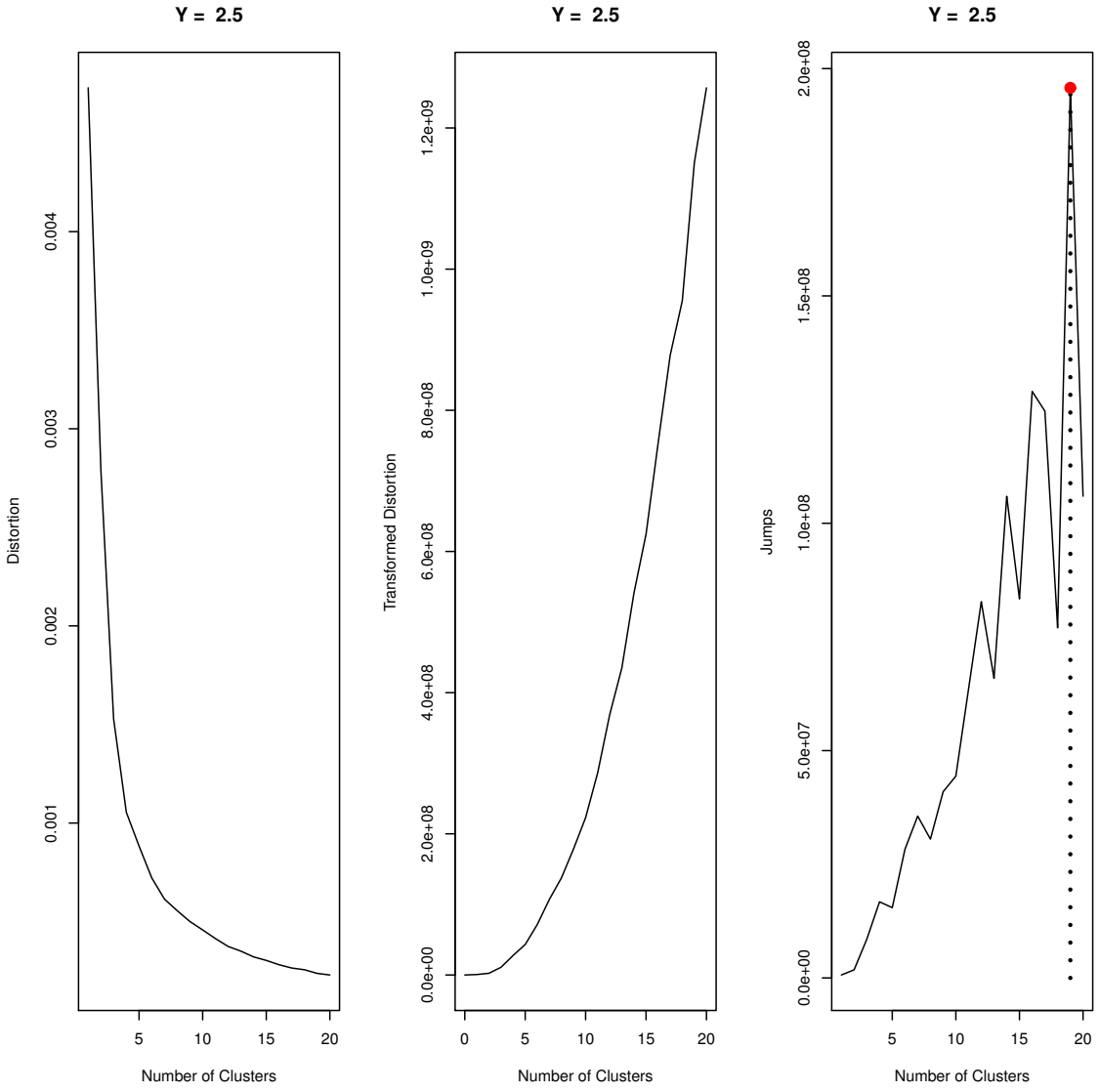

Figure 2.8: Bundestag dataset. Number of clusters K plotted against left: "Distortion", i.e., $\frac{S_K}{p}$, middle: $\left(\frac{S_K}{p}\right)^{-Y}$ with $Y = \frac{p}{2} = 2.5$, right: $D(K)$ from (2.3).

Chapter 3

Dissimilarities

An intuitive definition of clustering is that observations in the same cluster should be similar (within-cluster homogeneity) and observations in different clusters should be dissimilar (between-cluster separation).

Many cluster analysis methods, including most hierarchical clustering methods introduced later in this chapter, are based on measures of dissimilarity for pairs of observations. K-means was a method for observations characterised by p real-valued variables. Methods based on dissimilarity measures can be applied to all kinds of data including potentially multivariate binary or categorical data such as the Veronica dataset, for which dissimilarity measures can be defined. There are also dissimilarity measures for data with variables of mixed type, or for other types of data such time series, texts, images or melodies. Dissimilarity measures can also be used to cluster variables.

The choice of a dissimilarity measure does not only depend on the type of data. For the same kind of data, one may use different dissimilarity measures, depending on the meaning of the data and the aim of clustering. Consider Figure 3.1. There are three observations on eight variables, $\mathbf{x}_1 = (1, 4, 5, 4, 2, 1, 1, 4), \mathbf{x}_2 = (2, 3, 2, 2, 3, 3, 3, 3), \mathbf{x}_3 = (7, 11, 11, 12, 9, 8, 8, 12)$. On every single variable, $\mathbf{x}_1$ is closer to $\mathbf{x}_2$ than to $\mathbf{x}_3$, so according to the Euclidean distance, $d_{L2}(\mathbf{x}_1, \mathbf{x}_2) < d_{L2}(\mathbf{x}_1, \mathbf{x}_3)$. This makes sense for applications in which observations indeed can be treated as similar if the values on the individual variables are similar. This is for example the case if these are 8 different measurements on a patient that are all related positively to a certain disease; the third patient may be quite ill and patients 1 and 2 may be more or less fine and very similar, particularly if there is some measurement error with variance of about 1, say. On the other hand, there are also similarities between $\mathbf{x}_1$ and $\mathbf{x}_3$. Both have higher values on V2-V4 and V8 and lower values on the other four variables. This does not hold for $\mathbf{x}_2$. The variables may refer to the same measurement on 8 different time points, and the researcher may be interested in seasonal patterns, according to which $\mathbf{x}_1$ and $\mathbf{x}_3$ should be defined to be similar and $\mathbf{x}_1$ should be quite different from $\mathbf{x}_2$. This would require a different dissimilarity measure.

It is even conceivable that for the very same variables depending on the aim of clustering two different similarity measures could be appropriate. For example the observations may be species of plants and the variables may state their frequency in 8 different places. Should species be treated as similar if they prefer the same places, regardless of how abundant they are in terms

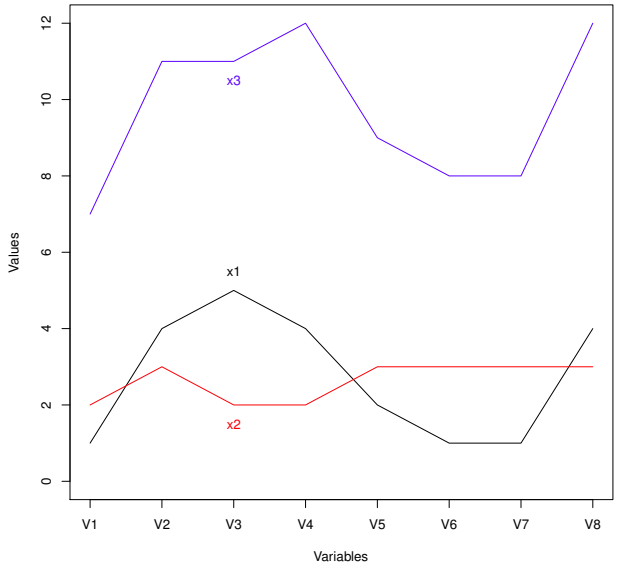

Figure 3.1: Parallel coordinates plot of three observations $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ on 8 variables V1-V8.

of absolute values? This depends on the aim of the analysis and has to be decided by a subject matter specialist; it cannot be "estimated" from the data.

Definition 3.1 *A* **dissimilarity** *is a function* $d: \mathcal{X}^2 \mapsto I\!R_0^+$, $\mathcal{X}$ *being the object space, so that* $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) \geq 0$ *and* $d(\mathbf{x}, \mathbf{x}) = 0$ *for* $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. *(Some literature also requires* $d(\mathbf{x}, \mathbf{y}) > 0$ *for* $\mathbf{x} \neq \mathbf{y}$.*)*
A dissimilarity fulfilling the triangle equality

$$d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z}), \ \ \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{X},$$

is called a **distance** *or* **metric**.

The triangle inequality is connected to Euclidean intuition and therefore seems to be a "natural" requirement, but in fact it is not required in all applications and sometimes not appropriate.

A vast number of dissimilarity measures has been proposed, some for rather general purposes. Chapter 3 in Everitt et al. (2011) gives a good overview of general purpose dissimilarities, with some more references in Hennig (2015).

Note that there is also the concept of **similarity** which has the opposite interpretation of a dissimilarity, namely larger values mean more similarity. Similarity can usually be constructed from dissimilarities (for example, max dissimilarity value minus dissimilarity gives a similarity and the other way round), therefore mostly only one of them is needed.

Both dissimilarities and similarities are called **proximities** in some literature.

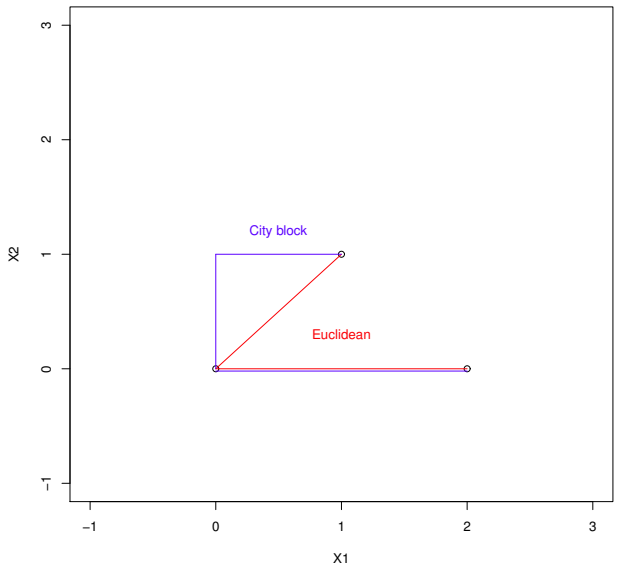

Figure 3.2: Euclidean (red) and Manhattan/city block distances (blue).

3.1 Continuous variables

Definition 3.2 *The* **Minkowski** (L_q)**-distance** *between two objects* $\mathbf{x}_i, \mathbf{x}_j$ *on* p *real-valued variables* $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ *is*

$$d_{Lq}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[q]{\sum_{l=1}^{p} d_l(x_{il}, x_{jl})^q}, \tag{3.1}$$

where $d_l(x, y) = |x - y|$.

Most often, the Euclidean distance d_{L2} and the **Manhattan/city block distance** d_{L1} are used. Using d_{Lq} with larger q gives the variables with larger d_l more weight, i.e., two observations are treated as less similar if there is a very large dissimilarity on one variable and small dissimilarities on the others than if there is about the same medium-sized dissimilarity on all variables, whereas d_{L1} gives all variable-wise contributions implicitly the same weight. Note that this does not hold for the Euclidean distance that corresponds to physical distances and is used as default choice in many applications; however, if the variables have fairly different meanings and different measurement units, d_{L1} may be more appropriate. Figure 3.2 illustrates the difference between d_{L1} and d_{L2}.

All the Minkowski distances are not scale equivariant, i.e., they will be dominated by variables with larger variation, and will change if variables are multiplied by constants, e.g., when measurement units are changed.

The Euclidean distance is rotation invariant, i.e., it doesn't change if the points in $\mathbb{R}^p$ are rotated, but this doesn't hold for the other L_q-distances.

Here is an alternative that is both scale and rotation invariant:

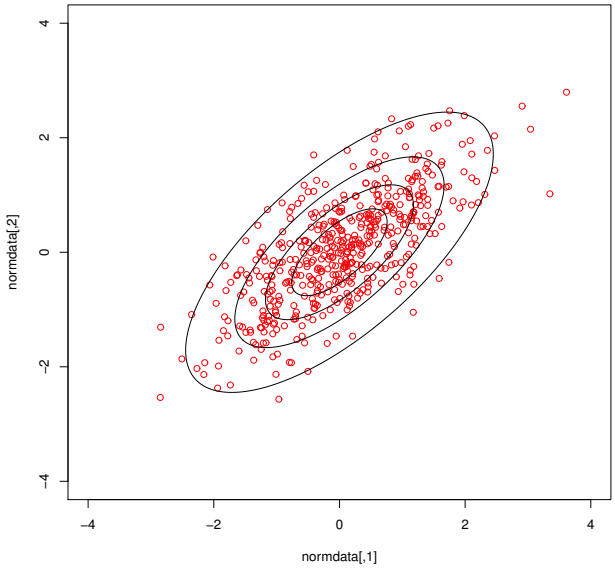

Figure 3.3: Points on ellipsoids have same Mahalanobis distance to center.

Definition 3.3 *The* **(squared) Mahalanobis distance** *is defined by*

$$d_M(\mathbf{x}_i, \mathbf{x}_j)^2 = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{x}_j), \tag{3.2}$$

where $\mathbf{S}$ *is a scatter matrix such as the sample covariance matrix.*

See Figure 3.3.

R's `dist`-function computes several standard distance measures, including Euclidean, Manhattan and Minkowski-distance. There is also a function `mahalanobis` that computes Mahalanobis-distances.

For example (see Figure 3.4):

```
dolive2 <- dist(olive,method="euclidean")
dolive1 <- dist(olive,method="manhattan")

# This shows you how similar they are:
plot(dolive1,dolive2,cex=0.3)

# Actually, it makes much more of a difference to scale the data.
solive <- scale(olive)
dolives2 <- dist(solive,method="euclidean")
dolives1 <- dist(solive,method="manhattan")

plot(dolive2,dolives2,cex=0.3) # For example.

# Note that dist produces an object of class dist.
```

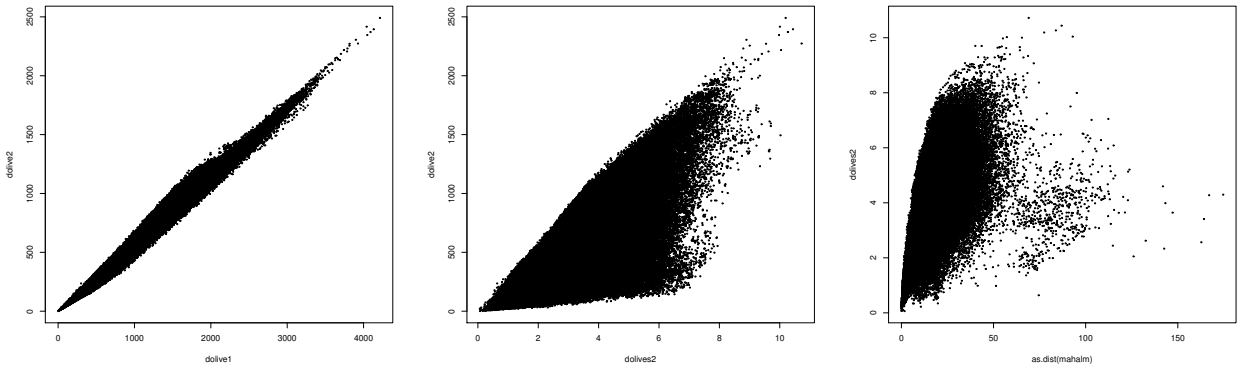

Figure 3.4: Olive oil dataset, comparing (a) Euclidean and Manhattan distance, (b) Euclidean distances on standardised and unstandardised data, (c) Euclidean distance on standardised data and Mahalanobis distance.

```
# This is a vector of all distances in one row.
# This is memory efficient, but sometimes it's more
# useful to have a distance matrix, so that it is
# easy to look up specific values:

dolivematrix <- as.matrix(dolives2)
# This gives the distance between observations 1 and 2.
dolivematrix[1,2]
# [1] 0.6644688

# A distance matrix can be transformed into a dist-object
# by as.dist.

# The mahalanobis command can only compute a vector of
# Mahalanobis distances, so producing all distances is
# more tedious; here's how to make a distance matrix:

mahalm <- matrix(0,ncol=572,nrow=572)
olivecov <- cov(olive)
for (i in 1:572)
  mahalm[i,] <- mahalanobis(olive,olive[i,],olivecov)

# Note that for the Mahalanobis distance it doesn't make
# a difference whether the dataset is scaled or not.
# Still it's quite different from the distance on
# standardised data.
plot(as.dist(mahalm),dolives2,cex=0.3)
```

3.2 Binary and categorical variables

If objects $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip}),\ i = 1, \ldots, n$, are represented by p categorical variables from a set $\mathcal{B}$ (for binary variables, $\mathcal{B} = \{0, 1\}$), the most straightforward way to define a dissimilarity is to just count the number of variables on which two objects $\mathbf{x}_1$ and $\mathbf{x}_2$ do not coincide, divided by p.

Definition 3.4

$$d_{SM}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{p}\sum_{j=1}^{p} 1(x_{1j} \neq x_{2j})$$

is called **simple matching distance** *(*$1(\bullet)$ *denotes the indicator function).*

However, often for binary data (e.g. in the case of geographical presence-absence data in ecology, particularly if there are much more absences than presences) common presences are important, whereas common absences are not. In such a situation, a different dissimilarity is often more suitable.

Definition 3.5

$$d_J(\mathbf{x}_1, \mathbf{x}_2) = 1 - \frac{\sum_{j=1}^{p} 1(x_{1j} = 1 \text{ and } x_{2j} = 1)}{\sum_{j=1}^{p} 1(x_{1j} = 1 \text{ or } x_{2j} = 1)}$$

is called **Jaccard distance**.

Example 3.6 *The difference between the two distances (both fulfill the triangle inequality) is illustrated by the following examples with* $p = 10$. *The most extreme examples are when there is only one presence each:* $\mathbf{x}_1 = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, $\mathbf{x}_2 = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$. *Then according to* d_{SM} *these are very similar,* $d_{SM}(\mathbf{x}_1, \mathbf{x}_2) = 0.2$, *because of the many joint zeroes. According to* d_J *these are as dissimilar as possible,* $d_J(\mathbf{x}_1, \mathbf{x}_2) = 1$, *the maximum, because the ones are in different places. Generally, observations with very few ones are similar according to* d_{SM}, *whereas* d_J *depends on whether the ones are in the same places. On the other hand,* $\mathbf{x}_1 = (1, 1, 1, 1, 1, 1, 1, 1, 1, 0)$, $\mathbf{x}_2 = (0, 1, 1, 1, 1, 1, 1, 1, 1, 1)$ *have* $d_{SM}(\mathbf{x}_1, \mathbf{x}_2) = d_J(\mathbf{x}_1, \mathbf{x}_2) = 0.2$.

For the Veronica data, joint presences of genetic information are important and joint absences are meaningless, and therefore the Jaccard distance is better than the simple matching distance.

There are many other distance measures for binary and categorical data.

The `dist`-function computes the Jaccard distance for binary observations with `method="binary"` (the name Jaccard doesn't appear on the help page, which is somewhat confusing). It doesn't compute the simple matching distance, which is implemented as function `sm` in package `nomclust`. For binary variables, `dist` can be used to compute the simple matching coefficient, because this is the same as the Manhattan distance divided by p.

```
jveronica <- dist(veronica,method="binary")
smveronica <- dist(veronica,method="manhattan")/p

plot(jveronica,smveronica,cex=0.3)
```

These are fairly similar, see Figure 3.5.

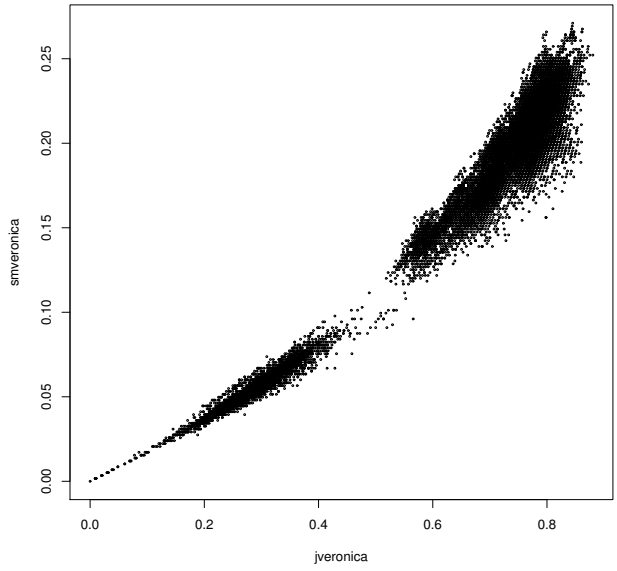

Figure 3.5: Veronica dataset, comparing Jaccard and simple matching distance.

3.3 Correlation dissimilarity

Sometimes researchers are interested in clustering variables. In the Veronica dataset one could as well be interested in clustering AFLP bands (genes) instead of plants. Also, in the Bundestag dataset, one could be interested in clustering parties in principle, although there are only five of them.

Often similarity between variables is connected to correlation. Two variables are seen as similar if they are highly correlated, which may well be the case even if they take quite different values. In the Bundestag dataset, for example, one may rather be interested in which parties are similar according to what their strong and weak constituencies are; it is less interesting and more obvious to analyse which parties rather have bigger and which parties rather have smaller percentage values overall.

For two variables $x_{.1}, x_{.2}$, for the empirical correlation coefficient $-1 \leq r(x_{.1}, x_{.2}) \leq 1$. A dissimilarity must be non-negative and zero for identical objects (variables), i.e., for correlation 1. This can be achieved by

$$d_C(x_{.1}, x_{.2}) = \frac{1}{2}(1 - r(x_{.1}, x_{.2})).$$

Have another look at Figure 3.1; if the variables are the objects and V1-V8 are the observations, the problem illustrated there can be tackled by d_C, which has $\mathbf{x}_1$ more similar to $\mathbf{x}_3$ than to $\mathbf{x}_2$ (there are also some time series dissimilarity measures that achieve the same thing).

Example 3.7 *This computes the correlation dissimilarity between parties in the Bundestag dataset:*

```
corp05 <- cor(p05)
# > corp05
#                SPD      UNION      GRUENE        FDP      LINKE
# SPD     1.00000000 -0.5662267  0.08935976 -0.3300797 -0.1337904
# UNION  -0.56622666  1.0000000 -0.15295824  0.3563825 -0.6220824
# GRUENE  0.08935976 -0.1529582  1.00000000  0.3386211 -0.3644767
# FDP    -0.33007972  0.3563825  0.33862105  1.0000000 -0.4741279
# LINKE  -0.13379043 -0.6220824 -0.36447674 -0.4741279  1.0000000

cordistp05 <- 0.5-corp05/2
# > cordistp05
#             SPD     UNION    GRUENE       FDP     LINKE
# SPD    0.0000000 0.7831133 0.4553201 0.6650399 0.5668952
# UNION  0.7831133 0.0000000 0.5764791 0.3218088 0.8110412
# GRUENE 0.4553201 0.5764791 0.0000000 0.3306895 0.6822384
# FDP    0.6650399 0.3218088 0.3306895 0.0000000 0.7370640
# LINKE  0.5668952 0.8110412 0.6822384 0.7370640 0.0000000
```

In some applications, variables should be treated as similar if they are strongly related, even if the relation is negative. This is achieved by

$$d_{CN}(x_{\cdot 1}, x_{\cdot 2}) = 1 - |r(x_{\cdot 1}, x_{\cdot 2})|.$$

3.4 Mixed type variables and missing values

In some applications, variables are of mixed type, i.e., some continuous, some binary or categorical, some perhaps ordinal or of other nonstandard type. Gower (1971) proposed a general formula that can accommodate different dissimilarity measures d_l, $l \in I\!N_p$ that one may use for different variables, or even different sets of variables.

$$d_G(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{l=1}^{p} w_l \delta_{ijl} d_l(x_{il}, x_{jl})}{\sum_{l=1}^{p} \delta_{ijl}}$$

for $i, j \in I\!N_n$, where w_l serves for variable standardisation and $\delta_{ijl} = 1$ except if x_{il} or x_{jl} are missing, in which case $\delta_{ijl} = 0$. This is a generalisation of d_{L1} and takes into account missing values by just leaving the corresponding variable out and rescaling the others. Gower recommended to use the weight w_l for standardization to $[0, 1]$-range, i.e., $w_l = \frac{1}{\max d_l}$, but it can be chosen in different ways.

3.5 Multidimensional Scaling

Multidimensional Scaling (MDS) is not really a part of cluster analysis, but it is a very useful tool for work with dissimilarities. I will only treat it very shortly and on a practical level. There is much more to MDS, see Borg and Groenen (2005).

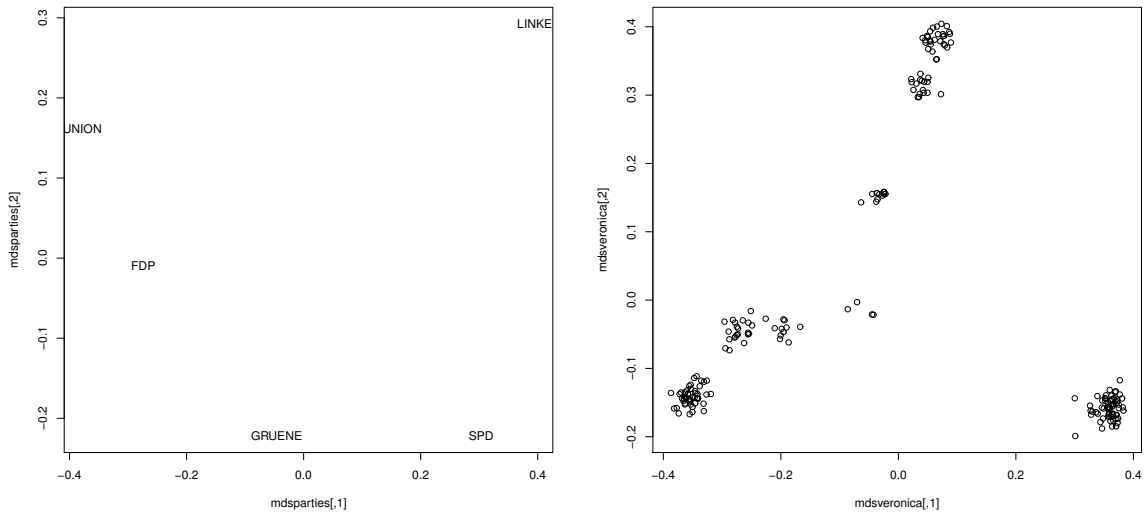

Figure 3.6: Classical 2-dimensional MDS for Bundestag parties with correlation dissimilarity (left) and Veronica plants with Jaccard distance (right).

The aim of MDS is to map objects that are characterised by a dissimilarity measure into Euclidean space so that the Euclidean distances in that space approximate in some sense optimally the dissimilarities. This means that the data can then be visualised by standard scatterplots which represent the dissimilarities approximately, i.e., given data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\}$ that are characterised by a dissimilarity measure d find $\mathbf{y}_1, \mathbf{y}_2, \ldots \mathbf{y}_n \in I\!R^p$ so that for all $i, j \in I\!N_n$:

$$d(\mathbf{x}_i, \mathbf{x}_j) \approx d_{L2}(\mathbf{y}_i, \mathbf{y}_j).$$

Example 3.8 *For the Bundestag parties, using the correlation dissimilarity:*

```
mdsparties <- cmdscale(cordistp05,k=2)
# k is the number of dimensions here.

plot(mdsparties,type="n")
text(mdsparties,labels=dimnames(p05)[[2]])
```

This produces the left side of Figure 3.6, the first dimension of which has some kind of left wing/right wing interpretation, although the right wing parties appear on the left (the orientation of the plot is meaningless regarding the quality of approximation of dissimilarities).

For visualising the Jaccard distance for the Veronica data:

```
mdsveronica <- cmdscale(jveronica,k=2)
plot(mdsveronica)
```

This dataset appears to be clearly clustered, see the right side of Figure 3.6.

Never forget, though, that this is only an approximation of the dissimilarities, and that in two dimensions the approximation may be quite weak.

One could run `cmdscale` *with, e.g.,* `k=5`*, and look at a* `pairs`*-plot to see more (for only five observation as in case of the Bundestag parties, there is not much to expect from* `k>2`*).*

3.6 Similarity between clusterings

Clusterings on a dataset can themselves be seen as data, and one may be interested in how similar different clusterings on a dataset are. As opposed to the other sections, for reasons of consistency with most literature, I focus on similarity measures here.

There are many applications for this.

- It may be of interest to see to what extent a clustering corresponds to a given external grouping, for interpreting and validating the clustering (such as the regional groupings for the Olive Oil and Bundestag datasets).
- If such an external grouping is interpreted as "truth", different clustering methods can be compared according to how similar to the truth their results are.
- The influence of certain features of the dataset may be investigated. For example, clusterings with all variables and with some variables left out can be compared to see how much difference the variables in question make. The same applies to clustering with all data and with certain outliers left out.
- There is some recent work on exploring the stability of a clustering by running a clustering method on various subsets of the data ("bootstrap"/"resampling"). Results can be compared on the data points these subsets have in common, which measures the stability of the clustering. This can even be used to decide the number of clusters as the number for which the clustering is most stable. See Leisch (2015) for an overview of such techniques.
- There are also certain applications where one wants to cluster clusterings based on a dissimilarity measure, although the is rarer than the applications of a similarity between clusterings mentioned above.

The most widely used similarity goes back to Rand (1971). This is a version of the simple matching distance for binary data.

Definition 3.9 *Let* $\mathcal{C}_1 = \{C_{11}, \ldots, C_{1K_1}\}$, $\mathcal{C}_2 = \{C_{21}, \ldots, C_{2K_1}\}$ *be partitions of a dataset* $\mathcal{D}$ *with* $|\mathcal{D}| = n$*. For every pair of points* $\mathbf{x}, \mathbf{y} \in \mathcal{D}$ *and* $j = 1, 2$*, define*

$$i_j(\mathbf{x}, \mathbf{y}) = 1(\mathbf{x}, \mathbf{y}) \text{ are in same cluster in } \mathcal{C}_j.$$

The **Rand index** *is the simple matching similarity between* i_1 *and* i_2 *over all pairs of points:*

$$R(\mathcal{C}_1, \mathcal{C}_2) = \frac{1}{n(n-1)/2} \sum_{\mathbf{x},\mathbf{y} \in \mathcal{D}} 1\left(i_1(\mathbf{x}, \mathbf{y}) = i_2(\mathbf{x}, \mathbf{y})\right).$$

The Rand index is the proportion of pairs of points that are either in the same cluster in both clusterings or in different clusters in both clusterings. As such, its computation does not rely on whether $K_1 = K_2$, and it does not require to match specific clusters in $\mathcal{C}_1$ with clusters in $\mathcal{C}_2$.

Unfortunately, the interpretation of values of R depends on the numbers and sizes of clusters in the two clusterings. Two random clusterings with large K_1, K_2 can be expected to have a rather high value of R. Therefore, people usually use an adjustment by Hubert and Arabie (1985).

Definition 3.10 *The* **adjusted Rand index** *(ARI) is defined as* $ARI(\mathcal{C}_1, \mathcal{C}_2) = \frac{R(\mathcal{C}_1,\mathcal{C}_2)-ER}{1-ER}$, *where* ER *is the expected value of* R *assuming that points are randomly assigned to clusters in such a way that numbers and sizes of clusters in* $\mathcal{C}_1, \mathcal{C}_2$ *are held constant.*

Theorem 3.11 *With the definition above,*

$$ARI(\mathcal{C}_1, \mathcal{C}_2) = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}},$$

where i *always runs over* $\mathbb{N}_{K_1}$ *and* j *runs over* $\mathbb{N}_{K_2}$, *with* $n_{ij} = |C_{1i} \cap C_{2j}|$, $a_i = \sum_{j=1}^{K_2} n_{ij}$ *and* $b_j = \sum_{i=1}^{K_1} n_{ij}$.

This was proved by Hubert and Arabie (1985).

The ARI can take values between -1 and 1. It takes value 1 if and only if the clusterings are identical, and 0 is its expected value for two independent random clusterings. Some experience is helpful for interpreting ARI values; certainly negative and very small ones do not indicate hardly any similarity between clusterings. Whether values between, say, 0.3 and 0.7 indicate "good similarity" depends on the standards required and on whether and what different ARI values are compared.

Example 3.12 *For the Olive Oil data, we may be interested in whether or not scaling of the data gives a better* K*-means clustering result. Assuming that the macro areas are some kind of true clustering with* $K = 3$, *we get:*

```
library(mclust) # This has the adjustedRandIndex command.

solive <- scale(olive)
olive3 <- kmeans(olive,3,nstart=100)
olive3s <- kmeans(solive,3,nstart=100)

adjustedRandIndex(olive3$cluster,olive3s$cluster)
# 0.4587804, these are somewhat different.

adjustedRandIndex(olive3$cluster,oliveoil$macro.area)
# 0.3182057
```

```
adjustedRandIndex(olive3s$cluster,oliveoil$macro.area)
# 0.448355, both OK but not great, with scaling clearly better

# The Sugar and James index suggests a higher number of clusters.
olive20 <- kmeans(solive,20,nstart=100)
adjustedRandIndex(olive20$cluster,oliveoil$macro.area)
# 0.166162, not helpful.
```

Meila (2015) lists more similarity indexes for clusterings. In the same way in which the Rand index uses the simple matching (dis)similarity, the Jaccard distance can also be used.

Chapter 4

Hierarchical clustering

4.1 Basic concepts

Hierarchical clustering is even older than K-means, and its origins are in Poland as well. Florek et al. (1951) introduced Single Linkage clustering in order to group objects based on a given dissimilarity. Research on hierarchical clustering can be seen as part of discrete mathematics rather than statistics; connecting it to statistical models is difficult. In many applications, however, hierarchical clustering is a competitor to K-means and clustering based on statistical models; it has its own characteristics that are sometimes useful.

The aim of hierarchical clustering is to set up a hierarchy of clusters, i.e., not only to partition the data into a fixed number of clusters but rather to have a sequence of groupings at different levels, with member sets of finer groupings being subsets of coarser groupings, in the same way in which human beings have traditionally thought about biological classification, in which species (such as the domestic cat) contain subspecies or breeds (such as the Persian cat), but are also part of bigger genuses (such as *felis*, small and medium-sized cats), families (such as *felidae*, general cats including lions, panthers etc.), orders (*carnivora* mammals), classes (mammals) etc. Here is the mathematical definition:

Definition 4.1 *A* **hierarchy** *is a sequence of partitions $\mathcal{C} = \bigcup_{j=1}^{m} \mathcal{C}_j$, where $\mathcal{C}_j$, $i = j, \ldots, m$ are partitions with $K_1 = |\mathcal{C}_1| > \ldots > K_m = |\mathcal{C}_m|$ ($|C|$ denoting the number of elements of C) so that for $C_j \in \mathcal{C}_j$ and $C_k \in \mathcal{C}_k$ with $j < k$ either $C_j \cap C_k = C_j$ or $C_j \cap C_k = \emptyset$, so that the sets on lower levels are partitions of the sets of the higher levels.*

Hierarchies can be visualised as trees ("dendrograms"), which show how the clusters in the "finer" lower level partitions are merged in order to arrive at the higher level partitions (whether the finer partitions are called of "lower" or "higher" level is somewhat arbitrary; here we use these terms in agreement with the most widespread convention to draw dendrograms). Figure 4.1 shows a hierarchy of sets and the corresponding dendrogram for a toy dataset.

There are two main types of methods to set up hierarchies:

agglomerative methods start from an initial low level hierarchy in which every observation is a cluster on its own, and proceed by merging clusters upward until everything is merged,

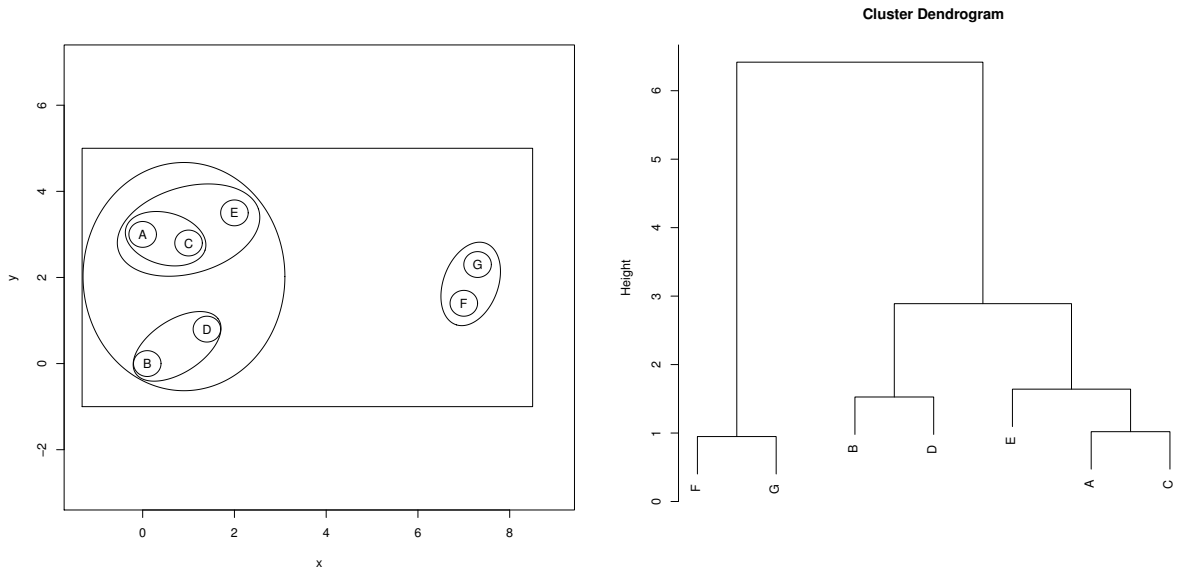

Figure 4.1: Hierarchy of sets and corresponding dendrogram (Average Linkage based on Euclidean distance) on toy dataset.

divisive methods start with the whole dataset as a single cluster and proceed downward by dividing clusters until everything is isolated.

For computational reasons, agglomerative methods are in far more widespread use, and we will only treat agglomerative methods here. The methods `diana` and `mona` in R's `cluster`-package are divisive, see Kaufman and Rousseeuw (1990).

Here is a general algorithm for agglomerative hierarchical clustering (**AAHC**). Required is a dissimilarity between sets, $D : (\mathcal{P}(\mathcal{X}))^2 \mapsto I\!R_0^+$, where $\mathcal{P}(\mathcal{X})$ is the set of subsets of the object set $\mathcal{X}$. Normally, this is based on a dissimilarity d on $\mathcal{X}$, see below, and $D(\{\mathbf{x}_i\}, \{\mathbf{x}_j\}) = d(\mathbf{x}_i\mathbf{x}_j)$ for $i, j \in I\!N_n$.

Initialisation. $k = 1$. Every object is a cluster on its own.

$$\mathcal{C}_1 = \{C_1, \ldots, C_n\} = \{\{\mathbf{x}_1\}, \ldots, \{\mathbf{x}_n\}\}, \; K_1 = n.$$

Step k.1. Find $C_i, C_j \in \mathcal{C}_k$ so that $D(C_i, C_j) = \min_{(C_l, C_m)} D(C_l, C_m)$.

Step k.2. Merge C_i, C_j:

$$\mathcal{C}_{k+1} = \mathcal{C}_k \cup \{C_i \cup C_j\} \setminus \{C_i, C_j\}$$

so that $K_{k+1} = K_k - 1$. Set $H_k = D(C_i, C_j)$, the level (dendrogram height) at which C_i and C_j are merged[1].

[1]It is assumed here that there is a unique pair of clusters C_i, C_j minimising D. Otherwise one would normally merge all candidate pairs at the same time, unless this would lead to merging triplets or bigger collections of clusters, in which case what has to be done depends on the choice of D.)

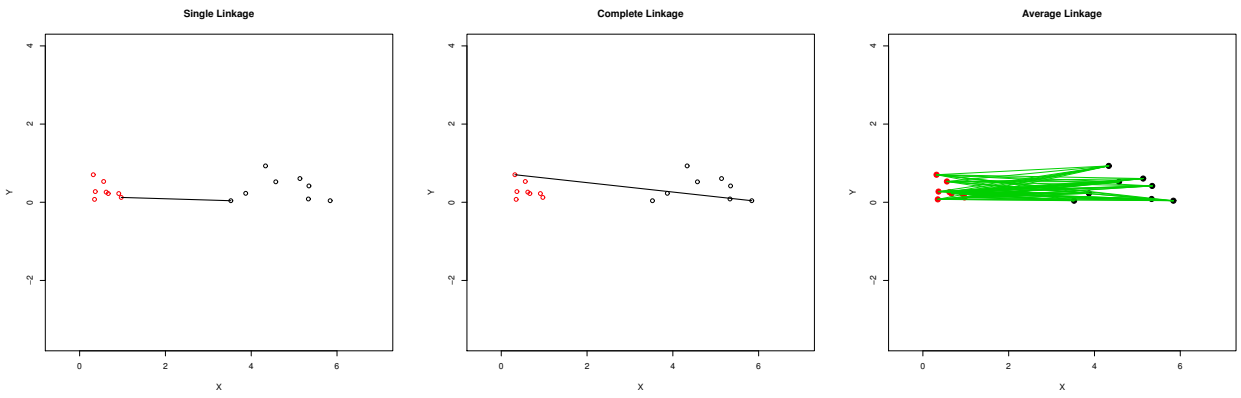

Figure 4.2: Single, Complete and Average Linkage dissimilarity between clusters.

Stop when everything is merged, $K_{k+1} = 1$, $\mathcal{C}_{k+1} = \{\mathcal{X}\}$.
Otherwise, increase k by 1 and go to Step k.1.

What is missing is the definition of D. This can be done in various ways, defining different methods for agglomerative hierarchical clustering.

The merging levels $H_1, \ldots, H_{K-1}$ correspond to the "Height" in Figure 4.1. For most choices of D, the AAHC is monotonic, i.e., $H_1 \leq \ldots \leq H_{K-1}$, which means that smaller clusters are always merged at higher levels than those at which they were produced. This coincides with the intuition that the clustering becomes coarser and coarser, and later stages of the algorithm work at higher levels. If this is not the case (as for the "centroid method" not treated here), drawing the dendrogram becomes less straightforward.

From a complete hierarchy or a dendrogram, a partition with K clusters can always be obtained by using $\mathcal{C}_i$ with $K_i = K$. This amounts to cutting the dendrogram so that there are K clusters. Equivalently, instead of fixing K, one can also fix the height H at which the dendrogram is cut, and choose $\mathcal{C}_i$ so that $H_i \geq H$ but $H_{i-1} \leq H$, which implicitly decides the number of clusters $K = K_i$. In some rare applications something is known about an acceptable dissimilarity within clusters, and in such cases it may be easier to fix H than K. R's function `cutree` cuts a partition out of a dendrogram. It has a parameter `k` for fixing K and `h` for fixing H, only one of which needs to be set.

For datasets with large n, dendrograms cannot be drawn at the lowest levels anymore, and storing all pairwise dissimilarities becomes problematic. Sometimes micro-clusters are constructed in large datasets before running the AAHC with these micro-clusters as objects, see also p.106 of Aggarwal and Reddy (2014) for some hierarchical methods for larger datasets.

4.2 Dissimilarities for clusters and standard methods

Here are three of the four most popular instances of the AAHC (the fourth one is Ward's method, see below).

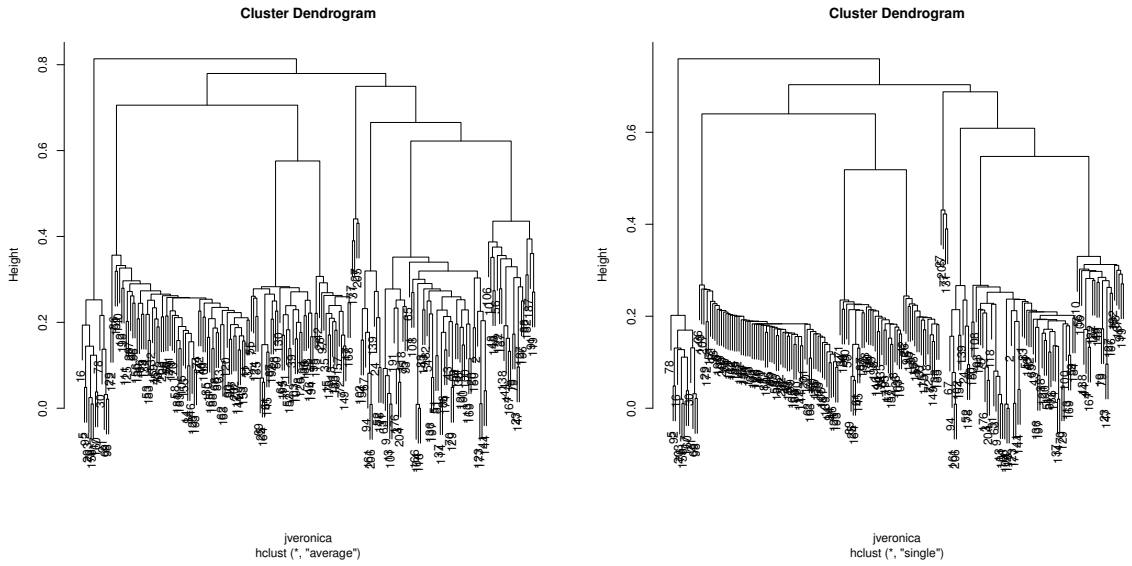

Figure 4.3: Dendrograms for Veronica data with Jaccard distance between individuals, Average Linkage (left) and Single Linkage (right).

Definition 4.2 *The AAHC with*

$$D(B,C) = \min_{\mathbf{x}\in B,\mathbf{y}\in C} d(\mathbf{x},\mathbf{y})$$

is called **Single Linkage** *(or nearest neighbour) clustering.*
The AAHC with

$$D(B,C) = \max_{\mathbf{x}\in B,\mathbf{y}\in C} d(\mathbf{x},\mathbf{y})$$

is called **Complete Linkage** *(or furthest neighbour) clustering.*
The AAHC with

$$D(B,C) = \frac{1}{|B||C|}\sum_{\mathbf{x}\in B,\mathbf{y}\in C} d(\mathbf{x},\mathbf{y})$$

is called **Average Linkage** *(or UPGMA for "Unweighted Pair Group Method with Arithmetic Mean") clustering.*

See Figure 4.2 for an example. Single Linkage was the method introduced by Florek et al. (1951). Although these methods look fairly similar, they can give quite different results in some situations.

Remark 4.3 *For all these methods the AAHC is monotonic. For Single Linkage this is obvious; merging two clusters, say $C_1^* = C_1 \cup C_2$ cannot produce a smaller dissimilarity between a point in C_1^* and any other cluster C_3 than there was at the lower level before between either C_1 or C_2 and C_3.*
Formally: Consider $\mathcal{C}_k$ with $C_1, C_2 \in \mathcal{C}_k$ and $\mathcal{C}_{k+1} = \mathcal{C}_k \cup \{C_1^*\} \setminus \{C_1, C_2\}$. Now

$$H_k = D(C_1,C_2) = \min_{\mathbf{x}\in C_i,\mathbf{y}\in C_j,\ C_i\neq C_j\in\mathcal{C}_k} d(\mathbf{x},\mathbf{y}).$$

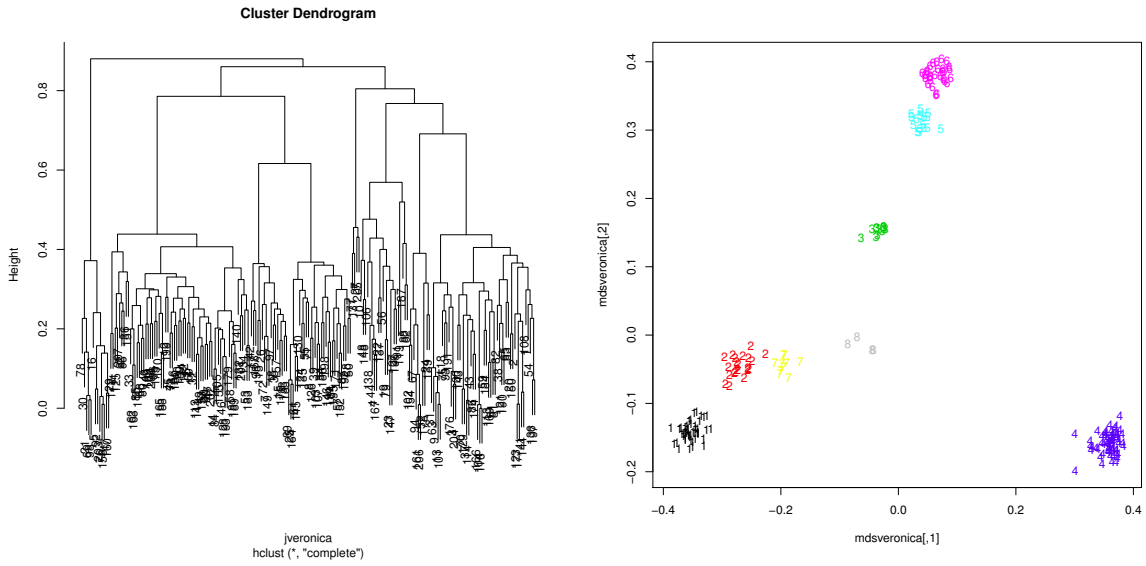

Figure 4.4: Left: Dendrograms for Veronica data with Jaccard distance between individuals, Complete Linkage. Right: MDS plot with 8-cluster solution from any of the Linkage methods

Then for any $C_3 \in \mathcal{C}_k \cap \mathcal{C}_{k+1}$, $D(C_1^*, C_3)$ is either equal to $D(C_1, C_3)$ or to $D(C_2, C_3)$ (because the minimising $\mathbf{x} \in C_1^*$ is either from C_1 or from C_2), and because $D(C_1, C_2)$ was the minimum of D within $\mathcal{C}_k$ and all clusters from $\mathcal{C}_{k+1}$ apart from C_1^* are also in $\mathcal{C}_k$, the minimum $D(C_i, C_j)$, $C_i, C_j \in \mathcal{C}_{k+1}$, i.e., H_{k+1}, must be $\geq D(C_1, C_2) = H_k$.

For Complete Linkage the argument works in the same way, again realising that the maximum $d(\mathbf{x}, \mathbf{y})$ for $\mathbf{x} \in C_1^*, \mathbf{y} \in C_3$ requires $\mathbf{x}$ to be from either C_1 or C_2, and therefore the corresponding $D(C_1^*, C_3)$ and actually all $D(C_i, C_j)$, $C_i, C_j \in \mathcal{C}_{k+1}$ had been in the competition at the lower level already.

For Average Linkage, the argument is different and is an exercise.

Remark 4.4 *In Complete Linkage, in $\mathcal{C}_k$, for all within-cluster dissimilarities there is $d(\mathbf{x}, \mathbf{y}) \leq H_{k-1}$. This can be useful in applications in which a clustering is required that cannot tolerate within-cluster dissimilarities above a certain upper bound. Such a clustering can be found by cutting the Complete Linkage at the corresponding height. There is therefore a "within-cluster homogeneity guarantee" in Complete Linkage.*

Proof *by complete induction: $\mathcal{C}_2$ is the first clustering to have two objects in the same cluster, say w.l.o.g. $C_1 = \{\mathbf{x}_1, \mathbf{x}_2\}$, and $H_1 = d(\mathbf{x}_1, \mathbf{x}_2)$ by definition. Now assume that for $\mathcal{C}_k$ all within-cluster dissimilarities are $\leq H_{k-1}$. As above, assume $\mathcal{C}_{k+1} = \mathcal{C}_k \cup \{C_1^*\} \setminus \{C_1, C_2\}$, with $H_k = D(C_1, C_2)$. This means that the maximum within-cluster dissimilarity in C_1^* is H_k, and all other clusters in $\mathcal{C}_{k+1}$ were in $\mathcal{C}_k$ already and therefore their maximum within-cluster dissimilarities are all $\leq H_{k-1} \leq H_k$ because of the monotonicity of Complete Linkage.*

Remark 4.5 *In Single Linkage, in $\mathcal{C}_k$, for all between-cluster dissimilarities there is $d(\mathbf{x}, \mathbf{y}) \geq H_{k-1}$. This is because pairs of clusters C_1, C_2 that have $d(\mathbf{x}, \mathbf{y}) < H_{k-1}$, $\mathbf{x} \in C_1, \mathbf{y} \in C_2$, must have been merged before stage k of the AAHC. There is therefore a "between-cluster separation guarantee" in Simple Linkage.*

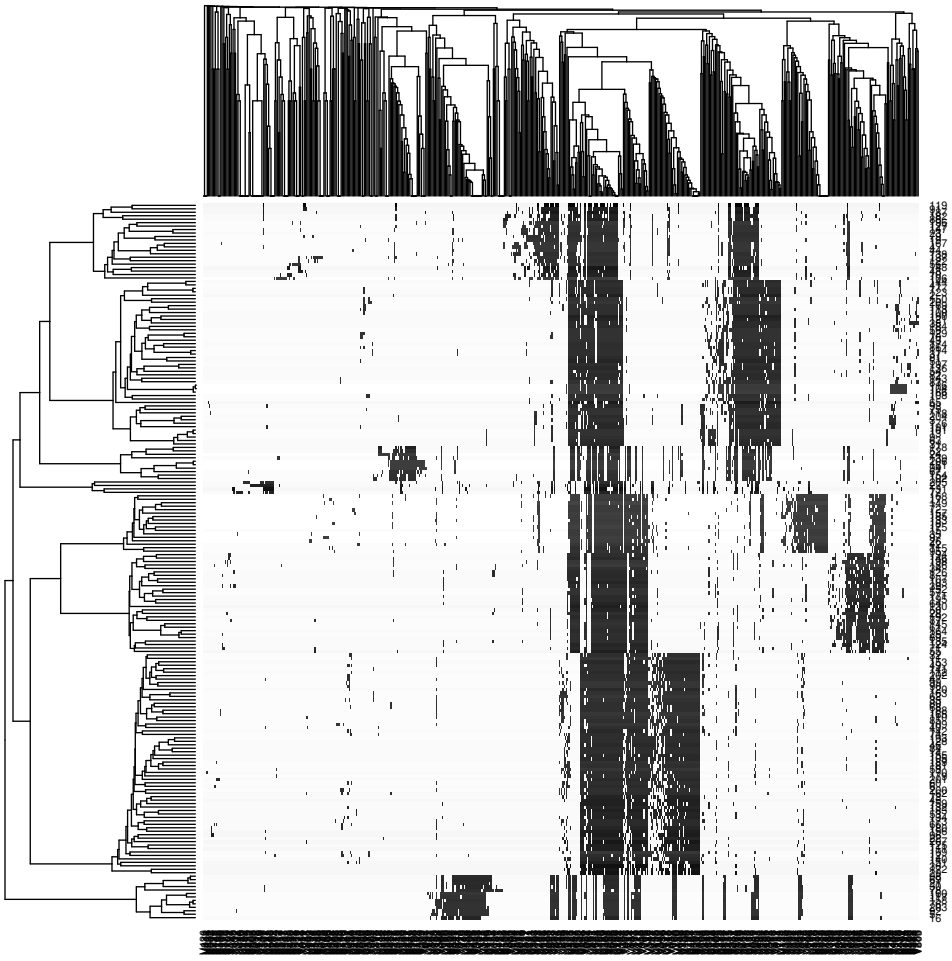

Figure 4.5: Heatmap for Veronica data with row and column order determined by Average Linkage clustering.

Looking at these results from a different angle, Complete Linkage enforces within-cluster homogeneity disregarding separation (Complete Linkage clusters are often not properly separated at all), and Single Linkage enforces between-cluster separation disregarding homogeneity (Single Linkage clusters can be extremely heterogeneous). Note that these statements are to be understood for cuts at fixed nontrivial numbers of clusters K; obviously, at the lowest and at the top level, both methods have the same clusterings.

Average Linkage does not come with straightforward guarantees like the above. It is a compromise between Single and Complete Linkage, between within-cluster homogeneity and between-cluster separation, and as such in practice often the preferred choice.

4.3 Examples

In R, hierarchical clustering is run using the `hclust`-function, which has a `method`-argument that can take (among others) the values `"average"`,`"single"` or `"complete"`.

The first example is the Veronica dataset with the Jaccard distance. An informal way of finding a good level at which to cut the clustering is to look at whether the appearance of the dendrogram "suggests" a value, e.g., by having a big interval between two heights. A formal method will be introduced in Chapter 5.

```
# Average Linkage
plantclust <- hclust(jveronica,method="average")

# Plot the dendrogram:
plot(plantclust)

# 8 looks like a good number of clusters.
plantclust8 <- cutree(plantclust,8)

# Visualisation using MDS:
plot(mdsveronica,col=plantclust8,pch=clusym[plantclust8])
# This looks good.

# Single Linkage
plantclusts <- hclust(jveronica,method="single")
plot(plantclusts) # 8 looks still OK
plantclusts8 <- cutree(plantclusts,8)
plot(mdsveronica,col=plantclusts8,pch=clusym[plantclusts8])
# The same as before.

# Complete Linkage
plantclustc <- hclust(jveronica,method="complete")
plot(plantclustc) # Could use 8 or 9.
plantclustc8 <- cutree(plantclustc,8)
plot(mdsveronica,col=plantclustc8,pch=clusym[plantclustc8])
# The same as before.
```

The 8-cluster solutions are identical here, which suggests a very strong clustering. The dendrograms can be seen in Figures 4.3 and 4.4. The clustering process below the 8-cluster level differs somewhat between the methods (with Complete Linkage, $K = 8$ is not a very obvious choice), but overall the methods are very similar here. Figure 4.4 also shows the MDS-plot with the 8-cluster solution, which looks very convincing.

One application of the dendrogram is that it can be used for ordering the rows and columns in the heatmap for visualisation. Setting up a dendrogram requires to order the points in such a way that clusters should be neighbours to each other at the stage at which they are merged.

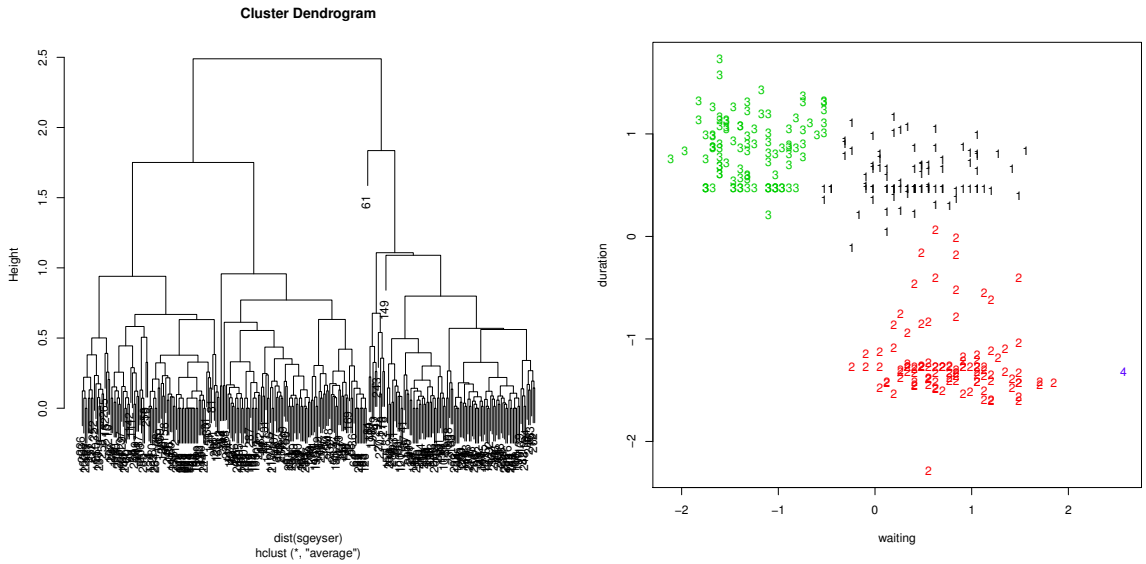

Figure 4.6: Average Linkage dendrogram and 4-cluster solution for Old Faithful Geyser data.

This order holds clustering information, although it is not uniquely defined (rotating within clusters produces different orders that could be used for dendrograms of the same hierarchy). It can be used as follows:

```
# I decided to use the Jaccard distance also for AFLP bands/genes:
vveronica <- dist(t(veronicam),method="binary")
varclust <- hclust(vveronica,method="average")
# As a clustering this is pretty messy, but still it
# can be used to impose an order of genes.

# This submits the two dendrograms to the heatmap:
heatmap(veronicam,Rowv=as.dendrogram(plantclust),
        Colv=as.dendrogram(varclust),
        col=grey(seq(1,0,-0.01)))
```

The result is in Figure 4.5.

The Geyser dataset is an example in which the different AAHC methods give quite different results.

```
# Average Linkage
geyclust <- hclust(dist(sgeyser),method="average")
plot(geyclust)
# 4 looks like a good K, will isolate one outlier
geyclust4 <- cutree(geyclust,4)
plot(sgeyser,col=geyclust4,pch=clusym[geyclust4])
```

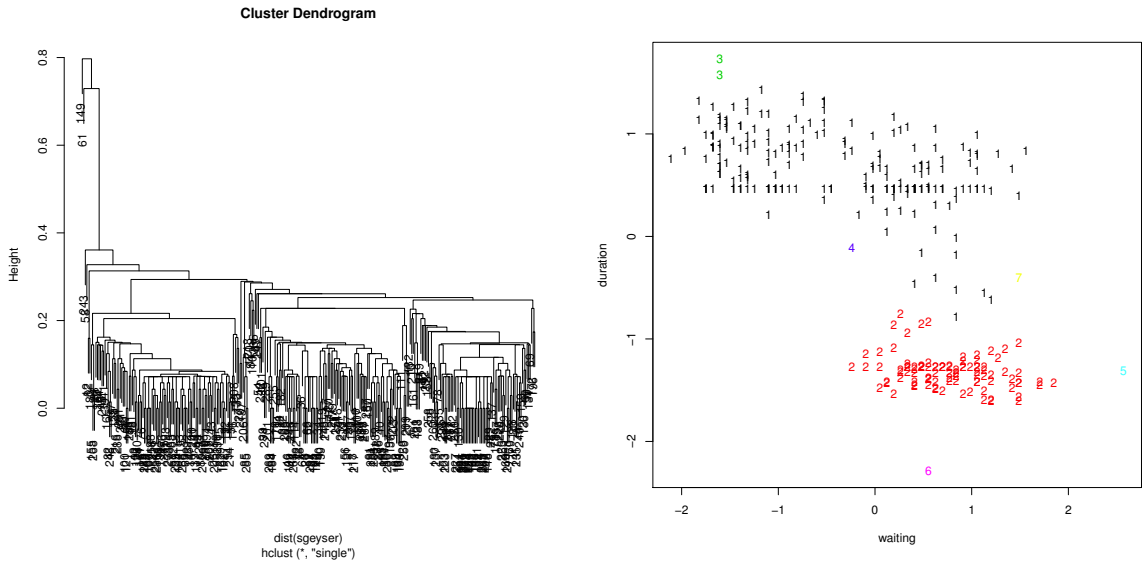

Figure 4.7: Single Linkage dendrogram and 7-cluster solution for Old Faithful Geyser data.

```
# Single Linkage
geyclusts <- hclust(dist(sgeyser),method="single")
plot(geyclusts)
# 7 looks like a good K
geyclusts7 <- cutree(geyclusts,7)
plot(sgeyser,col=geyclusts7,pch=clusym[geyclusts7])

# Complete Linkage
geyclustc <- hclust(dist(sgeyser),method="complete")
plot(geyclustc)
# 5 looks like a good K
geyclustc5 <- cutree(geyclustc,5)
plot(sgeyser,col=geyclustc5,pch=clusym[geyclustc5])
```

The results can be seen in Figures 4.6, 4.7 and 4.8. For numbers of cluster smaller than 7, Single Linkage only isolates outliers by producing one- and two-point clusters. The main bulk of the data is a single very heterogeneous cluster (but separated from the outliers). One needs to use $K = 7$ or larger to still have a split within the main bulk of the points. This happens very often with Single Linkage (putting quite heterogeneous clusters together because there are some connecting points is often called "chaining" in the literature, and Single Linkage is prone to this).

Apart from the somewhat odd decision to isolate one outlier with a large waiting time but not the other one with the shortest duration, Average Linkage looks more or less fine.

The Complete Linkage dendrogram does not give a particularly strong indication about the number of clusters. Complete Linkage will not isolate outliers as long as this requires to put together parts of the dataset that are more heterogeneous (in terms of the maximum distance)

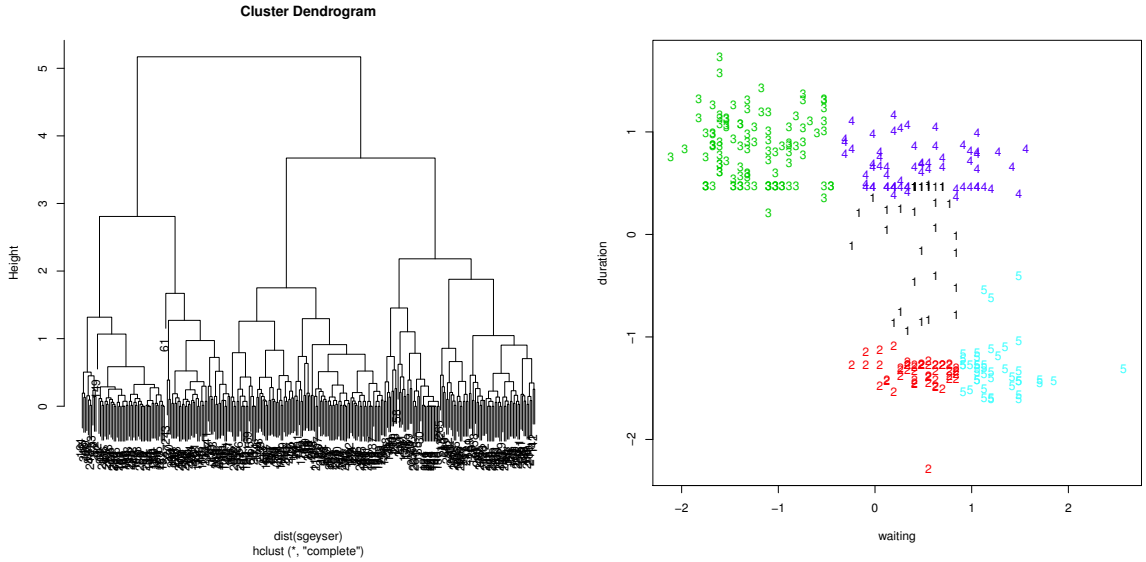

Figure 4.8: Single Linkage dendrogram and 5-cluster solution for Old Faithful Geyser data.

than what is obtained by merging the outliers with some parts of the dataset.

The 5-cluster solution illustrates well that Complete Linkage tries to keep maximum distances down in all clusters at the same time. For the 4-cluster solution, Complete Linkage will merge clusters 1 and 4 in Figure 4.8, and for the 3-cluster solution, clusters 2 and 5 will be merged, which seems like a fairly good solution (the integrated outliers stop clusters 2 and 5 from merging earlier, or with cluster 1).

An issue with the AAHC in general is that points once merged can never be separated again at a later stage, which may lead to trouble if the global structure at a higher level is not really consistent with what looked like the best local structure at a lower level to the algorithm.

4.4 Ward's method

A different way of using the AAHC is Ward's method (Ward (1963)). Ward proposed to use an objective function for agglomerative hierarchical clustering, so that in each step clusters are merged in order to give the best possible value of the objective function. What is today known as Ward's method is the use of the K-means criterion $S(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K)$ from Definition 2.2 as objective function.

This amounts to running the AAHC with

$$D(B, C) = D_k(B, C) = S(\mathcal{C}^*, \mathbf{m}_1^*, \ldots, \mathbf{m}_{K_{k+1}}^*),$$

where D_k depends on the step k and the clustering $\mathcal{C}_k$ in which B and C occur, $\mathcal{C}^*$ is produced from $\mathcal{C}_k$ by merging B and C, and $\mathbf{m}_1^*, \ldots, \mathbf{m}_{K_{k+1}}^*$ are the cluster means of the clusters in $\mathcal{C}^*$.

Ward's method is therefore a hierarchical version of K-means. It produces a local optimum (among the clusterings that can be constructed by merging clusters of the previous level) of S at each level of the hierarchy. A disadvantage of Ward's method is that the value of S

for given K achieved is in many cases not as good as what can be achieved by running the K-means algorithm several times. On the other hand there are two advantages. Firstly, the algorithm produced a hierarchy (in case that this is required). Secondly, the algorithm works in a deterministic way and is not dependent on random initialisation. Results will reproduce without fixing the random numbers. One could use Ward's method to generate a deterministic starting point for the K-means algorithm, although this does not usually improve on Ward's solution substantially.

In the given form, Ward's method cannot be computed for general dissimilarities; as K-means it relies on the Euclidean space and observations characterised by variables, so that means can be computed. However, it can be shown that for a given clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ with means $\mathbf{m}_1, \ldots, \mathbf{m}_K$:

$$S(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K) = \sum_{k=1}^{K} \frac{1}{2|C_k|} \sum_{\mathbf{x}_i, \mathbf{x}_j \in C_k} d_{L2}^2(\mathbf{x}_i, \mathbf{x}_j). \tag{4.1}$$

This means that the K-means criterion can be computed from Euclidean dissimilarities without knowing the variables. The R-function `hclust` with `method="ward.D2"` computes Ward's AAHC from dissimilarities. These have to be Euclidean distances D_{L2} to compute what is normally referred to as Ward's method, but one can also use other more general dissimilarity measures d; the method then uses the right side of (4.1) with d instead of d_{L2} as D. The R-function `agnes` in package `cluster` computes Ward's method from n data points with p variables with `method="ward"`. Note that all these implementations compute D_k in a computationally simplified but equivalent way.

For the Bundestag data:

```
# kmeans with 5 clusters
set.seed(12345)
kmbundestag5 <- kmeans(p05,5,nstart=100)

# Ward's method
wbundestag <- hclust(dist(p05),method="ward.D2")
# plot(wbundestag) # Dendrogram not shown
# Same as wbundestag <- agnes(p05,method="ward")

# With K=5:
wbundestag5 <- cutree(wbundestag,5)

table(kmbundestag5$cluster,wbundestag5)
#   wbundestag5
#     1  2  3  4  5
#  1 27  0  0 55  0
#  2 72  0  0  0  0
#  3  0  0  0 16 22
#  4  8  0 36  0  0
#  5  0 63  0  0  0
```

```
# Fairly different.

adjustedRandIndex(kmbundestag5$cluster,wbundestag5)
# [1] 0.6362054

library(fpc)
# This can be used to compute S for any clustering:
kmb <- cluster.stats(dist(p05),kmbundestag5$cluster)
kmb$within.cluster.ss
# 1.319956
# This is the same as kmbundestag5$tot.withinss
wmb <- cluster.stats(dist(p05),wbundestag5)
wmb$within.cluster.ss
# 1.534854
# Quite a bit worse.
```

Chapter 5

PAM and ASW - further methods for dissimilarities

5.1 Partitioning Around Medoids

Partitioning Around Medoids (PAM) (Kaufman and Rousseeuw (1990)) is a clustering method for dissimilarity data that has a strong connection to K-means. As K-means, PAM aims at finding centroid objects for each of a given fixed number of K clusters. These centroid objects cannot be mean vectors, because the computation of mean vectors requires Euclidean space and cannot be done for general dissimilarity data. Instead, for PAM, the centroid objects ("medoids") are members of the dataset.

Definition 5.1 *Let $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\}$ with objects $\mathbf{x}_i$ from some set $\mathcal{X}$ characterised by a dissimilarity measure $d: \mathcal{X}^2 \mapsto \mathbb{R}_0^+$.*

The **Partitioning Around Medoids (PAM) clustering** *of $\mathcal{D}$ for given fixed K is defined by choosing $\mathbf{m}_1^{PK}, \ldots, \mathbf{m}_K^{PK} \in \mathcal{D}$ and $c^{PK}(1), \ldots, c^{PK}(n)$ in such a way that they minimise*

$$T(\mathcal{C}, \mathbf{m}_1, \ldots, \mathbf{m}_K) = \sum_{i=1}^{n} d(\mathbf{x}_i, \mathbf{m}_{c(i)}).$$

This is obviously the very same idea as in Definition 2.2.

Compared with K-means, obviously PAM is more flexible. Particularly, PAM can also be run for Euclidean data, using the squared Euclidean distance (as K-means) but also the unsquared Euclidean distance, the Manhattan distance or the Mahalanobis distance. When run with the squared Euclidean distance, it is worse than K-means, because mean vectors are better in terms of minimising T which then becomes S from Definition 2.2 than data objects.

Using the unsquared Euclidean or Manhattan distance, PAM has the advantage of not penalising large within-cluster distances as strongly as K-means, so that it is somewhat better at finding non-spherical clusters, as long as the deviation from a spherical shape is not all too strong. This can also incur a problem, though, if large distances are too easily integrated into a cluster, see the Veronica-example in Section 5.5.

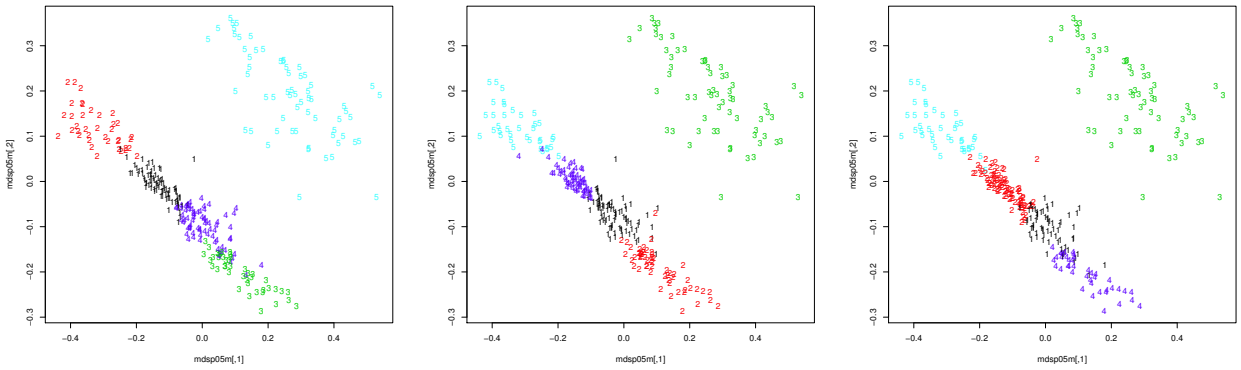

Figure 5.1: MDS based on Manhattan distance for Bundestag data with 5-means clustering (right), PAM-clustering with Euclidean distances (middle), PAM-clustering with Manhattan-distances (left).

5.2 Computation of PAM

In principle, PAM could be computed using the same idea as in Section 2.2, but Kaufman and Rousseeuw (1990) proposed a different algorithm, which is deterministic and will usually get closer to the global optimum than the algorithm in Section 2.2. It is much slower, though.

Build phase Step 0 Start with $k = 1$.

Step 1 For fixed $(\mathbf{m}_1^{Ik}, \ldots, \mathbf{m}_{k-1}^{Ik}) = (\mathbf{m}_1^{I(k-1)}, \ldots, \mathbf{m}_{k-1}^{I(k-1)})$, choose

$$\mathbf{m}_k^{Ik} = \arg\min_{\mathbf{m}_k \in \mathcal{D}} \sum_{i=1}^{n} d(\mathbf{x}_i - \mathbf{m}_{c^k(i)}),$$

where $c^k(i) = \arg\min_{j \in I\!N_k} d(\mathbf{x}_i - \mathbf{m}_j^{Ik})$.

End if $k = K$, otherwise $k = k + 1$, go to Step 1.

Swap phase Step 0 $T^{(0)} = T((c^k(1), \ldots, c^k(n)), (\mathbf{m}_1^*, \ldots, \mathbf{m}_K^*) = (\mathbf{m}_1^{IK}, \ldots, \mathbf{m}_K^{IK}), q = 1$.

Step 1 For all pairs (i, k) so that $\mathbf{x}_i \notin \{\mathbf{m}_1^*, \ldots, \mathbf{m}_K^*\}$ and $\mathbf{m}_k^* \in \{\mathbf{m}_1^*, \ldots, \mathbf{m}_K^*\}$ compute $T_{ik} = T(\mathcal{C}^{ik}, \mathbf{m}_1^{ik}, \ldots, \mathbf{m}_K^{ik})$, where $(\mathbf{m}_1^{ik}, \ldots, \mathbf{m}_K^{ik})$ are $(\mathbf{m}_1^*, \ldots, \mathbf{m}_K^*)$ but with $\mathbf{m}_k^*$ replaced by $\mathbf{x}_i$, and $\mathcal{C}^{ik}$ assigns every object to the closest centroid in $\{\mathbf{m}_1^{ik}, \ldots, \mathbf{m}_K^{ik}\}$.

Step 2 $(j, l) = \arg\min_{(i,k)} T_{ik}$, $T^{(q)} = T_{jl}$.

End if $T^{(q)} \geq T^{(q-1)}$. Otherwise $(\mathbf{m}_1^*, \ldots, \mathbf{m}_K^*) = (\mathbf{m}_1^{jl}, \ldots, \mathbf{m}_K^{jl})$, $q = q + 1$, Step 1.

PAM is fitted by the R-function `pam` in package `cluster`, see below. The algorithm is too slow for bigger datasets, and also, generally, the size of dissimilarity objects or matrices grows quadratically with n so that any dissimilarity based algorithm has a hard time for large n ($n > 4000$, say).

For larger datasets, there is a version `clara` ("clustering large applications") of `pam`, which works by applying PAM to several small enough subsets of the dataset and classifying all further points to their closest centroids, picking the best solution among these according to T. `clara` works for Euclidean data only, in which case it is not necessary to store all dissimilarities.

5.3 Example for PAM

PAM is fitted by the R-function `pam` in package `cluster`, which implements the methods from the book Kaufman and Rousseeuw (1990). Here I just quickly show its basic use with the Bundestag data; there will be more in Section 5.5.

```
library(cluster)
bundestagk5 <- kmeans(p05,5,nstart=100)

# By default, if pam is called with a dataset that is not a
# dist-object, the Euclidean distance is used:
bundestagp5 <- pam(p05,5)

# There's a bit of a difference between k-means and pam
# based on Euclidean distances
adjustedRandIndex(bundestagk5$cluster,bundestagp5$cluster)
[1] 0.7249983

# Alternatively, pam can be started with a dist-object, which allows
# computing it with the Manhattan-distance:
p05manhattan <- dist(p05,method="manhattan")
bundestagp5m <- pam(p05manhattan,5)

# Somewhat surprisingly, with Manhattan distance the clustering
# is more similar to K-means than to PAM with Euclidean
# distance. This is atypical.
adjustedRandIndex(bundestagp5$cluster,bundestagp5m$cluster)
[1] 0.6607979

adjustedRandIndex(bundestagk5$cluster,bundestagp5m$cluster)
[1] 0.8863854

# One can visualise this with an MDS-plot based on the
# Manhattan distance.
mdsp05m <- cmdscale(p05manhattan,2)

plot(mdsp05m,col=bundestagk5$cluster,pch=clusym[bundestagk5$cluster])
plot(mdsp05m,col=bundestagp5$cluster,pch=clusym[bundestagp5$cluster])
plot(mdsp05m,col=bundestagp5m$cluster,pch=clusym[bundestagp5m$cluster])
```

The plots can be seen in Figure 5.1. Note that the Manhattan distance-based clustering has a somewhat better chance of looking well, because the MDS is based on Manhattan distances. In any case, the methods agree on separating the constituencies in which the LINKE party is strong (mostly from the former communist GDR), and how to cluster the rest of the data is rather ambiguous anyway.

5.4 Average Silhouette Width

For estimating the number of clusters K using PAM, Kaufman and Rousseeuw (1990) recommend the Average Silhouette Width (ASW). The ASW is less strongly connected to PAM than the Sugar & James-index is to K-means, though. The Sugar & James-index is based on direct investigation of the K-means objective function S, whereas the ASW has its own rationale that is only loosely connected to PAM. It is not based on centroids and can rather be seen as a general purpose index for dissimilarities that can also be used to choose K for cutting hierarchical dendrograms.

Definition 5.2 *Let $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\}$ with objects $\mathbf{x}_i$ from some set $\mathcal{X}$ characterised by a dissimilarity measure $d : \mathcal{X}^2 \mapsto \mathbb{R}_0^+$. For a given clustering $\mathcal{C}_K$ characterised by labels $c(1), \ldots, c(n) \in \mathbb{N}_K$, with $n_k = \sum_{i=1}^n 1(c(i) = k)$, $k \in \mathbb{N}_K$ as usually, the* **silhouette width** *of $\mathbf{x}_i$, $i \in \mathbb{N}_n$, is*

$$s(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))}, \text{ where}$$
$$a(i) = \frac{1}{n_{c(i)}-1} \sum_{c(j)=c(i),\ j\neq i} d(\mathbf{x}_i, \mathbf{x}_j), \ b(i) = \min_{k\neq c(i)} \frac{1}{n_k} \sum_{c(j)=k} d(\mathbf{x}_i, \mathbf{x}_j).$$

In words, this is the difference between $b(i)$, the average dissimilarity of $\mathbf{x}_i$ and all the points in the closest cluster to which $\mathbf{x}_i$ does not belong, and $a(i)$, the average dissimilarity of $\mathbf{x}_i$ and all the points of the cluster $c(i)$ to which it belongs, standardised by the maximum of the two. The latter implies $-1 \leq s(i) \leq 1$. The idea is that if this is positive and big, $\mathbf{x}_i$ sits very well and clearly within its own cluster, whereas if this is very small or negative, $\mathbf{x}_i$ should rather be in another cluster, or at least its cluster membership is not very clear. The ASW averages this over all points.

Definition 5.3

$$ASW(\mathcal{C}_K) = \frac{1}{n} \sum_{i=1}^{n} s(i)$$

is called **Average Silhouette Width**. *The* **ASW-estimate of** K *(assuming that $2 \leq K \leq k_{max}$ for some pre-chosen k_{max}) is*

$$K_{ASW} = \underset{k=2,\ldots,k_{max}}{\arg\max}\ ASW(\mathcal{C}_k).$$

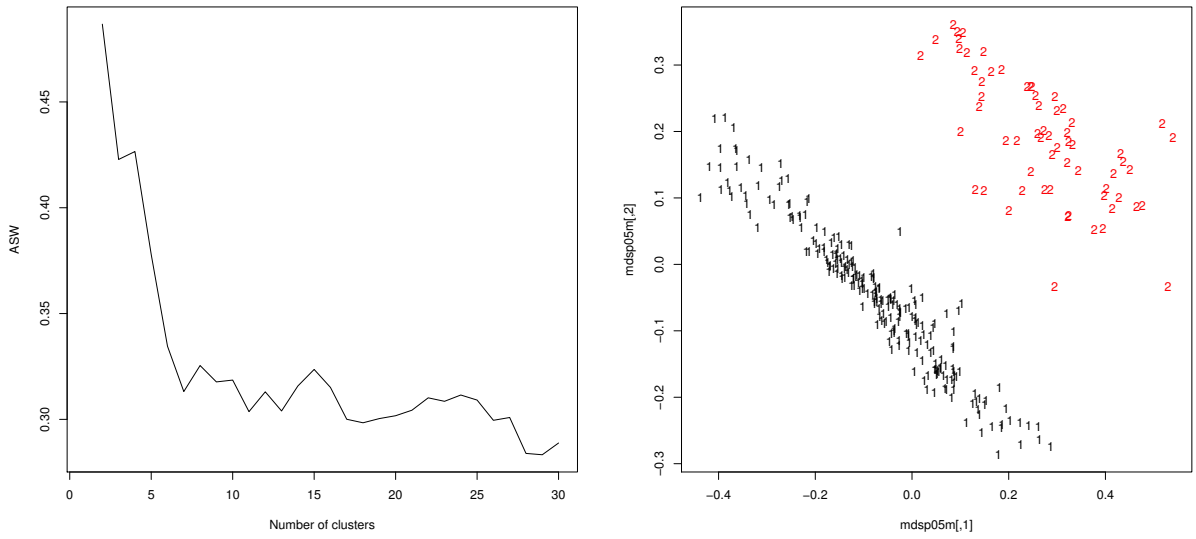

Figure 5.2: Left: ASW-values for Bundestag-data and PAM-clustering with Manhattan-distance. Right: MDS based on Manhattan distance and ASW-optimal PAM-clustering with $K = 2$.

The definition of K_{ASW} maximising the ASW is heuristic; there is no theory confirming under which circumstances this really yields a good K; the rationale rather is that if the ASW is accepted as a good measurement of the quality of a clustering, maximising it will give the best one.

The ASW cannot be computed for a homogeneous dataset, i.e., a "clustering" with $K = 1$, because $b(i)$ is not defined if there is no cluster to which an observation does not belong. Hennig and Lin (2015) argue that one can test whether $K = 1$ by simulating data from a null model for which $K = 1$, and comparing the distribution of the ASW for larger K for such data with the ASW observed in the dataset to be investigated. This can also be used to investigate how the distribution of the ASW changes over various values of K.

5.5 Examples for the ASW

The ASW is computed by function `silhouette` in library `cluster`. This function can also be used to produce a diagnostic plot of the silhouettes $s(i)$. As first example, estimate K for the Bundestag data with Manhattan distance (the Sugar & James-index and K-means produced too many clusters here for reasonable interpretation, and this was also unstable).

```
pasw <- NA
pclusk <- list()
psil <- list()

# Look at K between 2 and 30:
```

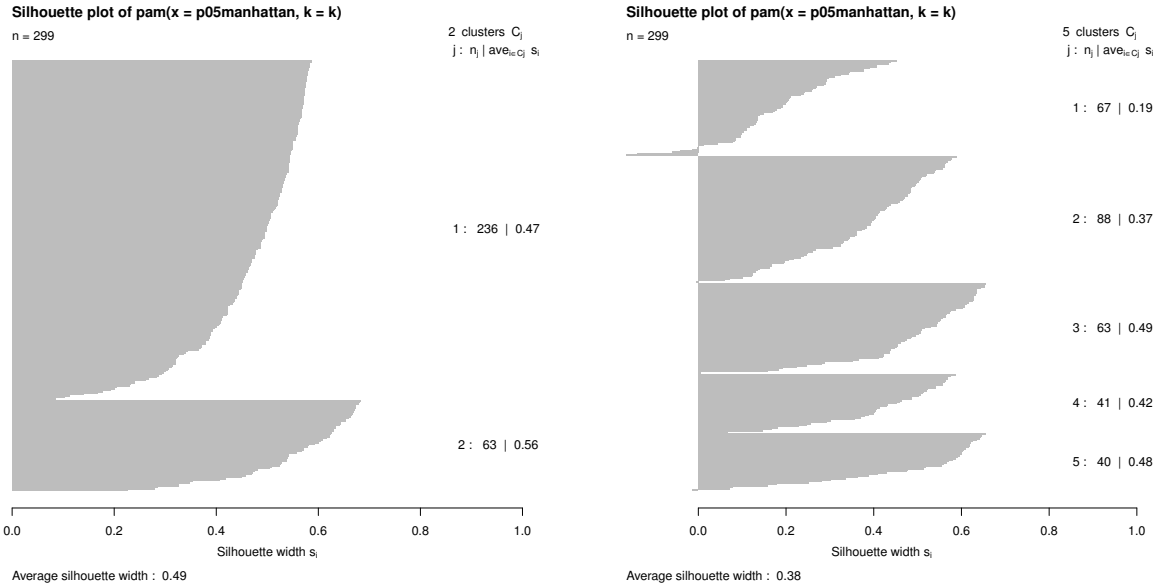

Figure 5.3: Silhouette plots for $K = 2$ and $K = 5$ for Bundestag-data and PAM-clustering with Manhattan-distance. The grey bars correspond to the values $s(i)$ ordered by clusters and by size of $s(i)$ within clusters.

```
for (k in 2:30){
  # PAM clustering:
  pclusk[[k]] <- pam(p05manhattan,k)
  # Computation of silhouettes:
  psil[[k]] <- silhouette(pclusk[[k]],dist=p05manhattan)
  # The silhouette function produces a lot of information,
  # from which the ASW needs to be extracted.
  pasw[k] <- summary(psil[[k]])$avg.width
}

# Plot the ASW-values against K:
plot(1:30,pasw,type="l",xlab="Number of clusters",ylab="ASW")

# Result in numbers:
# > pasw
# [1]        NA 0.4867459 0.4228157 0.4265560 0.3779154 0.3345530
# 0.3131574
#  [8] 0.3254449 0.3177023 0.3185734 0.3036558 0.3130183 0.3039691
# 0.3157744
# [15] 0.3235948 0.3151002 0.3000481 0.2983881 0.3003727 0.3017265
# 0.3043202
# [22] 0.3101771 0.3085357 0.3115082 0.3091192 0.2995540 0.3008825
# 0.2839130
# [29] 0.2832789 0.2888274
```

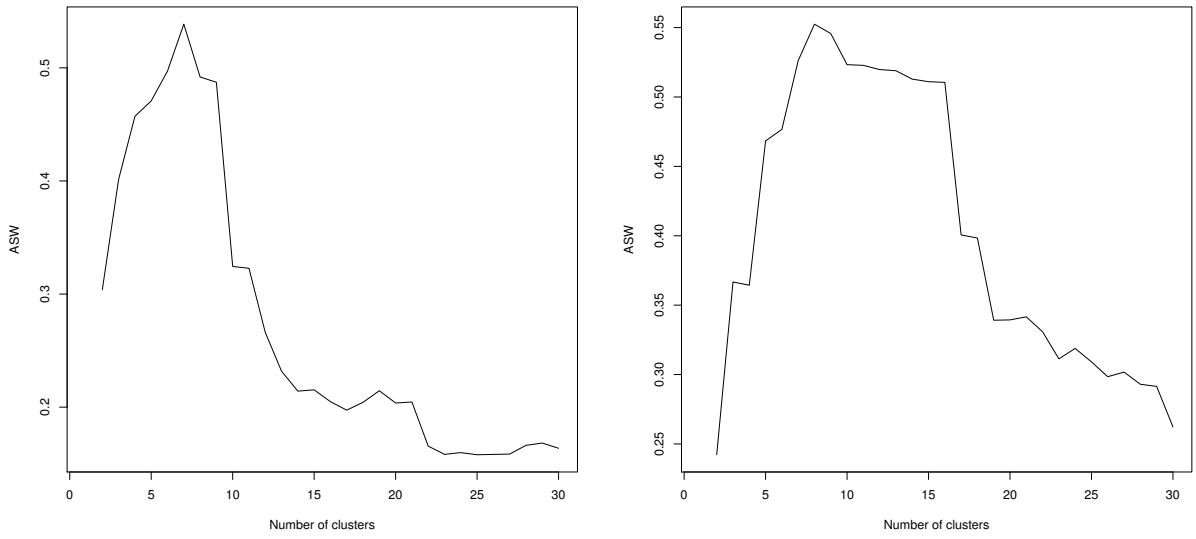

Figure 5.4: ASW-values for Veronica data with Jaccard-distance, PAM (left) and Average Linkage (right).

```
# Best value at K=2.

# MDS-plot:
plot(mdsp05m,col=pclusk[[2]]$cluster,pch=clusym[pclusk[[2]]$cluster])

# Silhouette plots for 2 and 5 clusters:
plot(psil[[2]])
plot(psil[[5]])
```

The results are in Figure 5.2 with Silhouette plots in Figure 5.3. Although $K_{ASW} = 2$ at the lower end of the value range for K should make the researcher wonder whether the dataset is clustered at all, the MDS-plot shows a clear and intuitive clustering at $K = 2$, namely the one separating the LINKE strongholds.

For the Veronica dataset, the ASW can be computed for both hierarchical clustering (Average Linkage is chosen here) and PAM.

```
# Jaccard-distance:
jveronica <- dist(veronica,method="binary")

# PAM:
pasw <- NA
pclusk <- list()
psil <- list()
for (k in 2:30){
  pclusk[[k]] <- pam(jveronica,k)
```

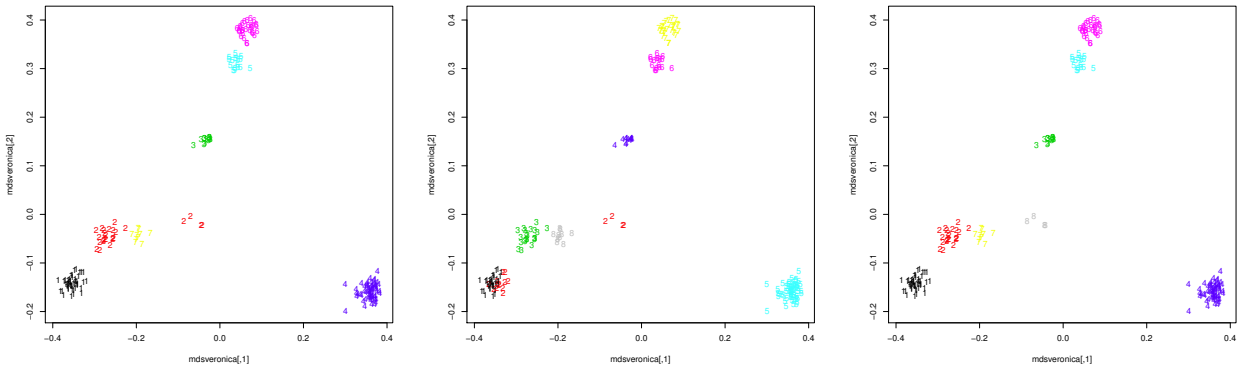

Figure 5.5: MDS-plots for Veronica data with Jaccard-distance, PAM, $K = 7$ (left), PAM, $K = 8$ (middle) and Average Linkage, $K = 8$ (right).

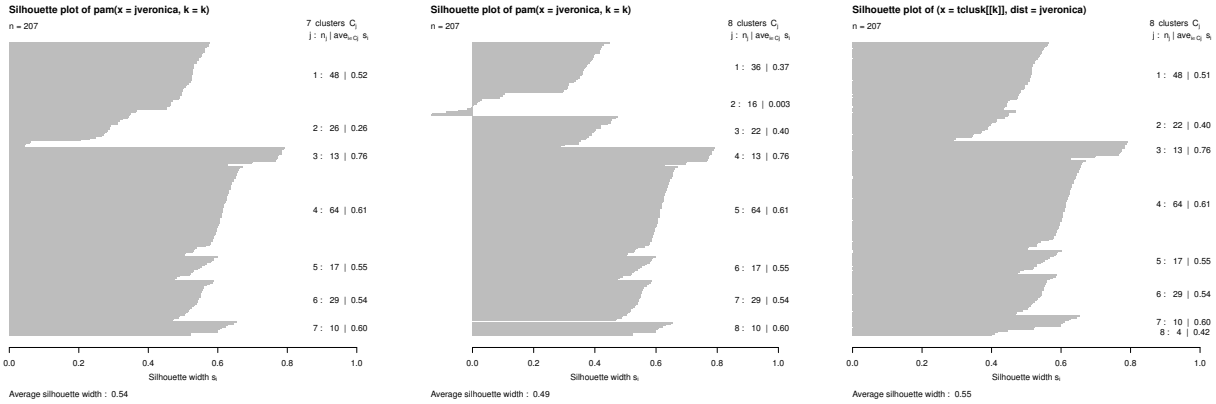

Figure 5.6: Silhouette-plots for Veronica data with Jaccard-distance, PAM, $K = 7$ (left), PAM, $K = 8$ (middle) and Average Linkage, $K = 8$ (right).

```
  psil[[k]] <- silhouette(pclusk[[k]])
  pasw[k] <- summary(psil[[k]])$avg.width
}
plot(1:30,pasw,type="l",xlab="Number of clusters",ylab="ASW")
pasw
# (gives the numbers; there's a maximum at K=7,
# ASW=0.5386146

# Average Linkage
plantclust <- hclust(jveronica,method="average")

tasw <- NA
tclusk <- list()
tsil <- list()
for (k in 2:30){
```

```
  tclusk[[k]] <- cutree(plantclust,k)
  tsil[[k]] <- silhouette(tclusk[[k]],dist=jveronica)
  tasw[k] <- summary(silhouette(tclusk[[k]],dist=jveronica))$avg.width
}
plot(1:30,tasw,type="l",xlab="Number of clusters",ylab="ASW")
tasw
# The maximum is at K=8, ASW=0.5524769

# The Average Linkage clustering at K=8 has a slightly better
# ASW. Actually it looks much better:
mdsveronica <- cmdscale(jveronica,k=2)
# PAM with K=7:
plot(mdsveronica,pch=clusym[pclusk[[7]]$cluster],col=pclusk[[7]]$cluster)
# PAM with K=8:
plot(mdsveronica,pch=clusym[pclusk[[8]]$cluster],col=pclusk[[8]]$cluster)
# Average Linkage with K=8:
plot(mdsveronica,pch=clusym[tclusk[[8]]],col=tclusk[[8]])
```

The results are in Figures 5.4 and 5.5. The Average Linkage results seems fine here, whereas PAM does not handle the smallest cluster appropriately; the PAM-criterion T for $K = 8$ gives a better value of T splitting up what is cluster no. 1 in the Average Linkage clustering, but this does not look convincing, and the ASW-value is worse than the value for $K = 7$.

The Silhouette-plots (Figure 5.6) show that cluster 2 for PAM with $K = 8$ looks bad. For $K = 7$ it looks slightly better but one could suspect that cluster 2 has two subclusters. For the Average Linkage-clustering, all clusters look fine:

```
plot(psil[[7]])
plot(psil[[8]])
plot(tsil[[8]])
```

Chapter 6

Mixture models

6.1 Basic concepts

Mixture models are the most popular clustering approach based on statistical models (K-means is used mostly ignoring its probability background). The term "model-based cluster analysis" is often used when mixture models are applied for clustering.

The general idea of mixture models is that the data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ (often but not necessarily from $\mathbb{R}^p$) are modelled as i.i.d. according to distributions from a parametric family $\mathcal{F} = \{P_\theta \text{ with density } f_\theta : \theta \in \Theta\}$, Θ being the parameter set. K clusters are modelled as being generated by densities $f_{\theta_1}, \ldots, f_{\theta_K}$ that occur with probabilities $0 \leq \pi_1, \ldots, \pi_K \leq 1$ with $\sum_{k=1}^K \pi_k = 1$.

Overall, in a **mixture model**, $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are modelled as i.i.d. with density

$$f_\eta(\mathbf{x}) = \sum_{k=1}^K \pi_k f_{\theta_k}(\mathbf{x}), \tag{6.1}$$

where $\eta = (\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K)$ is the parameter vector.

Equivalently, connecting this to (2.1), the mixture model can be written down in two steps, introducing random variables $\gamma(1), \ldots, \gamma(n)$ giving cluster labels:

$$\begin{aligned} i \in \mathbb{N}_n : \ \gamma(i) \text{ i.i.d.} &\sim \text{Multinomial}(1, \pi_1, \ldots, \pi_K), \\ \mathbf{x}_i | \gamma(i) = k &\sim f_{\theta_{\gamma(i)}}, \end{aligned}$$

which implies that (6.1) is the marginal density for the $\mathbf{x}_i$, $i \in \mathbb{N}_n$ ("$\mathbf{x}_i$" is here and below used as a short hand for "random variable $\mathbf{X}_i$ takes value $\mathbf{x}_i$"). Bayes' formula then yields:

$$P(\gamma(i) = k | \mathbf{x}_i) = \frac{\pi_k f_{\theta_k}(\mathbf{x}_i)}{\sum_{j=1}^K \pi_j f_{\theta_j}(\mathbf{x}_i)}, \tag{6.2}$$

which is helpful for classifying the observations, see below.

The parameters η are, for fixed K, usually estimated by ML, i.e.,

$$\hat{\eta} = (\hat{\pi}_1, \ldots, \hat{\pi}_K, \hat{\theta}_1, \ldots, \hat{\theta}_K) = \arg\max_\eta \prod_{i=1}^n \left(\sum_{k=1}^K \pi_k f_{\theta_k}(\mathbf{x}_i) \right). \tag{6.3}$$

The estimated parameters can then be used in (6.2) to estimate probabilities for an observation $\mathbf{x}_i,\ i \in I\!N_n$ having been generated by mixture component k, which means belonging to cluster k interpreting (as is normally done) the mixture components as clusters:

$$p_{ik} = \hat{P}(\gamma(i) = k|\mathbf{x}_i) = \frac{\hat{\pi}_k f_{\hat{\theta}_k}(\mathbf{x}_i)}{\sum_{j=1}^{K} \hat{\pi}_j f_{\hat{\theta}_j}(\mathbf{x}_i)}.$$

If a partition is required, cluster labels can be estimated by maximising this:

$$\hat{\gamma}(i) = \arg\max_{k \in I\!N_k} p_{ik}.$$

The number of cluster K can be estimated by general model selection techniques. The most widely used one in model-based cluster analysis is the **Bayesian Information Criterion** (BIC). Let

$$\hat{L}_{n,K,\Theta} = \prod_{i=1}^{n} \left(\sum_{k=1}^{K} \hat{\pi}_k f_{\hat{\theta}_k}(\mathbf{x}_i) \right)$$

with ML-estimators $\hat{\eta}$ assuming the number of clusters K plugged in. Because models with smaller K are nested in models with larger K (by choosing, e.g., one or more $\pi_i = 0$), $\hat{L}_{n,K,\Theta}$ will increase monotonically in K. The BIC penalises $\hat{L}_{n,K,\Theta}$ with a function of the complexity (i.e., the number of free fitted parameters) of the model so that the complexity is traded off against $\hat{L}_{n,K,\Theta}$. This can also be used to compare parametric models with different numbers of parameters per mixture component.

Definition 6.1 *The* **Bayesian Information Criterion** *is*

$$BIC(n, K, \Theta) = 2 \log \hat{L}_{n,K,\Theta} - v(K, \Theta) \log(n),$$

where $v(K, \Theta)$ is the number of free parameters in the model with K mixture components in parameter space Θ.

The BIC-optimal model (and more specifically the BIC-optimal K) is the one maximising $BIC(n, K, \Theta)$.

This can also be used to compare parametric models with different numbers of parameters per mixture component, hence the dependence on Θ. $K = 1$ (i.e., a homogeneous model for unclustered data) can be included in the comparisons by the BIC. Note that in the literature sometimes the BIC is defined as $-2 \log \hat{L}_{n,K,\Theta} + v(K, \Theta) \log(n)$, which then would have to be minimised ("small is good"), whereas I use the "large is good"-version here, following the `mclust`-package.

The BIC was originally introduced by Schwarz (1978). For mixture models, there is a consistency result for the BIC as estimator of K under very restrictive conditions by Keribin (2000). Such results are phenomenally difficult to prove.

6.2 The Gaussian mixture model

The most popular mixture modelling for clustering is the Gaussian mixture model, in which the Gaussian distribution models clusters. There are many recent publications about mixtures of other parametric models, but this is not treated here. In the Gaussian case, (6.1) becomes

$$f_\eta(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \varphi_{\mathbf{a}_k,\Sigma_k}(\mathbf{x}) \tag{6.4}$$

with $\eta = (\pi_1, \ldots, \pi_K, (\mathbf{a}_1, \Sigma_1), \ldots, (\mathbf{a}_K, \Sigma_K))$.

Figure 6.1 shows some data generated by 4 different Gaussian mixtures. Gaussian populations are flexible with elliptical shapes. If interpreted as clusters, they may generate large within-cluster distances. They may not necessarily be well separated, and mixtures of $K > 1$ Gaussians may be unimodal. In some applications this may be desired; consider the two datasets at the bottom of Figure 6.1; despite the fact that the Gaussian populations are not separated and the mixture is unimodal, the data may still point at two real populations that differ in a meaningful way; and a cluster analyst may want to find these.

On the other hand, considering e.g. the upper right dataset in Figure 6.1, a mixture of two not well separated Gaussians may just be a homogeneous distribution with a slightly non-Gaussian shape, and an interpretation of the Gaussian components as meaningful clusters may not be justified. More generally, almost every distribution, homogeneous or inhomogeneous, can be approximated by a mixture of Gaussians if only enough Gaussians are used, so some care is required when interpreting Gaussian mixture components as clusters. Actually, mixture models are also sometimes used for density estimation with the aim of fitting the dataset without interpreting mixture components as clusters.

A comprehensive overview paper for Gaussian mixtures for clustering and other purposes is Fraley and Raftery (2002).

Remark 6.2 *The* `pdfCluster`*-package in R provides a method for clustering based on high density regions without parametric assumptions, see Azzalini and Menardi (2014). The function* `mergenormals` *in the* `fpc`*-package implements methods to merge Gaussian components in a mixture model that are not separated enough, see Hennig (2010).*

6.3 Covariance matrix models

The R-package `mclust` (developed by the authors of Fraley and Raftery (2002)) fits Gaussian mixtures. It allows to fit a number of models that are constrained versions of model (6.4), making certain assumptions for the covariance matrices. For example, the assumption $\forall k \in I\!N_K : \Sigma_k = b\mathbf{I}_p$ fits a "K-means style" spherical Gaussian mixture not suffering from the inconsistency problem of K-means mentioned in Remark 2.7. Another possible constraint is to assume all covariance matrices to be equal: $\forall k \in I\!N_K : \Sigma_k = \Sigma$.

Such constraints are sometimes useful because the most general model with unconstrained covariances may have too many parameters and may be numerically unstable, particularly if p

and K are large. Also, in some applications constraints may help to make the clusters better interpretable; for example, the K-means assumption will be much better at avoiding large within-cluster distances than the fully flexible model.

The `mclust`-system of constraints is based in spectral decomposition of the covariance matrix:

$$\Sigma_k = \lambda_k D_k A_k D_k^T,\ k \in I\!N_K,$$

where

- $(\lambda_{k1}, \ldots, \lambda_{kp})$ are the eigenvalues of Σ_k,
- $\lambda_k = \prod_{i=1}^{p} (\lambda_{ki})^{1/p}$ is the "hypervolume",
- D_k is the matrix of eigenvectors, the direction of which determines the "orientation",
- $A_k = \frac{1}{\lambda_k} \text{diag}(\lambda_{k1}, \ldots, \lambda_{kp})$ is called "shape" of Σ_k with $\det A_k = 1$.

These are illustrated in Figure 6.2. One or more of these can be assumed equal between clusters. The shape can be assumed to be the unit matrix.

`mclust` has a coding system for constraints. "V" means "variable" (i.e., flexible), "E" means "equal", "I" means "unit matrix". Constrained models are defined by three letter codes for volume, shape, and orientation:

```
From ?mclustModelNames:

*univariate mixture*
  "E": equal variance (one-dimensional)
  "V": variable variance (one-dimensional)

*multivariate mixture*
 "EII": spherical, equal volume
 "VII": spherical, unequal volume
 "EEI": diagonal, equal volume and shape
 "VEI": diagonal, varying volume, equal shape
 "EVI": diagonal, equal volume, varying shape
 "VVI": diagonal, varying volume and shape
 "EEE": ellipsoidal, equal volume, shape, and orientation
 "EEV": ellipsoidal, equal volume and equal shape
 "VEV": ellipsoidal, equal shape
 "VVV": ellipsoidal, varying volume, shape, and orientation
```

`mclust` allows the user to specify one or more models using the parameter `modelNames`. The default (in case `modelNames` is not specified) is to fit all models and to select the optimal one using the BIC.

6.4 Computation: the EM-algorithm

Assume K and the covariance matrix model to be fixed (for applying the BIC, this is done for a range of values of K and models). Fitting the Gaussian mixture by ML means maximising

$$\log L_{n,K,\Theta}(\eta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \varphi_{\mathbf{a}_k,\Sigma_k}(\mathbf{x}_i) \right).$$

Again, there is no straightforward analytic solution, and an algorithm is needed that computes a local optimum.

This is done in `mclust` by the so-called EM-algorithm (Dempster et al. (1977)). This is a general principle for finding an ML-estimator in case of incomplete information. Sometimes the EM-algorithm is referred to as "clustering method", by which people usually mean that the EM-algorithm is used to fit a Gaussian mixture model.

The general principle of the EM-algorithm is as follows: Let $\tilde{\mathbf{y}} = \mathbf{y}_1, \ldots, \mathbf{y}_n$ be the unobserved complete data and $\mathcal{D} = T(\tilde{\mathbf{y}})$ be the observed data.

The missing information in the mixture model are the cluster memberships $\gamma(i), \ldots, \gamma(n)$, i.e., $\mathbf{y}_i = (\gamma(i), \mathbf{x}_i)$.

In general, the EM-algorithm attempts to maximise $L(\eta) = \sum_{i=1}^{n} \log f_\eta(\mathbf{x}_i)$. Define $L_{n,c}(\eta) = \sum_{i=1}^{n} \log f_{\eta,c}(\mathbf{y}_i)$, where $f_{\eta,c}$ is the density for complete information. The algorithm is initialised by a starting parameter η_0, $t = 1$, and proceeds by iterating:

E-step Compute the **E**xpected complete likelihood.

$$q(\eta|\eta_{t-1}) = E_{\eta_{t-1}}(L_{n,c}(\eta)|T = \mathcal{D}).$$

M-step **M**aximise the conditional likelihood.

$$\eta_t = \arg\max_{\eta} q(\eta|\eta_{t-1}),\ t = t+1.$$

Dempster et al. (1977) proved that both steps never decrease $L(\eta)$.

In the Gaussian mixture model, $\eta = (\pi_1, \ldots, \pi_K, \mathbf{a}_1, \ldots \mathbf{a}_K, \Sigma_1, \ldots, \Sigma_K)$.

The complete loglikelihood with $\gamma(i),\ i \in I\!N_n$ known is

$$L_{n,K,\Theta,c}(\eta) = \sum_{i=1}^{n} \sum_{k=1}^{K} 1(\gamma_i = k)(\log \pi_k + \log \varphi_{\mathbf{a}_k,\Sigma_k}(\mathbf{x}_i)).$$

E-step:

$$E_{\eta_{t-1}}(L_{n,K,\Theta,c}(\eta)|T = \mathcal{D}) =$$
$$= \sum_{i=1}^{n} \sum_{k=1}^{K} P(\gamma(i) = k|\eta_{t-1}, \mathbf{x}_i)(\log \pi_k + \log \varphi_{\mathbf{a}_k,\Sigma_k}(\mathbf{x}_i)),$$
$$p_{ik}^{(t-1)} = P\{\gamma(i) = k|\eta_{t-1}, \mathbf{x}_i\} = \frac{\pi_k^{(t-1)} \varphi_{\mathbf{a}_k^{(t-1)},\Sigma_k^{(t-1)}}(\mathbf{x}_i)}{\sum_{h=1}^{K} \pi_h^{(t-1)} \varphi_{\mathbf{a}_h^{(t-1)},\Sigma_h^{(t-1)}}(\mathbf{x}_i)}.$$

M-step: maximise

$$\sum_{i=1}^{n}\sum_{k=1}^{K} p_{ij}^{(t-1)}(\log\pi_k + \log\varphi_{\mathbf{a}_k,\Sigma_k}(\mathbf{x}_i)).$$

Assume fully flexible covariance matrices, i.e., model "VVV" (in other cases it is more difficult). It is then possible to maximise separately

$$\sum_{i=1}^{n}\sum_{k=1}^{K} p_{ik}^{(t-1)}\log\pi_k \;\Rightarrow \pi_k^{(t)} = \frac{1}{n}\sum_{i=1}^{n} p_{ik}^{(t-1)},$$

$$\sum_{i=1}^{n}\sum_{k=1}^{K} p_{ik}^{(t-1)}\log\varphi_{\mathbf{a}_k,\Sigma_k}(\mathbf{x}_i),$$

which yields weighted Gaussian ML estimators for $(\mathbf{a}_k, \Sigma_k),\ k \in I\!N_K$:

$$\mathbf{a}_k^{(t)} = \frac{1}{\sum_{i=1}^{n} p_{ik}^{(t-1)}}\sum_{i=1}^{n} p_{ik}^{(t-1)}\mathbf{x}_i,\ \Sigma_k^{(t)} = \frac{1}{\sum_{i=1}^{n} p_{ik}^{(t-1)}}\sum_{i=1}^{n} p_{ik}^{(t-1)}(\mathbf{x}_i - \mathbf{a}_k^{(t)})(\mathbf{x}_i - \mathbf{a}_k^{(t)})'.$$

These can be iterated until convergence, i.e., until there is (almost) no improvement in $L_{n,K,\Theta}(\eta)$ anymore.

`mclust` initialises the algorithm by running a Ward-style hierarchical clustering applied to the loglikelihood.

6.5 Example

The core command in the `mclust`-package for fitting Gaussian mixtures is `Mclust`. This is applied to the Olive Oil data.

```
# Fit all models with K between 1 and 15.
# In mclust, the number of clusters is called G.
# This takes quite some time. May choose a smaller range
# or fit only one or a few models, see below, to make
# it faster.
molive <- Mclust(olive,G=1:15)

summary(molive)
# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust VVE (ellipsoidal, equal orientation) model with 8 components:
#
# log.likelihood   n  df       BIC       ICL
#      -20585.88 572 163 -42206.68 -42239.66
```

```
#
# Clustering table:
#   1   2   3   4   5   6   7   8
# 193  49  69  61  65  57  33  45

# The optimal clustering is VVE with K=8.

# The cluster labels are in molive$classification.

# A bit more information about the BIC-values:
summary(molive$BIC)
# Best BIC values:
#              VVE,8      VVV,3       VVV,6
# BIC      -42206.68 -42671.283 -42712.2220
# BIC diff      0.00   -464.607   -505.5457

# Diagnostic plots:
plot(molive)

# Compare VVE,8-clustering with regions:

adjustedRandIndex(molive$classification,oliveoil$region)
# [1] 0.8808006
# Very good!

# The VVV,3-model is second best and could be good for macro.areas.
# Fit model for fixed K and fixed model:
molive3 <- Mclust(olive,G=3,modelNames="VVV")

adjustedRandIndex(molive3$classification,oliveoil$macro.area)
# [1] 0.8141643
# Also quite good.

# Diagnostic plots:
plot(molive3)
```

The `plot`-command applied to an `Mclust`-output offers four diagnostic plots. The first plot visualises the BIC-values for the different fitted models. The second plot plots the clusters in a matrix plot. The third plot plots "uncertainties", i.e., $1 - p_{i\hat{\gamma}(i)}$. If $p_{i\hat{\gamma}(i)}$ is large, it means that the probability that $\mathbf{x}_i$ is from cluster $\hat{\gamma}(i)$ is estimated to be very high. Otherwise there is large uncertainty about whether $\hat{\gamma}(i)$ is really the correct cluster for $\mathbf{x}_i$. These points are darker in the uncertainty plot. The fourth plot gives density contours of the estimated mixture density. For the results for the Olive Oil data, see Figure 6.3.

6.6 Remarks

Remark 6.3 *An important theoretical issue with mixture models is* **identifiability**. *A mixture model is identifiable if it is not possible (up to permutation of the numbers of the mixture components) to obtain the same probability model with two or more different choices of parameters. If this were the case, an arbitrarily large dataset could still not tell the different parameters (which may imply different clusterings) apart.*

The Gaussian mixture model is identifiable, but not all mixture models are.

Remark 6.4 *A problem with some of the more flexible covariance matrix models is that a mixture component can be chosen in such a way that it fits one point precisely, which means that the covariance matrix degenerates to a zero matrix and the likelihood degenerates to infinity. Consequently, one normally looks for local optima of the likelihood for which the smallest eigenvalue of a covariance matrix is bounded away from zero. In practice, still sometimes "spurious clusters" occur which are very small and meaningless but produce a very large likelihood because their covariance matrix has an eigenvalue close to zero.*

Remark 6.5 *Some of the covariance matrix models (e.g., "VVV" and "EEE") are scale and rotation invariant, i.e., the clusters do not change if the dataset is rotated, or the scale of the variables is changed with different variables treated potentially differently. Other models such as the K-means type "EII" are not scale equivariant, and some ("VII") are not rotation equivariant.*

Consequently, picking the best model using the BIC is not invariant in any sense, although rotation and scaling often do not make much of a difference in practice in case that the BIC picks an invariant model.

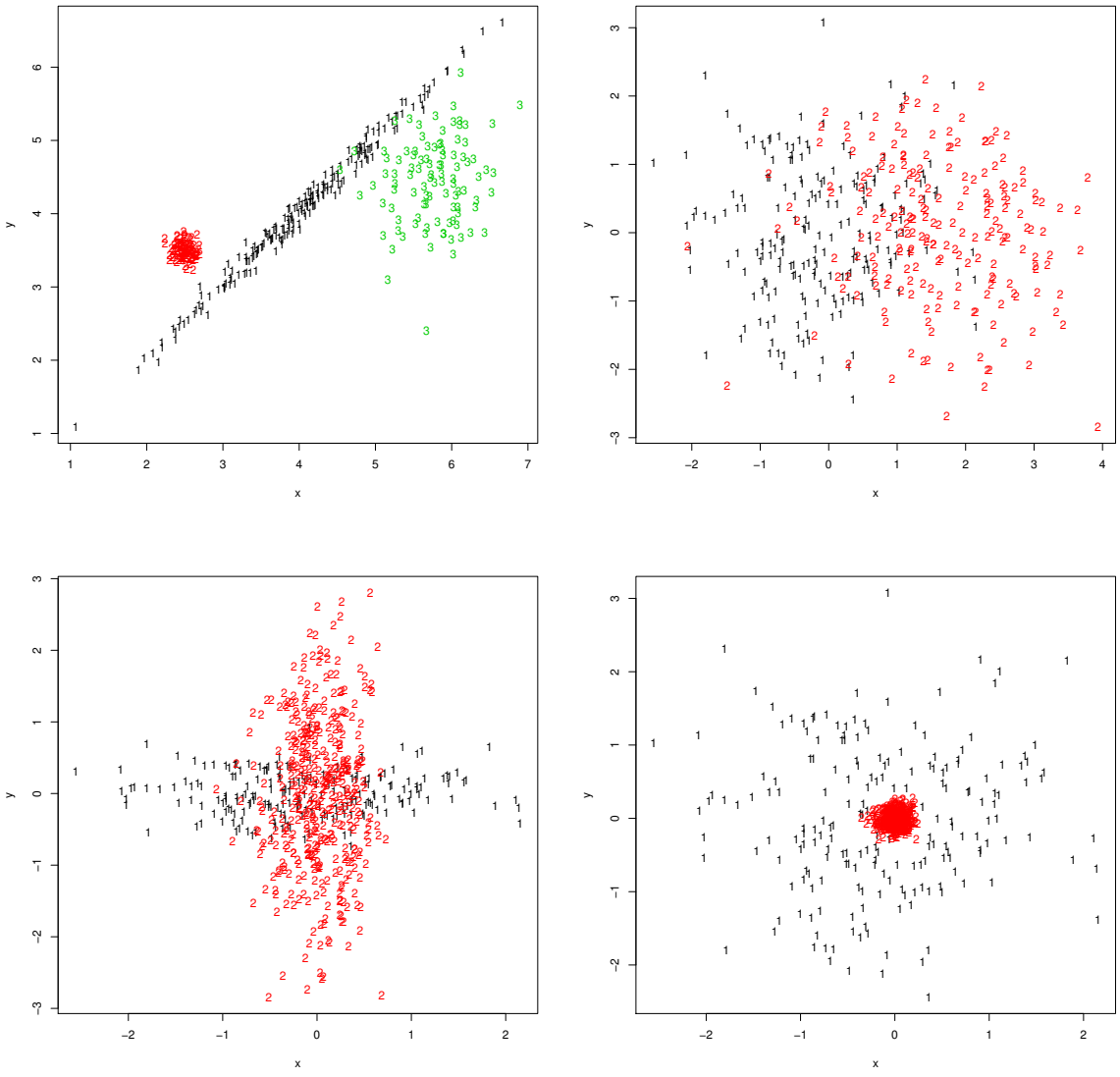

Figure 6.1: Data from 4 Gaussian mixtures (colours and numbers indicate the mixture components from which observations were generated).

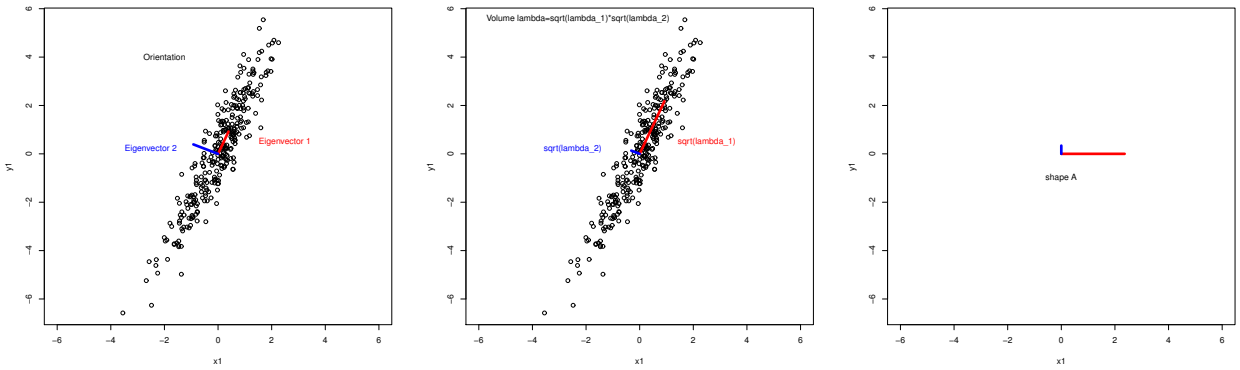

Figure 6.2: Illustration of orientation, volume and shape of covariance matrices of Gaussian distributions.

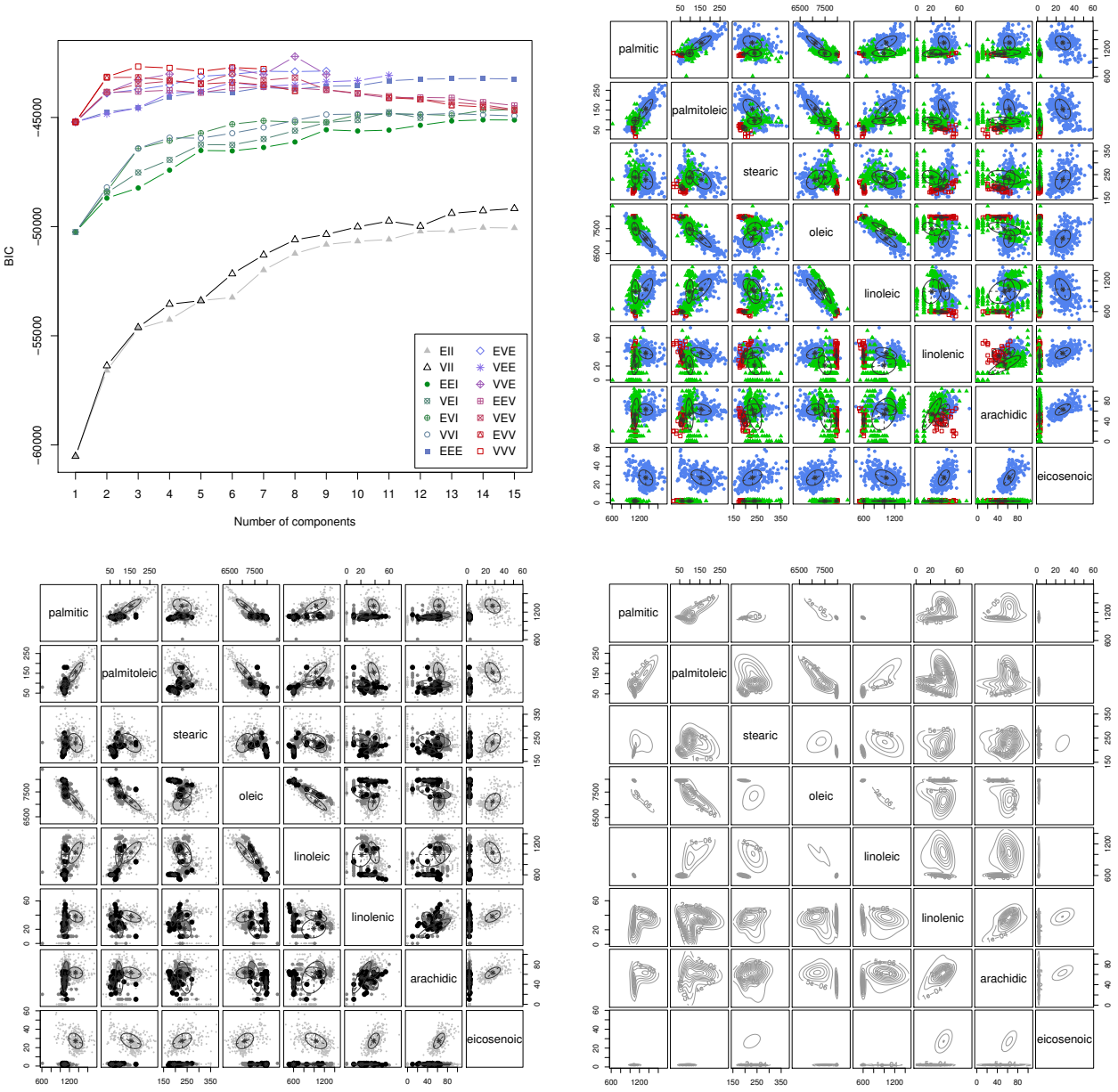

Figure 6.3: "mclust"-plots for Olive Oil data: Upper left: BIC values for different K and covariance matrix models from "plot(molive)". The following plots are for model "VVV" with $K = 3$, i.e., from "plot(molive3)". Upper right: Clustering, lower left: uncertainty plot, lower right: density plot.

Bibliography

Aggarwal, C. C. and C. K. Reddy (Eds.) (2014). *Data Clustering: Algorithms and Applications*. CRC Press, Boca Raton (FL).

Azzalini, A. and A. W. Bowman (1990). A look at some data on the old faithful geyser. *Applied Statistics 39*, 357–365.

Azzalini, A. and G. Menardi (2014). Clustering via nonparametric density estimation: the r package pdfcluster. *Journal of Statistical Software 57*, 1–26.

Borg, I. and P. Groenen (2005). *Modern Multidimensional Scaling: theory and applications (2nd ed.)*. Springer, New York.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statisrtical Society, Series B 39*(1), 1–38.

Everitt, B. S., S. Landau, M. Leese, and D. Stahl (2011). *Cluster Analysis, 5th ed.* Wiley, New York.

Florek, K., J. Lukaszewicz, J. Perkal, and S. Zubrzycki (1951). Sur la liaison et la division des points dun ensemble fini. *Colloquium Mathematicae 2*, 282–285.

Fraley, C. and A. E. Raftery (2002). Model-based clustering, discriminant analysis and density estimation. *Journal of the American Statistical Association 97*, 611–631.

Gordon, A. D. (1999). *Classification, 2nd ed.* CRC Press, Boca Raton (FL).

Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics 27*, 857–874.

Halkidi, M., M. Vazirgiannis, and C. Hennig (2015). Method-independent indices for cluster validation and estimating the number of clusters. In C. Hennig, M. Meila, F. Murtagh, and R. Rocci (Eds.), *Handbook of Cluster Analysis*, pp. 595–618. Taylor & Francis.

Hartigan, J. A. and M. A. Wong (1979). A k-means clustering algorithm. *Applied Statistics 28*, 100–108.

Hennig, C. (2010). Methods for merging gaussian mixture components. *Advances in Data Analysis and Classification 4*, 3–34.

Hennig, C. (2015). Clustering strategy and method selection. In C. Hennig, M. Meila, F. Murtagh, and R. Rocci (Eds.), *Handbook of Cluster Analysis*, pp. 703–733. Taylor & Francis.

Hennig, C. and C.-J. Lin (2015). Flexible parametric bootstrap for testing homogeneity against clustering and assessing the number of clusters. *Statistics and Computing 25*, 821–833.

Hennig, C., M. Meila, F. Murtagh, and R. Rocci (2015). *Handbook of Cluster Analysis*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. Taylor & Francis.

Hubert, L. and P. Arabie (1985). Comparing partitions. *Journal of Classification 2*(2), 193–218.

Jain, A. K. and R. C. Dubes (1988). *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs (NJ).

Kaufman, L. and P. Rousseeuw (1990). *Finding Groups in Data*. Wiley, New York.

Keribin, C. (2000). Consistent estimation of the order of mixture models. *Sankhya Series A 62*, 49–66.

Leisch, F. (2015). Resampling methods for exploring cluster stability. In C. Hennig, M. Meila, F. Murtagh, and R. Rocci (Eds.), *Handbook of Cluster Analysis*, pp. 637–652. Taylor & Francis.

Martinez-Ortega, M. M., L. Delgado, D. C. Albach, J. A. Elena-Rossello, and E. Rico (2004). Species boundaries and phylogeographic patterns in cryptic taxa inferred from aflp markers: Veronica subgen. pentasepalae (scrophulariaceae) in the western mediterranean. *Systematic Biology 29*, 965–986.

Meila, M. (2015). Criteria for comparing clusterings. In C. Hennig, M. Meila, F. Murtagh, and R. Rocci (Eds.), *Handbook of Cluster Analysis*, pp. 619–636. Taylor & Francis.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association 66*, 846–850.

Schwarz, G. E. (1978). Estimating the dimension of a model. *Annals of Statistics 6*, 461–464.

Steinhaus, H. (1957). Sur la division des corps matriels en parties. *Bulletin of the Polish Academy of Sciences 4*, 801–804.

Sugar, C. A. and G. M. James (2003). Finding the number of clusters in a data set: An information theoretic approach. *Journal of the American Statistical Association 98*, 750–763.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association 58*, 236–244.