Die approbierte Originalversion dieser Dissertation ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

DISSERTATION

Robust Clustering and Dimension Reduction: Methods, Algorithms and Implementation

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

Ao. Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Filzmoser
E107
Institut für Statistik und Wahrscheinlichkeitstheorie

eingereicht an der Technischen Universität Wien
bei der Fakultät für Mathematik und Geoinformation

von
Mag.rer.soc.oec. Dipl.-Ing. Heinrich Fritz
Matrikelnummer: 0325693
Landstr. Hauptstr. 104/4
1030 Wien

Wien, am 11.September 2011

Acknowledgements

First of all, I want to thank my supervisor Prof. Peter Filzmoser from the Vienna University of Technology, for his motivation and support throughout the last four years during my PhD studies and for continuously guiding me along the way from the Bachelors to the Masters and finally through the PhD program. In many discussions throughout several years he kept motivating and inspiring me with new ideas and insights, which finally brought this project to a successful end.
Further I want to express my thanks to Prof. Luis Ángel Garcia Escudero and Prof. Agustín Mayo Iscar from the University of Valladolid, Spain, who played an important role during the realization of major chapters of this thesis. Through their technical support, and their expertise in writing scientific papers, this work noticeably gained importance.
I also would like to thank Prof. Christophe Croux from the Katholieke Universiteit Leuven, Belgium for his theoretical contribution and his support in writing the papers published in the scope of this thesis.
Last but not least I want to mention my friends who kept motivating me until the end, by continuously asking about the thesis' status, as well as my family and parents, who believed in me and supported me throughout all these years - financially and mentally and thus gave me the chance to finally achieve this objective.

Abstract

Considering information extraction with respect to real data samples, classical statistical estimation generally suffers from the difference between the assumptions such methods are based on, and the circumstances observed in reality. Due to unexpected behaviour and irregularities of the monitored process itself, or problems during data acquisition, basic prerequisites as e.g. normality are not likely to be observed in real scenarios. In general, robust methods address this issue and help to identify and assess such irregularities, and thus allow proper estimation even on heavily contaminated data. The downside of robustness is loss of statistical efficiency, which is reduced by developing adjustable methods, which can be individually tuned. Thus, depending on the actual occurrence of irregularities a compromise between robustness and statistical efficiency is achieved.

For the primary assessment of observed data, an estimation of the basic structure gives a first idea of what information stands behind those numbers. The robust methods for clustering and dimension reduction discussed and developed in this thesis provide a flexible and expansive framework for this purpose. They allow to parameterise the structure of a wide range of possible data configurations, whereas its robustness properties reduce the influence of abnormalities.

In particular, a sparse variant of an already existing approach for robust principal component analysis is developed, which combines elements of projection-pursuit and LASSO regression. So far no method is available which combines robustness and sparseness properties for principal component analysis, thus its publication is expected to be received positively by the statistical community. Further, a maximum likelihood method for trimmed clustering is developed, whereas various methodical and algorithmic improvements of existing solutions are implemented. The resulting method combines the characteristics of already existing methods and expands these approaches to one general solution.

The theoretical presentation and development of the methods in this work is accompanied by a discussion of algorithmic aspects and an implementation in the programming environment R. The core implementation, however, is not restricted to R and can be exported to other environments, which makes the methods easily available for a wide range of users beyond the R community.

Kurzfassung

Betrachtet man den Prozess der Informationsextrahierung aus Echtdaten im Kontext der klassischen Statistik, so stößt man schnell an deren Grenzen, da die Annahmen klassischer Methoden oft mit den realen Gegebenheiten im Widerspruch stehen.
Unregelmäßigkeiten innerhalb eines beobachteten Prozesses, sowie Probleme während der Datenerfassung und deren Weiterverarbeitung, resultieren oft in unerwarteten Ergebnissen und darin, dass Echtdaten selten einfachen Voraussetzungen, wie beispielsweise Normalität genügen.
An diesem Punkt setzen robuste Methoden an, die in der Lage sind Verunreinigungen der Daten zu erkennen und zu beseitigen, um auch auf Basis von stark kontaminierter Daten gültige Schätzungen zu liefern. Der Nachteil dieser angestrebten Robustheit gegenüber Ausreißern ist jedoch die geringere statistische Effizienz entsprechender Schätzverfahren. Dieser Verlust wird reduziert, indem Methoden herangezogen werden, deren Robustheit individuell an die jeweilige Verunreinigung anpassbar ist, womit ein Kompromiss zwischen Robustheit und statistischer Effizienz gefunden wird.
Untersucht man einen bestimmten Datensatz, so gibt die Schätzung grundlegender mathematischer Strukturen in den Daten einen ersten Eindruck über die Information die hinter den ermittelten Zahlen steht. Die robusten Methoden der Clusteranalyse und Dimensionsreduzierung, die in dieser Arbeit entwickelt und diskutiert werden, bilden ein flexibles und weitreichendes Gerüst für diesen Zweck. Diese Methoden ermöglichen es viele verschiedenartige Strukturen zu verstehen und parametrisch abzubilden, wobei der Einfluss von Ausreißern reduziert wird.
Im Einzelnen wird eine Variante eines bereits existierenden Ansatzes für robuste Hauptkomponentenanalyse entwickelt, die jedoch sogenannte "sparse", also dünn besetzte Ladungsmatrizen liefert. Dieser Ansatz kombiniert Elemente von Projection Pursuit basierten Algorithmen, sowie der LASSO Regression. Die vorgestellte Methode ist die erste, die robuste Eigenschaften mit dünn besetzten Ladungsmatrizen im Zusammenhang mit Hauptkomponenten kombiniert. Weiters wird eine „Maximum-Likelihood" Methode für getrimmtes Clustering entwickelt, wobei mehrere algorithmische Verbesserungen von existierenden Methoden implementiert werden. Die resultierende Methode verfügt über Eigenschaften mehrerer existierender Ansätze und erweitert diese zu einer einzelnen generellen Lösung.
Die theoretische Diskussion der angesprochenen Methoden in dieser Arbeit wird von einer Reihe algorithmischer Details und einer Implementierung für die Programmierumgebung R begleitet. Die Kernimplementierung der Algorithmen wurde jedoch unabhängig von R gestaltet, und kann deshalb auch in anderen Umgebungen wie beispielsweise Matlab genutzt werden.

Contents

1 Introduction **1**
- 1.1 General Ideas of Robust Estimation . . . 1
- 1.2 General Aspects of Robust Methods . . . 2
- 1.3 Outlier Identification . . . 3
  - 1.3.1 Types of Outliers . . . 3
- 1.4 Robust Location and Covariance Estimation . . . 4
  - 1.4.1 Methods . . . 5
  - 1.4.2 Example . . . 6
- 1.5 Robust PCA . . . 8
  - 1.5.1 Methods . . . 8
  - 1.5.2 Example . . . 11
- 1.6 Robust Sparse PCA . . . 11
  - 1.6.1 Algorithms . . . 12
  - 1.6.2 Example . . . 12
- 1.7 Robust Cluster Analysis . . . 13
  - 1.7.1 Methods . . . 14
  - 1.7.2 Example . . . 17
- 1.8 Outline of the Thesis . . . 19

2 A comparison of algorithms for the multivariate L_1-median **21**
- 2.1 Introduction . . . 21
- 2.2 Algorithms for computing the L_1-median . . . 22
  - 2.2.1 General optimization procedures . . . 23
  - 2.2.2 Problem specific algorithms . . . 24
  - 2.2.3 Implementation . . . 26
  - 2.2.4 Convergence . . . 26
- 2.3 Comparison of the algorithms . . . 27
  - 2.3.1 Adjusting tolerance levels for convergence . . . 28
  - 2.3.2 Uncorrelated data . . . 30
  - 2.3.3 Correlated data . . . 33
  - 2.3.4 High-dimensional data with low sample size ($p \gg n$) . . . 33
  - 2.3.5 Degenerated situations . . . 35
  - 2.3.6 Real data examples . . . 37

2.3.7 Runtime as a function of the sample size 38
2.4 Conclusions 39

3 Exploring High-dimensional Data With Robust Principal Components 41
3.1 Introduction 41
3.2 PCA for high-dimensional data 42
3.2.1 Example 44
3.3 Orthogonal distance and score distance 44
3.4 Exploring the multivariate data structure 46
3.5 Summary 48

4 Robust Sparse Principal Component Analysis 49
4.1 Introduction 49
4.2 Method 51
4.3 Algorithm 52
4.4 Selection of λ 54
4.5 Simulation experiments 56
4.6 Real data examples 60
4.7 Concluding remarks 66

5 tclust: An R Package for a Trimming Approach to Cluster Analysis 68
5.1 Introduction to robust clustering and tclust 68
5.2 Trimming and the spurious outliers model 71
5.3 Constraints on the cluster scatter matrices 72
5.3.1 Constraints on the eigenvalues 73
5.3.2 Constraints on the determinants 74
5.3.3 Equal scatter matrices 74
5.3.4 Example 74
5.4 Numerical output 77
5.5 Algorithms 78
5.6 Comparison with other robust clustering proposals 79
5.7 Selecting the number of groups and the trimming size 81
5.8 Graphical displays 86
5.9 Swiss Bank notes data 89
5.10 Conclusion 91

6 A fast algorithm for robust constrained clustering 93
6.1 Introduction 93
6.2 Constrained robust clustering and TCLUST 95
6.3 Algorithm 97
6.4 Simulation Study 101
6.5 Conclusions 105

7 Implementational Details **109**
7.1 Introduction 109
7.2 Architecture 111
7.2.1 Application Layer 111
7.2.2 Module Interface (Environment) 112
7.2.3 Module Interface (**C++**) 113
7.2.4 Module Implementation 113
7.2.5 SMat - Simple Matrix Classes 113
7.2.6 MEAL - Declaration 114
7.2.7 MEAL - Implementation 115
7.3 Conclusions 115

Chapter 1

Introduction

1.1 General Ideas of Robust Estimation

All statistical methods are based on certain assumptions on the data they are supposed to be applied on. These assumptions improve the efficiency of the methods and in many cases lead to nice mathematical properties of a considered solution. Next to assumptions as normality or independency, classical methods often assume that all observations belong to the same distribution. Such classical methods are very elegant from a theoretical point of view, and can usually be calculated with little computational effort. Thus they were quite popular when computational power was still dear and scarce. However, in practical applications such assumptions cannot always be fulfilled, leading to inappropriate results which do not correctly reflect the actual information contained in the data.
A class of algorithms which addresses some of the shortcomings of classical methods in practical applications is formed by robust methods, which do not assume homogeneously distributed data. Methods of this class show a certain resistance to irregular or outlying observations, and only refer to the bulk of the data for any kind of estimation. In practice, many reasons may cause irregularities in a data set, as for example heterogeneous data generating processes or errors and problems in the chain where the obtained data is measured or processed.
The resistance to irregular or outlying observations of a method is quantified by its *breakdown point* (see Donoho and Huber, 1983), usually ranging from 0% (e.g. mean) up to 50% (e.g. median), which means that asymptotically half of the data can be contaminated arbitrarily without obtaining completely arbitrary results. As any estimation in this context is supposed to refer to the majority of the data, breakdown points beyond 50% are not considered.

1.2 General Aspects of Robust Methods

Throughout the last century until now, the theoretical properties of robust methods have been investigated extensively (see the "Methods"-sections in the following), but only during the last decades the computational power has been available for practically applying these methods to a large extent. However, apart from the obvious advantage of robustness these methods also yield drawbacks, as in general they are statistically not as efficient as their classical counterparts. This results from the fact, that many robust methods implicitly or explicitly downweight the influence of apparently irregular observations, which results in a loss of information. Thus, when applying a robust method onto a non-contaminated data set, the result's variance is expected to be larger than when applying a corresponding classical method. For reducing the effect of this clear drawback, many robust methods provide tuning parameters as e.g. a trimming level, which enables the user to adjust the method's breakdown point. Thus, theoretically a solution can be obtained which also depends on the actual contamination level in the data. However, in general such methods need some human supervision, as these tuning parameters have to be found in an interactive process. In some contexts such parameters may be chosen automatically, but it shows, that usually human supervision is inevitable (see 5.7) when several closely related tuning parameters have to be chosen.
A further disadvantage of robust methods is that from a computational point of view they are significantly more complex than according classical methods. Many robust methods internally are based on sort algorithms (e.g. Q_n, see Rousseeuw and Croux, 1993), and thus do not scale linearly anymore.
Further, there are only few robust estimators, which provide an explicit definition to a solution as e.g. the median. Many robust estimators can only be computed numerically, that is, by optimizing a certain objective function. This is still straightforward, if such an objective function is convex (as for e.g. the L_1-median, see Weber, 1909; Weiszfeld, 1937), but gets computationally more intense, if this cannot be assured anymore (e.g. tclust, see García-Escudero *et al.*, 2008). In such a case several randomly initialized *runs* of an algorithm are computed, each converging at a local optimum, and the result yielding the best objective function's value is chosen in the following, assuming that the global optimum was found. Anyway, for this class of robust methods it is impossible to evaluate whether a solution's quality can still be increased, other than by trying even more random initializations of the algorithm. Thus, at some point an approximate result has to be considered as sufficient, knowing that better solutions to the given problem likely exist.

1.3 Outlier Identification

A basic approach for obtaining a robust estimation without referring to a particular robust method, is the identification of irregular observations, their removal and the subsequent application of a classical method to the remaining data set. Thus an instrument for the identification of such irregular observations will be discussed. Another motivation for the identification of outliers is to get a better understanding of a particular data set, as e.g. the reason for irregular observations can be investigated and tracked back to its origin.
A measurement usually used for quantifying the outlyingness of an observation $\boldsymbol{x} \in \mathbb{R}^p$ is the Mahalanobis distance, defined as

$$MD(\boldsymbol{x}) = \sqrt{(\boldsymbol{x} - \boldsymbol{m})^\top \boldsymbol{C}^{-1} (\boldsymbol{x} - \boldsymbol{m})} \tag{1.3.1}$$

with $\boldsymbol{C}$ as a covariance estimate, and $\boldsymbol{m}$ a center estimate. The square of this distance is approximately χ^2 distributed, with p degrees of freedom and thus an observation with Mahalanobis distance larger than a threshold $\sqrt{\chi^2_{p;1-\alpha}}$, with $1-\alpha$ the corresponding quantile (e.g. 0.95) is considered as outlier (compare 3.3).
In the following, robust covariance and location estimates will be considered in this identification process, as classical covariance and location estimates are likely to be influenced such that outliers cannot be spotted appropriately anymore and are thus *masked* or *swamped* (Maronna *et al.*, 2006). After discussing the necessary robust location and covariance estimators an example on outlier identification is given in Section 1.4.2. Further, an example of outlier detection in high-dimensional data sets is discussed in Chapter 3.

1.3.1 Types of Outliers

A basic categorization for outliers is given in Figure 1.1 (a), where the difference between multivariate and univariate outliers is shown by means of a normally distributed data set containing some contamination. Univariate outliers (+) are easy to spot, as they can be identified by only considering one single variable. In this example the outliers point out clearly when considering the data set projected onto the x-axis, whereas when considering a projection onto the y-axis these outliers are among the regular observations (∘) and thus are indistinguishable from them. On the other hand, the multivariate outliers (×) which obviously do not belong to the regular observations either cannot be identified that easily by only considering one single variable. For getting hold of this type of outliers the whole multivariate structure has to be considered, and thus a robust location and covariance estimate is needed in advance.
As there are numerous reasons for the existence of outliers, it is not straightforward to give a complete categorization here, but another type shall be

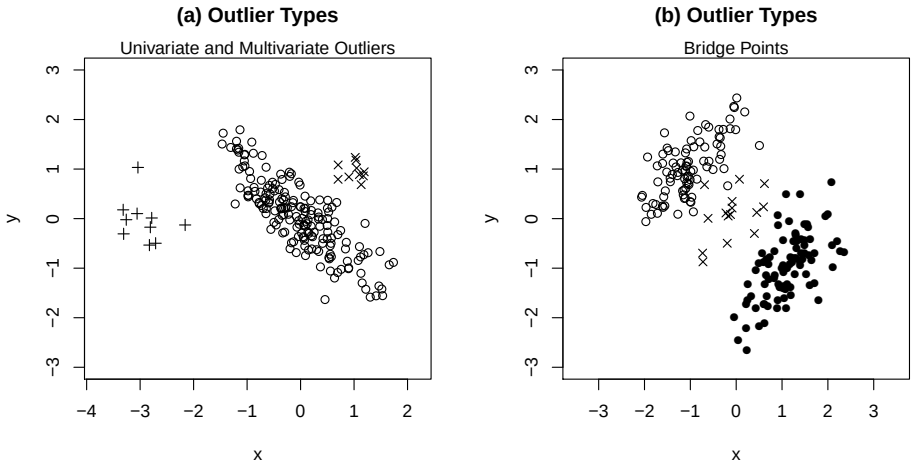

Figure 1.1: Some different outlier types: Multivariate outliers (×) and univariate outliers (+) in (a), and bridge points (×) in between two clusters (○, ●) in (b).

mentioned in this context, which is mainly relevant in robust cluster analysis (see Chapter 5). In this field of robust statistics, the idea of a homogeneous data source is dropped completely, and the parameters of several distributions are estimated by considering the cluster structure which arises due the present inhomogeneity in the considered data. In this context so called *bridge points* (see García-Escudero *et al.*, 2008) are outlying observations, which can be located in the center of a data structure, as in Figure 1.1 (b) between two clusters (○, ●), which makes them difficult to identify. This type of outliers can influence cluster algorithms such, that two or more clusters are joined artificially, and thus none of the involved clusters can be identified appropriately.

1.4 Robust Location and Covariance Estimation

As many methods in robust statistics are based on an initial covariance estimate, which is processed further (e.g. principal component analysis, see Jolliffe, 1986), a robust covariance and location estimate is the key for the robustification of several classical statistical methods. Further, classical algorithms may be applied onto cleaned data sets, where according to Section 1.3 outlying observations have been identified and removed previously. As for this purpose a robust covariance matrix and location estimate is needed in advance, corresponding methods play a key role in robust statistics.
Although a simple approach to robust location estimation would be the component wise application of the median, the robustification of the classi-

cal covariance estimator is not straightforward, as the previously discussed masked and multivariate outliers are not easy to spot. Thus several approaches have been developed for this purpose:

1.4.1 Methods

Depending on the relevance in this work, several robust methods are presented and discussed more extensively throughout this introductory chapter. A rough overview of related methods, can be found at the end of each "Methods" section, this however is not intended to be a complete listing of robust methods.

- L_1-median

The L_1-median, or spatial median, is an outstanding multivariate robust estimator of location, as some of its properties have already been investigated a long time ago (see Weber, 1909; Weiszfeld, 1937). As an alternative to the component-wise median this location and orthogonally equivariant estimator yields a 50% breakdown point (see Lopuhaä and Rousseeuw, 1991). The objective is to find a center which minimizes the sum of Euclidean distances to all observations. In Chapter 2 the properties of different approaches for the computation of this location estimator are investigated by comparing their precision, runtime and behaviour, especially in high-dimensional and difficult data constellations.

- fast MCD

This algorithm (see Rousseeuw and Van Driessen, 1999) is a computationally efficient implementation of the Minimum Covariance Determinant (MCD) method as introduced in Rousseeuw (1984) with attractive robustness properties.
The idea is to find a subset of h observations of a data set containing n observations in p dimensions, with $[(n + p + 1)/2] \leq h \leq n$, such that the corresponding empirical covariance matrix has minimum determinant compared to all other possible subsets of same size.
The obvious drawback of the method, the large number of subsets which have to be checked, is addressed in the fast MCD algorithm. For a given partition of the data set the algorithm applies a so called *concentration step* (C-step) which finds a further partition yielding a smaller covariance determinant. These C-steps are repeated until no better partition can be found and thus the algorithm has converged in a local optimum. The algorithm is applied several times with different initial subsets which are chosen randomly, hoping that one of the found local optima is globally the best possible solution. A more detailed explanation of very similar algorithms and their relation to fast MCD is given in Chapters 5 and 6.

A restriction of this method is that there is no possibility of applying it to high-dimensional ($p >> n$) data sets. This is due to the condition, that at least $[(n + p + 1)/2]$ observations must be considered in each step, which conflicts with $p \geq n$. Moreover, the determinant of the empirical covariance matrix of any subsample is always zero, making the optimization impossible.

- Further Approaches

The M-estimator (Maronna, 1976; Campbell, 1980) has been one of the first approaches to robust covariance estimations, which however lacks robust performance in high dimensional data sets. The Stahel-Donoho method (Stahel, 1981; Donoho, 1982) examines the outlyingness of each observation and in the following estimates a weighted covariance matrix. This algorithm's performance and the application of different weight functions were investigated in Maronna and Yohai (1995). Further the S-estimator was proposed in Davies (1987) and Rousseeuw and Leroy (1987), as well as the minimum volume ellipsoid (MVE) method in Rousseeuw (1984, 1985), as a predecessor of the MCD estimator. Finally, Maronna and Zamar (2002) proposed the orthogonalized Gnanadesikan-Kettenring (OGK) method, as an enhanced version of the pairwise Gnanadesikan-Kettenring covariance estimator (Gnanadesikan and Kettenring, 1972).

1.4.2 Example

Figure 1.2 compares the application of classical covariance estimation to the fast MCD estimator on a generated data set drawn from a bivariate normal distribution with 10% contamination. In panel (a) the deviance of the estimated covariance structure (solid) from the theoretical covariance (dashed) is clearly visible, whereas in (b), the estimated covariance structure appears almost unaffected by the added outliers.
The ellipses in Figures 1.2 (a) and (b) are closely related to the Mahalanobis distance, as they enclose all points with $MD^2 < \chi^2_{2;1-\alpha}$, with $\alpha = 0.05$. Under normality a fraction α of data points is expected to be located outside this boundary, which explains the appearance of few regular observations outside of the theoretical ellipse.
For identifying the outliers as discussed in Section 1.3, Figure 1.2 (c) shows the squared Mahalanobis distances calculated with classical covariance and location estimates. The group of outliers points out, as the corresponding Mahalanobis distances are apparently larger than those of the regular observations. However, the squared Mahalanobis distances of about half of the outliers do not exceed the chosen threshold of $\chi^2_{p;0.95}$ and are thus not distinguishable from the remaining observations. This is caused by an effect called *masking*, as outliers strongly influence classical location and covariance estimates, yielding improper Mahalanobis distances, such that irregular

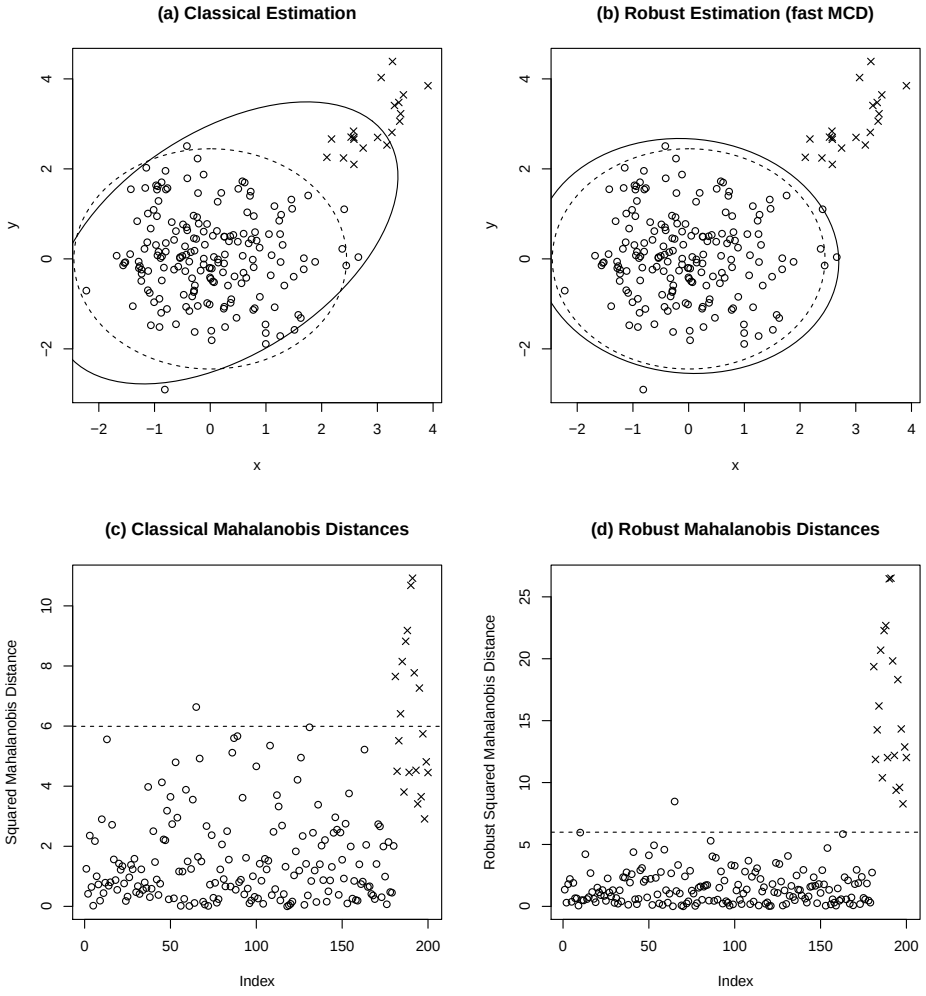

Figure 1.2: Comparing classical covariance estimation (a) to the robust fast MCD algorithm (b). Outliers (×) and regular observations (∘) are displayed next to the estimated (solid) and theoretical (dashed) covariance structure. The classical and robust squared Mahalanobis distances are shown in (c) and (d) respectively, including the $\chi^2_{p;0.95}$ threshold (dashed).

observations appear as if they were regular.
In order to bypass this issue the fast MCD algorithm is applied and hence robust (squared) Mahalanobis distances are calculated. This is shown in Figure 1.2 (d), where the outlying observations can be clearly distinguished, just by considering a threshold of $\chi^2_{p;0.95}$.

1.5 Robust PCA

Principal component analysis (PCA) is a method for finding a number of $k \leq p$ orthogonal directions in a data set, which maxmimize the variance of the data projected onto them (see Jolliffe, 1986). These directions are called principal components (PCs), which give an uncorrelated representation of the original data structure. Thus they can be described by an orthogonal projection matrix $\boldsymbol{\Gamma}$, also called the loadings matrix, containing the directions and an additional vector $\boldsymbol{\lambda}$ which is proportional to the squared length of each component. Each element λ_j of this vector corresponds to the variance of the data projected onto the jth principal component. Further a diagonal matrix $\boldsymbol{\Lambda}$ is considered, which holds the values of $\boldsymbol{\lambda}$ in its diagonal. As usually much less than p components already explain the majority of the variance in a data set, a number of components $k < p$ already represents an approximation of the original data set's structure. Thus PCA can be used as a dimension reduction technique, which is especially useful when high-dimensional data sets are analyzed, as an intuitive interpretation for such data sets is hard to find. Further this method faciliates the interpretation of a data set, as linear combinations of the most important variables and their relations to each other are pointed out.
A classical mathematical approach to principal component analysis is quite straightforward, as a covariance matrix $\boldsymbol{\Sigma}$ is decomposed such that

$$\boldsymbol{\Sigma} = \boldsymbol{\Gamma}\boldsymbol{\Lambda}\boldsymbol{\Gamma}^\top, \tag{1.5.1}$$

with $\boldsymbol{\Gamma}$ an orthogonal matrix, and $\boldsymbol{\Lambda}$ a diagonal matrix as described above. This decomposition corresponds to an eigenvalue decomposition of $\boldsymbol{\Sigma}$, with matrix $\boldsymbol{\Gamma}$ holding the eigenvectors in its columns, and $\boldsymbol{\Lambda}$ holding the corresponding eigenvalues in its diagonal. Further a scores matrix $\boldsymbol{Z}$ is considered, which represents an original centered data matrix $\boldsymbol{X}$ in the space of the principal components:

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{\Gamma}. \tag{1.5.2}$$

For centering the data matrix a location estimate is needed, which in the classical sense would usually be the column-wise mean. However, if robustness is an issue, the corresponding robust location estimator, the column-wise median or L_1-median (see Chapter 2) can be considered here.

1.5.1 Methods

- Naive approach

Due to the close relation of PCA to covariance estimation, the computation of robust PCs seems obvious, as using a robust covariance estimate in Equation 1.5.1 yields robust PCs. However, as explained in the following, there are more specific approaches for the direct computation of (robust) PCs,

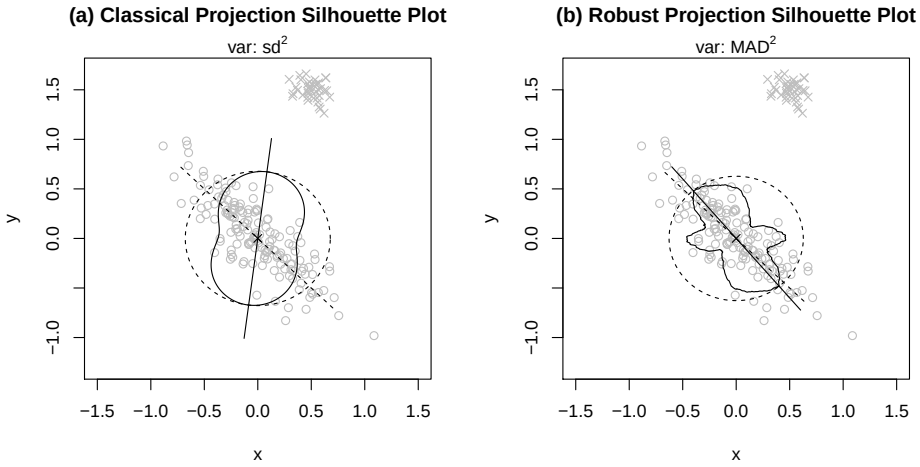

Figure 1.3: Classical (a) and robust (b) projection silhouette plots of a normally distributed data set featuring a 20% outlier portion. Also the estimated (solid) and theoretical (dashed) first PCs are included.

which avoid the prior estimation of a (robust) covariance matrix, which comes in handy when high-dimensional data sets are considered.

- Grid algorithm

This algorithm, as introduced in Croux *et al.* (2007), is based on projection pursuit: A direction is searched, which maximizes a variance estimate of the data projected onto it, resulting in the first PC. Any subsequent PC is computed by applying the same algorithm onto the data projected into the orthogonal space of the already found PCs (see 4.3). By simply considering a robust variance estimator as the squared median absolute deviation (MAD^2) or the squared Q_n estimator (see Rousseeuw and Croux, 1993), this algorithm directly leads to robust PCs.
The issue of the enormous amount of potential candidate directions which have to be considered in this setting is addressed by reducing the problem to a series of simple optimizations in the two-dimensional space. There, a fixed number of candidate directions aligned on a grid can be tested easily, which finally reduces the search of the optimal direction to a series univariate problems. The two-dimensional space which is examined within each iterative step is spanned by a linear combination of variables which converges towards the desired PC, and alternately by the single variables of the data set. A detailed presentation of this algorithm is given in Chapter 4, where a more flexible method based on this approach is developed.
Major advantages of this method are, that it is easily applicable to high

dimensional data sets, and that the computation can be stopped after a certain number of components has been computed. This plays an important role when processing high-dimensional data sets, as there it is usual, that only a small number of PCs already explain a major portion of the total variance, and thus the computational effort can be reduced considerably.
Figure 1.3 illustrates the algorithm's concept by means of a classical and a robust *projection silhouette* plot of a two-dimensional data set. The used data set has been drawn from a bivariate normal distribution, whereas a contamination level of 20% is simulated.
The silhouette (solid) is aligned around a previously estimated center $\boldsymbol{m}$. The distance of each point $\boldsymbol{x}_i$ on the silhouette to the center $\boldsymbol{m}$ represents the square root of the variance of the considered data set projected onto the direction through $\boldsymbol{m}$ and $\boldsymbol{x}_i$. The Grid algorithm chooses the direction which yields the largest distance s_{max} of the silhouette to the center. In this two-dimensional example the chosen direction (solid) is returned as first PC, and its orthogonal direction as the second component. The dashed circle with radius s_{max} helps to visualize the maximum extent of the silhouette. Note that this method works independently from the choice of center $\boldsymbol{m}$, which is only needed for the visualization of the silhouette.
In Figure 1.3 (a) the classical variance estimate is used and a strong difference between the chosen direction (solid) and the theoretical first principal component of the uncontaminated data set (dashed) can be observed, which is apparently caused by the outlier group located at the top of the figure. However, considering the substantial contamination of the data set, the first component is estimated quite precisely when using the MAD^2 as variance estimate in panel (b).

- Further Approaches

Hubert *et al.* (2005) propose the ROBPCA method which combines dimension reduction techniques with some aspects of the MCD estimator for obtaining a robust estimation of the PCs. Moreover, this work introduces exploratory tools as for example particular diagnostic plots, which are used in Chapter 4.
Croux and Ruiz-Gazen (1996) introduced the fundament of the Grid algorithm, the so called CR-method, which is also a projection pursuit approach. However, the candidate directions are not aligned on a grid, but directions through a center and the different observations are considered. In the following the algorithm has been revised in Croux and Ruiz-Gazen (2005) improving its numerical stability.
As already mentioned, at this point all algorithms for robust covariance estimation may be used for obtaining a corresponding robust estimation of the principal components.

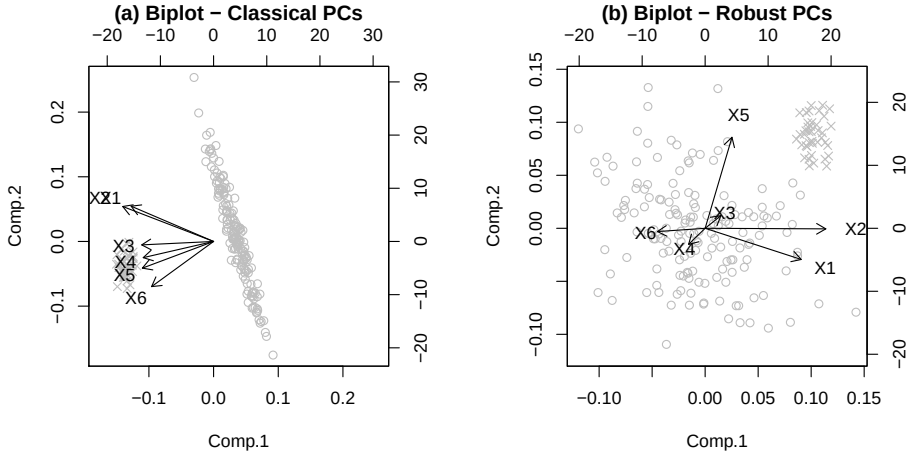

Figure 1.4: A classical (a) and a robust (b) Biplot of a simulated data set featuring a 20% contamination level. Regular observations (∘) and outliers (×) are represented by different symbols.

1.5.2 Example

The difference of classical and robust PCA shall be illustrated using a generated six-dimensional normally distributed data set, featuring a 20% outlier portion. The corresponding Biplots (see Gabriel, 1971; Gower and Hand, 1996) are given in Figure 1.4. The Biplot based on classical components in panel (a) shows the clear influence of the contamination, as the arrows which represent the loadings of the variable X1 through X6 onto the first and second PC point straight to the added outliers. On the other hand, the robust estimation results in panel (b) computed with the Grid algorithm using MAD^2 as variance estimate do rather refer to the structure of the uncontaminated data, as no major influence of the outliers can be observed in this situation.

1.6 Robust Sparse PCA

A quite recent approach for improving the interpretability of PCs is the so called sparse principal component analysis. Additionally to the maximization of the variance projected onto the PCs, another aspect gains importance: in order to facilitate the interpretation of PCs, the loadings matrix is supposed to contain a certain amount of zero values.
Thus, in the best case a PC is a linear combination of a number of variables significantly smaller than p. Especially when exploring high-dimensional data sets this is advantageous, as additionally to the possible dimension

reduction the number of values to interpret can be reduced even further.
As two completely contrary objectives – the maximization of the explained variance in $k < p$ components, and a high number of zeros in the loadings matrix – are tried to be achieved, a certain trade-off has to be made at this point. The relative importance of these aims is defined by introducing tuning parameters to this method, for controlling the result's level of sparseness. For choosing such tuning parameters, methods have been developed recently (Farcomeni, 2009; Leng and Wang, 2009; Guo *et al.*, 2010). A BIC type criterion for this purpose is discussed in Chapter 4.

1.6.1 Algorithms

- Sparse Grid algorithm

The Sparse Grid algorithm as discussed in Chapter 4 is very closely related to the standard Grid algorithm and thus does not depend on any prior estimation of a loadings matrix. The Grid algorithm maximizes a variance estimate of a given data set projected onto a single direction, whereas the objective function of the sparse method additionally considers the sparseness of the examined direction. According to the standard Grid algorithm, robustness is introduced by considering a robust variance estimate in the objective function (e.g. MAD^2 or Q_n^2).

- Further Approaches

Farcomeni (2009) considers sparse PCA as a combinatorial variable selection problem which is solved by a conventional branch and bound algorithm. Jolliffe *et al.* (2003) proposed the SCoTLASS method which reduces PCA to a linear regression problem and then introduces sparseness by the LASSO regression method (Tibshirani, 1996), which was originally developed for variable selection in the context of regression models. Zou *et al.* (2006) extend this approach by additionally considering the elastic net regression method (Zou and Hastie, 2005), a generalization of LASSO. Guo *et al.* (2010) propose a method for fused sparse loadings, which is especially capable of capturing block structures in the loadings matrix.
However, in contrast to the sparse Grid algorithm none of these approaches has been initially designed as robust method, and all of them depend on a prior estimation of the (non sparse) principal components.

1.6.2 Example

Biplots of different sparse PCs are given in Figure 1.5. The PCs are computed applying the sparse Grid algorithm using the MAD^2 as variance estimate, on the same generated data set as used in Section 1.5.2, consisting of variables X1 to X6.

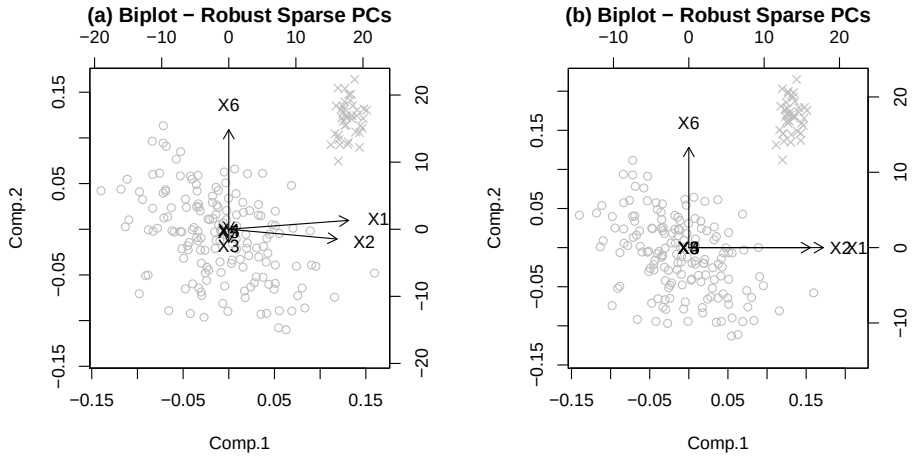

Figure 1.5: Two biplots based on robust sparse PCs of a simulated data set featuring a 20% contamination level, computed with the spare Grid algorithm. Different choices of the algorithm's tuning parameters yield a lower level of sparseness in plot (a) than in plot (b). Regular observations ($\circ$) and outliers ($\times$) are represented by different symbols.

In panel (a) a rather low level of sparseness has been chosen, resulting in 8 nonzero loadings in the first two PCs, whereas the plot in panel (b) refers to very sparse PCs yielding only 3 nonzero loadings in the first two PCs. The first component in panel (a) is apparently influenced by variables X1 and X2, and also contains a small loading of variable X5, whereas in panel (b) the first PC solely contains variables X1 and X2. The second PC in panel (a) contains loadings of all variables but X4, but in the more sparse example in panel (b) the second PC is equal to variable X6.
The costs for improving the principal components' interpretability by decreasing the number of non-zero loadings is the loss of explained variance. Thus the first two non-sparse PCs explain 70% of the total variance, the explained variance of the PCs in panel (a) drops to 60%, and to 57% when considering the even more sparse PCs in panel (b). However, if sparseness and thus interpretability of the components have priority, the decrease of 13% of explained variance compared to the decrease of the number of non-zero loadings from 12 to 3 is surely acceptable.

1.7 Robust Cluster Analysis

Cluster analysis is a set of statistical methods which detect similarities among observations, and group them together to a limited number of homogeneous clusters with a possibly large extent of heterogeneity among them.

Thus, these methods are not based on the assumption that the majority of the observations is identically distributed. This is contrary to the methods discussed so far, as they share the assumption, that a given data set consists of one main structure followed by more than half of the observations, whereas the influence of possible outliers which are differently distributed is down-weighted.
Clustering methods are divided into two main categories:

- **Partitional clustering** methods divide a set of observations into a fixed number of disjoint subsets (clusters). Usually assumptions on the geometric extension of the clusters are made (i.e. assumptions on the distributions and/or their parameters, see 5.3).

- **Hierarchical clustering** methods (Ward, 1963) do not make clear assignments of observations to clusters, but represent the relations between the observations in a tree structure. Only the subsequent interpretation of this hierarchic structure yields a clear cluster assignment. The found clusters however do not necessarily underlie any distribution assumptions.

Throughout this work the family of partitional clustering methods is considered. In this context robustness is usually introduced by identifying and trimming the most outlying observations (see e.g. Cuesta-Albertos *et al.*, 1997).
The parameters of such a robust clustering approach, the trimming level α and the number of groups k are usually chosen in advance. Approaches for selecting them exist and are based on multiply applying the algorithm with different values for α and k, comparing the results and then choosing the solution which fits best. For comparing different results obtained by varying these parameters either some criterion may be computed (see Neykov *et al.*, 2007), or exploratory methods may be applied (see 5.7).
Methods for choosing the number of clusters do still require human supervision, as they sometimes deliver unintended results (see 5.6). Approaches for automatically selecting the number of clusters and the trimming level simultaneously do not exist yet, as for a given problem there might exist more than one plausible answer to the question of the number of clusters and the trimming level. For example, the question whether some outliers are irregular observations or are already forming their own cluster is subject to the user's interpretation and view of the problem (see 5.7).

1.7.1 Methods

- tk-means

Cuesta-Albertos *et al.* (1997) proposed tk-means as a robust extension of the simpler k-means approach (see MacQueen, 1967). The idea is to split n

observations into $k+1$ disjoint partitions (clusters) $R_0, \ldots, R_k$, solving the minimization

$$\underset{R_0,\ldots,R_k}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{\boldsymbol{x}_i \in R_j} \|\boldsymbol{x}_i - \boldsymbol{m}_j\|, \tag{1.7.1}$$

with $\boldsymbol{m}_j$ being the component-wise mean of all observations belonging to R_j, and $\|\cdot\|$ the L_2 norm. Note, that observations in partition R_0 are not considered in this optimization and are thus trimmed, which yields a certain robustness of the method. The number of observations in R_0 is defined as $\#R_0 = \lceil \alpha n \rceil$, with α, the chosen trimming level. Apparently, when minimizing (1.7.1) the most outlying observations end up in R_0, as they yield the largest distance to all cluster centers $\boldsymbol{m}_j$. The original k-means approach did not consider an additional group R_0 for the $\lceil \alpha n \rceil$ most distant observations, and thus does not have any robustness properties. As the objective function in Equation (1.7.1) considers the Euclidean distance of each observation to its group's center the method assumes spherically structured clusters. Further the cluster sizes are not considered in the objective function, thus all clusters are given the same weight, implicitly assuming that the cluster sizes are equal.
An algorithm for this method can be formulated, as after randomly choosing k cluster centers $\boldsymbol{m}_j$ a so called E-step is applied: For each observations $\boldsymbol{x}_i$ a value d_j is computed, which represents the pertinence of an observation $\boldsymbol{x}_i$ to its closest cluster

$$d_i = \max_{j=1,\ldots,k} \frac{1}{\|\boldsymbol{x}_i - \boldsymbol{m}_j\|}. \tag{1.7.2}$$

The observations yielding the smallest $\lceil \alpha n \rceil$ values of vector $\boldsymbol{d}$ are then assigned to R_0, whereas each remaining observation $\boldsymbol{x}_i$ is assigned to the cluster which yields the lowest Euclidean Distance $\|\boldsymbol{x}_i - \boldsymbol{m}_j\|$ for $j = 1, \ldots, k$.
In a second M-step the cluster centers $\boldsymbol{m}_j$ are re-estimated

$$\boldsymbol{m}_j = \frac{1}{n} \sum_{\boldsymbol{x}_i \in R_j} \boldsymbol{x}_i$$

for $j = 1, \ldots, k$. E- and M-steps are applied alternately until a specified number of iterations is reached, or the algorithm converges. Convergence is determined by comparing the change of the cluster centers $\boldsymbol{m}_j$ between two iterations. Due to the non-convex objective function the algorithm converges at a local optimum and thus has to be started several times with different initial centers for obtaining a reasonable solution.

- tclust

The tclust method (see García-Escudero *et al.*, 2008) can be seen as an extension of tk-means, as some properties of the fast MCD algorithm are

added to this approach. This results in an EM-method with trimming based on the objective function

$$\underset{R_0,\ldots,R_k}{\operatorname{argmax}} \sum_{j=1}^{k} \sum_{\boldsymbol{x}_i \in R_j} \log\left(p_j \phi\left(\boldsymbol{x}_i; \boldsymbol{m}_j, \boldsymbol{S}_j\right)\right),$$

with $p_j = \#R_j / \left(n - \#R_0\right)$ as the cluster weights, ϕ the p-dimensional normal distribiution's density function and $\boldsymbol{S}_j$, the cluster scatter matrices. This type of objective function eliminates the discussed disadvantages of tk-means, but unfortunately yields a new problem, as it is unbounded (Maronna and Jacovkis, 1974). A single cluster scatter matrix with rank lower than p would yield an infinite objective function's value. Thus so called *spurious clusters* (see Gallegos and Ritter, 2005; García-Escudero *et al.*, 2008), clusters lying on a hyperplane of dimension lower than p, would be preferred by such a method, which is usually not desired. This problem is resolved by constraining the cluster scatter matrices, such that they always have full rank, which is done by restricting the relative range of their eigenvalues or determinants. This goes back to Hathaway (1985), who used related constraints in the context of mixture fitting methods. Appropriate constraints for obtaining a bounded objective function, and an algorithm for imposing them are discussed in Chapters 5 and 6, as well as an implementation of the complete tclust method.

Defining an algorithm for the tclust method is quite straightforward, as its basic structure is very similar to the mentioned tk-means algorithm, yielding the following differences:

In the M-step the cluster scatter matrices $\tilde{\boldsymbol{S}}_j$ are computed additionally

$$\tilde{\boldsymbol{S}}_j = \frac{1}{n_j} \sum_{i \in R_j} \left(\boldsymbol{x}_i - \boldsymbol{m}_i\right)\left(\boldsymbol{x}_i - \boldsymbol{m}_i\right)^\top,$$

with $n_j = \#R_j$, the number of observations assigned to cluster j. Subsequently the estimated cluster scatter matrices are constrained by a given restriction for avoiding spurious clusters (see García-Escudero *et al.*, 2008, and Chapters 5 and 6).

Further the vector $\boldsymbol{d}$ is computed differently (compare Equation 1.7.2), as here each cluster's constrained scatter matrix $\boldsymbol{S}_j$ is taken into account when quantifying the pertinence of an observation $\boldsymbol{x}_i$ to a cluster j

$$d_i = \max_{j=1,\ldots,k} \left(p_j \phi\left(\boldsymbol{x}_i; \boldsymbol{m}_j, \boldsymbol{S}_j\right)\right). \tag{1.7.3}$$

Corresponding to the tk-means method, the E- and M-steps are executed until convergence. Several runs of the algorithm, each one randomly initialized, are necessary for obtaining a proper solution, as each run converges in a local optimum.

The selection of the parameters α and k is suggested to be done by exploratory tools as discussed in Chapter 5 and García-Escudero *et al.* (2011).

- tle

Neykov *et al.* (2007) describe a classification algorithm which is closely related to tclust, with the difference that singular cluster scatter matrices are avoided by restricting the number of observations in each cluster, rather than by imposing any constraints on the cluster scatter matrices. In contrast to tclust this restriction is defined strictly and cannot be controlled by the user. Apart from this difference these methods optimize the same objective function, although tle allows for different distribution families, whereas tclust focuses on normally distributed clusters. Further, tle robustly solves regression problems of mixed components, which is not considered in tclust. For selecting the number of clusters k, Neykov *et al.* (2007) discuss a BIC type criterion.

- mclust

The mclust approach by Banfield and Raftery (1993) and Fraley and Raftery (1998) performs similarly as tlcust and tle, but is not a trimming method in the classical sense, as background noise is tried to be explained by a uniformly distributed component, which yields problems especially when outliers follow a specific pattern (see 5.6).
In combination with the tclust method also a BIC type criterion is used for choosing the number of clusters, however this approach is misleading if e.g. the data set features structured noise.

- Further Approaches

Gallegos (2002) introduces a clustering method similar to tclust which restricts all cluster scatter matrices' determinants to be equal. Also Gallegos and Ritter (2005) present a closely related approach which restricts the cluster scatter matrices by averaging them, which is the robustified version of a method by Friedman and Rubin (1967). The completely different EMMIX method (McLachlan and Peel, 2000) is an approach for finding multivariate t-distributed clusters.

1.7.2 Example

Figure 1.6 shows the clustering results for the tk-means and tclust algorithms applied to a data set of observations drawn from two different, slightly overlapping normal distributions forming a "T"-shape, including a 10% outlier portion. The two drawn groups contain 100 observations each, yielding a number of 22 simulated outliers. Both algorithms were advised to search for $k = 2$ clusters and to consider a contamination level of 10%. The solution in panel (a) is calculated using tk-means and shows clearly how this method assumes spherical clusters. The resulting covariance ellipsoids (solid) reduce

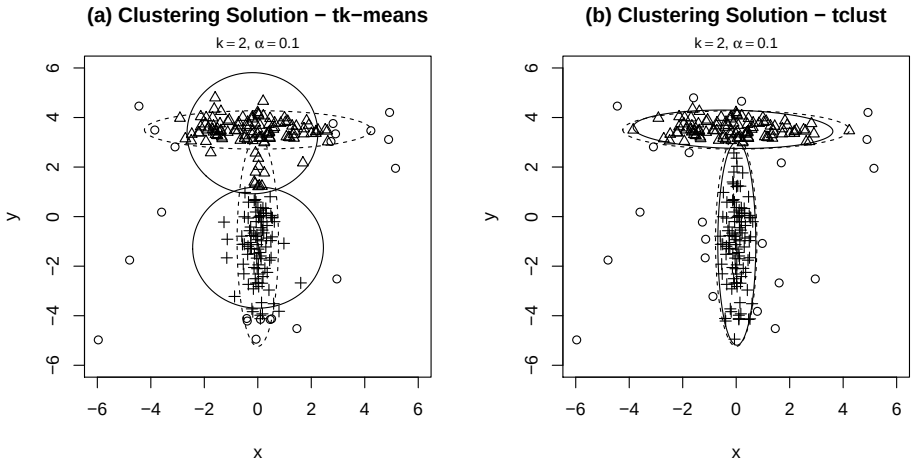

Figure 1.6: Classification result for tk-means (a) and tclust (b) computed on a data set drawn from two different normal distributions forming a "T"-shape, containing a 10% outlier portion, including the estimated covariance ellipsoids (solid) and the theoretical structure (dashed). The identified outliers ($\circ$) and the cluster assignment of each observation ($\triangle, +$) are illustrated by means of different symbols.

to circles, which apparently differ from the theoretical covariance structure (dashed), which strongly influences the cluster assignment. Referring to the *upper* cluster as to the horizontal bar and to the *lower* cluster as to the vertical bar of the mentioned "T"-shape, the upper cluster's structure is not found correctly, as some observations to its left and right are trimmed wrongly. Moreover some observations originally belonging to the lower cluster are incorrectly assigned to the upper cluster, which again is caused by assuming spherical cluster structures. The tclust algorithm however is more flexible and allows almost arbitrary covariance structures. In Figure 1.6 (b) it captures the "T"-structure quite well, as the theoretical and estimated covariance ellipsoids of both clusters are almost identical. Slight differences can be explained by the sampling error, and the fact that some of the outliers in the original data set overlap with the clusters. These observations are thus assigned to one of the clusters, whereas some regular observations in the outer regions of the clusters are incorrectly classified as outliers. This can be resolved by adjusting the trimming level appropriately.
However, as already mentioned, the question of the *real* contamination level or number of clusters cannot be answered clearly without having more information on the theoretical structure of the data set. Considering single observations in the outer regions of the clusters as outliers or as regular observations is both valid, as well as the introduction of a third big cluster,

holding all outliers, yielding the parameter constellation $k = 3$, $\alpha = 0$.

1.8 Outline of the Thesis

In this work the properties of existing multivariate robust methods are investigated in terms of efficiency, precision or runtime. New robust methods are developed and compared to existing approaches with similar properties. This first, introductory chapter presents the most important methods in the context of this work and gives an overview on related publications in the literature. The second chapter focuses on the investigation of the properties of various L_1-median algorithms, followed by a chapter on robust PCA in the context of high-dimensional data. The fourth chapter introduces the first robust and sparse approach to PCA in literature. A theoretical approach to robust trimmed clustering accompanied by the presentation of an according implementation is given in the fifth chapter, followed by a chapter discussing some algorithmic details of the same clustering approach. The last chapter gives an insight in the implementational structure of the related software packages.
All developed methods are implemented in **R** (Development Core Team, 2010), whereas the runtime-critical elements have been exported to **C++**, combining the flexibility and speed of both programming environments.

Chapter 2 investigates the properties of various algorithms for the computation of the L_1-median. Four general purpose optimizers (Nelder and Mead, 1965; Nocedal and Wright, 2006; Fletcher and Reeves, 1964; Dennis and Schnabel, 1983) are compared to two more specific algorithms, particularly developed for this purpose (Vardi and Zhang, 2000; Hössjer and Croux, 1995). Next to the numerical stability and the convergence behaviour, several simulation examples compare the speed of the algorithms, as well as their performance on degenerated and high-dimensional data sets.

Fritz H, Filzmoser P, Croux C (2011). A comparison of algorithms for the multivariate L_1-median. *Computational Statistics*, pp. 1–18.

Chapter 3 examines the possibility of outlier detection in high-dimensional data sets by means of robust PCA, using the projection pursuit based Grid algorithm (Croux *et al.*, 2007). Diagnostic plots based on distance-distance plots (Hubert *et al.*, 2005) are developed, which are used for examining a high-dimensional real data set.

Filzmoser P, Fritz H (2007). Exploring high-dimensional data with robust principal components. In S Aivazian, P Filzmoser, Y Kharin (eds.), *Proceedings of the Eighth International Conference on Computer Data Anal-*

ysis and Modeling, volume 1, pp. 43–50. Belarusian State University, Minsk.

Chapter 4 develops a PCA approach, which combines both robustness and sparseness properties in one single method, being the first of its kind. The algorithm is an extension of the well established Grid algorithm (Croux *et al.*, 2007) and gains its additional properties only by modifications of the objective function. Simulation studies on generated and real data sets, including high-dimensional data are computed and discussed for investigating the algorithm's performance.

Croux C, Filzmoser P, Fritz H (2011). Robust Sparse Principal Component Analysis. Submitted to Technometrics.

Chapter 5 discusses the tclust method introduced in García-Escudero *et al.* (2008) and presents the first working implementation of the method. Apart from that several exploratory tools are presented for validating the algorithm's output and for choosing the according tuning parameters, like the number of clusters and the trimming level. Several examples considering generated and real data sets are presented for discussing the algorithm's performance.

Fritz H, García-Escudero L, Mayo-Iscar A (2011). tclust: An R Package for a Trimming Approach to Cluster Analysis. Submitted to Journal of Statistical Software.

Chapter 6 is a theoretical consideration of the tclust algorithm, discussing a fast algorithm for imposing the necessary constraints on the cluster scatter matrices. An approximate solution to this problem which has been firstly given in García-Escudero *et al.* (2008) is revised, resulting in an exact and computationally easy feasible algorithm. A simulation study based on generated data sets investigates the algorithm's characteristics, such as precision and convergence behaviour.

Fritz H, García-Escudero L, Mayo-Iscar A (2011). A fast algorithm for robust constrained clustering. Submitted to Metrika.

Chapter 7 concludes with insights in the structure of the software released in the context of this work. The programs are available in **R**, but the core algorithms are implemented independently in **C++**, such that an application in other environments as e.g. MATLAB (2010) or as a stand-alone implementation is feasible easily. Only two slim interface layers have to be implemented for embedding the algorithm in the corresponding environment.

Unpublished manuscript.

Chapter 2

A comparison of algorithms for the multivariate L_1-median

Summary: The L_1-median is a robust estimator of multivariate location with good statistical properties. Several algorithms for computing the L_1-median are available. Problem specific algorithms can be used, but also general optimization routines. The aim is to compare different algorithms with respect to their precision and runtime. This is possible because all considered algorithms have been implemented in a standardized manner in the open source environment R. In most situations, the algorithm based on the optimization routine NLM (non-linear minimization) clearly outperforms other approaches. Its low computation time makes applications for large and high-dimensional data feasible.
Keywords: Algorithm Multivariate median Optimization Robustness

Co-authors: Peter Filzmoser, Christophe Croux

2.1 Introduction

A prominent generalization of the univariate median to higher dimensions is the geometric median, also called L_1-median or spatial median. For a data set $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ with each $\boldsymbol{x}_i \in \mathbb{R}^p$, the L_1-median $\hat{\boldsymbol{\mu}}$ is defined as

$$\hat{\boldsymbol{\mu}}(\boldsymbol{X}) = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{\mu}\| \tag{2.1.1}$$

where $\|\cdot\|$ denotes the Euclidean norm. In words, the L_1-median is the point for which the sum of the Euclidean distances to n given data points is

minimal. This problem is formulated in an even more general form by Weber (1909) (*Fermat-Weber problem*), as he refers to location issues in industrial contexts. If the data points are not collinear, the solution to problem (2.1.1) is unique (Weiszfeld, 1937). The L_1-median has several further attractive statistical properties, like:

(a) Its breakdown point is 0.5 (Lopuhaä and Rousseeuw, 1991), i.e., only if more than 50% of the data points are contaminated, the L_1-median can take values beyond all bounds.

(b) It is location and orthogonally equivariant, that is for any $\boldsymbol{b} \in I\!R^p$ and orthogonal $p \times p$ matrix $\boldsymbol{L}$,

$$\hat{\boldsymbol{\mu}}(\boldsymbol{L}\boldsymbol{X} + \boldsymbol{b}) = \boldsymbol{L}\hat{\boldsymbol{\mu}}(\boldsymbol{X}) + \boldsymbol{b},$$

with $\boldsymbol{L}\boldsymbol{X} + \boldsymbol{b} = \{\boldsymbol{L}\boldsymbol{x}_1 + \boldsymbol{b}, \ldots, \boldsymbol{L}\boldsymbol{x}_n + \boldsymbol{b}\}$.

Property (a) makes this estimator very attractive from a robustness point of view. According to property (b), the L_1-median is orthogonal equivariant, but not affine equivariant. Orthogonal equivariance is already sufficient for many situations, like for principal component analysis (PCA). Therefore several authors consider the L_1-median in the context of robust PCA (e.g., Croux and Ruiz-Gazen, 2005; Croux *et al.*, 2007). Note that the L_1-median can be extended beyond the multivariate setting to functional spaces (Gervini, 2008) and Hilbert spaces (e.g. Chaudhuri, 1996; Debruyne *et al.*, 2010).
An iterative algorithm for finding the numerical solution of the L_1-median has been proposed by Weiszfeld (1937). Several other algorithms will be outlined in Section 2. The goal of this paper is to compare the algorithms with respect to their precision and runtime (Section 3). For such a comparison, a unified implementation of the algorithms has been made using C++ code embedded in the R-library `pcaPP` (Development Core Team, 2010), see Section 2.3. The final Section 4 concludes.

2.2 Algorithms for computing the L_1-median

For the computation of the estimate $\hat{\boldsymbol{\mu}}$ we have to minimize the convex function

$$S(\boldsymbol{\mu}) = \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{\mu}\|. \tag{2.2.1}$$

The algorithms tested and examined here can generally be divided into two categories:

2.2.1 General optimization procedures

The minimization of (2.2.1) can be done by numerical algorithms, developed for general, non-linear optimization purposes. One can either evaluate the target function $S(\boldsymbol{\mu})$ on several points, or additionally use the first derivative of $S(\boldsymbol{\mu})$,

$$\frac{\partial S(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = -\sum_{i=1}^{n} \frac{\boldsymbol{x}_i - \boldsymbol{\mu}}{\|\boldsymbol{x}_i - \boldsymbol{\mu}\|}, \tag{2.2.2}$$

or the Hessian matrix

$$\frac{\partial^2 S(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}\, \partial \boldsymbol{\mu}^t} = \sum_{i=1}^{n} \left(\frac{1}{\|\boldsymbol{x}_i - \boldsymbol{\mu}\|}\, \boldsymbol{I}_p - \frac{(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^t}{\|\boldsymbol{x}_i - \boldsymbol{\mu}\|^3} \right), \tag{2.2.3}$$

where $\boldsymbol{I}_p$ is the $p \times p$ identity matrix. Since all algorithms are implemented in the software environment R (Development Core Team, 2010), we considered general unconstrained non-linear optimization algorithms which are accessible in this environment for our purpose:

- **NM:** Nelder and Mead (1965) proposed a simplex method for minimizing functions of p variables, also known as downhill simplex method. Values of the target function are compared at $p+1$ points, whereas the point with highest target value is replaced in each iteration. Rather many iterations are needed till convergence, but as this is one of the most common simplex algorithms, it is included in this comparison.

- **BFGS:** Broyden, Fletcher, Goldfarb and Shanno proposed a quasi-Newton algorithm searching for a stationary point of a function by local quadratic approximation (see, e.g., Nocedal and Wright, 2006). In contrast to *real* Newton methods, this algorithm does not require an analytical computation of the exact Hessian matrix, as this is approximated internally by the algorithm. Since in high-dimensional situations the computation of the exact Hessian matrix can be quite time consuming, algorithms that approximate the Hessian matrix internally are to be preferred in this context.

- **CG:** Conjugate gradient algorithms are iterative methods, known for their low memory requirements. Quasi-Newton methods usually converge after fewer iterations, but a single iteration of a conjugate gradient method is computationally less intensive. In the subsequent simulations, the version of Fletcher and Reeves (1964) is applied. The gradient information in equation (2.2.2) is used by this method.

- **NLM:** Non-linear minimization can be carried out by a Newton-type algorithm (Dennis and Schnabel, 1983), using the gradient information from equation (2.2.2). It provides two options regarding the Hessian matrix:

an analytical representation can be provided, or the Hessian matrix is again approximated internally by secant methods. Approximating the Hessian matrix turned out to be the faster approach, particularly when high-dimensional data sets are processed. Hence in subsequent simulations the Hessian matrix is always approximated rather than analytically calculated. From the three available optimization types introduced in Dennis and Schnabel (1983), *Line Search*, *Double Dogleg*, and *More-Hebdon*, the first method is applied here, as in this context its convergence characteristics turned out to be most reliable among these three.

2.2.2 Problem specific algorithms

For the specific problem of computing the L_1-median, several algorithms have been proposed in the literature. Here we mention the algorithm of Weiszfeld (1937), which is the basis for an improved version by Vardi and Zhang (2000). Another algorithm, based on the steepest descent, has been introduced by Hössjer and Croux (1995), and will also be examined here.

- Weiszfeld (1937) formulated an iterative approach for solving the Fermat-Weber problem for data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and n non-negative weights. In this paper all weights are equal. A current (or initial) solution $\hat{\boldsymbol{\mu}}_l$ is improved by calculating a scaled sum of all observations:

$$T_0(\boldsymbol{\mu}) = \frac{\sum_{i=1}^{n} \frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i - \boldsymbol{\mu}\|}}{\sum_{i=1}^{n} \frac{1}{\|\boldsymbol{x}_i - \boldsymbol{\mu}\|}} \tag{2.2.4}$$

$$\hat{\boldsymbol{\mu}}_{l+1} = \begin{cases} T_0(\hat{\boldsymbol{\mu}}_l) & \text{if } \hat{\boldsymbol{\mu}}_l \notin \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \\ \hat{\boldsymbol{\mu}}_l & \text{if } \hat{\boldsymbol{\mu}}_l \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \end{cases} \tag{2.2.5}$$

  This algorithm converges given that $\hat{\boldsymbol{\mu}}_l \notin \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ for every iteration step $l \in \mathbb{N}$.

- **VaZh:** Vardi and Zhang (2000) improved the behavior of Weiszfeld's algorithm in case that $\hat{\boldsymbol{\mu}}_l \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ appears in any iteration. The resulting algorithm is, apart from the treatment of this particular issue, quite similar, and for this reason no comparison is made with Weiszfeld's original algorithm later on in the simulation study. The proposal of Vardi

and Zhang (2000) is as follows:

$$T_1(\boldsymbol{\mu}) = \frac{\sum\limits_{\boldsymbol{x}_i \neq \boldsymbol{\mu}} \frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i - \boldsymbol{\mu}\|}}{\sum\limits_{\boldsymbol{x}_i \neq \boldsymbol{\mu}} \frac{1}{\|\boldsymbol{x}_i - \boldsymbol{\mu}\|}} \tag{2.2.6}$$

$$\eta(\boldsymbol{\mu}) = \begin{cases} 1 & \text{if } \boldsymbol{\mu} = \boldsymbol{x}_i,\ i = 1, \ldots, n \\ 0 & \text{else} \end{cases} \tag{2.2.7}$$

$$R(\boldsymbol{\mu}) = \sum_{\boldsymbol{x}_i \neq \boldsymbol{\mu}} \frac{\boldsymbol{x}_i - \boldsymbol{\mu}}{\|\boldsymbol{x}_i - \boldsymbol{\mu}\|} \tag{2.2.8}$$

$$\gamma(\boldsymbol{\mu}) = \min\left(1, \frac{\eta(\boldsymbol{\mu})}{\|R(\boldsymbol{\mu})\|}\right) \tag{2.2.9}$$

$$\hat{\boldsymbol{\mu}}_{l+1} = (1 - \gamma(\hat{\boldsymbol{\mu}}_l))\, T_1(\hat{\boldsymbol{\mu}}_l) + \gamma(\hat{\boldsymbol{\mu}}_l)\hat{\boldsymbol{\mu}}_l. \tag{2.2.10}$$

The case $\hat{\boldsymbol{\mu}}_l \notin \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ implies $\gamma(\hat{\boldsymbol{\mu}}_l) = 0$, and the algorithm behaves exactly as in (2.2.4). Otherwise, if $\hat{\boldsymbol{\mu}}_l \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, the sum in (2.2.6) is calculated as in (2.2.4), but only over the $\boldsymbol{x}_i \neq \hat{\boldsymbol{\mu}}_l$, whereas observation $\boldsymbol{x}_i = \hat{\boldsymbol{\mu}}_l$ is added afterwards in (2.2.10), applying weight $\gamma(\hat{\boldsymbol{\mu}}_l)$.

- **HoCr:** Hössjer and Croux (1995) proposed an approach which combines a steepest descent algorithm with step halving. The current solution is improved by stepping into the direction with steepest descent of the target function. If the target function increases after this step, the step size is being halved until the target function decreases. If the step size shrinks to zero without finding a better solution, the algorithm has converged. A detailed description of the algorithm including pseudocode is given by the authors. Their algorithm can also be used for the rank based extension of the L_1-median they propose.

For reasons of completeness, let us mention two other algorithms proposed in the literature for computing the L_1-median. Gower (1974) proposed a steepest descent algorithm combined with a bisection algorithm. It is somehow similar to the HoCr algorithm, but the use of the bisection method instead of step-halving considerably increases the computation time. Bedall and Zimmermann (1979) proposed to use the Newton-Raphson procedure with the exact expression (2.2.3) for the Hessian matrix, and is similar to the NLM method with analytical second derivative. It turned out to be much slower than the NLM procedure. The algorithms of Gower (1974) and Bedall and Zimmermann (1979) are included in the R-package `depth`. Due to their similarity with algorithms already included in the simulation study, and since they are not competitive in terms of computation speed, we do not report their performance in this paper.

2.2.3 Implementation

Due to efficiency issues (runtime and memory), all methods are implemented in C++ and are embedded in the R-library `pcaPP` (version 1.8-1). The algorithms based on general optimization methods are simply wrapping the available routines in R, see Table 2.1. The routine for non-linear minimization

Algorithm	R - Optimizer	`pcaPP` - Routine
Nelder and Mead (NM)	`nmmin`	`l1median_NM`
Broyden, Fletcher, Goldfarb and Shanno (BFGS)	`vmmin`	`l1median_BFGS`
Conjugate gradient (CG)	`cgmin`	`l1median_CG`
Non-linear minimization (NLM)	`optif9`	`l1median_NLM`
Vardi and Zhang (VaZh)		`l1median_VaZh`
Hössjer and Croux (HoCr)		`l1median_HoCr`

Table 2.1: Implementation of the optimization methods as wrapper functions.

(`optif9`) originates from the UNCMIN-Fortran package by R.B. Schnabel which implements Newton and quasi-Newton algorithms for unconstrained minimization, see Dennis and Schnabel (1983), Schnabel *et al.* (1985). All other mentioned routines are C-implementations which come along with R. By default the component-wise median is used as starting value for all mentioned algorithms.
The more specific L_1-median routines (Vardi and Zhang, 2000; Hössjer and Croux, 1995) are transcripts of the routines published in the R-library `robustX`, whereas the original implementation of the algorithm by Hössjer and Croux (1995) was made available by the authors. The implementation of the algorithm of Vardi and Zhang (2000) is changed slightly, as the original algorithm crashes if $\boldsymbol{\mu} = \boldsymbol{x}_i$ for more than one $i \in \{1, \ldots, n\}$, see equation (2.2.7). Although this might be a rare situation, equation (2.2.7) needs to be changed to

$$\eta(\boldsymbol{\mu}) = \sum_{\boldsymbol{x}_i = \boldsymbol{\mu}} 1 \tag{2.2.11}$$

in order to stabilize the algorithm in such degenerated cases.

2.2.4 Convergence

When comparing different ways of solving a problem, one major issue is to create equal and thus fair conditions for each approach which is to be examined. Here each algorithm can be tested providing exactly the same input, whereas the output quality can be compared likewise by checking the

value of the target function $S(\hat{\boldsymbol{\mu}})$ of (2.2.1). However, it is not as easy to control the precision of the results of each particular algorithm. Each of the mentioned algorithms provides an input control parameter τ, which is a tolerance level influencing the convergence behavior, and is mainly used to specify the desired precision. Unfortunately this tolerance level is interpreted differently, as shown in Table 2.2.

Algorithm	Convergence criterion	Based on
NM BFGS CG	$S(\hat{\boldsymbol{\mu}}_{l-1}) - S(\hat{\boldsymbol{\mu}}_l) \leq \tau\,(S(\hat{\boldsymbol{\mu}}_l) + \tau)$	Relative change of $S(\hat{\boldsymbol{\mu}})$
NLM	$\bigtriangledown := \frac{\left\lvert \frac{\partial S(\hat{\boldsymbol{\mu}}_l)}{\partial \hat{\boldsymbol{\mu}}_l} \right\rvert_{\max\{\|\hat{\boldsymbol{\mu}}_l\|,1\}}}{\max\{\lvert S(\hat{\boldsymbol{\mu}}_l)\rvert,1\}}$ $\max_i \bigtriangledown_i \leq \tau$ with $\bigtriangledown = (\bigtriangledown_1, \ldots, \bigtriangledown_p)^t$	Maximum scaled gradient
VaZh	$\|\hat{\boldsymbol{\mu}}_{l-1} - \hat{\boldsymbol{\mu}}_l\|_1 \leq \tau \|\hat{\boldsymbol{\mu}}_l\|_1$	Relative change of $\hat{\boldsymbol{\mu}}$ (L_1-norm)
HoCr	$\|\hat{\boldsymbol{\mu}}_{l-1} - \hat{\boldsymbol{\mu}}_l\|_2 \leq \tau$	Change of $\hat{\boldsymbol{\mu}}$ (L_2-norm)

Table 2.2: Convergence criteria for different optimization routines.

In order to still get comparable results, we have to choose τ appropriately. This is done by examining first, which tolerance level τ leads to comparable precision in a particular situation, and then these tolerance levels are applied in all subsequent simulation settings. Details are provided in the next section.

2.3 Comparison of the algorithms

In order to test the performance of the algorithms, several types of artificial and real data samples are chosen. On a particular data set, the L_1-median is computed with each algorithm discussed above and the resulting estimations are compared by considering the values of the target function (2.2.1). The smallest resulting target value is used as reference, whereas deviations from this reference value indicate worse approximations of the L_1-median. Throughout this paper, we use this *deviation* as a measure of precision of an algorithm. The best algorithm(s) always yield a deviation of exactly zero. In the following simulated scenarios, each simulation is performed 100 times with data sets sampled from the same distribution. As measure for overall precision the 95% quantile of the resulting deviations is considered, referred to as the *95% deviation quantile*. The motivation for this measure is that it reflects the precision of the vast majority of runs, but does not account for single runs where the convergence behavior was different just by chance. Additionally, the (median) runtime in seconds is reported for each

algorithm. All simulations are computed on an AMD Athlon 64 X2 4200+ Processor at 2.2 GHz.
In order to stop non converging algorithms from freezing the process, the maximum number of iterations is set to 500 for each algorithm. If not stated differently, the simulated data sets consist of $n = 1000$ observations and $p = 100$ variables. The covariance matrix $\boldsymbol{C}$ of the distribution used to generate the data is diagonal with diagonal elements $p+1-i$, for $i = 1, \ldots, p$. Outliers are generated by randomly selecting observations from the simulated data matrix, multiplying them with a value of 10, and shifting them in each coordinate by the value 10.

2.3.1 Adjusting tolerance levels for convergence

Before comparing any results of the different algorithms, their convergence criteria shall be examined and their tolerance levels need to be adjusted, such that they deliver equal precision for a particular simulation setting. For that purpose a simple simulation is performed with several tolerance levels τ, monitoring the resulting precisions. This allows to obtain a tolerance level for each algorithm leading to "optimal" precisions. In the subsequent simulations, these tolerance levels are used and so the resulting figures are directly comparable. The algorithms are applied to 100 different p-variate normally distributed data sets (we used $n = 1000$ and $p = 100$ in the following) with center $\mathbf{0}$, covariance matrix $\boldsymbol{C}$ (see above) and without outliers. The tolerance level is altered between 1e-1 and 1e-20. Figure 2.1 shows the 95% deviation quantiles of each algorithm computed over different tolerance levels. Note that a 95% deviation quantile of exactly 0 has been truncated to 1e-15 for being able to use logarithmic scales. A value of 1e-15 (or below) is reached at a tolerance level of 1e-5 for HoCr, at 1e-6 for VaZh, at 1e-9 for CG, and at 1e-11 for NLM. This high precision cannot be reached for the algorithms BFGS and NM within the considered range of the tolerance levels; NM seems to be totally unaffected by the choice of the tolerance level. Also changing the different tuning parameters of the algorithm NM does not lead to better results in a simulation as presented here.
Considering the computation time of the algorithms with respect to the used tolerance level τ (Figure 2.2), we note that the runtime of the algorithm NLM seems to be rather unaffected by the specified value of τ. On the other hand, CG shows a large increase in runtime when τ is raised from 1e-11 to 1e-13. On the whole, the relative ranking of the different algorithms remains about the same for different levels of τ. Hence, the faster algorithms when requiring a high tolerance level will also be the faster ones when only a rude approximation is needed. In particular this implies that it will not be advantageous to use the output of one algorithm as starting value for another algorithm. For this reason, we stick to the coordinatewise median as a starting value for all the algorithms we consider.

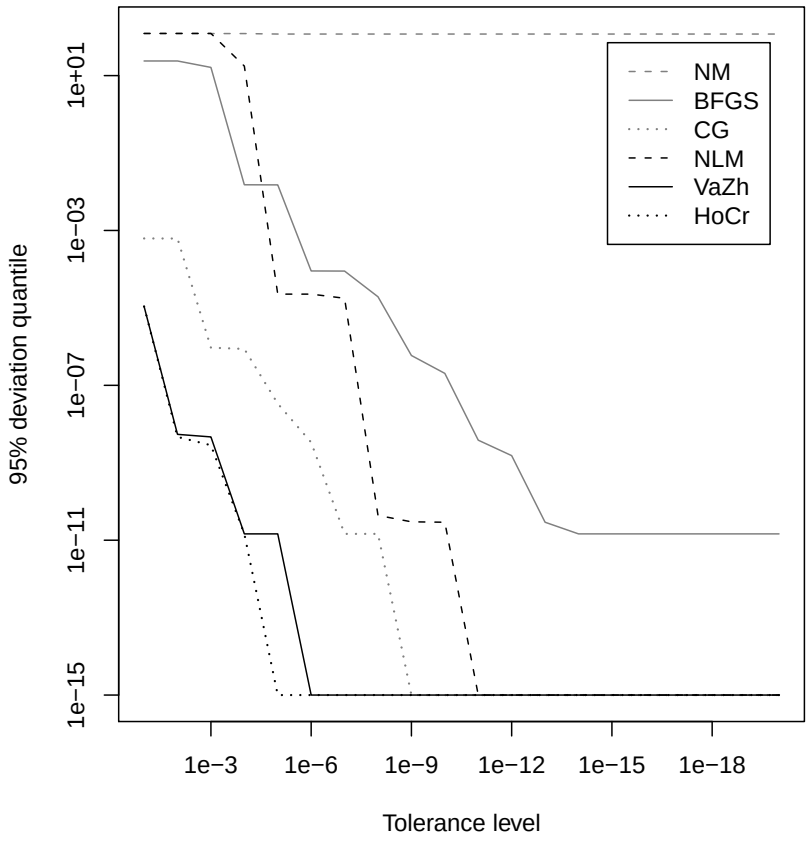

Figure 2.1: The 95% deviation quantiles of the L_1-median target values as a function of the tolerance level τ.

Table 2.3 shows the chosen tolerance levels for subsequent simulations. The largest level of τ at which each algorithm reaches its best performance (zero deviation) is chosen and then scaled down by a factor 1e3, compensating for different simulation scenarios, as the data structure might slightly influence the convergence behavior. With respect to the high increase of the runtime of the algorithm CG when τ excesses 1e-11, this algorithm's tolerance level is set to 1e-10. As method NM does not seem to be affected by the tolerance level, the choice of $\tau =$ 1e-10 is admittedly arbitrary.

NM	BFGS	CG	NLM	VaZh	HoCr
1e-10	1e-17	1e-10	1e-14	1e-9	1e-8

Table 2.3: Tolerance levels τ as used for each particular algorithm.

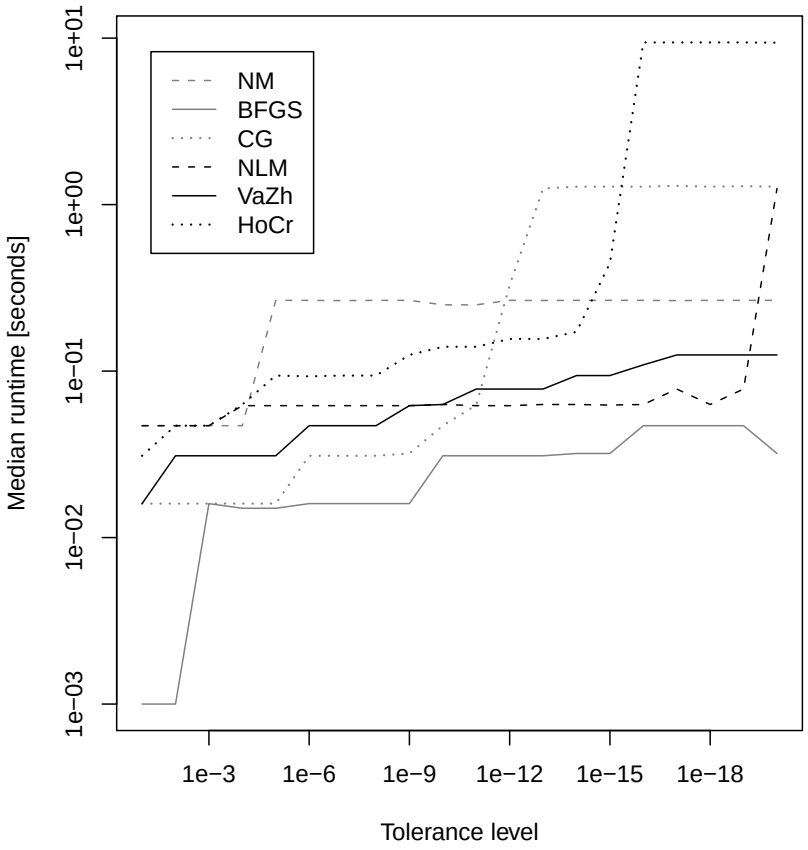

Figure 2.2: Median runtimes [seconds] of the L_1-median computation as a function of the tolerance level τ.

By selecting the tolerance level τ as outlined above, we aim at optimizing the precision of the algorithm, as measured by the 95% quantile of the deviation. This is not implying that all considered algorithms end up with the same precision, but it does turn out that the best performing algorithms have precisions very close to each other. We then discriminate between these best performing algorithms by comparing their run times. An alternative strategy might have been to control the running times, and then compare the attained precision. We did not pursue this alternative strategy since, besides several other implementation issues, a tolerance level τ still needs to be specified, and it is not so clear how to do this.

2.3.2 Uncorrelated data

The observations are drawn randomly from p-variate normal (N_p) and log-normal (LN_p) distributions, respectively, with center $\mathbf{0}$ and covariance ma-

trix $\boldsymbol{C}$. Simulations on multivariate t-distributed data have been computed too, but as the algorithms perform quite similar as on normally distributed data, these results are omitted here. In order to get an impression on how the algorithms are relatively affected by outliers, the percentage of outliers, p_{out}, varies between 0% and 40% . Table 2.3.2 shows the 95% deviation quantiles of each algorithm, for the given distributions and different percentages of outliers. For $p_{out} = 0$ (and normally distributed data) this is the same simulation setting as before when the tolerance levels have been chosen, and it is not surprising that apart from BFGS and NM all algorithms return the same precision, yielding a 95% deviation quantile of 0. As the concept of the L_1-median may downweight but never completely trims any observation, added outliers always slightly influence the estimation. However, increasing p_{out} does not influence the algorithms' relative performance, as none of the methods can be identified as particularly sensitive to a different outlier proportion among the tested candidates. The runtime behavior in this simulation set-

Distribution	p_{out}	NM	BFGS	CG	NLM	VaZh	HoCr
$N_p(\mathbf{0}, \mathbf{C})$	0 %	115	1.46e-11	0	0	0	0
	10 %	118	2.91e-11	0	0	0	0
	20 %	133	1.46e-12	0	0	0	0
	30 %	147	0	0	0	0	0
	40 %	146	5.82e-11	0	0	0	0
$LN_p(\mathbf{0}, \mathbf{C})$	0 %	1.66	1.19e-13	0	0	0	0
	10 %	3.51	4.77e-13	0	0	0	0
	20 %	5.98	9.09e-13	0	0	0	0
	30 %	9.60	9.09e-13	0	0	0	0
	40 %	16.20	1.82e-12	0	0	0	0

Table 2.4: 95% deviation quantiles of L_1-median algorithms applied to uncorrelated data.

ting is shown in Table 2.3.2. It is not possible to point out one algorithm which is always the fastest while delivering the best approximation quality. On normally distributed data, BFGS is the fastest algorithm, but as seen before it is not as accurate as other methods. The runtimes of the algorithms seem to be quite independent from the added amount of outliers, and they are all in about the same range (apart from NM being always the slowest choice which obviously is caused by convergence issues). Applying the algorithms on log-normally distributed data, however, yields the best results for the NLM algorithm. In terms of computation time, NLM is the only algorithm which is not heavily influenced by the distribution type, nor by the amount of outliers added. For the algorithms VaZh, HoCr and CG, the data configuration has a notable effect since their runtimes increase up to a factor of 6 (CG) when highly contaminated data are processed. The algorithm of Nelder and Mead (NM) never converges within the 500 allowed iterations, which explains its high deviation values and longer runtimes.

Distribution	p_{out}	NM	BFGS	CG	NLM	VaZh	HoCr
$N_p(\mathbf{0}, \mathbf{C})$	0 %	0.265	0.032	0.047	0.063	0.062	0.094
	10 %	0.266	0.046	0.125	0.063	0.062	0.078
	20 %	0.265	0.032	0.047	0.063	0.062	0.078
	30 %	0.265	0.031	0.046	0.063	0.062	0.078
	40 %	0.265	0.031	0.047	0.063	0.062	0.078
$LN_p(\mathbf{0}, \mathbf{C})$	0 %	0.266	0.172	0.047	0.063	0.062	0.078
	10 %	0.266	0.172	0.281	0.063	0.062	0.078
	20 %	0.265	0.187	0.109	0.078	0.078	0.109
	30 %	0.265	0.141	0.211	0.078	0.110	0.141
	40 %	0.265	0.109	0.297	0.078	0.218	0.250

Table 2.5: Median runtimes [seconds] of the L_1-median algorithms applied to uncorrelated data.

Finally, to check the robustness of our findings with respect to the chosen performance measures, we show in Table 2.3.2 the average of the deviations over the simulated samples, as an alternative to the 95% quantile deviation we used before. In line with the findings from Table 2.3.2, we see that the NM algorithm is not competitive, and also BFGS does worse than the other considered algorithms. For the other algorithms, the average deviations are very small, and there is hardly any difference between them, confirming the conclusions made from the corresponding table containing the 95% quantile deviations. Furthermore, Table 2.3.2 presents the mean, instead of the median, runtime. Comparing Tables 2.3.2 and 2.3.2 shows that the mean and median runtime remain very close to each other. In the remainder of this paper, we only report the 95% quantile deviations and median runtimes.

Distribution	p_{out}	NM	BFGS	CG	NLM	VaZh	HoCr
$N_p(\mathbf{0}, \mathbf{C})$	0 %	91.1	3.2e-12	1.46e-13	1.46e-13	0	1.46e-13
	10 %	93.1	1.75e-12	0	0	2.91e-13	0
	20 %	114	1.46e-12	0	0	5.82e-13	0
	30 %	122	2.33e-12	5.82e-13	5.82e-13	0	5.82e-13
	40 %	116	6.98e-12	0	0	0	0
$LN_p(\mathbf{0}, \mathbf{C})$	0 %	1.42	5e-14	2.27e-15	2.27e-15	0	2.27e-15
	10 %	3.05	2.36e-13	9.09e-15	9.09e-15	1.82e-14	9.09e-15
	20 %	5.51	3.73e-13	9.09e-15	9.09e-15	3.64e-14	9.09e-15
	30 %	9.09	4.73e-13	1.82e-14	1.82e-14	9.09e-15	1.82e-14
	40 %	15.4	4.18e-13	3.64e-14	3.64e-14	0	3.64e-14

Table 2.6: Average deviation of the L_1-median target value applied to uncorrelated data.

Distribution	p_{out}	NM	BFGS	CG	NLM	VaZh	HoCr
$N_p(\mathbf{0}, \mathbf{C})$	0 %	0.262	0.039	0.062	0.064	0.062	0.094
	10 %	0.262	0.039	0.146	0.064	0.061	0.086
	20 %	0.260	0.039	0.090	0.065	0.062	0.083
	30 %	0.262	0.036	0.100	0.064	0.062	0.076
	40 %	0.261	0.031	0.105	0.065	0.061	0.074
$LN_p(\mathbf{0}, \mathbf{C})$	0 %	0.262	0.177	0.049	0.068	0.062	0.076
	10 %	0.262	0.177	0.245	0.070	0.062	0.086
	20 %	0.261	0.185	0.172	0.072	0.079	0.106
	30 %	0.262	0.143	0.318	0.072	0.114	0.142
	40 %	0.262	0.102	0.544	0.074	0.212	0.252

Table 2.7: Average runtimes [seconds] of the L_1-median algorithms applied to uncorrelated data.

2.3.3 Correlated data

In this section we investigate the effect of a correlation structure within the simulated data on the performance of the algorithms. Therefore, we use a covariance matrix $\boldsymbol{C'}$ for data generation, with elements 1 in the diagonal, and numbers c as off-diagonal elements. The values of c are set to 0.5, 0.9, and 0.99 among the different simulation scenarios. In addition, the effect of the data distribution and the influence of outliers is observed. Table 2.3.3 shows the precision of the algorithms. Their performance does not seem to be really influenced by the correlation level c, the only difference is observed for the algorithm HoCr which performs slightly worse for log-normally distributed highly correlated data.
According to Table 2.3.3, the only algorithms unaffected by the correlation level are NM and NLM. NM again did not converge which explains the constant runtime of 265 milliseconds. NLM however converges almost within the same time as when applied on uncorrelated data, which is outstanding among all tested methods. The algorithms CG, HoCr and VaZh require more computation time for highly correlated data.

2.3.4 High-dimensional data with low sample size ($p \gg n$)

Here we generate data according to $N_p(\mathbf{0}, \boldsymbol{I}_p)$ and $LN_p(\mathbf{0}, \boldsymbol{I}_p)$, respectively, for $n = 10$ and $p = 100$. As previously, contamination is added and its percentage is varied. Since the rank of the generated p-dimensional data is n, it is possible to reduce the dimensionality to n without any loss of information. This is done by singular value decomposition (SVD) as follows (compare Serneels *et al.*, 2005). Let $\boldsymbol{X}$ be the simulated data matrix, then $\boldsymbol{X}^t = \boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^t$ is the SVD of $\boldsymbol{X}^t$, with $\boldsymbol{U}$ an $n \times n$ orthogonal matrix, $\boldsymbol{S}$ a diagonal matrix including the n singular values in its diagonal, and $\boldsymbol{V}$ a $p \times n$ orthogonal matrix. The matrix $\tilde{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{S}$ has only size $n \times n$, but it contains

Distribution	p_{out}	r	NM	BFGS	CG	NLM	VaZh	HoCr
$N_p(\mathbf{0}, \mathbf{C}')$	0 %	0.5	19.2	1.82e-12	0	0	0	0
		0.9	23.3	1.82e-12	0	0	0	0
		0.99	31.0	1.82e-12	0	0	0	0
	30 %	0.5	33.2	7.28e-12	0	0	0	0
		0.9	22.1	7.28e-12	0	0	0	0
		0.99	57.8	7.28e-12	0	0	0	0
$LN_p(\mathbf{0}, \mathbf{C}')$	0 %	0.5	3.09	1.14e-13	0	0	0	0
		0.9	2.83	1.14e-13	0	0	0	1.14e-13
		0.99	4.34	1.14e-13	0	0	0	2.27e-13
	30 %	0.5	3.62	9.09e-13	0	0	0	0
		0.9	3.13	9.09e-13	0	0	0	0
		0.99	3.42	9.09e-13	0	0	0	9.09e-13

Table 2.8: 95% deviation quantiles of L_1-median algorithms applied to correlated data.

Distribution	p_{out}	c	NM	BFGS	CG	NLM	VaZh	HoCr
$N_p(\mathbf{0}, \mathbf{C}')$	0 %	0.5	0.265	0.078	0.110	0.063	0.156	0.219
		0.9	0.265	0.094	0.296	0.078	0.281	0.422
		0.99	0.265	0.125	0.344	0.078	0.468	0.688
	30 %	0.5	0.265	0.063	0.297	0.078	0.156	0.188
		0.9	0.265	0.078	0.593	0.078	0.250	0.328
		0.99	0.265	0.110	0.718	0.078	0.422	0.547
$LN_p(\mathbf{0}, \mathbf{C}')$	0 %	0.5	0.266	0.157	0.141	0.078	0.156	0.203
		0.9	0.265	0.156	0.211	0.078	0.282	0.359
		0.99	0.266	0.140	0.375	0.078	0.453	0.570
	30 %	0.5	0.265	0.094	0.539	0.078	0.172	0.203
		0.9	0.265	0.156	0.609	0.078	0.265	0.313
		0.99	0.266	0.141	1.258	0.078	0.406	0.500

Table 2.9: Median runtimes [seconds] of the L_1-median algorithms applied to correlated data.

the same information as $\boldsymbol{X}$. Using this dimension reduction, we can apply the L_1-median algorithms to both $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$, and compare the resulting estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\tilde{\boldsymbol{\mu}}}$, respectively. This can be done by back-transforming the column vector $\hat{\tilde{\boldsymbol{\mu}}}$ to the original space with $\boldsymbol{V}\hat{\tilde{\boldsymbol{\mu}}}$, and comparing the results with the Euclidean distance

$$\delta = \|\hat{\boldsymbol{\mu}} - \boldsymbol{V}\hat{\tilde{\boldsymbol{\mu}}}\|. \tag{2.3.1}$$

This difference between estimations in transformed and original space is computed 100 times for each simulation setting, and the 95% distance quantile is reported in Table 2.3.4. As the algorithm NLM provided the best results in previous simulations, it is rather expected that it also performs well in here. Indeed, this is confirmed, as NLM gives the smallest distances between the estimations computed in the original and in the transformed space. The only

algorithm which sticks out again due to its instability and inaccuracy is NM, which has to be stopped after 500 iterations without reaching convergence.

Distribution	p_{out}	NM	BFGS	CG	NLM	VaZh	HoCr
$N_p(\mathbf{0}, \boldsymbol{I}_p)$	0 %	3.98	2.71e-07	2.93e-07	7.31e-12	8.49e-10	9.41e-08
	10 %	3.81	2.37e-07	2.81e-07	7.41e-12	8.27e-10	1.13e-07
	20 %	4.42	2.32e-07	3.21e-07	1.14e-11	1.04e-09	1.21e-07
	30 %	4.92	3.54e-07	3.57e-07	1.53e-11	1.06e-09	1.66e-07
	40 %	5.71	4.6e-07	4.1e-07	1.77e-11	1.51e-09	2.07e-07
$LN_p(\mathbf{0}, \boldsymbol{I}_p)$	0 %	0.0401	2.46e-09	2.57e-09	6.44e-15	8.41e-12	3.74e-10
	10 %	0.0376	2.27e-09	2.58e-09	8.19e-15	8.07e-12	1.22e-09
	20 %	0.0424	3.29e-09	2.88e-09	1.47e-14	1.17e-11	1.99e-09
	30 %	0.0464	5.24e-09	3.69e-09	2.78e-14	1.79e-11	3.56e-09
	40 %	0.0591	6.06e-09	6.08e-09	4.97e-14	1.41e-10	7.46e-09

Table 2.10: 95% distance quantiles of L_1-median algorithms applied to the original and the transformed data sets.

2.3.5 Degenerated situations

In this subsection we study the behavior of the algorithms at two particular data structures. First we consider the case of collinear data, secondly the case where more than half of the data points are coinciding. In these settings the exact value of the minimum of the objective function (2.2.1) is known. The deviation of an algorithm is now computed relative to the exact optimum.

- If n is even, and all observations are collinear, the L_1-median is not uniquely defined, as any point between the 2 innermost observations is a proper solution (this is in analogy to the univariate median). As such a data structure has rank one, the application of the classical median is possible yielding the same result as the L_1-median. Therefore the data matrix has to be projected onto the line given by two arbitrary (different) observations, which in this case coincides with the first principal component. The median is computed in this one-dimensional subspace, whereas the result is transformed back into the original space afterwards.

  Simulations on data sets generated with a covariance matrix with arbitrary values (finite and positive) for the variances, and off-diagonal elements $c_{ij} = \sqrt{(c_{ii}c_{jj})}\ \forall i \neq j$ (perfect collinearity of all variables) and arbitrary (finite) mean vectors result in Table 2.3.5. Apart from two occurrences where VaZh gives slightly better results, the algorithms CG, NLM, VaZh and HoCr show equal performance and return an exact solution. BFGS still returns good results but seems to have numerical problems due to its slight imprecision. NM seems to deliver

arbitrary results, which are caused by convergence problems. The runtime comparison in Table 2.3.5 shows that BFGS is always the fastest algorithm, followed by NLM which is as fast as in the first simulation setting. The runtimes of CG, VaZh and HoCr increased by a factor 2-4 compared to Table 2.3.2.

Distribution	p_{out}	NM	BFGS	CG	NLM	VaZh	HoCr
$N_p(\mathbf{0}, \mathbf{C})$)	0 %	6.42e+03	9.09e-13	0	0	0	0
	10 %	6.01e+03	3.64e-12	0	0	0	0
	20 %	5.66e+03	1.82e-13	0	0	0	0
	30 %	5.36e+03	7.28e-12	0	0	0	0
	40 %	5.27e+03	7.28e-12	7.28e-12	7.28e-12	0	7.28e-12
$LN_p(\mathbf{0}, \mathbf{C})$	0 %	1.08e+04	1.82e-12	9.09e-14	9.09e-14	0	1.82e-12
	10 %	1.08e+04	3.82e-12	0	0	0	0
	20 %	1.12e+04	7.28e-12	0	0	0	0
	30 %	1.23e+04	1.46e-11	0	0	0	0
	40 %	1.40e+04	0	0	0	0	0

Table 2.11: 95% error quantiles of L_1-median estimations applied to collinear data.

Distribution	p_{out}	NM	BFGS	CG	NLM	VaZh	HoCr
$N_p(\mathbf{0}, \mathbf{C})$	0 %	0.266	0.047	0.094	0.078	0.258	0.446
	10 %	0.266	0.047	0.125	0.078	0.250	0.422
	20 %	0.266	0.047	0.195	0.078	0.250	0.406
	30 %	0.265	0.047	0.172	0.078	0.250	0.391
	40 %	0.266	0.047	0.125	0.078	0.282	0.422
$LN_p(\mathbf{0}, \mathbf{C})$	0 %	0.266	0.047	0.109	0.093	0.235	0.422
	10 %	0.266	0.054	0.148	0.093	0.266	0.453
	20 %	0.266	0.047	0.133	0.093	0.250	0.382
	30 %	0.266	0.047	0.125	0.079	0.328	0.469
	40 %	0.265	0.039	0.125	0.094	0.406	0.578

Table 2.12: Median runtimes [seconds] of the L_1-median algorithms applied to collinear data.

- If more than $n/2$ observations are concentrated in one point, say $\boldsymbol{y}$, the solution of the L_1-median is $\hat{\boldsymbol{\mu}} = \boldsymbol{y}$. A simulation setting as in Section 2.3.2 but with $n/2 + 1$ observations set to $\boldsymbol{y} = \mathbf{1}$ (vector of ones) has been carried out, expecting the resulting L_1-median estimation to be $\hat{\boldsymbol{\mu}} = \mathbf{1}$. All algorithms are able to find the known center exactly, thus a table showing deviation quantiles can be omitted. However, the computation times differ for some algorithms (Table 2.3.5). Although NM converges to the exact solution in this setting, it converges slowly, resulting in a considerably higher value for the runtime. Also NLM is unable to detect this degenerated situation immediately. All other

algorithms stop after a single iteration and thus converge so quickly, that the used method for measuring time is not able to record such short periods.

The algorithms HoCr and VaZh have to be pointed out here, as they can handle such occurrences separately: both algorithms are based on the distances of each observation to the current L_1-median estimation, which is calculated during each iteration. By simply checking how many observations have zero distance to the current L_1-median estimation, the method can detect such cases and stop the iteration. A different return code is provided in order to inform the user about such rare occurrences.

Distribution	p_{out}	NM	BFGS	CG	NLM	VaZh	HoCr
$N_p(\mathbf{0}, \mathbf{C})$	0 %	0.265	0.000	0.000	0.047	0.000	0.000
	10 %	0.266	0.000	0.000	0.047	0.000	0.000
	20 %	0.266	0.000	0.000	0.047	0.000	0.000
	30 %	0.266	0.000	0.000	0.047	0.000	0.000
	40 %	0.266	0.000	0.000	0.047	0.000	0.000
$LN_p(\mathbf{0}, \mathbf{C})$	0 %	0.266	0.000	0.000	0.047	0.008	0.000
	10 %	0.265	0.000	0.000	0.047	0.000	0.000
	20 %	0.265	0.000	0.000	0.047	0.000	0.000
	30 %	0.266	0.000	0.000	0.047	0.000	0.000
	40 %	0.266	0.000	0.000	0.047	0.000	0.000

Table 2.13: Median runtimes [seconds] of the L_1-median algorithms applied to the degenerated data structure.

Although the situations examined here would rarely occur, it is important to keep in mind that there might be further data configurations which are not so easy to handle.

2.3.6 Real data examples

For testing the L_1-median algorithms on real data, the data sets `bhorizon`, `chorizon`, `humus`, `moss`, `bssbot` and `bsstop`, from the R-library `mvoutlier` are used. These data contain the concentration of chemical elements in the soil of certain regions of Europe. Depending on the considered data set, 600 to 800 observations are available for 30-110 variables. Overall, the data are usually right-skewed, and therefore the algorithms were additionally run on the log-transformed data. Moreover, there are many equal values (especially for `bsstop`) which are caused by detection limit problems (see Reimann *et al.*, 2008). Table 2.3.6 gives the deviations as returned by the different algorithms. Since the tolerance levels have been adjusted before, all methods perform similar, again with the exception NM. A comparison

of runtimes in Table 2.3.6 shows that NLM is constantly the fastest approach, outperforming all other algorithms in terms of computation time, while maintaining the highest accuracy.

Data set	log	NM	BFGS	CG	NLM	VaZh	HoCr
`bhorizon`	no	1.46e+03	0	0	0	0	0
	yes	1.73	0	0	0	0	0
`chorizon`	no	4.99e+04	0	0	0	0	0
	yes	95.3	0	0	0	0	0
`humus`	no	501	0	0	0	2.33e-10	0
	yes	1	4.55e-13	0	0	0	0
`moss`	no	317	0	0	0	0	0
	yes	0.225	2.27e-13	0	0	0	0
`bssbot`	no	80.5	0	0	0	0	0
	yes	32.1	4.55e-13	0	0	0	0
`bsstop`	no	71.5	0	0	0	0	0
	yes	35.5	0	0	0	0	0

Table 2.14: Deviations of L_1-median algorithms applied to real data sets.

Data set	log	NM	BFGS	CG	NLM	VaZh	HoCr
`bhorizon`	no	0.047	0.000	0.062	0.016	0.063	0.046
	yes	0.047	0.031	0.125	0.016	0.016	0.016
`chorizon`	no	0.172	0.062	0.219	0.047	0.094	0.156
	yes	0.156	0.078	0.063	0.031	0.063	0.078
`humus`	no	0.031	0.016	0.016	0.000	0.047	0.046
	yes	0.046	0.016	0.094	0.016	0.031	0.016
`moss`	no	0.031	0.000	0.016	0.000	0.031	0.031
	yes	0.031	0.015	0.016	0.000	0.032	0.031
`bssbot`	no	0.109	0.016	0.047	0.016	0.078	0.094
	yes	0.109	0.031	0.031	0.016	0.031	0.047
`bsstop`	no	0.094	0.015	0.047	0.016	0.063	0.047
	yes	0.094	0.062	0.328	0.015	0.047	0.032

Table 2.15: Runtimes [seconds] of the L_1-median algorithms applied to real data sets.

2.3.7 Runtime as a function of the sample size

In this subsection we report median runtimes as a function of the sample size n. Since the ranking of the different algorithms with respect to their runtime does not depend much on the actual data configuration, we report results for the uncorrelated data case, as in subsection 2.3.2. We generate from a multivariate normal distribution, with $p = 100$, and where $\log_{10}(n)$ varies from 2 to 5. Table 2.3.7 reports the median runtimes over 100 simulated samples. We see that the NLM algorithm is the fastest for all considered

sample sizes. The runtimes suggest that the computational complexity of the different algorithms is $O(n \log n)$.

n	NM	BFGS	CG	NLM	VaZh	HoCr
100	0.017	0.012	0.005	0.006	0.012	0.009
1000	0.239	0.117	0.064	0.071	0.122	0.147
10000	2.335	1.739	10.204	0.694	1.336	2.577
1e+05	23.096	15.179	141.382	7.121	13.361	30.646

Table 2.16: Median runtimes [seconds] of the L_1-median algorithms applied to uncorrelated data with varying sample size n and fixed number of variables $p = 100$.

2.4 Conclusions

For computing the L_1-median, several specific algorithms can be found in the literature. In addition, the solution can also be found by general optimization routines, which are widely available in modern software packages. A fair comparison of these approaches is only possible by a unified software implementation. This has been done in the R-library `pcaPP` (version 1.8-1), using C++ code for the implementation. The implemented algorithms have been tested for various data configurations. Optimization based on NLM leads to the smallest value of the target function in all considered settings, which makes it the first choice as an appropriate algorithm for computing the L_1-median. Moreover, the computation time of this approach turned out to be to a large extent unaffected by the used distribution family, the outlier proportion added, the convergence criterion, and even by high correlation values, whereas other algorithms tend to have convergence issues in some simulation settings. In the tested situations NLM never showed any problems with convergence, and hence it provides a stable, fast and reliable approach, returning results with the highest precision among the tested algorithms.
Considering the tested data sets and their dimensions, it might be surprising that the absolute runtime of the algorithms is very low. Thus, whenever a robust data center needs to be computed, the L_1-median is an attractive estimator as long as affine equivariance is not required. It is definitely more attractive than the component-wise median, which has the same breakdown point, but which is not orthogonal equivariant.
Concluding, we recommend the algorithm NLM, implemented as `l1median_NLM` in the R-library `pcaPP` (version 1.8-1), as this particular approach turned out to deliver results of highest precision in combination with very low computation time, widely unaffected by the underlying data structure.

Acknowledgments

We would like to thank Martin Mächler from the ETH-Zürich for providing different L_1-median algorithms in the R-library `robustX`.

Chapter 3

Exploring High-dimensional Data With Robust Principal Components

Summary: For high-dimensional data of low sample size it is difficult to compute principal components in a robust way. We mention an algorithm which is highly precise and fast to compute. The robust principal components are used to compute distances of the observations in the (sub-)space of the principal components and distances to this (sub-)space. Both distance measures retain valuable information about the multivariate data structure. Plotting the magnitudes of the distance measures helps to reveal important multivariate data information.

Co-authors: Peter Filzmoser

3.1 Introduction

Principal component analysis (PCA) is frequently used in an exploratory context to gain first insight into multivariate data. The method is fast to compute, and also the geometric concept of PCA is easy to understand. Moreover, for the purpose of graphically inspecting the multivariate data the data requirements (distributional assumptions, etc.) are very low. These features make PCA attractive for researchers and practitioners working in various fields. Pairwise plots of the first few principal components (PCs) often allow to reveal the multivariate data structure in a way that relations among the observations as well as data groups and structures can be discovered.
PCA is also often used to identify data outliers. Surprisingly, in many applications this even works, although it is well known that the directions of the

PCs are determined by the eigenvectors of the covariance matrix of the data. Using the empirical covariance matrix as an estimator of the covariance results in an unbounded influence function of the eigenvalues and eigenvectors (Croux and Haesbroeck, 2000) which means that the eigendecomposition can be completely misled even in case of very small amount of contamination. On the other hand, huge outliers can attract a PC, and the analyst will be able to discover such outliers by inspecting the corresponding component. In that way, PCA does not focus on revealing the main variability of the underlying data structure, but it still is able to find interesting phenomena in the data. The analyst would then usually remove this outlier and proceed with a new PCA on the reduced data to finally reveal the multivariate data structure.
Less extreme outliers or multivariate outliers are not necessarily visible as isolated points at the projections on the PCs. Even worse, they are able to spoil the PC directions, and the goal of exploring the relevant data structure by the first few PCs may completely fail (Croux and Ruiz-Gazen, 2005). This can be avoided if PCA is robustified. The easiest possibility is to estimate the covariance matrix in a robust way, using well known estimators like the MCD (Rousseeuw and Van Driessen, 1999). The eigenvectors of this robust covariance matrix will be resistant to outliers in the data and point in directions of high variability of the main data cloud.
Exploring data by PCA and robustifying PCA as mentioned above is possible for "tall" data where n, the number of observations, is (much) larger than p, the number of variables. However, there are many applications in chemometrics, marketing, biostatistics, etc. where p is much larger than n. Robust covariance estimation e.g. using the MCD does no longer work in this case and one needs to consider other alternatives. Moreover, projecting the data on the first two PCs is often still very uninformative because of the high dimensionality of the data.
In this paper we will briefly describe methods and algorithms for robust PCA in the case $p >> n$ (Section 2). At the basis of a diagnostics plot for PCA (Hubert et al., 2005) we construct a plot in Section 3 that allows to reveal multivariate data structure. As a motivating example we use a data set of 21 NIR spectra in 268 dimensions (Swierenga et al., 1999).

3.2 PCA for high-dimensional data

A very appealing approach for robust PCA in high dimensions is based on the projection-pursuit (PP) principle (Li and Chen, 1985). This approach uses the initial definition of PCA of finding a direction where the projected data points have maximal variance. Subsequent directions have the same goal of maximizing the variance of the projected data, but they are supposed to be orthogonal to previously found directions. Robustifying this approach

is in fact very easy, because the measure of variance simply needs to be robust. The MAD or the Qn estimators have been suggested for this purpose (Croux and Ruiz-Gazen, 2005). The difficult task, however, is to develop an algorithm for solving the maximization problem.
In the case $p >> n$ the computational complexity can be reduced considerably by first performing a singular value decomposition which allows a reduction of the dimensionality from p to only n dimensions without any loss of information (see e.g. Stanimirova et al., 2004).
The algorithm of Croux and Ruiz-Gazen (2005) uses candidate directions for finding the first PC that consist of directions from the center of the data cloud to each single data point. The resulting n directions are then evaluated by computing the (robust) variance of the projected data points, and the direction corresponding to the maximum of the variance is an approximation of the direction of the first PC. The search is then continued analogously in the subspace orthogonal to the found direction. The identified directions are approximations of the eigenvectors of a covariance-based approach, and the corresponding maximal variances are approximations of the eigenvalues. Recently it was pointed out (Croux et al., 2007) that the algorithm of Croux and Ruiz-Gazen (2005) has serious drawbacks:

(a) The estimated eigenvalues corresponding to the k-th PC with $k > n/2$ are exactly zero if the MAD, the Qn, or any other highly robust scale measure is used. This artefact is also called *implosion* of the scale measure. Usually, this drawback has no real consequences because one is mainly interested in the first few PCs. However, if a robust covariance matrix should be estimated with the robust PCs, the result could be seriously biased.

(b) Especially for $p >> n$ the algorithm is not very precise. This can be verified by comparing the theoretical maximum which are the eigenvalues of the eigendecomposition of the sample covariance matrix with the resulting estimated eigenvalues of this algorithm using the classical variance measure.

Both disadvantages are solved by the so-called GRID algorithm (Croux et al., 2007). The idea is to search for the optimal direction only in a plane on a regular grid. The plane is first spanned by two variables, but later on also information of the other variables is used by taking linear combinations with the remaining variables. The resulting directions are highly precise: the estimated eigenvalues are in general considerably higher than for the algorithm of Croux and Ruiz-Gazen (2005), and they come very close to the true maximum. The GRID algorithm has been implemented in the library *pcaPP* of the statistical software *R*.

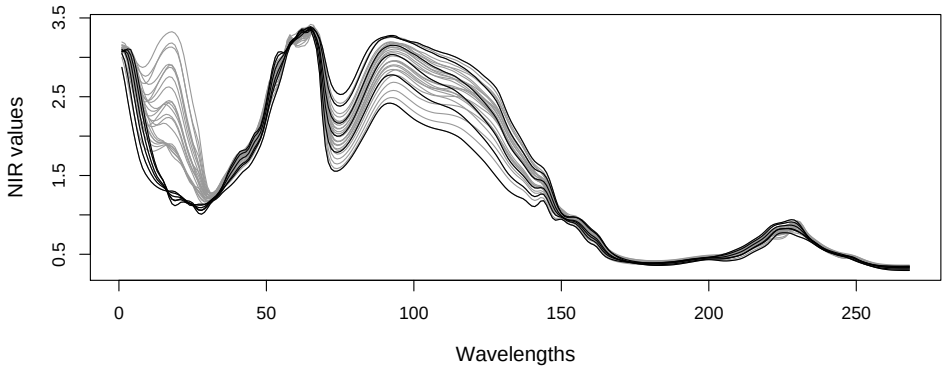

Figure 3.1: NIR spectra of PET yarns; the highlighted (black) spectra have deviating structure for small wavelengths.

3.2.1 Example

We apply robust PCA on a data set described in Swierenga et al. (1999) of 21 NIR spectra of PET yarns, measured at 268 wavelengths. The data set is available in *R* in the package *pls* as data set *NIR*. Swierenga et al. (1999) used these data together with 28 corresponding densities to construct a robust multivariate calibration model. Here we will only consider the NIR data, and our goal is to gain insight into their multivariate data structure using robust PCA. Figure 3.1 shows the 28 spectra. We highlighted some spectra with somewhat abnormal behavior at small wavelengths.
Robust PCA using the GRID algorithm was applied to the NIR data set. To obtain robust components the MAD (medIan absolute deviation) was used as scale measure. Figure 3.2 shows the plot of first versus second PC (left) and first versus third PC (right). The highlighted spectra from Figure 3.1 are visualized as dark points. The first 3 PCs include 98.8% of the total variability. However, it is not obvious from these plots that the highlighted spectra are somehow different. It is possible that this difference is expressed in the remaining 25 PCs which, on the other hand, only contain a bit more than 1% of the total variability.

3.3 Orthogonal distance and score distance

Hubert et al. (2005) used the *orthogonal distance* (OD) and the *score distance* (SD) as diagnostic tools in the context of PCA. For a sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in $\mathcal{R}^p$ the OD is defined as

$$\mathrm{OD}_i^{(k)} = \|\boldsymbol{x}_i - \hat{\boldsymbol{\mu}} - \boldsymbol{P}^{(k)}\boldsymbol{t}_i^{(k)}\| \tag{3.3.1}$$

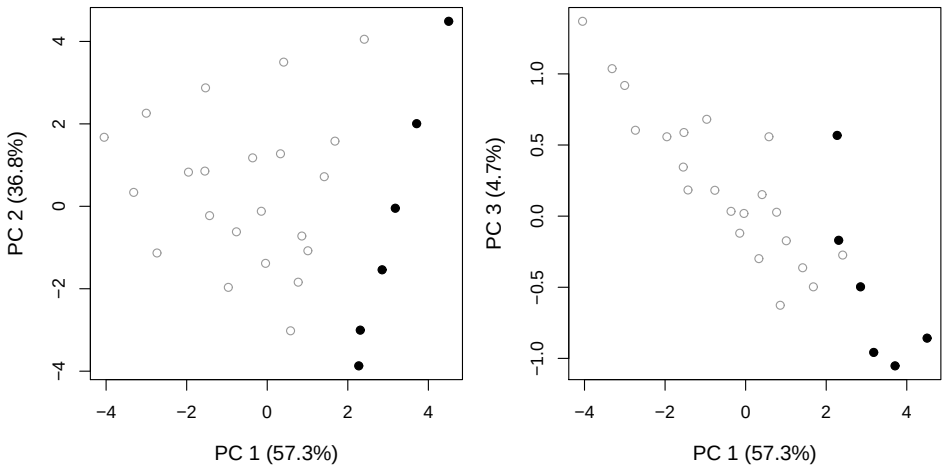

Figure 3.2: Robust PCA for the NIR data using the GRID algorithm with the MAD as scale measure.

and the SD as

$$\mathrm{SD}_i^{(k)} = \left[\sum_{j=1}^{k} \frac{\|\boldsymbol{t}_i^{(k)}\|^2}{l_j} \right]^{1/2} \tag{3.3.2}$$

for $i = 1, \ldots, n$. Here, $\hat{\boldsymbol{\mu}}$ is the estimated center of the data, the matrix $\boldsymbol{P}^{(k)}$ contains the first k estimated eigenvectors in its columns, l_j are the estimated eigenvalues, and $\boldsymbol{t}_i^{(k)}$ is the ith score vector in the space of the first k principal components ($1 \leq k \leq r$) with r being the rank of the data. Both the OD and the SD depend on the number of PCs considered. The OD describes the orthogonal distance of an observation to the space spanned by the first k PCs, whereas the SD is a Mahalanobis-like measure of distance of an observation within the PC space. Samples with large OD and SD can have severe leverage to a classical PCA.

Hubert et al. (2005) introduced a diagnostic plot for PCA by plotting SD versus OD. Critical thresholds for SD and OD allow to identify outlying observations. Figure 3.3 shows this diagnostic plot for the NIR data when $k = 2$ PCs (left) and $k = 3$ PCs are used to compute the distances. Again, the black points refer to the spectra with unusual behavior. We used similar critical values as in Hubert et al. (2005): for the SD a quantile of the chi-squared distribution with k degrees of freedom (we used $\sqrt{\chi^2_{k;0.975}}$), and for OD we also took the Wilson-Hilferty approximation for the scaled chi-squared distribution which assumes that the ODs to the power of 2/3 are approximately normally distributed. The parameters μ and σ of the normal distribution can be estimated by the median and MAD of the values $\mathrm{OD}^{2/3}$, and the critical value can be taken as $(\hat{\mu} + \hat{\sigma} z_{0.975})^{3/2}$, with $z_{0.975}$ being the

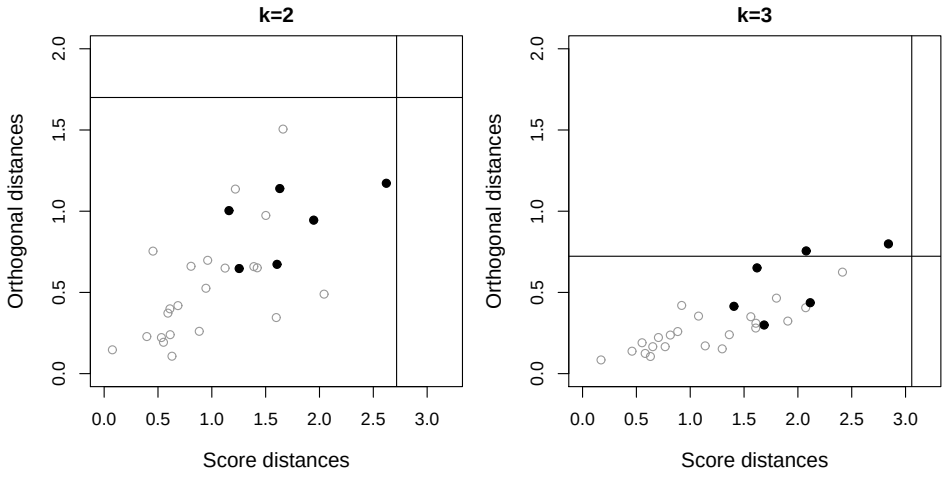

Figure 3.3: Diagnostic plots to the PCA from Figure 3.2 for 2 (left) and 3 (right) components.

97.5% quantile of the standard normal distribution. The critical values in Figure 3.3 for $k = 3$ allow to identify two observations as "outliers" because their OD is larger than the critical value. Note that our goal is not necessarily outlier detection using the PCs but rather to learn about the multivariate data structure. These plots, however, do not reveal any special phenomenon like groups of deviating data points.
It is possible, that such a diagnostic plot would reveal deviating data points in a better way if more PCs were used for computing the OD and SD. Hubert et al. (2005) suggested to take as many PCs such that about 90% of the total variability are explained. This would correspond to the right plot in Figure 3.3.

3.4 Exploring the multivariate data structure

It is easy to show that

$$\mathrm{SD}_i^{(k)} \leq \mathrm{SD}_i^{(k+1)} \qquad \text{and} \qquad \mathrm{OD}_i^{(k)} \geq \mathrm{OD}_i^{(k+1)}$$

for $1 \leq k < r$. These properties can also be observed in Figure 3.3. In fact, the SD is just the projection of the Mahalanobis distance on the space spanned by the first k PCs, see Equation (3.3.1). Using all PCs for computing the SD is exactly the Mahalanobis distance. Naturally, the distances in a higher dimensional space are increasing. Also for OD the above property is obvious because for increasing dimension of the PC space the orthogonal distances to the data space have to decrease, see Equation (3.3.2). On the

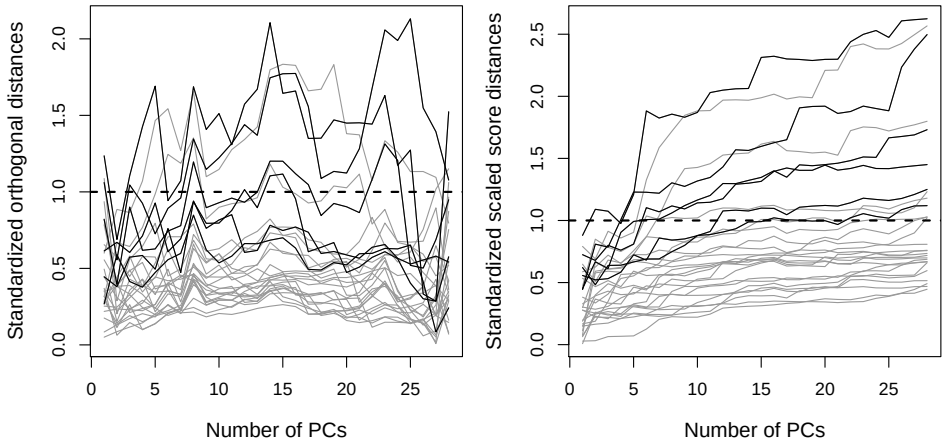

Figure 3.4: Standardized OD (left) and SD (right) for each observation using $k = 1, \ldots, 28$ PCs. The dashed lines indicate the 97.5% critical values.

other hand, also the critical values for OD and SD change for an increasing number k of PCs. Moreover, it is possible that an observation has large OD and small SD for k PCs, but small OD and large SD for $k+1$ PCs. This can be the case if the observation is far away from the space spanned by k PCs but very close to PC number $k+1$.
Thus, studying the OD and SD for various values of k might provide more insight into the multivariate data structure. Since it will be rather difficult to use a diagnostic plot like in Figure 3.3 for several or all values of k we prefer to present two separate plots for the OD and for the SD. Following the suggestions of Maronna and Zamar (2002), we multiply the SD with the scaling factor $\sqrt{\chi^2_{k;0.5}}/\text{median}(SD_i^{(k)})$ which improves the chi-square approximation significantly. Afterwards, we divide the OD and the scaled SD by their critical values corresponding to the 97.5% quantiles or the respective distribution. OD and SD are thus standardized, and a comparison with the critical threshold 1 can be easily done. Figure 3.4 shows the resulting plots for the number of PCs ranging from 1 to 28. Each line in the plot corresponds to the standardized OD (left) or SD (right) of an observation. The black lines are the atypical observations from Figure 3.1. The dashed lines refer to the critical threshold 1. Most of the atypical observations are only exceeding the threshold if more than 3 PCs are taken.
The visualization of Figure 3.4 can be improved by observing if an observation exceeds the critical value at a given number of PCs or not. This information is shown in Figure 3.5 for the OD (left) and for the SD (right). A white square indicates that the observation did not exceed the critical value for given k. Light gray, dark gray and black boxes are plotted if the

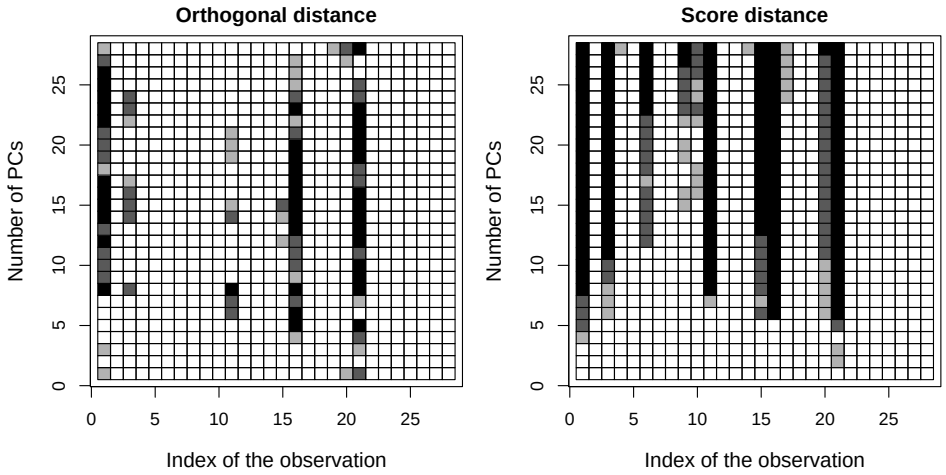

Figure 3.5: OD (left) and SD (right) for each observation using $k = 1, \ldots, 28$ PCs. The different gray levels indicate that the 97.5% (light gray), 99% (dark gray), or 99.9% (black) critical value was exceeded.

observations exceed the corresponding 97.5%, 99%, or 99.9% critical values, respectively. The atypical observations visible in Figure 3.1 refer to the indices 1, 3, 6, 10, 15, and 21, which are also atypical in the plot of the OD and/or SD in Figure 3.5. There are additional outstanding observations which refer to further deviating structure in some of the NIR spectra. Thus, the plot provides an impression about the multivariate data structure.

3.5 Summary

For high-dimensional data it is not trivial how to compute robust principal components. We propose to use the GRID algorithm (Croux et al., 2007) which is very precise and results in highly robust PCs. The estimated eigenvectors and eigenvalues from this approach can be used to compute orthogonal and score distances which are indications of outlyingness of the observations. We introduced a new plot of these distance measures computed for various numbers of principal components. Deviations of observations from the main data structure in a certain sub-space spanned by k PCs are indicated in the plot. Since the dimension of the sub-space changes with the number of PCs, the plot helps to reveal the multivariate data structure. In the presentation we will also give other real data examples where this plot gives insight into the data structure. Moreover, we will demonstrate that not only the robustness of PCA is important for an informative plot, but also the precision of the PCA algorithm.

Chapter 4

Robust Sparse Principal Component Analysis

Summary: A method for principal component analysis is proposed that is sparse and robust at the same time. The sparsity delivers principal components that have loadings on a small number of variables, making them easier to interpret. The robustness makes the analysis resistant to outlying observations. The principal components correspond to directions that maximize a robust measure of the variance, with an additional penalty term to take sparseness into account. We propose an algorithm to compute the sparse and robust principal components. The method is applied on several real data examples, and diagnostic plots for detecting outliers and for selecting the degree of sparsity are provided. A simulation experiment studies the loss in statistical efficiency by requiring both robustness and sparsity.

Keywords: dispersion measure, projection-pursuit, outliers, variable selection

Co-authors: Christophe Croux, Peter Filzmoser

4.1 Introduction

Principal component analysis (PCA) is a standard tool for dimension reduction of multivariate data. PCA searches for linear combinations of the variables, called principal components (PC), that summarize well the data. The PCs correspond to directions maximizing the variance of the data projected on them (see, e.g. Jolliffe, 2002). The transformation matrix defining the principal components is called the loadings matrix, and it may be used to interpret the PCs. In general, PCA does not deliver well interpretable components. Good interpretability of PCs is related to rather large or small

(absolute) values in the loadings matrix yielding either quite strong or quite weak contributions of the variables to the PC. Loadings matrices with many values exactly equal to zero, which we call *sparse loadings matrices*, are preferred, since the interpretation of a particular principal component does not require to consider all variables, but only a small subset. This yields a *sparse PCA*, which is especially helpful for analyzing high dimensional data sets. In this paper we introduce a method for PCA that yields both sparse and *robust* results. Outliers frequently occur in multivariate data sets, and any multivariate procedure should take the possible presence of outliers into account.

Different approaches for computing sparse loadings matrices have been proposed in the literature. Vines (2000) and Anaya-Izquierdo *et al.* (2011) use a restriction on the loadings to integers. Jolliffe *et al.* (2003) introduced the SCoTLASS, related to the Lasso estimator (Tibshirani, 1996). Here the principal components maximize the variance but under an upper bound on the sum of the absolute values of the loadings. It is shown that such an approach yields better results than a two-step procedure, where after a standard PCA rotation techniques are performed (Jolliffe, 1995). Zou *et al.* (2006) use the elastic net to obtain a version of sparse PCA. Modifications and improvements of this method are made in Leng and Wang (2009). Finally, Guo *et al.* (2010) introduce a fusion penalty to capture block structures within the variables. All these methods, however, are not robust to outliers. This paper proposes a PCA method that is robust and sparse at the same time. Several robust, but non sparse, PCA methods have been introduced in the literature (see, e.g., Filzmoser, 1999; Hubert *et al.*, 2005; Maronna, 2005), and robustness properties were investigated (Croux and Haesbroeck, 2000). Here we focus on the *projection-pursuit* approach to PCA, where the PCs are extracted from the data by searching for directions that maximize a robust measure of variance of the data projected on it (Li and Chen, 1985; Croux and Ruiz-Gazen, 2005). Using a robust measure of variance avoids that the PCs are attracted by the outliers, since outliers inflate the standard non-robust variance. An efficient algorithm for computing the projection-pursuit based PCs is the *Grid algorithm*, introduced in Croux *et al.* (2007). The Grid algorithm is very precise, and an implementation is available in the R package `pcaPP` (Filzmoser *et al.*, 2010). Up to the best of our knowledge, the PCA method we propose is the first one combining the properties of robustness and sparsity.

The paper is organized as follows: Section 2 defines the robust sparse principal components as the solution of a non convex optimization problem. Section 3 shows how the Grid algorithm can be extended to find an approximate solution of this problem. The selection of tuning parameters is discussed in Section 4. Simulation results are presented in Section 5, and real data examples are shown in Section 6. The final Section 7 concludes.

4.2 Method

Given n multivariate observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$, collected in the rows of the data matrix $\boldsymbol{X}$. The first PCA direction is given by

$$\boldsymbol{a}_1 = \underset{\|\boldsymbol{a}\|=1}{\operatorname{argmax}} V(\boldsymbol{a}^t\boldsymbol{x}_1, \ldots, \boldsymbol{a}^t\boldsymbol{x}_n), \tag{4.2.1}$$

where V is a variance measure. In the standard non-robust case, V is the empirical variance (Var), and the resulting optimal direction $\boldsymbol{a}_1$ corresponds to the first eigenvector of the sample covariance matrix. Equation (4.2.1) is the projection-pursuit formulation for finding the first PC, with V being the projection-pursuit index. Robust PCA directions can easily be obtained by taking a robust variance measure for V, like the squared Median Absolute Deviation (MAD) or the squared Q_n estimator. The Q_n estimator was proposed in Rousseeuw and Croux (1993) and is, for a univariate data set $y_1, \ldots, y_n$, defined as the first quartile of all pairwise distances $|y_i - y_j|$, for $1 \leq i < j \leq n$. Croux and Ruiz-Gazen (2005) showed that using the Q_n^2 estimator as projection index yields robust and efficient estimates for the principal components. In the remainder of this paper, we use the Q_n^2 as robust variance estimator.
Suppose the first $j-1$ PCA directions have already been found ($j > 1$), then the jth direction ($j \leq p$) is defined as

$$\boldsymbol{a}_j = \underset{\|\boldsymbol{a}\|=1, \boldsymbol{a}\perp\boldsymbol{a}_1, \ldots, \boldsymbol{a}\perp\boldsymbol{a}_{j-1}}{\operatorname{argmax}} V(\boldsymbol{a}^t\boldsymbol{x}_1, \ldots, \boldsymbol{a}^t\boldsymbol{x}_n), \tag{4.2.2}$$

imposing an orthogonality constraint to all previously found directions. The jth principal component is then the vector containing the PCA *scores*

$$z_{ij} = \boldsymbol{a}_j^t\boldsymbol{x}_i \quad \text{for } i = 1, \ldots, n. \tag{4.2.3}$$

The loadings matrix for the first k PCs is denoted by $\boldsymbol{A}_k$, and contains in its columns the optimal directions or loadings vectors $\boldsymbol{a}_j$, for $1 \leq j \leq k$. The loadings determine the contribution of each variable to the principal components. The matrix containing the principal component scores is then

$$\boldsymbol{Z}_k = \boldsymbol{X}\boldsymbol{A}_k. \tag{4.2.4}$$

Sparsity can be imposed on the PCA directions by adding an L_1 penalty in the objective function. As such, Jolliffe *et al.* (2003) introduced the SCoTLASS criterion,

$$\max_{\|\boldsymbol{a}\|=1, \boldsymbol{a}\perp\boldsymbol{a}_1, \ldots, \boldsymbol{a}\perp\boldsymbol{a}_{j-1}} \boldsymbol{a}^t\hat{\boldsymbol{\Sigma}}\boldsymbol{a}, \quad \text{subject to } \|\boldsymbol{a}\|_1 \leq t, \tag{4.2.5}$$

for obtaining the jth PCA direction, with $1 \leq j \leq p$. Here, $\hat{\boldsymbol{\Sigma}}$ is the empirical covariance matrix, and the L_1 norm $\|\boldsymbol{a}\|_1 = \sum_{j=1}^p |\boldsymbol{a}_j|$ takes the sum of the

absolute values of the components of the vector $\boldsymbol{a}$. It is more convenient to work with the dual formulation of the above problem, given by

$$\max_{\|\boldsymbol{a}\|=1, \boldsymbol{a}\perp\boldsymbol{a}_1,\ldots,\boldsymbol{a}\perp\boldsymbol{a}_{j-1}} \boldsymbol{a}^t\hat{\boldsymbol{\Sigma}}\boldsymbol{a} - \lambda_1\|\boldsymbol{a}\|_1, \tag{4.2.6}$$

where λ_1 is a tuning parameter. The larger λ_1, the more the components of $\boldsymbol{a}$ are shrunken towards zero. Due to the use of the L_1 penalty, some of the loadings will even become exactly zero, similar as for the Lasso estimator in regression. The approach of Jolliffe *et al.* (2003) requires an estimated covariance matrix $\hat{\boldsymbol{\Sigma}}$ as input of the maximization problem (4.2.5), which can be solved using the algorithm detailed in Trendafilov and Jolliffe (2006) or in Journée *et al.* (2010).
An obvious way to sparse robust PCA would be to replace the empirical covariance matrix by a robust covariance estimator, as is often done in robust multivariate data analysis (Hubert *et al.*, 2008). However, computing robust covariance matrices in high dimensions, and particularly if $p > n$, is cumbersome –the estimator may even not exist– and time consuming. We therefore propose to stick to the projection-pursuit approach, where the PCs are directly obtained without using a prior covariance estimation. Adding the L_1 constraint in definition (4.2.1) for finding the first PCA direction yields

$$\tilde{\boldsymbol{a}}_1 = \operatorname*{argmax}_{\|\boldsymbol{a}\|=1} V(\boldsymbol{a}^t\boldsymbol{x}_1,\ldots,\boldsymbol{a}^t\boldsymbol{x}_n) - \lambda_1\|\boldsymbol{a}\|_1. \tag{4.2.7}$$

The vector $\tilde{\boldsymbol{a}}_1$ is the first sparse PCA direction, and its sparsity is controlled by the tuning parameter λ_1. Setting $\lambda_1 = 0$ results in the unconstrained first PCA direction $\boldsymbol{a}_1$, but for increasing values of λ_1, sparsity gains importance compared to robust variance maximization. Similarly, the jth sparse PCA direction ($1 < j \leq p$) is defined by

$$\tilde{\boldsymbol{a}}_j = \operatorname*{argmax}_{\|\boldsymbol{a}\|=1, \boldsymbol{a}\perp\tilde{\boldsymbol{a}}_1,\ldots,\boldsymbol{a}\perp\tilde{\boldsymbol{a}}_{j-1}} V(\boldsymbol{a}^t\boldsymbol{x}_1,\ldots,\boldsymbol{a}^t\boldsymbol{x}_n) - \lambda_j\|\boldsymbol{a}\|_1, \tag{4.2.8}$$

with λ_j a tuning parameter, possibly different from λ_1. Definition (4.2.7) and (4.2.8) are very elegant and simple, and maintain the basic interpretation of the principal components: we look for directions maximizing a robust variance, under the constraint the loadings should not become too large. If $V = \text{Var}$, then definitions (4.2.6) and (4.2.7) are the same. Note that most often one does not need all possible PCs, but only the first few. An advantage of the projection-pursuit approach is that the estimators are computed sequentially, reducing the computation time for small values of k.

4.3 Algorithm

Computing the projection-pursuit based PCs requires to find the optimal directions in (4.2.1) and (4.2.2) over a p-dimensional space. For general projection indices V it is not possible to find analytical solutions for the optimal

directions. Moreover, since V may be not differentiable in its arguments, using gradient based methods is not always possible. Several proposals to find good approximations of the projection-pursuit based PCs, applicable for any choice of the projection index V, have been made (Hubert *et al.*, 2002; Croux and Ruiz-Gazen, 2005; Croux *et al.*, 2007). In this paper we extend the Grid algorithm of Croux *et al.* (2007) for obtaining sparse solutions, i.e. to solve (4.2.7) and (4.2.8). The algorithm is fast to compute and accurate even for larger dimension. It is available in the R package pcaPP (Filzmoser *et al.*, 2010). Below we give an outline of the algorithm.
Let k be the number of sparse PCs that need to be computed. Assume that the first $j-1$ sparse PCA directions $\tilde{\boldsymbol{a}}_{j-1}$ are already obtained and are collected in the first $j-1$ columns of the loadings matrix $\tilde{\boldsymbol{A}}_{j-1}$, with $1 \leq j \leq k-1$. Now we want to compute $\tilde{\boldsymbol{a}}_j$. For notational consistency, set $\tilde{\boldsymbol{A}}_0^{\perp}$ equal to the identity matrix. For $j > 1$, let $\tilde{\boldsymbol{A}}_{j-1}^{\perp}$ be a matrix containing in its columns an orthonormal basis for the subspace orthogonal to the space spanned by the first $j-1$ sparse PCA directions. Denote $\boldsymbol{x}_i^{(j-1)} = (\tilde{\boldsymbol{A}}_{j-1}^{\perp})^t \boldsymbol{x}_i$, for $i = 1, \ldots, n$, belonging to the lower-dimensional space $\mathbb{R}^{p-j+1}$. Solving the maximization problem (4.2.8) is then equivalent to maximizing the objective function

$$f(\boldsymbol{a}) = V(\boldsymbol{a}^t \boldsymbol{x}_1^{(j-1)}, \ldots, \boldsymbol{a}^t \boldsymbol{x}_n^{(j-1)}) - \lambda_j \|\tilde{\boldsymbol{A}}_{j-1}^{\perp} \boldsymbol{a}\|_1, \tag{4.3.1}$$

under the restriction that $\|\boldsymbol{a}\| = 1$. As sparseness relates to the components of a direction in the space of the original variables, and not to the lower dimensional space $\boldsymbol{a}$ belongs to, we need to back-transform the vector $\boldsymbol{a}$ to the original space before taking the L_1 norm.
For optimizing (4.3.1) the Grid algorithm is used. The basic idea of this algorithm is to reduce the problem to a sequence of optimizations in a two-dimensional plane under a unit norm constraint. This boils down to a sequence of maximizations of a function over the unit circle, which is simply a univariate maximization problem that can be solved by means of a grid search over $[-\pi, \pi]$. Consider the optimization of (4.3.1) for a given value of $1 \leq j \leq k$. We take the following steps:

1. Sort the columns of $\boldsymbol{X}^{(j)}$, where the rows of $\boldsymbol{X}^{(j)}$ contain the vectors $\boldsymbol{x}_i^{(j-1)}$, in descending order of their projection index V. Then the first variable has the largest value for V and its corresponding loadings vector $\boldsymbol{a} = (1, 0, \ldots, 0)$ serves as a first approximation of the solution. The vector $\boldsymbol{a}$ has $p-j+1$ components.

2. For $l = 1, \ldots, maxiter$, perform an iteration step in which all components of the vector $\boldsymbol{a}$ are updated

   - For $1 \leq i \leq p-j+1$, update the ith component, a^i, of the current

best approximation $\boldsymbol{a}$ by finding the angle γ^* maximizing

$$f\left(a^1 b(\gamma), \ldots, a^{i-1} b(\gamma), \cos\gamma, a^{i+1} b(\gamma), \ldots, a^{p-j+1} b(\gamma)\right),$$

where γ ranges in the interval $[\arccos(a^i) - \pi/(2^{l-1}), \arccos(a^i) + \pi/(2^{l-1})]$, and where $b(\gamma) = \sin(\gamma)/\sqrt{1-(a^i)^2}$ is such that the unit norm condition holds. This function is maximized by a grid search using $Ngrid$ evaluation points. The updated value of a^i is then simply $\cos\gamma^*$.

Note that if the iteration step l increases, we perform a more restricted search in the plane, since we assume that we are already close enough to the solution. Since $Ngrid$ remains constant, we are increasing the precision in every iteration step.

The procedure is said to converge when the absolute change of the optimal direction $\boldsymbol{a}$ between two iterations drops below a prespecified tolerance level. The procedure always stops if the maximum number of iterations ($maxiter$) is reached. In our implementation, we take $Ngrid = 25$ and $maxiter = 10$ by default. Finally, the optimal sparse direction $\boldsymbol{a}$ found for the jth PC by the grid algorithm has to be back-transformed into the original space, yielding $\tilde{\boldsymbol{a}}_j = \tilde{\boldsymbol{A}}_{j-1}^{\perp}\boldsymbol{a}$.

4.4 Selection of λ

The tuning parameter λ_j regulates the degree of sparseness. The larger λ_j, the less weight is given to the robust variance measure V in the objective function (4.2.8), for $j = 1, \ldots, k$. To make the relative importance of the penalty term in (4.2.8) comparable across the different PCs, i.e. to have a similar degree of sparsity over the different principal components, we take

$$\lambda_j := \lambda\, \mathcal{V}(\boldsymbol{X}^{(j)}), \tag{4.4.1}$$

where the matrix $\boldsymbol{X}^{(j)}$ is defined in the previous section, and contains the data vectors projected on the orthogonal complement of the space spanned by the first $j-1$ optimal directions. Furthermore, $\mathcal{V}$ denotes the total robust variance of a data matrix, and is for any n by p matrix $\boldsymbol{Y}$ defined as

$$\mathcal{V}(\boldsymbol{Y}) = \sum_{i=1}^{p} V(\boldsymbol{y}_i), \tag{4.4.2}$$

where $\boldsymbol{y}_i$ stands for the ith column of $\boldsymbol{Y}$ and V is the robust variance measure used as projection index. Using (4.4.1), there is only one tuning parameter λ to be selected. The penalty term λ_j decreases with increasing j, along with the value of the projection index V for the jth principal component.

We propose to select the λ to minimize a BIC type criterion (see also Guo *et al.*, 2010; Leng and Wang, 2009)

$$\text{BIC}(\lambda) = \frac{\widetilde{\text{RV}}}{\text{RV}} + \text{df}(\lambda)\frac{\log(n)}{n}, \tag{4.4.3}$$

where $\widetilde{\text{RV}}$ and RV refer to the total robust variance of the residuals matrix obtained from a sparse PCA and an unconstrained PCA. The first term in the BIC is a measure for the quality of the fit, while the second term penalizes for model complexity. Here, $\text{df}(\lambda)$ is the number of non-zero loadings when using λ as the penalty parameter, as in Guo *et al.* (2010). The calculation of $\widetilde{\text{RV}}$ and RV is immediate, since they are given by

$$\widetilde{\text{RV}} = \mathcal{V}(\boldsymbol{X} - \boldsymbol{X}\tilde{\boldsymbol{A}}_k\tilde{\boldsymbol{A}}_k^t) \text{ and } \text{RV} = \mathcal{V}(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{A}_k\boldsymbol{A}_k^t),$$

where $\boldsymbol{X}$ stands for the data matrix, and $\boldsymbol{A}_k$ and $\tilde{\boldsymbol{A}}_k$ denote the loadings matrices containing the first k PC directions (in the columns) for unconstrained and constrained PCA, respectively. Note that, for $V = \text{Var}$, the BIC criterion in (4.4.3) equals the one in Guo *et al.* (2010). In practice, the selection of λ is carried out by minimizing the $\text{BIC}(\lambda)$ over a grid $[0, \lambda_{max}]$, where λ_{max} results in full sparseness of the sparse PCA solution with k components (i.e. every loadings vector contains only one non-zero element).

Besides λ, one also needs to choose the number of components k. Appropriate selection of k is an old and common problem in principal components analysis, and many proposals have been made for it. In this paper we select the number k from the scree-plot of an unconstrained PCA (see Cattell, 1966). Such a scree-plot represents the percentage of explained (robust) variance (EV) by the PCs versus the number of principal components. Mathematically, the explained (robust) variance is given by

$$EV_k = \frac{\mathcal{V}(\boldsymbol{Z}_k)}{\mathcal{V}(\boldsymbol{X})}, \tag{4.4.4}$$

with $\boldsymbol{Z}_k$ the matrix containing the principal component scores, see (4.2.4). For $V = \text{Var}$, EV_k equals the ratio of the sum of the k largest eigenvalues to the sum of all eigenvalues of the sample covariance matrix. Since we are concerned about ease of interpretation and sparsity we do not want to select a higher number of components when running the Sparse PCA, and maintain the same number of PCs. For this value of k, a selected λ should result in a sparser loadings matrix, at the price of limited reduction in explained (robust) variance. In Section 6 we present the so-called *tradeoff curve*, where the percentage of explained variance of the k sparse PCs is plotted as a function of λ. This tradeoff curve is a graphical tool, in addition to the BIC, for selecting an appropriate value of λ.

4.5 Simulation experiments

In this section we present two simulation experiments. The sparse method should (i) result in increased estimation precision when the true loadings matrix is sparse, and (ii) succeed in detecting those variables that do not contribute to the principal components, i.e. true zero loadings are exactly estimated as zero. We contrast the standard approach, with $V = \text{Var}$, with the robust approach, with $V = Q_n^2$. If no outliers are present, then the two properties above hold for both approaches. But it will be shown that, in presence of outliers, the standard sparse method does not meet its objectives anymore.

Experiment 1

We generate data sets of $n = 50$ observations in $p = 10$ dimensions. The true loadings matrix is

$$\boldsymbol{A} = \begin{pmatrix} \sqrt{0.5} & 0 & \sqrt{0.5} & 0 & 0 & \cdots & 0 \\ \sqrt{0.5} & 0 & -\sqrt{0.5} & 0 & 0 & \cdots & 0 \\ 0 & \sqrt{0.5} & 0 & \sqrt{0.5} & 0 & \cdots & 0 \\ 0 & \sqrt{0.5} & 0 & -\sqrt{0.5} & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 1 & & 0 \\ \vdots & \vdots & \vdots & \vdots & & \ddots & \\ 0 & 0 & 0 & 0 & 0 & & 1 \end{pmatrix}$$

and the eigenvalues are $\boldsymbol{l} = (1, 0.5, 0.1, \ldots, 0.1)$. The observations are generated from a multivariate normal distribution $N_{10}\left(\boldsymbol{0}, \boldsymbol{A}\boldsymbol{L}\boldsymbol{A}^t\right)$, with a diagonal matrix $\boldsymbol{L}$ holding the values of $\boldsymbol{l}$ in its diagonal. Contamination is added by replacing a portion of p_{out} observations by outliers, generated from the distribution $N_{10}\left(\boldsymbol{\mu}_{out}, \boldsymbol{I}_{10}\right)$ with $\boldsymbol{\mu}_{out} = (2, 4, 2, 4, 0, -1, 1, 0, 1, -1)^t$. Note that the outliers are not very far from the center of the model distribution. From the generated data set the loadings matrix is estimated, with $k = 2$. The resulting $\hat{\boldsymbol{A}}_2$ is compared to the true $\boldsymbol{A}_2$, containing the first two columns of $\boldsymbol{A}$, by computing the angle φ between the subspaces spanned by columns of the matrices.

Both the standard and the robust sparse PCA procedure are applied to $m = 100$ simulated data sets. Figure 4.1 pictures the average value of φ over the m simulations, as a function of the tuning parameter λ. Different outlier proportions, ranging from no contamination to 40% of outliers are considered.

If no outliers are present ($p_{out} = 0$, solid line), we get the expected pattern. Starting with $\lambda = 0$ (i.e. non sparse PCA) the estimation error decreases until a minimum is reached at about $\lambda = 1.2$. Penalizing the loadings further yields again an increasing estimation error. If the true model is sparse (here about 80% of the true loadings are zero) sparse estimation methods indeed

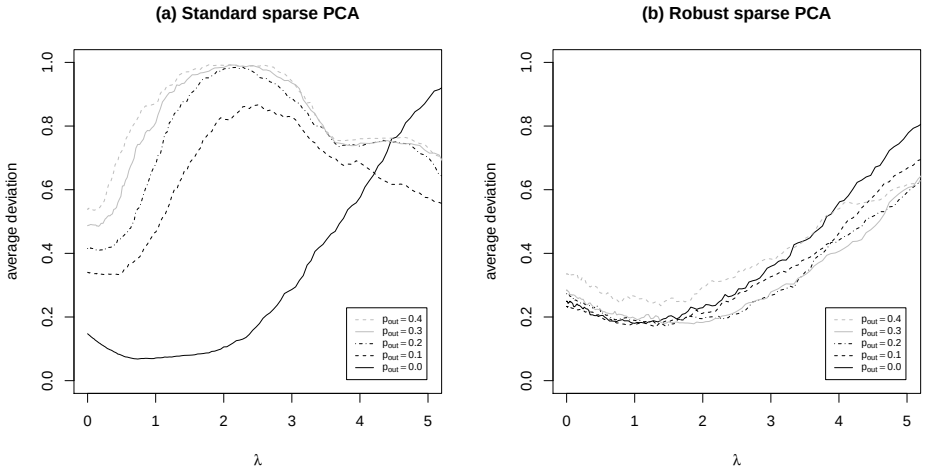

Figure 4.1: Average deviation between estimated and true loadings for (a) the standard and (b) robust sparse PCA methods for different levels of contamination p_{out} and different values of λ.

may improve the precision of the maximum likelihood method. For the robust sparse method a similar pattern is observed. Note that there is a slight loss in precision using the robust instead of the standard method. However, the robust method remains fairly accurate under contamination, as can be seen from the other curves in Figure 4.1 (b). This is in contrast with the standard method, where the estimation error increases substantially and supersedes those of the robust counterpart by a large amount. Finally, note that in presence of outliers the advantage of penalizing disappears for the standard method, since $\lambda = 0$ yields the smallest average deviation φ. This does not happen for robust sparse PCA.

Experiment 2

We consider the same design as introduced by Zou *et al.* (2006), and subsequently used by Farcomeni (2009) and Guo *et al.* (2010) in the same context of sparse PCA. We have $n = 20$ observations and $p = 10$ variables driven by two latent variables

$$U_1 \sim N(0, 290), \quad U_2 \sim N(0, 300),$$

where $\varepsilon \sim N(0, 1)$, and U_1, U_2 and ε are independent. The observed variables are constructed as

$$X_j = \begin{cases} U_1 + \varepsilon_j, & \text{if } 1 \leq j \leq 4 \\ U_2 + \varepsilon_j, & \text{if } 5 \leq j \leq 8 \\ -0.3U_1 + 0.925U_2 + \varepsilon + \varepsilon_j, & \text{if } j = 9, 10. \end{cases}$$

The error terms ε and ε_j, for $1 \leq j \leq 10$, are i.i.d. $N(0,1)$. The first two principal components correspond to U_2 and U_1, respectively, and in this order. The first block of variables, X_1 to X_4 is expected to have a high loading on the second PC, but zero loadings on the first one. The second block, X_5 to X_8, should have important loadings on the first PC, but a zero loading on the second one. The remaining variables X_9 and X_{10} have a more important loading on the first PC than on the second one, and a sparse PCA could shrink this second loading to zero. We will add outliers generated from the distribution $N(\boldsymbol{\mu}_{out}, \sigma^2_{out}\boldsymbol{I}_{10})$, with $\boldsymbol{\mu}_{out} = (0, -100, 100, 50, 0, 100, -100, 50, 75, -75)^t$, and $\sigma^2_{out} = 20$. These added data are not univariate outliers, and hence are not detectable by making boxplots of the individual variables, but they do not follow the factor structure described above.

We generate $m = 100$ samples according to the simulation design, using outlier portions 0%, 10%, and 20%, and apply the standard and the robust version of the sparse PCA algorithm. For every sample, an optimal value of the tuning parameter was selected according to the BIC criterion. Then loadings of each of the 10 variables on the first two PCs are computed, as well as the percentage of explained (robust) variance EV. The reported values correspond to the median and median absolute deviation (MAD, between parenthesis) over the 100 replications, and are presented in Table 4.1, in a similar way as in (Guo *et al.*, 2010, Table 1).

Without contamination (0 %), the results are according to the expectations, and very much comparable to those of Guo *et al.* (2010). For both the standard and the robust sparse method, we get that variables X_5 through X_{10} are solely represented in the first PC, variables X_1 to X_4 in the second PC , and the loadings of the last two variables for the second PC are also shrunken to zero. When adding contamination it is seen from Table 4.1 that the standard PCA gets distorted, and does not succeed in retrieving the sparsity in the data generating process. The robust method, however, still delivers sparse solutions. The price the robust method pays for the resistance with respect to outliers is an increased variability, as measured by the MAD values.

The standard sparse PC directions are attracted by the outliers and do no longer explain the actual structure of the majority of observations. As we can see from the last row of Table 4.1, the explained variance by the first principal component increases substantially with an increasing level of contamination. This is a misleading outcome, since it is only caused by the use of the sample variance estimator, which gets inflated due to the outliers. It is not meaning that the PCs are more representative for the bulk of the data. When using robust sparse PCA, we see that the percentage of explained variance remains about the same when the outliers are added.

		Standard Estimation						Robust Estimation					
		PC 1			PC 2			PC 1			PC 2		
		0%	10%	20%	0%	10%	20%	0%	10%	20%	0%	10%	20%
Block 1	X_1	0	0	0	0.5	0.14	0.12	0	0	0	0.46	0.34	0.3
		(0)	(0.01)	(0)	(0.01)	(0.15)	(0.15)	(0)	(0)	(0)	(0.12)	(0.24)	(0.28)
	X_2	0	-0.41	-0.42	0.5	0.15	0.14	0	0	0	0.46	0.3	0.26
		(0)	(0.03)	(0.03)	(0.01)	(0.16)	(0.16)	(0)	(0.10)	(0.12)	(0.10)	(0.26)	(0.22)
	X_3	0	0.42	0.42	0.5	0.15	0.15	0	0	0	0.44	0.28	0.25
		(0)	(0.03)	(0.02)	(0.01)	(0.15)	(0.13)	(0)	(0.07)	(0)	(0.12)	(0.40)	(0.36)
	X_4	0	0.21	0.2	0.5	0.13	0.13	0	0	0	0.47	0.38	0.33
		(0)	(0.03)	(0.03)	(0.01)	(0.15)	(0.15)	(0)	(0.1)	(0)	(0.10)	(0.27)	(0.31)
Block 2	X_5	0.41	0.01	0	0	-0.25	-0.27	0.39	0.33	0.32	0	0	0
		(0.02)	(0.02)	(0)	(0)	(0.25)	(0.22)	(0.01)	(0.15)	(0.11)	(0)	(0.13)	(0.06)
	X_6	0.42	0.44	0.43	0	-0.24	-0.28	0.38	0.33	0.36	0	0	0
		(0.02)	(0.03)	(0.02)	(0)	(0.25)	(0.17)	(0.14)	(0.27)	(0.25)	(0)	(0.17)	(0.21)
	X_7	0.41	-0.4	-0.41	0	-0.24	-0.27	0.39	0.23	0.26	0	0	0
		(0.02)	(0.03)	(0.03)	(0)	(0.36)	(0.3)	(0.12)	(0.27)	(0.35)	(0)	(0.25)	(0.14)
	X_8	0.42	0.22	0.21	0	-0.26	-0.28	0.4	0.35	0.34	0	0	0
		(0.02)	(0.03)	(0.03)	(0)	(0.23)	(0.18)	(0.12)	(0.21)	(0.22)	(0)	(0.13)	(0.18)
	X_9	0.39	0.33	0.32	0	-0.33	-0.34	0.31	0.31	0.22	0	0	0
		(0.04)	(0.03)	(0.02)	(0)	(0.13)	(0.1)	(0.18)	(0.24)	(0.32)	(0)	(0.1)	(0.25)
	X_{10}	0.39	-0.3	-0.3	0	-0.33	-0.36	0.3	0.25	0.22	0	0	0
		(0.03)	(0.03)	(0.03)	(0)	(0.2)	(0.13)	(0.19)	(0.21)	(0.33)	(0)	(0.11)	(0.21)
	EV (%)	61.4	67.4	80.3	35.7	21.2	13	58.9	51.2	50.4	31.9	30.1	28.3
		(10.9)	(5.75)	(3.9)	(9.3)	(4.9)	(3.8)	(12.5)	(8.9)	(8.9)	(11.4)	(7.1)	(6)

Table 4.1: Second simulation experiment: simulated loadings of the 10 variables on the first two PCs using standard and robust sparse PCA. The last line presents the percentage of explained variance EV. The reported values are the median and MAD (in parenthesis) over 100 simulation runs. The different columns correspond to no outliers ("0%"), and 10% and 20% of outliers in the data.

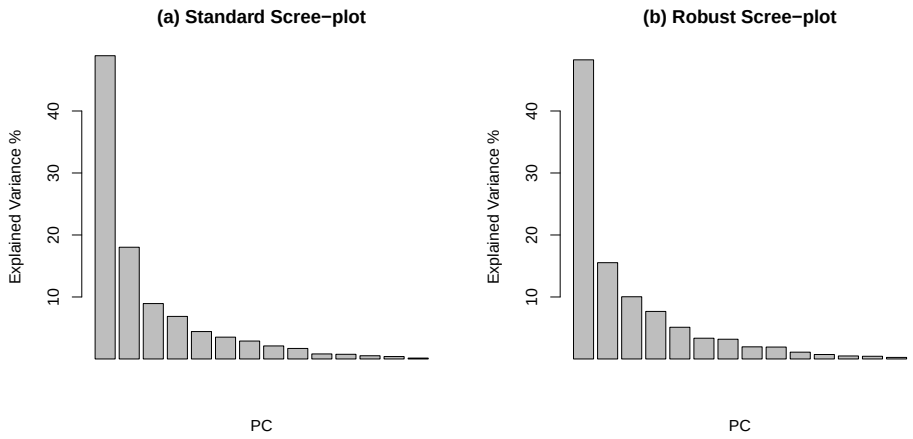

Figure 4.2: Scree-plots for a (a) standard and (b) robust PCA ($\lambda = 0$) for the car data set.

4.6 Real data examples

The method is used for two differently structured data sets. The first example has $n > p$ and shows how the robust method is capable of spotting groups of outliers. The second example points out the method's applicability on high-dimensional data sets, where $p > n$.

Example 1

The car data set (Kibler *et al.*, 1989) consists of 26 variables containing technical and insurance-related data for 205 different car models. Only continuous variables, and observations without missing values are considered here, resulting in a data set of size 195×14. To make the scale of the variables comparable, we divide each column of the data matrix by its standard deviation (if $V = \text{Var}$) or by a robust scale measure (if $V = Q_n$). Figure 4.2 gives a scree-plot for non-sparse standard and robust PCA, which plots the explained variance, as defined in (4.4.4), versus the number of components. Based on this scree-plot we decide to retain the first four PCs, explaining about 80% of the total (robust) variance, for both approaches.
Figure 4.3 shows the *tradeoff curve*, discussed in Section 4, plotting the percentage of explained variance as a function of λ. The explained variances are computed over a grid of 100 different values of the tuning parameter λ, ranging from $\lambda = 0$ (no sparseness) up to full sparseness (exactly one non-zero loading per PC). This plot illustrates how an increase in sparseness affects, and in general will lower, the explained variance. The idea is that the selected λ should be such that the sharpest decline of tradeoff curve occurs afterwards. The selected λ should be close to the end of the first, relatively

flat, part of the tradeoff curve. Using the BIC criterion from equation (4.4.3), minimized over the same grid of 100 values, we get $\lambda = 2.36$, corresponding to the vertical dashed line in the plot. From the tradeoff curve we conclude that this is an acceptable value. The sharper decline of the tradeoff curve occurs for a tuning parameter larger than 3.

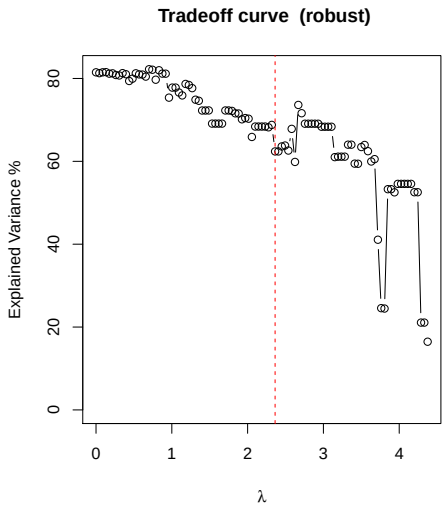

Figure 4.3: Tradeoff curve for robust sparse PCA computed on the car data set. The dashed line represents the λ selected by the BIC criterion.

Table 4.2 shows the resulting loadings for robust non-sparse PCA and robust sparse PCA, derived with $\lambda = 2.36$. By adding the penalty term in the objective function, the number of non-zero loadings is reduced from 56 to 16, whereas the total amount of explained variance in the first four PCs drops from 81% to 64%. We do find this decrease in explained variance acceptable, given the gained sparsity in the loadings matrix. This could facilitate interpretation, in particular for the higher order principal components. For instance, the fourth principal component is uniquely determined by `peak-rpm`.
Further exploratory data analysis can be done by making distance-distance plots (see Hubert *et al.*, 2002). Such a plot presents two different distance measures: the score distance of each observation in the space of the first k PCs, and the orthogonal distance of each observation to this space. Using cut-off values for both types of distances, outliers can be identified that do not follow the pattern the majority of the data follows. For details on the construction of these plots, we refer to Hubert *et al.* (2002). Figure 4.4 shows distance-distance plots for the car data, using standard and robust PCA, and their sparse versions, resulting in four different plots. As before, the first $k = 4$ PCs are retained, and λ is selected according to the BIC. The robust

Table 4.2: Loadings of the variables on the first four robust non-sparse ($\lambda = 0$) and robust sparse ($\lambda = 2.36$) PCs of the car data set.

	Robust PCA				Robust sparse PCA			
	PC1	PC2	PC3	PC4	PC1	PC2	PC3	PC4
symboling	-0.03	-0.04	0.03	-0.17	0	0	0	0
wheel-base	0.24	0.25	0.08	0.16	0	0.50	0	0
length	0.29	0.18	-0.05	0.04	0.24	0	0.85	0
width	0.26	0.16	0.14	0.03	0.21	0	0	0
height	0.08	0.39	-0.26	0.32	0	0.87	0	0
curb-weight	0.24	0.13	0.12	0.00	0.32	0	0	0
bore	0.24	0.16	-0.25	0.04	0.21	0	0.03	0
stroke	0.00	-0.24	0.29	-0.58	0	0	0	0
compression-ratio	-0.47	0.61	0.49	-0.11	-0.45	0	0.53	0
horsepower	0.36	-0.01	0.16	-0.20	0.43	0	0	0
peak-rpm	0.08	-0.38	0.60	0.64	0	0	0	1.00
city-mpg	-0.31	0.04	-0.02	0.14	-0.30	0	0	0
highway-mpg	-0.33	0.07	-0.04	0.14	-0.35	0	0	0
price	0.33	0.31	0.34	-0.12	0.40	0	0.06	0
EV %	49.20	15.54	10.12	5.97	45.73	8.32	6.03	4.16
Cumulative EV %	49.20	64.74	74.85	80.82	45.73	54.05	60.08	64.24

distance-distance plot (Figure 4.4 b) points out a very distinct outlier group (denoted by symbols $\times$) which in fact represents all car-models running on diesel. The robust sparse model (Figure 4.4 d) is also able to clearly identify this particular group of outliers. In contrast, when considering the standard non-sparse (Figure 4.4 a) and sparse (Figure 4.4 c) distance-distance plots, these outliers cannot be identified, since their presence is masked by the use of a non-robust diagnostic measure. We conclude that in this example only the robust procedure allows to detect the group of outliers, and that adding the sparsity condition did not affected the diagnostic power of the robust distance-distance plot.

Example 2

The yarn data set (see Swierenga *et al.*, 1999) contains near-infrared (NIR) spectra of 21 PET yarns of different density. 268 different wavelengths were measured, yielding a data set of size 21×268. As the algorithm discussed in Section 3 computes one (sparse) PC at a time and may stop after computing the kth component, it is especially useful in high-dimensional applications, where the actual information is restricted to a comparatively low-dimensional subspace. Due to this characteristic, computation time can be reduced tremendously, as in such settings usually only a few PCs are important. In the data set $k = 2$ PCs already explain more than 85% of the total (robust) variance, thus the iteration can be stopped after obtaining the first two principal components, rather than computing all $\min(n, p)$ loadings vectors. In this particular example this reduces computation time by 90%

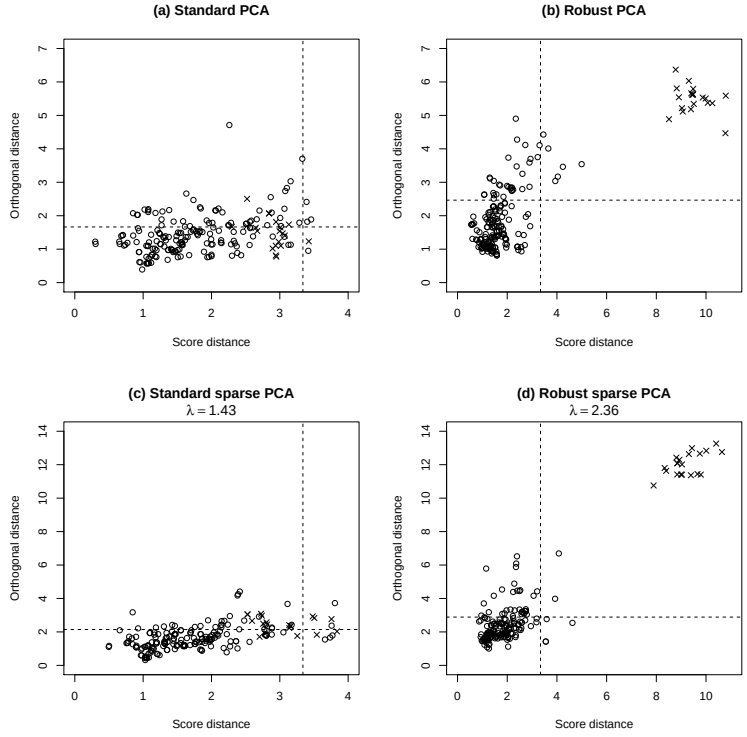

Figure 4.4: Distance-distance plots for standard and robust PCA and their sparse versions. In the robust plots vehicles running on diesel (×) are clearly distinguishable from vehicles using gasoline (○).

(from 41 to 4 seconds for standard and from 135 to 13 seconds for robust PCA on an AMD Athlon x64 X2 4200+ running at 2.2GHz).
Figure 4.5 shows the spectral lines of the 21 observations (black). Three spectral intervals A, B and C are pointed out, as the variables in these areas show a higher variance than in other regions. In interval B the single yarns are grouped together to 5 "clusters", whereas in region A and C this pattern cannot be observed and the yarns are more homogeneously structured. We add three outlying spectra (see Figure 4.5, in grey) in order to test the algorithm's robustness properties in high-dimensional scenarios.
We start by selecting an appropriate value for the number k of PCs to retain. The screeplot in Figure 4.6 conforms that $k = 2$ is a good choice, explaining most of the (robust) variance. Note that the large value for EV_1 for the standard method is mainly due to the fact that the sample variance is inflated by the outliers. The screeplot for standard PCA on the data set without the outlying spectra does resemble Figure 4.6 (b). Then, we use the tradeoff curve in Figure 4.7 for selecting a value of λ keeping a sufficiently large percentage of explained variance. For robust PCA we take $\lambda = 16.02$,

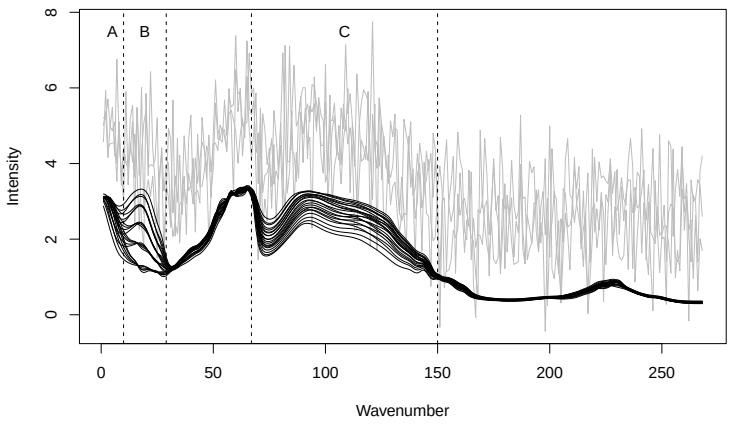

Figure 4.5: The yarn data set. The NIR spectrum of 21 different PET yarns (black), with intensity measured at 268 wavelengths. Outliers (in grey) were added for challenging the robust sparse PCA estimator.

a value at the end of the flat part of the curve and well before the sharp decrease in the tradeoff curve. For that value of λ we explain still 85% of the robust variance. The BIC criterion gives us a value of 19.55, which is not that different, but leads to a too large loss of explained robust variance. For standard PCA we take $\lambda = 12.77$ explaining 75% the total variance.
Figure 4.8 shows the loadings of the 268 variables, labeled with wavenumbers one to 268 for standard and robust PCA, and their sparse version. For standard PCA, the loadings in general do not seem to contain any interpretable structure and are heavily influenced by the outliers. The first standard sparse PC (panel b, dashed line), does hardly contain any zeros, whereas the second (panel d, dashed line) does only contain 11 non-zero loadings. However, this second sparse standard PC does not point out specific spectral ranges, but is mainly made up of single spikes, describing the outlier's random pattern. In contrast to this, robust PCA shows distinct features in all four plots. The first non-sparse robust PC (panel a, solid line) points out a peak at the spectrum's lower end. This peak is even much more clearly detected by the robust sparse model (panel b, solid line) and corresponds to the spectral range B in which the yarns reveal a rather "clustered" structure. Most of the loadings outside of the interval B are reduced to zero, illustrating that a sparse approach make interpretation easier. The second robust PC (panel c, solid line) is mainly made up of the wavelengths in spectral ranges A and C, corresponding to the wavelengths with high variability but without "cluster structure" among the yarns. Wavelengths outside of these intervals A and C contribute less to the second PC, as their

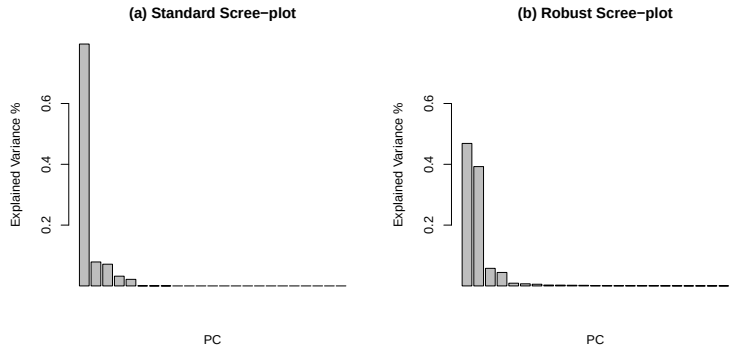

Figure 4.6: Scree-plots for a (a) standard and (b) robust PCA ($\lambda = 0$) for the yarn data set.

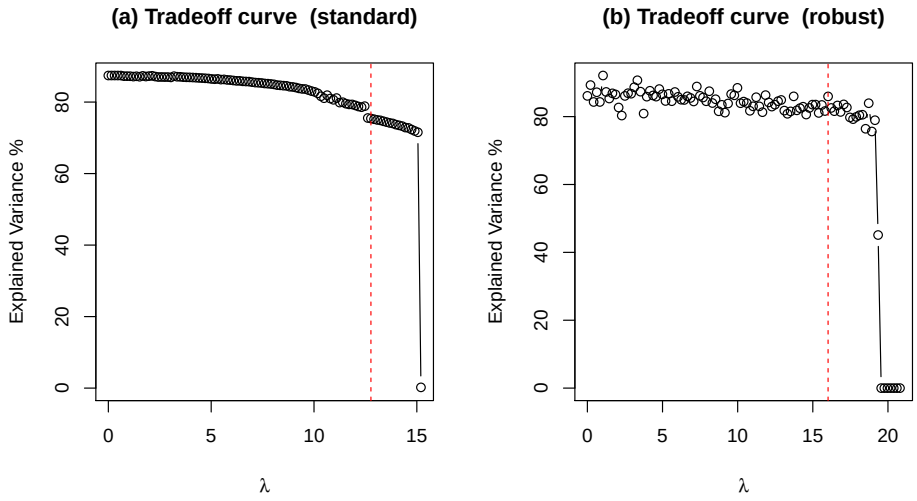

Figure 4.7: Tradeoff curves for standard and robust sparse PCA computed on the yarn data set. The dashed lines represent the selected value of the tuning parameter λ.

(absolute) loadings are quite low. The loadings of the second sparse robust PC (panel d, solid line) do even much better in separating the wavelengths in intervals A or C from the others; almost all loadings outside of these ranges are exactly equal to zero. As we can see from the tradeoff curve in Figure 4.7 (b), the sparse robust solution only explains 1% less variance than the non-sparse ($\lambda = 0$), whereas the number of non-zero loadings decreases from $2 \times 268 = 536$ to 159. Despite the noise added by the three outlying spectra, the robust sparse method is capable of finding distinct structures in the data.

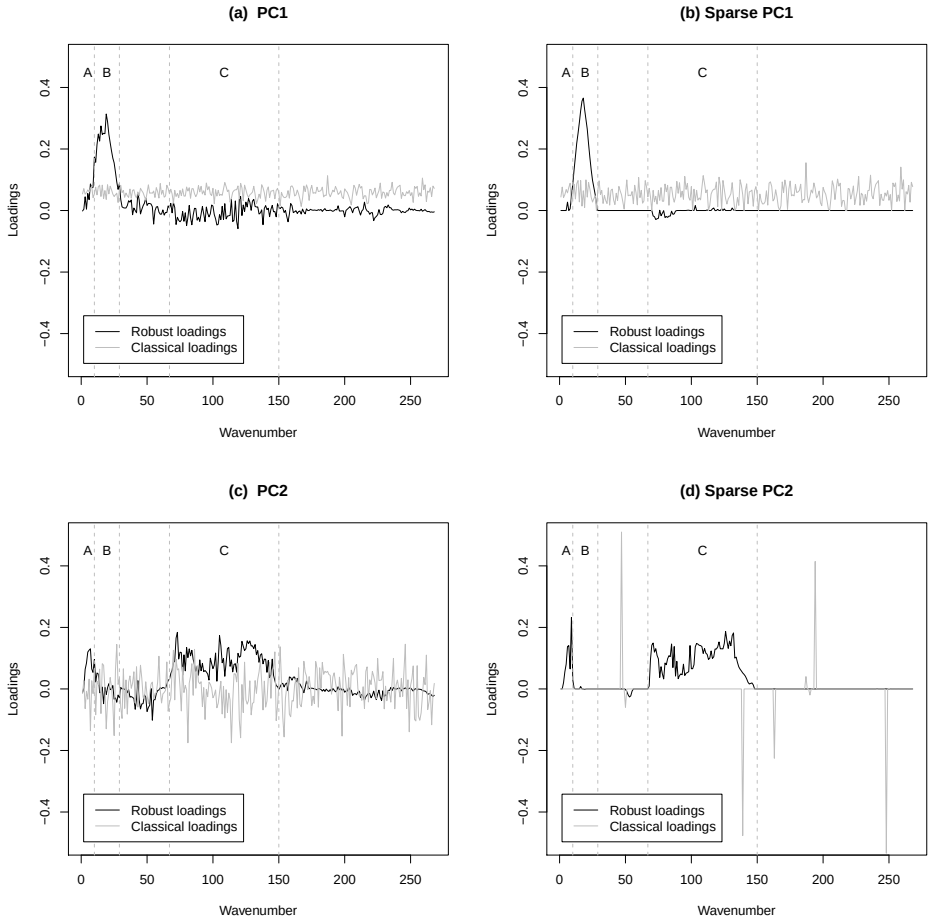

Figure 4.8: Loadings of the 268 variables on the first two principal components using standard (grey) and robust (black) PCA. Results are given for both sparse (right) and non-sparse (left) PCA for the yarn data set.

4.7 Concluding remarks

Sparse PCA delivers components that can be considered as a compromise between maximizing the variance and simplifying the interpretability. Robust sparse PCA also has the goal of simple interpretability, but the determination of the PCA directions is not affected by outlying observations. The proposed approach is based on the idea of projection-pursuit, maximizing a robust variance for finding the directions. Projection-pursuit based PCA has the further advantage that the components are extracted sequentially, which allows to stop the algorithm after a desired number of components. This is especially attractive for the analysis of data in high dimensions, with possibly fewer observations than variables.

The optimal level of the tuning parameter λ, optimal in terms of both interpretability and explained variance, can be determined by an information criterion like the BIC criterion introduced in equation (4.4.3). This criterion can be used for determining the sparsity parameter jointly for all extracted PCs. The simulations and the data examples have demonstrated that the robust sparse PCs can be accurately estimated with the Grid algorithm, that the results are resistant with respect to data outliers, and that the resulting sparsity patterns are useful. The tradeoff curve, visualizing the tradeoff between explained variance and sparsity, can be used as an exploratory tool for obtaining more guidance on an optimal sparsity level. An implementation of the algorithm is available in the R package pcaPP (Filzmoser *et al.*, 2010).
There are several questions we did not address and which are left for future research. For instance, one could think of a joint selection criterion for the number of principal components and the tuning parameter λ, as opposed to the two-step approach followed in this paper. Another limitation of the paper is that we only considered the L_1 norm in the constraint on the loadings. In regression analysis one frequently uses the L_2 norm, e.g. Maronna (2011) for regularized robust regression, but this will not lead to sparse solutions. Using the L_0 norm, though, does yield sparsity (see Farcomeni, 2009). Finally, one could consider to add a supplementary penalty on the norm of the score vectors, given in (4.2.3), to get both sparse loadings coefficients and score vectors, as in Witten *et al.* (2009). This would yield a sparse variant of robust low-rank approximations of a data matrix, as in Maronna and Yohai (2008).
A naive approach to robust sparse PCA would be to estimate a sparse robust covariance matrix, and then compute the eigenvectors of it. While sparse robust covariance matrices have recently been proposed (Croux *et al.*, 2010), this is not a useful approach since the eigenvectors will not inherit the sparsity of the matrix. A projection-pursuit approach, as undertaken in this paper, avoids this pitfall. Projection-pursuit approaches to sparse discriminant analysis and sparse canonical correlation analysis were recently proposed (see Witten and Tibshirani, 2011; Lykou and Whittaker, 2010), and robust version of these methods can be obtained along similar lines as in this paper.

Chapter 5

tclust: An R Package for a Trimming Approach to Cluster Analysis

Summary: Outlying data can heavily influence standard clustering methods. At the same time, clustering principles can be useful when robustifying statistical procedures. These two reasons motivate the interest in developing feasible robust model-based clustering approaches. With this in mind, an R package for performing non-hierarchical robust clustering, called **tclust** is presented here. Instead of trying to "fit" noisy data, a proportion α of the most outlying observations is trimmed. The **tclust** package efficiently handles different cluster scatter constraints. Graphical exploratory tools are also implemented to help the user make sensible choices for the trimming proportion as well as the number of clusters to search for.

Keywords: Model-based clustering, trimming, heterogeneous clusters
Co-authors: Luis A. García-Escudero, Agustín Mayo-Iscar

5.1 Introduction to robust clustering and tclust

Methods for cluster analysis are basically aimed at detecting homogeneous clusters with large heterogeneity among them. As happens with other (non-robust) statistical procedures, clustering methods may be heavily influenced by even a small fraction of outlying data. For instance, due to outlying observations, two or more clusters might artificially be joined or "spurious" non-informative clusters may be made up by only a few outlying observations (see, e.g. García-Escudero and Gordaliza, 1999; García-Escudero *et al.*, 2010). Therefore, the application of robust methods in this context is very advisable, especially in fully automatic clustering (unsupervised learning)

problems. Certain relations between cluster analysis and robust methods (Rocke and Woodruff, 2002; Hardin and Rocke, 2004; García-Escudero *et al.*, 2003; Woodruff and Reiners, 2004) are also a motivation for the interest of robust clustering techniques. For instance, robust clustering techniques can be used to handle "clusters" of highly concentrated outliers which are especially dangerous in (non-robust) estimation. García-Escudero *et al.* (2010) provides a recent survey of robust clustering methods.

The **tclust** package for the **R** environment for statistical computing (**R** Development Core Team, 2010a) implements different robust non-hierarchical clustering algorithms where trimming plays a key role. This package is available at `http://CRAN.R-project.org/package=tclust`. As trimming allows to remove a fraction α of the "most outlying" data, the strong influence of outlying observations can be avoided and robustness naturally arises. This trimming approach to clustering has been introduced in Cuesta-Albertos *et al.* (1997), Gallegos (2002), Gallegos and Ritter (2005) and García-Escudero *et al.* (2008). Trimming also serves to highlight interesting anomalous observations.

Trimming is not a new concept in statistics. For instance, the widely used trimmed mean for one-dimensional data removes a proportion $\alpha/2$ of the largest, and a proportion $\alpha/2$ of the smallest observations before computing the mean. However, it is not straightforward to extend this philosophy to cluster analysis, because most of these problems are of multivariate nature. Moreover, it is often the case that "bridge points" lying between clusters ought to be trimmed. Instead of forcing the statistician to define the regions to be trimmed in advance, the procedures implemented in **tclust** take the whole data structure into account in order to decide which parts of the sample should be discarded. By considering this type of trimming, these procedures are even able to trim outlying bridge points. The "self-trimming" philosophy behind these procedures is exactly the same as adopted by some well-known high breakdown-point methods (see, e.g., Rousseeuw and Leroy, 1987).

As a first example of this trimming approach, let us consider the trimmed k-means method introduced in Cuesta-Albertos *et al.* (1997). The function `tkmeans` from the **tclust** package implements this method. In the following example, this function is applied to a bivariate data set based on the Old Faithful geyser called `geyser2` that accompanies the **tclust** package. The code given below creates Figure 5.1.

```
R > library ("tclust")
R > data ("geyser2")
R > clus <- tkmeans (geyser2, k = 3, alpha = 0.03)
R > plot (clus)
```

In the data set `geyser2`, we are searching for $k = 3$ clusters and a proportion $\alpha = 0.03$ of the data is trimmed. The clustering results are shown in Figure

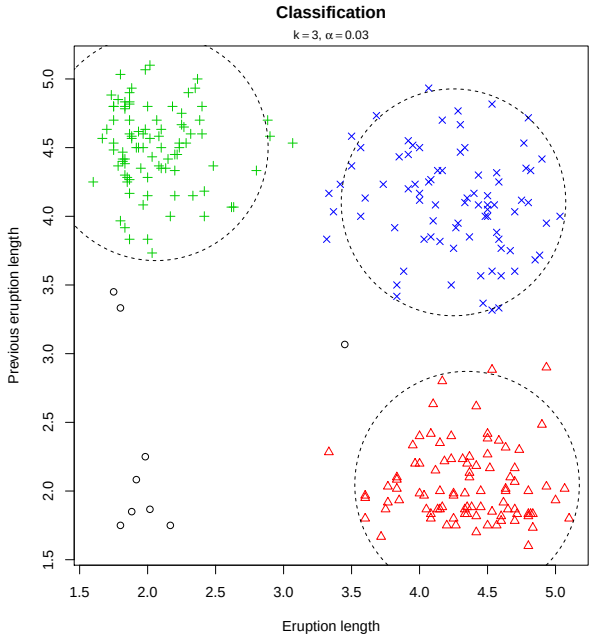

Figure 5.1: Trimmed k-means results with $k = 3$ and $\alpha = 0.03$ for the bivariate Old Faithful Geyser data. Trimmed observations are always denoted by the symbol "○" (in all the figures in this work).

5.1. Among this 3% of trimmed data, we can see 6 anomalous "short followed by short" eruptions lengths. Notice that an observation situated between the clusters is also trimmed.

The package presented here adopts a "crisp" clustering approach, meaning that each observation is either trimmed or fully assigned to a cluster. In comparison, mixture approaches estimate a cluster pertinence probability for each observation. Robust mixture alternatives have also been proposed where noisy data is tried to be fitted through additional mixture components. For instance, package **mclust** (Banfield and Raftery, 1993; Fraley and Raftery, 1998) and the Fortran program **emmix** (McLachlan and Peel, 2000) implement such robust mixture fitting approaches. Mixture fitting results can be easily converted into a "crisp" clustering result by converting the cluster pertinence probabilities into 0-1 probabilities. Contrary to these mixture fitting approaches, the procedures implemented in the **tclust** package simply remove outlying observations and do not intend to fit them at all. Package **tlemix** (see Neykov *et al.*, 2007) also implements a closely related trimming approach. As described in Section 5.3, the **tclust** package focuses on offering adequate cluster scatter matrix constraints leading to a wide range of clustering procedures, depending on the chosen constraint,

and avoiding the occurrence of spurious non-interesting clusters.
The outline of the paper is as follows: In Section 5.2 we briefly review the so-called "spurious outliers" model and show how to derive two different clustering criteria from it. Different constraints on the cluster scatter matrices and their implementation in the **tclust** package are commented in Section 5.3. Section 5.4 presents the numerical output returned by this package. Some brief comments concerning the implemented algorithms are given in Section 5.5. Some comments about how **tclust** performs compared to other robust clustering approaches are given in Section 5.6. Section 5.7 shows some graphical outputs that help us make sensible choices for the number of clusters and trimming proportion. Other useful plots summarizing the robust clustering results are shown in Section 5.8. Finally, Section 5.9 applies the **tclust** package to a well-know real data set.

5.2 Trimming and the spurious outliers model

Gallegos (2002) and Gallegos and Ritter (2005) propose the "spurious outliers model" as a probabilistic framework for robust crisp clustering. Let $f(\cdot;\mu,\Sigma)$ denote the probability density function of the p-variate normal distribution with mean μ and covariance matrix Σ. The "spurious-outlier model" is defined through "likelihoods" like

$$\left[\prod_{j=1}^{k}\prod_{i\in R_j} f(x_i;\mu_j,\Sigma_j)\right]\left[\prod_{i\in R_0} g_i(x_i)\right] \tag{5.2.1}$$

with $\{R_0,...,R_k\}$ being a partition of the set of indices $\{1,2,...,n\}$ such that $\#R_0 = \lceil n\alpha\rceil$. R_0 are the indices of the "non-regular" observations generated by other (not necessarily normal) probability density functions g_i. "Non-regular" observations can be clearly considered as "outliers" if we assume certain sensible assumptions for the g_i (see details in Gallegos, 2002; Gallegos and Ritter, 2005). Under these assumptions, the search of a partition $\{R_0,...,R_k\}$ with $\#R_0 = \lceil n\alpha\rceil$, vectors μ_j and positive definite matrices Σ_j maximizing (5.2.1) can be simplified to the same search (of a partition, vectors and positive definite matrices) by just maximizing

$$\sum_{j=1}^{k}\sum_{i\in R_j} \log f(x_i;\mu_j,\Sigma_j). \tag{5.2.2}$$

Notice that observations x_i with $i\in R_0$ are not taken into account in (5.2.2). Maximizing (5.2.2) with $k=1$ yields the Minimum Covariance Determinant (MCD) estimator (Rousseeuw, 1985).
Unfortunately, the direct maximization of (5.2.2) is not a well-defined problem when $k>1$. It is easy to see that (5.2.2) is unbounded without any

`restr`	`equal.weights = TRUE`	`equal.weights = FALSE`
`"eigen"`	*k-means* Cuesta-Albertos *et al.* (1997)	García-Escudero *et al.* (2008)
`"deter"`	Gallegos (2002)	This work
`"sigma"`	*Friedman and Rubin* (1967) Gallegos and Ritter (2005)	This work

Table 5.1: Clustering methods handled by **tclust**. Names in cursive letters are untrimmed ($\alpha = 0$) methods.

constraint on the cluster scatter matrices Σ_j. The `tclust` function from the **tclust** package approximately maximizes (5.2.2) under different cluster scatter matrix constraints which will be shown in Section 5.3.
The maximization of (5.2.2) implicitly assumes equal cluster weights. In other words, we are ideally searching for clusters with equal sizes. The function `tclust` provides this option by setting the argument `equal.weights = TRUE`. The use of this option does not guarantee that all resulting clusters exactly contain the same number of observations, but the method hence prefers this type of solutions.
Alternatively, different cluster sizes or cluster weights can be considered by searching for a partition $\{R_0, ..., R_k\}$ (with $\#R_0 = \lceil n\alpha \rceil$), vectors μ_j, positive definite matrices Σ_j and weights $\pi_j \in [0, 1]$ maximizing

$$\sum_{j=1}^{k} \sum_{i \in R_j} (\log \pi_j + \log f(x_i; \mu_j, \Sigma_j)). \tag{5.2.3}$$

The (default) option `equal.weights = FALSE` is used in this case. Again, the scatter matrices also have to be constrained such that the maximization of (5.2.3) becomes a well-defined problem. Note that equation (5.2.3) simplifies to (5.2.2) when assuming `equal.weights = TRUE` and all weights are equally set to $\pi_j = 1/k$.

5.3 Constraints on the cluster scatter matrices

As already mentioned, the function `tclust` implements different algorithms aimed at approximately maximizing (5.2.2) and (5.2.3) under different types of constraints which can be applied on the scatter matrices Σ_j. The type of constraint is specified by the argument `restr` of the `tclust` function. Table 5.1 gives an overview of the different clustering approaches implemented by the `tclust` function depending on the chosen type of constraint.
Imposing constraints is compulsory because maximizing (5.2.2) or (5.2.3) without any restriction is not a well-defined problem. Notice that an almost degenerated scatter matrix Σ_j would cause trimmed log-likelihoods

(5.2.2) and (5.2.3) to tend to infinity. This issue can cause a (robust) clustering algorithm of this type to end up finding "spurious" clusters almost lying in lower-dimensional subspaces. Moreover, the resulting clustering solutions might heavily depend on the chosen constraint. The strength of the constraint is controlled by the argument `restr.fact` ≥ 1 in the `tclust` function. The larger `restr.fact` is chosen, the looser is the restriction on the scatter matrices, allowing for more heterogeneity among the clusters. On the contrary, small values of `restr.fact` close to 1 imply very "equally scattered" clusters. This idea of constraining cluster scatters to avoid spurious solutions goes back to Hathaway (1985), who proposed it in mixture fitting problems.
Also arising from the spurious outlier model, other types of constraints have recently been introduced by Gallegos and Ritter (2009, 2010). These (closely related) constraints also serve to avoid degeneracy of trimmed likelihoods but they are not implemented in the current version of the **tclust** package.

5.3.1 Constraints on the eigenvalues

Based on the eigenvalues of the cluster scatter matrices, a scatter similarity constraint may be defined. With $\lambda_l(\Sigma_j)$ as the eigenvalues of the cluster scatter matrices Σ_j and

$$M_n = \max_{j=1,\ldots,k} \max_{l=1,\ldots,p} \lambda_l(\Sigma_j) \text{ and } m_n = \min_{j=1,\ldots,k} \min_{l=1,\ldots,p} \lambda_l(\Sigma_j) \qquad (5.3.1)$$

as the maximum and minimum eigenvalues, the restriction `restr = "eigen"` constrains the ratio M_n/m_n to be smaller or equal than a fixed value `restr.fact` ≥ 1. A theoretical study of the properties of this approach with `equal.weights = FALSE` can be found in García-Escudero *et al.* (2008).
This type of constraint limits the relative size of the axes of the equidensity ellipsoids defined through the obtained Σ_j when assuming normality. This way we are simultaneously controlling the relative group sizes and also the deviation from sphericity in each cluster.
Setting `equal.weights = TRUE`, `restr = "eigen"` and `restr.fact = 1` implies the most constrained case. In this case, the `tclust` function tries to solve the trimmed k-means problem as introduced by Cuesta-Albertos *et al.* (1997). This problem simplifies to the well-known k-means clustering criterion when no trimming is done (i.e. `alpha = 0`). The `tkmeans` function directly implements this most constrained application of the `tclust` function.

5.3.2 Constraints on the determinants

Another way of restricting cluster scatter matrices is constraining their determinants. Thus, if

$$M_n = \max_{j=1,...,k} |\Sigma_j| \text{ and } m_n = \min_{j=1,...,k} |\Sigma_j|$$

are the maximum and minimum determinants, we attempt to maximize (5.2.2) or (5.2.3) by constraining the ratio M_n/m_n to be smaller or equal than a fixed value `restr.fact`. This is done in the function `tclust` by using the option `restr = "deter"`.
Now, this type of constraint limits the relative volumes of the mentioned equidensity ellipsoids, but not the cluster shapes. The use of this type of constraint is particularly advisable when affine equivariance is required because this property is satisfied when `restr = "deter"`.
The untrimmed case `alpha = 0`, `restr = "deter"` and `restr.fact = 1` was already outlined in Maronna and Jacovkis (1974), as the only sensible way to avoid (Mahalanobis distance modified) k-means type algorithms to return clusters of a few almost collinear observations. The possibility of trimming data was also considered in Gallegos (2002) who implicitly assumed $|\Sigma_1| = ... = |\Sigma_k|$ (and so `restr.fact = 1`). The package presented here extends her approach to more general cases (`restr.fact` > 1).

5.3.3 Equal scatter matrices

Among the methods considered, **tclust** also implements a stronger type of constraint by setting `restr = "sigma"` which forces all cluster scatter matrices to be the same: $\Sigma_1 = ... = \Sigma_k$. This is known as the "determinantal" criterion and it goes back to Friedman and Rubin (1967). The trimmed version of this approach was introduced by Gallegos and Ritter (2005). The argument `restr.fact` is ignored when applying this type of constraint.

5.3.4 Example

In this example, we examine the influence of different constraints by applying the function `tclust` to the so-called `M5data` data set. This data set, which accompanies the **tclust** package, has been generated following the simulation scheme M5 introduced in García-Escudero *et al.* (2008). Thus it is a bivariate mixture of three simulated gaussian components with very different scatters and a clear overlap between two of these components. A 10% proportion of outliers is also added in the outer region of the bounding rectangle enclosing the three gaussian components. See Figure 5.2 for a graphical representation and García-Escudero *et al.* (2008) for more details on the structure of this `M5data` data set. Executing the following code yields Figure 5.3.

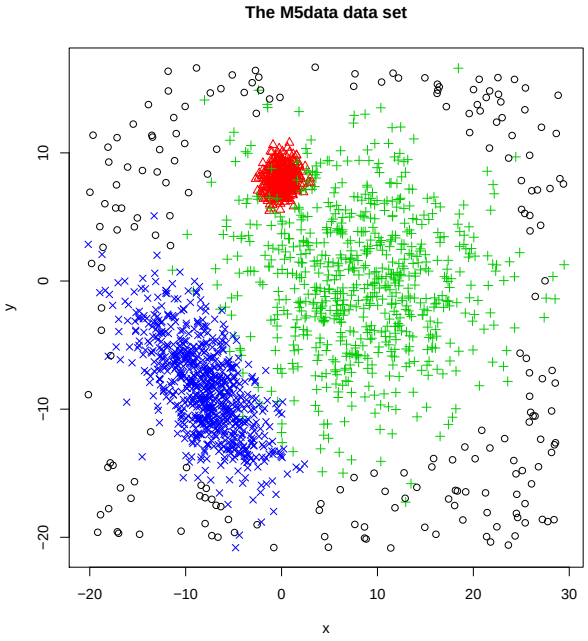

Figure 5.2: A scatter plot of the `M5data` data set. Different symbols are used for the data points generated by each of the three bivariate normal components and "○" for the added outliers.

```
R > data ("M5data")
R > x <- M5data[, 1:2]
R > res.a <- tclust (x, k = 3, alpha = 0.1, restr.fact = 1,
    +  restr = "eigen", equal.weights = TRUE)
R > res.b <- tclust (x, k = 3, alpha = 0.1, restr.fact = 1,
    + restr = "sigma", equal.weights = TRUE)
R > res.c <- tclust (x, k = 3, alpha = 0.1, restr.fact = 1,
    + restr = "deter", equal.weights = TRUE)
R > res.d <- tclust (x, k = 3, alpha = 0.1, restr.fact = 50,
    + restr = "eigen", equal.weights = FALSE)
R > plot (res.a, main = "/r")
R > plot (res.b, main = "/r")
R > plot (res.c, main = "/r")
R > plot (res.d, main = "/r")
```

Although different constraints are imposed, we are searching for $k = 3$ clusters and the trimming proportion is set to $\alpha = 0.1$ in all the cases. Note that only the clustering procedure introduced in García-Escudero *et al.* (2008), shown in Figure 5.3,(d), with a sufficiently large value of `restr.fact` approximately returns the three original clusters in spite of the very different

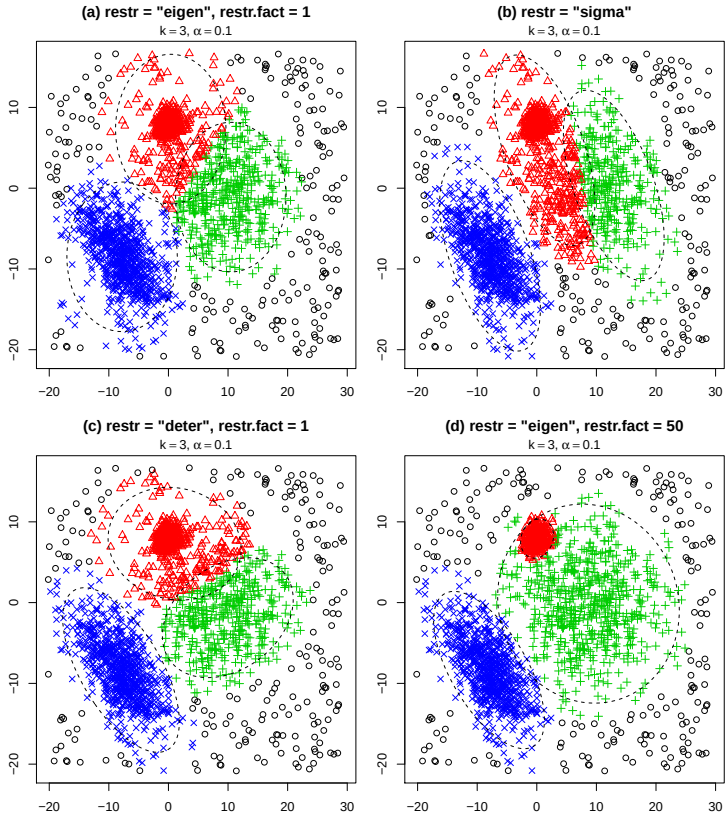

Figure 5.3: Results of the clustering processes for the `M5data` data set for different constraints on the cluster scatter matrices and the parameters $\alpha = 0.1$ and $k = 3$. Different colors and symbols represent each observation's individual cluster assignment.

cluster scatters and different cluster sizes. Moreover, this clustering procedure adequately handles the severe overlap of two clusters. The value `restr.fact = 50` has been chosen in this case because the eigenvectors of the covariance matrices of the three gaussian components satisfy restriction (5.3.1) for this value. Due to their underlying assumptions, the other three clustering methods (trimmed k-means in Figure 5.3,(a), Gallegos and Ritter (2005) in (b), Gallegos (2002) in (c)) return rather similarly structured clusters. In fact, we found spherical clusters in (a), clusters with the same scatter matrix in (b) and clusters with the same cluster scatter matrix determinant in (c). The `M5data` is perhaps a very "extreme" situation and restriction settings in (a), (b) and (c) can be useful (and easier to be interpreted) with not so extreme data sets and where the assumptions implied by these restriction settings hold.

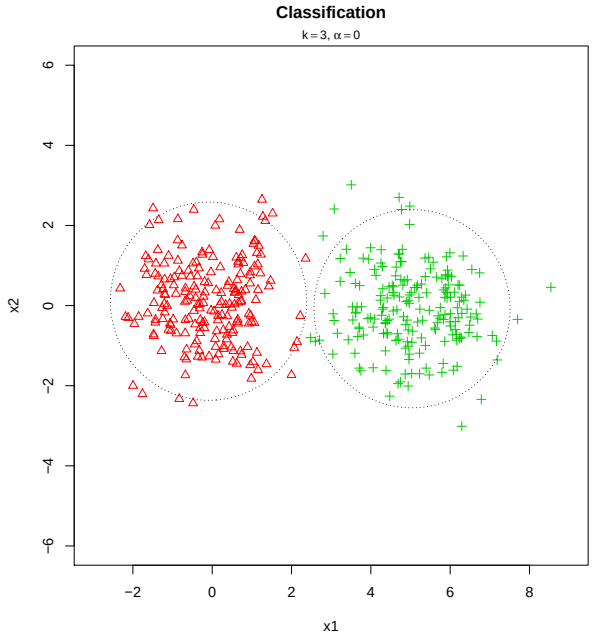

Figure 5.4: Applying `tclust` with $k = 3$ and $\alpha = 0$ on a simulated data set which originally consists of 2 clusters when `equal.weights = FALSE`.

5.4 Numerical output

The function `tclust` returns an S3 object containing the cluster centers μ_j by columns (`$centers`), scatter matrices Σ_j as an array (`$cov`), the weights (`$weights`), the number of observations in each group (`$size`) and the maximum value found for the trimmed log-likelihood objective function (5.2.2) or (5.2.3) (`$obj`). The vector `$cluster` provides the cluster assignment of each observation, whereas an artificial cluster "`0`" (without location and scatter information) is introduced which holds all trimmed data points.
Sometimes equations (5.2.2) and (5.2.3) maximize with some clusters remaining empty (see Figure 5.4). In this case, only information on the non-empty groups is returned. Notice that, if we are searching for k clusters, empty clusters can be found when a clustering solution for a number of clusters strictly smaller than k attains a higher value for (5.2.2) or (5.2.3) than the best solution found with k clusters. In this case, artificial empty clusters may be defined by considering sufficiently remote centers μ_j and scatter matrices Σ_j satisfying the specific constraints that are assigned to these empty clusters. They are chosen such that $f(\cdot; \mu_j, \Sigma_j)$ gives almost null density to all the observations in the sample. These artificially added centers and scatter matrices are not returned as output by the `tclust` function and a warning is issued. For instance, let us consider the following code

```
R > set.seed (10)
R > x <- rbind (rmvnorm (200, c (0, 0), diag (2)),
    +           rmvnorm (200, c (5, 0), diag (2)))
R > clus <- tclust (x, k = 3, alpha = 0, restr.fact = 1)
  Warning messages:
  1: In .tclust.warn(warnings, ret) :
    1 empty cluster has been detected - try reducing k.
  ...
R > plot (clus)
```

Although we are searching for $k = 3$ clusters, Figure 5.4 and the issued warning show that only 2 clusters are found. Notice that $k = 2$ is surely a more sensible choice for the number of clusters than $k = 3$ for this generated data set. Therefore, the detection of empty clusters, or clusters with few data points, can be helpful, providing valuable tools for making sensible choices for k as we will see in Section 5.7. On the other hand, the detection of empty clusters is very unlikely to happen when the argument `equal.weights = TRUE` is provided in the call to `tclust`.

5.5 Algorithms

The maximization of (5.2.2) or (5.2.3) considering different cluster scatter matrix constraints is not straightforward because of the combinatorial nature of the associated maximization problems.
The algorithm presented in García-Escudero *et al.* (2008) can be adapted to approximately solve all these problems. The methods implemented in **tclust** could be seen as Classification EM algorithms (Schroeder, 1976; Celeux and Govaert, 1992), whereas a certain type of "concentration" steps (see the fast-MCD algorithm in Rousseeuw and Van Driessen, 1999) is also applied. In fact, the concentration steps applied by the package **tclust** can be considered as an extension of those applied by the batch-mode k-means algorithm (Steinhaus, 1956; Forgy, 1965). It can be seen that the target function always increases throughout the application of concentration steps, whereas several random start configurations are needed in order to avoid ending trapped in local maxima. Therefore, `nstart` random initializations and `iter.max` concentration steps are considered. The probability that the algorithm converges close to the global optimum maximizing (5.2.2) or (5.2.3) increases with larger values of `nstart` and `iter.max`. The drawback of high values of `nstart` and `iter.max` obviously is the increasing computational effort.
In the concentration step, the centers and scatter matrices are updated by considering the cluster sample means and cluster sample covariance matrices. New cluster assignments are obtained by gathering the "closest" observations to the new centers. Mahalanobis distances, based on the computed cluster sample covariance matrices, are used in order to decide which are the closest

observations to each center. If needed, in the updating step, the cluster sample covariance matrices are modified as little as possible but in such a way that they satisfy the desired constraints (García-Escudero *et al.*, 2008). The main idea behind these "constrained" concentration steps is to replace the eigenvalues of the sample covariance matrices by optimally truncated eigenvalues, which satisfy the desired constraint. A more detailed description of the algorithm applied by `tclust` and the way the restrictions are forced onto the cluster scatter matrices can be found in Fritz *et al.* (2011).

5.6 Comparison with other robust clustering proposals

In this section, we briefly compare the performance of the clustering procedures implemented in the **tclust** package with respect to other robust clustering proposals in the literature.
The Partitioning Around Medoids (PAM) clustering method (Kaufman and Rousseeuw, 1990) has been proposed as a robust alternative to k-means clustering. It can be seen that the effect of the 6 anomalous "short followed by short" eruptions lengths in the lower left corner of Figure 5.1 do not affect the position of the k-medoid centers (see Figure 5.5,(a)) too much, nor the resulting clusters. However, in Figure 5.5,(b), we see that the clustering results with $k = 3$ are strongly affected when moving these 6 anomalous points toward a more distant position. On the other hand, in Figure 5.5,(c), we can see that these outlying data points do not affect the trimmed k-means based clustering at all once that they are trimmed.
In fact, only one single outlier placed in a very remote position is able to completely break down the PAM method (García-Escudero and Gordaliza, 1999). This also happens when applying **emmix**, which has a breakdown point of zero (Hennig, 2004). The **emmix** approach is able to obtain appropriate clustering results for the two data sets made of mixtures of symmetric and asymmetric t components as those shown in Figure 5.6. These two data sets contain three main clusters with some distant observations in the heavy tails of these t components, which would be considered as outliers when assuming normality. When applying the classification EM algorithm without trimming to these data sets, we are not able to find the three cluster structures and two main clusters are artificially joined together. However, when considering $\alpha = 0.05$, **tclust** perfectly avoids the harmful effect of the observations in the tails and still discovers the three clusters. In fact, almost all non-trimmed observations are correctly clustered in both, symmetric and asymmetric, cases. Any $\alpha > 0$ discarding the most outlying observations would give similar results. Moreover, it may be seen that the shape of the elliptical clusters are essentially discovered in the case of the symmetric-elliptical t components. In this example, we see that applying **tclust** to

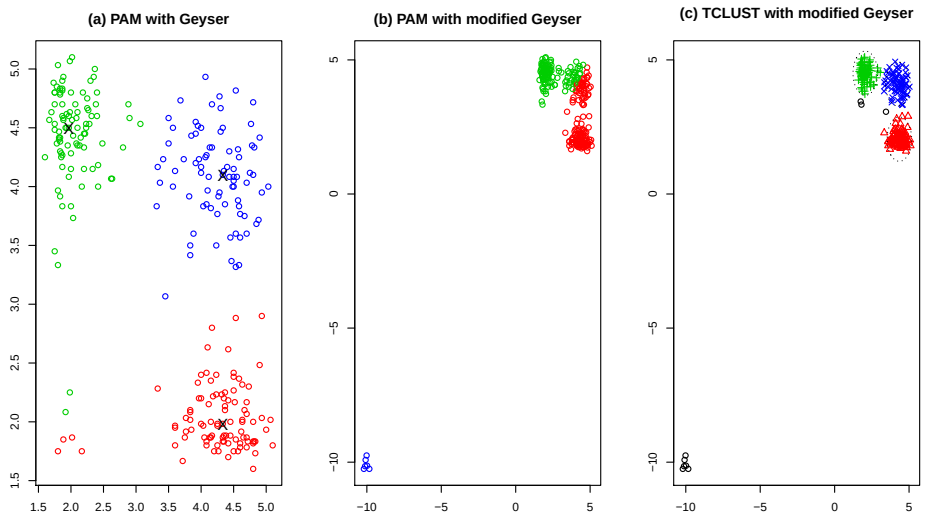

Figure 5.5: PAM's clustering results for the `geyser2` data with 3-medoids denoted by the symbol "×" in (a). PAM's clustering results for a modified `geyser2` data set in (b) and when applying `tkmeans` with $k = 3$ and $\alpha = 0.03$ in (c).

data sets including non-normally distributed components as those in Figure 5.6 may result in proper clustering solutions, this however cannot be guaranteed if the underlying distributions differ too much from normality. The closely related **tlemix** package allows to consider other non-normal models by taking advantage of the flexibility provided by the **FlexMix** package (Leisch, 2004). On the other hand, **tclust** focuses on normally distributed components and on the implementation of appropriate cluster scatter matrix constraints while **tlemix** does not. The **tlemix** mainly controls the minimum number of observations in a cluster.

The widely used **mclust** package considers a uniformly distributed component for explaining outlying data points. As we can see, this uniform component successfully accommodates the 10% "background noise" as seen in Figure 5.7,(a). However, it is not able to cope with a more structured noise pattern like the "helix" in Figure 5.7,(c) which also accounts for 10% of the data, although the information of a 10% contamination level was passed to **mclust**. Alternatively, the **tclust** package with $k = 2$ and $\alpha = 0.1$ properly discovers the outlying data points without trying to fit them.

Since groups of outliers may be considered as further clusters, it could be argued that robust clustering problems can always be solved by increasing the number of groups we are searching for. However, as explained in García-Escudero *et al.* (2010), this is not necessarily the best strategy. Firstly, sometimes the researcher fixes the number of clusters in advance, not being

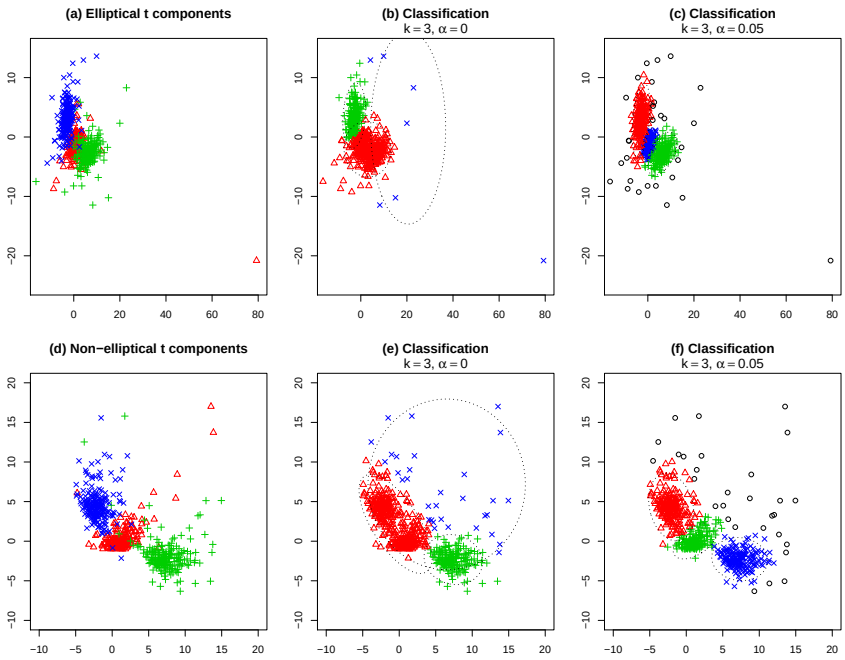

Figure 5.6: A data set made up of three elliptical t components is shown in (a) and three asymmetrical t components in (d). The associated clustering results when applying the `tclust` function with $k = 3$ and $\alpha = 0$ are shown in (b) and (e) and with $k = 3$ and $\alpha = 0.05$ in (c) and (f).

aware of the presence of a small amount of outlying observations. Secondly, it could lead to a quite large number of clusters when very scattered outliers are present in the data set.
A clear limitation of **tclust** is that it is not applicable on high-dimensional data sets, as the method in its current definition definitely needs a data set containing more observations than dimensions.

5.7 Selecting the number of groups and the trimming size

Perhaps one of the most complex problems when applying cluster analysis is the choice of the number of clusters, k. In some cases one might have an idea of the number of clusters in advance, but usually k is completely unknown. Moreover, in the approach proposed here, the trimming proportion α has also to be chosen without knowing the true contamination level.
As we will see through the following example, the choices for k and α are related problems that should be addressed simultaneously. It is important to see that a particular trimming level implies a specific number of clusters

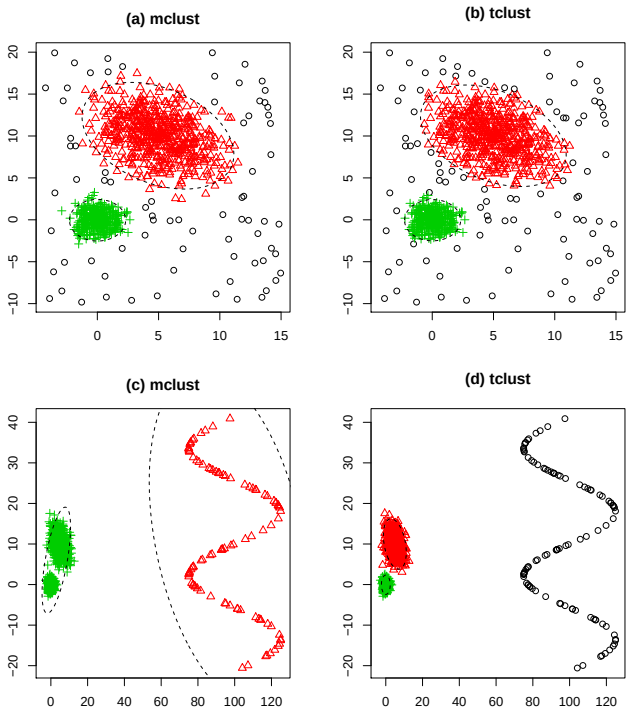

Figure 5.7: Clustering results for two simulated data sets when applying **mclust** with $k = 2$ in (a) and (c) and **tclust** with $k = 2$ and $\alpha = 0.1$ in (b) and (d).

and vice versa. This dependency can be explained as entire clusters tend to be trimmed completely when increasing α. On the other hand, when choosing α too low, groups of outliers might form new spurious clusters and thus it appears that the number of clusters found in the data set is higher. Moreover, the simultaneous choice of k and α depends on the type of clusters we are searching for and on the allowed differences between cluster sizes. These two aspects can be controlled by the choice of arguments `restr` and `restr.fact`.

To demonstrate the relation between α, k and `restr.fact`, let us consider `restr = "eigen"` and the data set in Figure 5.8 which could either be interpreted as a mixture of three components (a) or a mixture of two components (b) with a 10% outlier proportion. Both clustering solutions shown in Figure 5.8 are perfectly sensible and the final choice of α and k only depends on the value given to `restr.fact`. The code used to obtain Figure 5.8 is the following:

```
R > sigma1 <- diag (2)          ## EigenValues: 1, 1
R > sigma2 <- diag (2) * 8 - 2  ## EigenValues: 8, 4
R > sigma3 <- diag (2) * 50     ## EigenValues: 50, 50
```

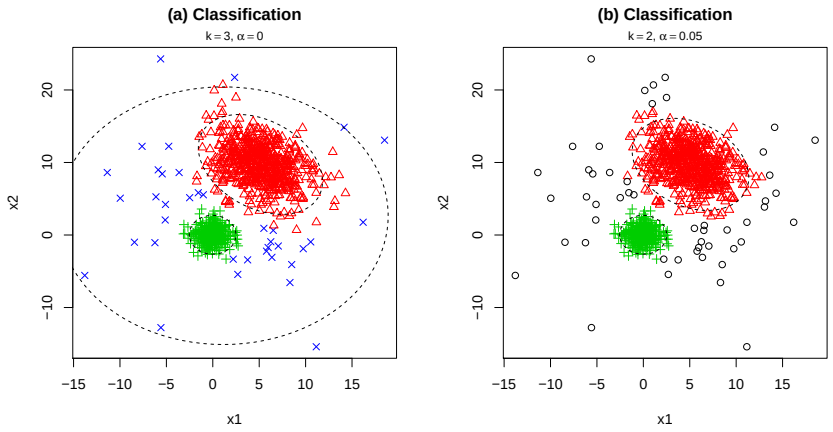

Figure 5.8: Clustering results for the simulated data set `mixt` with $k = 3$, $\alpha = 0$ and `restr.fact` $= 50$ (a) and $k = 2$, $\alpha = 0.1$ and `restr.fact` $= 8$ (b).

```
R > mixt <- rbind (
    + rmvnorm (360, mean = c (0.0,  0), sigma = sigma1),
    + rmvnorm (540, mean = c (5.0, 10), sigma = sigma2),
    + rmvnorm (100, mean = c (2.5,  5), sigma = sigma3))
R > plot (tclust (mixt, k = 3, alpha = 0.00, restr.fact = 50))
R > plot (tclust (mixt, k = 2, alpha = 0.05, restr.fact = 12))
```

Considering `sigma1` and `sigma2`, the quotient of the largest and smallest eigenvalue is 8, whereas the maximal quotient is 50 if we consider `sigma1`, `sigma2` and `sigma3`. Thus `restr.fact` $= 8$ would allow to consider two clusters while `restr.fact` $= 50$ would also allow to assume three groups there. Although the proportion of "contaminated" data is equal to 10%, the trimming level must be reduced to 5%, when considering $k = 2$, because the third (more scattered) gaussian component partially overlaps with the other two components.

Let us assume first that `restr` and `restr.fact` have been fixed in advance by the researcher who applies the robust clustering method. Even with this information and assuming $\alpha = 0$, choosing the appropriate number of clusters is not an easy task. The careful monitoring of the maximum value attained by log-likelihoods like those in (5.2.2) and (5.2.3) while changing k has traditionally been applied as a method for choosing the number of clusters when $\alpha = 0$. Moreover Bryant (1991) stated that the use of "weighted" log-likelihoods (5.2.3) is preferred to the use of log-likelihoods assuming equal weights (5.2.2). Notice that increasing k always causes the maximized log-likelihood (5.2.2) to increase too, and this could lead to "overestimate" the appropriate number of clusters (see García-Escudero *et al.*, 2011).

In this trimming framework, let us consider $\mathcal{L}^{\Pi}_{\texttt{restr.fact}}(\alpha, k)$ as the maximum value reached by (5.2.3) for each combination of a given set of values for k and α. García-Escudero *et al.* (2011) propose to monitor the "classification trimmed likelihoods" functionals

$$(\alpha, k) \mapsto \mathcal{L}^{\Pi}_{\texttt{restr.fact}}(\alpha, k)$$

while altering α and k, which yields an exploratory graphical tool for making sensible choices for parameters α and k. In fact, it is proposed to choose the number of clusters as the smallest value of k such that

$$\mathcal{L}^{\Pi}_{\texttt{restr.fact}}(\alpha, k+1) - \mathcal{L}^{\Pi}_{\texttt{restr.fact}}(\alpha, k) \tag{5.7.1}$$

is (close to) 0 except for small values of α. Once the number of clusters is fixed, a good choice for the trimming level is the first α_0 such that (5.7.1) is (close to) 0 for every $\alpha \geq \alpha_0$. Although we are convinced that monitoring the classification trimmed likelihoods functionals is very informative, no theoretical statistical procedures are available yet for determining when (5.7.1) can be formally considered as "close to 0".

The function `ctlcurves` in package **tclust** approximates the classification trimmed likelihoods by successively applying the `tclust` function for a sequence of values of k and α. A default value `restr.fact = 50` is considered but, if desired, other values of `restr.fact` can be passed to `tclust` via `ctlcurves` too.

For instance, the following code applied to the previously simulated `mixt` data set

```
R > plot (ctlcurves (mixt, k = 1:4, alpha =
    + seq (0, 0.2, by = 0.05)))
```

results in Figure 5.9. This figure shows that increasing k from 2 to 3 is needed when $\alpha = 0$, as the objective functions value differs noticeably between $k = 2$ and $k = 3$. On the other hand, increasing k from 2 to 3 is not needed anymore as the third (more scattered) "cluster" vanishes when trimming 5% of the most outlying observations. Thus, there is no discernable difference of the objective functions value with $\alpha \geq \alpha_0 = 0.05$ and $k \geq 2$. Increasing k from 3 to 4 is not needed in any case.

The previously described procedure for making sensible choices for parameters k and α requires an active role from the researcher. The type of restriction and the allowed restriction factor, which do not necessarily depend on the given data set, must be specified in advance. For instance, some specific clustering applications like "location-facilities" problems require almost spherical clusters that can be obtained by setting `restr = "eigen"` and a `restr.fact` close to 1. The researcher's decision on the restriction consequently modifies the proper determination of parameters k and α.

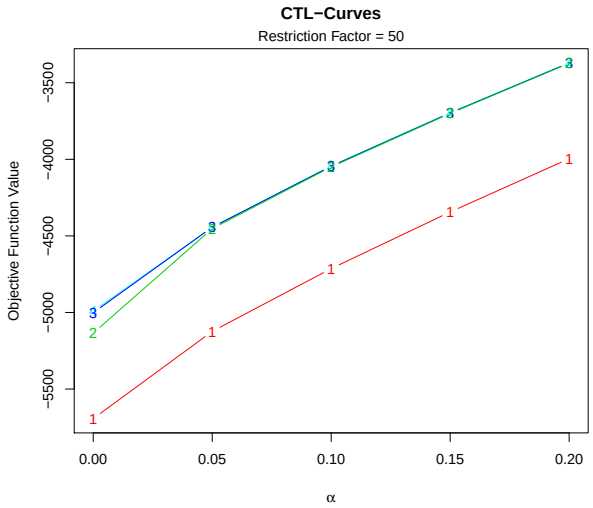

Figure 5.9: Classification trimmed likelihoods with $k = 1, ..., 4$, $\alpha = 0$, 0.05, ..., 0.2 and `restr.fact` $= 50$ for the `mixt` data set in Figure 5.8.

Due to the important role of the statement of `restr` and `restr.fact`, some general guideline for fixing them will be given here. For instance, as already commented, fixing `restr = "deter"` is recommended when only the relative cluster sizes shall be constrained, or when affine equivariance is clearly needed. On the other hand, using `restr = "eigen"` is advised when we want to simultaneously constrain relative cluster sizes and shapes.
With respect to the choice of `restr.fact`, we recommend to initially use large values when applying `ctlcurves`, thus, providing high flexibility to the clustering method. The default value `restr.fact = 50` is suggested for `ctlcurves`, as it worked well with a lot of data sets (especially if the variables have been properly standardized through the `scale` argument in the `tclust` function). The so obtained "sensible" values for k and α and their associated clustering solutions must be explored carefully. For instance, `tclust` issues a warning when the returned clustering solution has been "artificially restricted" by the algorithm, as shown in Section 5.9. This means, that the values M_n and m_n (see Sections 5.3.1 and 5.3.2) derived from the returned scatter matrices satisfy $M_n/m_n =$ `restr.fact`, because the algorithm has forced the chosen constraint, since the (unconstrained) group sample covariance matrices do not satisfy $M_n/m_n \leq$ `restr.fact`. In this situation, if no specific constraints are required, `restr.fact` may be increased stepwise until this warning disappears. Moreover, printing the object returned by the `ctlcurves` function points out all "artificially restricted" solutions for each computed combination of parameters k and α. In this way, if desired, we can easily search for clustering solutions which are not artificially restricted

and do not contain spurious clusters. Finally, the exploratory tools in Section 5.8 also help to evaluate whether all these parameters are reasonably chosen.
Note that arguments `nstart` and `iter.max` may be provided in the call to `ctlcurves` and they are internally passed to function `tclust`.
The curves presented in García-Escudero *et al.* (2003) can be considered as precedents of those we obtain by using the `ctlcurves` function. Trimmed likelihoods have also been taken into account in Neykov *et al.* (2007) for choosing k and α by using a BIC criterion.

5.8 Graphical displays

As seen in previous examples, the package **tclust** provides functions for visualizing the computed cluster assignments. One-dimensional, two-dimensional and higher-dimensional cases are visualized differently:

$p = 1$**:** The one-dimensional data set with the corresponding cluster assignments is displayed along the x-axis. Setting the argument `jitter = TRUE` jitters the data along the y-axis in order to increase the visibility of the actual data structure. Additionally, a (robust) scatter estimation of each cluster is also displayed.

$p = 2$**:** Tolerance ellipsoids are plotted additionally in order to visualize the estimated cluster scatter matrices.

$p > 2$**:** The first two Fisher's canonical coordinates are displayed in this case, which are computed based on the estimated cluster scatter matrices. Notice that trimmed observations are not taken into account when computing these coordinates, since they have been completely discarded. The implementation of these canonical coordinates is derived from the function `discrcoord` as implemented in the package **fpc** (Hennig, 2010).

A simple example demonstrates how the `plot` function works in different dimensions. The code:

```
R > geyser1 <- geyser2[, 1, drop = FALSE]
R > geyser3 <- cbind (geyser2, rnorm (nrow (geyser2)))
R > plot (tkmeans (geyser1, k = 2, alpha = 0.03), jitter = TRUE)
R > plot (tkmeans (geyser3, k = 3, alpha = 0.03))
```

yields Figure 5.10. For demonstrating the different plotting modes, we have selected one single variable from `geyser2` to obtain a one-dimensional data set (`geyser1`), and, added an additional normally distributed variable to `geyser2`, yielding a three-dimensional data set (`geyser3`). Figure 5.10 plots

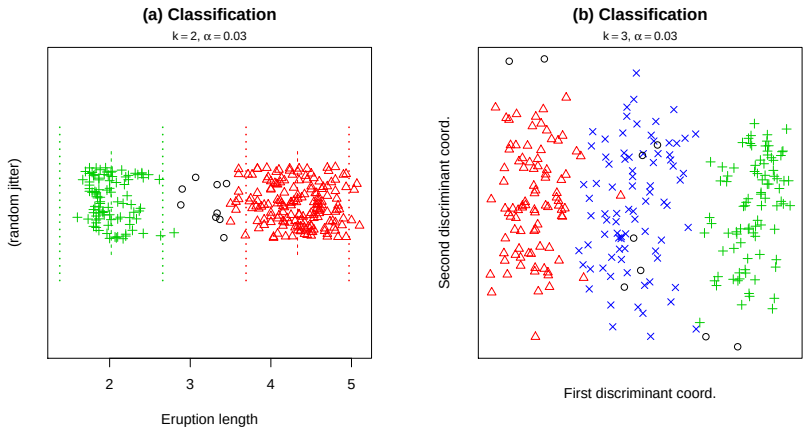

Figure 5.10: Trimmed k-means clustering results for `geyser1` (one-dimensional) in (a) and for `geyser3` (three-dimensional) in (b). These two data sets are based on `geyser2`. $k = 2$ is fixed in (a) and $k = 3$ in (b) while $\alpha = 0.03$ is fixed in both cases.

the results of the trimmed k-means robust clustering method for these two generated data sets.
Given a `tclust` object, some additional exploratory graphical tools can be applied in order to evaluate the quality of the cluster assignments and the trimming decisions. This is done by applying the function `DiscrFact`.
Let $\widehat{R} = \{\widehat{R_0}, \widehat{R_1}, ..., \widehat{R_k}\}$, $\widehat{\theta} = (\widehat{\theta_1}, ..., \widehat{\theta_k})$ and $\widehat{\pi} = (\widehat{\pi_1}, ..., \widehat{\pi_k})$ be the values obtained by maximizing (5.2.2) or (5.2.3) (we set $\widehat{\pi_j} = 1/k$ when maximizing (5.2.2)). $D_j(x_i; \widehat{\theta}, \widehat{\pi}) = \widehat{\pi_j}\phi(x_i, \widehat{\theta_j})$ is a measure of the degree of affiliation of observation x_i with cluster j. These values can be ordered as $D_{(1)}(x_i; \widehat{\theta}, \widehat{\pi}) \leq ... \leq D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi})$. Thus the quality of the assignment decision of a non trimmed observation x_i to the cluster j with $D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi}) = D_j(x_i; \widehat{\theta}, \widehat{\pi})$ can be evaluated by comparing its degree of affiliation with cluster j to the best second possible assignment. That is

$$\mathrm{DF}(i) = \log\left(D_{(k-1)}(x_i; \widehat{\theta}, \widehat{\pi})/D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi})\right) \text{ for } x_i \text{ not trimmed.}$$

Let $x_{(1)}, ..., x_{(n)}$ be the observations in the sample after being sorted according to their $D_{(k)}(\cdot; \widehat{\theta}, \widehat{\pi})$ values, i.e., $D_{(k)}(x_{(1)}; \widehat{\theta}, \widehat{\pi}) \leq ... \leq D_{(k)}(x_{(n)}; \widehat{\theta}, \widehat{\pi})$. It is not difficult to see that $x_{(1)}, ..., x_{(\lceil n\alpha \rceil)}$ are the trimmed observations which are not assigned to any cluster. Nevertheless, it is possible to compute the degree of affiliation $D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi})$ of a trimmed observation x_i to its nearest cluster. Thus, the quality of the trimming decision on this observation can be evaluated by comparing $D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi})$ to $D_{(k)}(x_{(\lceil n\alpha \rceil+1)}; \widehat{\theta}, \widehat{\pi})$, with $x_{(\lceil n\alpha \rceil+1)}$ being the non-trimmed observation with smallest value of

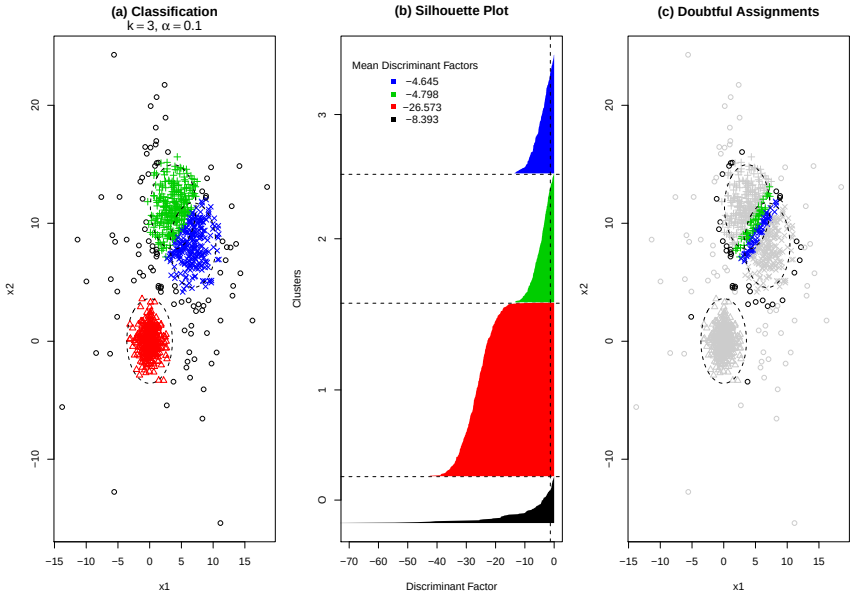

Figure 5.11: Graphical displays based on the DF(i) values for a `tclust` cluster solution obtained with $k = 3$, $\alpha = 0.1$, `restr.fact` $= 1$ and `equal.weights = TRUE` for the `mixt` data set.

$D_{(k)}(\cdot; \widehat{\theta}, \widehat{\pi})$. That is

$$\text{DF}(i) = \log\left(D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi}) / D_{(k)}(x_{(\lceil n\alpha \rceil + 1)}; \widehat{\theta}, \widehat{\pi})\right) \text{ for } x_i \text{ trimmed.}$$

Hence, discriminant factors DF(i) ≤ 0 are obtained for every observation in the data set, whether trimmed or not.
Observations with large DF(i) values (i.e. values close to zero) indicate doubtful assignments or trimming decisions. The use of this type of discriminant factors was already suggested in Van Aelst *et al.* (2006) in a clustering problem without trimming. "Silhouette" plots (Rousseeuw, 1987) can be used for summarizing the obtained ordered discriminant factors. Clusters in the silhouette plot with many large DF(i) values indicate the existence of not very "well-determined" clusters. The most "doubtful" assignments with DF(i) larger than a log(`threshold`) value are highlighted by the function `DiscrFact`.
Figure 5.11 shows the result of applying the `DiscrFact` function to a clustering solution found for the `mixt` data set appearing in Figure 5.8. The following code is used to obtain this figure:

```
R > clus.w <- tclust (mixt, k = 3, alpha = 0.1, restr.fact = 1,
    + equal.weights = TRUE)
R > discr.clus.w <- DiscrFact (clus.w, threshold = 0.1)
R > plot (discr.clus.w)
```

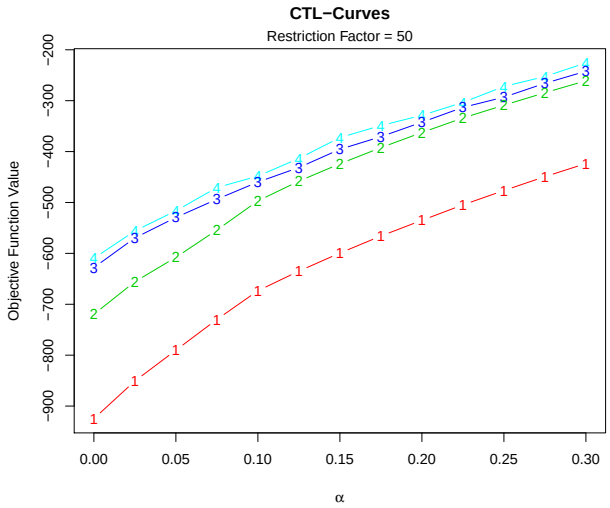

Figure 5.12: Classification trimmed likelihoods for $k = 1, ..., 4$ and $\alpha = 0$, .025, ..., .3 when `restr.fact` $= 50$ for the "Swiss Bank notes" data set.

The choice `threshold` $= 0.1$ means that a decision on a particular observation x_i is considered as doubtful, if the quality of the second best possible decision $(D_{(k-1)}(x_i; \widehat{\theta}, \widehat{\pi})$ or $D_{(k)}(x_{(\lceil n\alpha \rceil + 1)}; \widehat{\theta}, \widehat{\pi})$ for trimmed observations) is larger than one tenth of the quality of the actually made decision $(D_{(k)}(x_i; \widehat{\theta}, \widehat{\pi}))$.
Although Figure 5.9 suggests to choose $k = 2$, k has been increased to 3 in order to show how such a change leads to doubtful cluster assignment decisions which can be visualized by `DiscrFact`. Figure 5.11,(a) simply illustrates the cluster assignments and trimming decisions. The mentioned silhouette plot is presented in (b), whereas the doubtful decisions are marked in (c). All observations with $\mathrm{DF}(i) \geq \log(0.1)$ are highlighted as they are plotted darker/in color. Most of the doubtful decisions are located in the overlapping area of the two artificially found clusters (highlighted symbols "×" and "+"). Some doubtfully trimmed observations (highlighted symbol "∘") are located in the boundaries of these two clusters.

5.9 Swiss Bank notes data

The well-known "Swiss Bank notes" data set includes 6 numerical measurements (six-dimensional data set) made on 100 genuine and 100 counterfeit old Swiss 1000-franc bank notes (Flury and Riedwyl, 1988). The following code can be used to obtain the classification trimmed likelihoods shown in Figure 5.12.

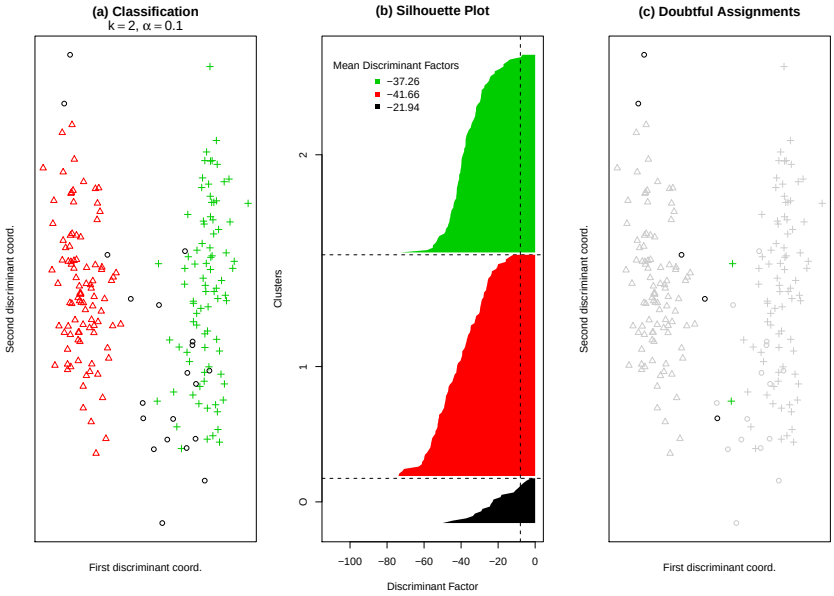

Figure 5.13: Clustering results with $k = 2$, $\alpha = 0.1$ and `restr.fact` $= 50$ summarized by the use of `DiscrFact` function for the "Swiss Bank notes" data set. The threshold value is chosen in order to highlight the 7 most doubtful cluster assignments.

```
R > data ("swissbank")
R > plot (ctlcurves (swissbank, k = 1:4, alpha = seq (0, 0.3,
    + by = 0.025)))
```

This figure indicates the clear existence of $k = 2$ main clusters ("genuine" and "forged" bills). Moreover, considering the clear difference between $\mathcal{L}^{\Pi}_{50}(0, 3)$ and $\mathcal{L}^{\Pi}_{50}(0, 2)$, we can see that a further cluster, i.e. $k = 3$, is needed when no trimming is allowed. This extra cluster can be justified by the heterogeneity of the group of forgeries (perhaps due to the presence of different sources of forged bills).
Considering Figure 5.12, the choice $k = 2$ and a value of α close to 0.1 also seem sensible. Notice that $\mathcal{L}^{\Pi}_{50}(\alpha, 3)$ is clearly larger than $\mathcal{L}^{\Pi}_{50}(\alpha, 2)$ for $\alpha < 0.1$ while these differences are not so big when $\alpha \geq 0.1$. We can even see smaller differences in the classification trimmed likelihood curves when increasing k from 3 to 4. However, these differences are less significant than those previously commented. More spurious clusters can be surely found but they have less entity and importance.
Figure 5.13 shows the clustering results with $k = 2$, $\alpha = 0.1$ and `restr.fact = 50` obtained by executing the code:

```
R > clus <- tclust (swissbank, k = 2, alpha = 0.1,
  + restr.fact = 50)
R > plot (DiscrFact (clus, threshold = 0.0001))
```

Notice that, in this example, we did not want to impose a specific constraint on the solution. Thus, the default parameter `restr.fact = 50` has initially been used in `ctlcurves`. After choosing the combination $\alpha = 0.10$ and $k = 2$, we could try to reduce the restriction factor which resulted in a warning:

```
R > tclust (swissbank, k = 2, alpha = 0.1, restr.fact = 40)
In .tclust.warn(warnings, ret):
  The result is artificially constrained due to restr.fact = 40.
```

Thus the choice `restr.fact = 50` seems appropriate as it does not artificially restrict the result, whereas a slightly smaller restriction factor (40) does. By examining the sizes of the obtained groups, we see that no spurious groups are found with `restr.fact = 50`:

```
R > clus$size
  [1] 95 85
```

We have used `restr = "eigen"` in this example, but `restr = "deter"` can be also successfully applied with smaller values of `restr.fact`.
We also use the function `DiscrFact` to summarize the obtained clustering results. The two first Fisher's canonical coordinates derived from the final cluster assignments are plotted. The threshold value 0.0001 is chosen in order to highlight the 7 most doubtful decisions.
Finally, Figure 5.14 shows a scatterplot of the fourth ("Distance of the inner frame to lower border") against the sixth variable ("Length of the diagonal") with the corresponding cluster assignments. We use the symbols "`G`" for the genuine bills and "`F`" for the forged ones. The 7 most doubtful decisions (i.e., the observations with largest $\mathrm{DF}(i)$ values that were highlighted in Figure 5.13,(c)) are surrounded by circles in this figure. We can see that "Cluster 1" essentially includes most of the "forged" bills while "Cluster 2" includes most of the "genuine" ones. Among the trimmed observations, we can find a subset of 15 forged bills following a clearly different forgery pattern that has been previously commented by other authors (see, e.g. Flury and Riedwyl, 1988; Cook, 1999). These most doubtful assignments include 5 "genuine" bills that have perhaps been wrongly trimmed.

5.10 Conclusion

We have presented a package called **tclust** for robust (non-hierarchical) clustering. As the package is implemented in a flexible manner, only the

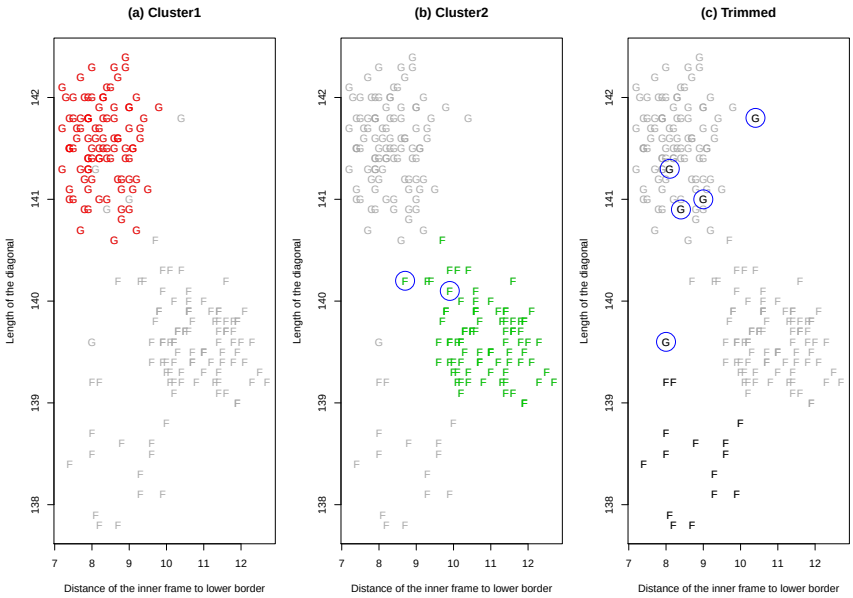

Figure 5.14: Clustering results with $k = 2$, $\alpha = .1$ and `restr.fact` $= 50$ for the "Swiss Bank notes" data set. Only the fourth and sixth variables are plotted. The 7 most doubtful decisions are rounded by a circle symbol.

restrictions on the cluster scatters have to be changed in order to carry out different robust clustering algorithms. Robustness is achieved by trimming a specific amount of observations which are identified as the "most outlying" ones.

This **R**-package implements robust clustering approaches which have already been described in the literature, whereas some of these approaches are extended to gain flexibility. The package also provides some graphical tools which on the one hand help to chose appropriate parameters (`ctlcurves`) and on the other hand help to estimate the adequacy of a particular clustering solution (`DiscrFact`).

The future work on this package focuses on implementing further types of scatter restrictions, making the algorithm even more flexible and on providing more numerical tools for automatically choosing the number of clusters and the trimming proportion.

Acknowledgements:

This research is partially supported by the Spanish Ministerio de Ciencia e Innovación, grant MTM2008-06067-C02-01, and 02 and by Consejería de Educación y Cultura de la Junta de Castilla y León, GR150.

Chapter 6

A fast algorithm for robust constrained clustering

Summary: The application of "concentration" steps is the main principle behind Forgy's k-means algorithm and Rousseeuw and van Driessen's fast-MCD algorithm. Although they share this principle, it is not completely straightforward to combine both algorithms for developing a clustering method which is not affected by a certain proportion of outlying observations and that is able to cope with non spherical groups or with groups with different weights. However, these approaches can be successfully combined by additionally controlling the relative cluster scatters in the concentration steps. In this way, the appearance of uninteresting spurious clusters is avoided. An algorithm which implements such "constrained concentration" steps in a computationally efficient way will be presented in this work.

Keywords: Cluster Analysis, Robustness, Trimming, k-means, MCD, Trimmed k-means

Co-authors: Luis A. García-Escudero, Agustín Mayo-Iscar

This research was partially supported by Ministerio de Ciencia y Tecnología and FEDER grant BFM2002-04430-C02-01 and by Consejería de Educación y Cultura de la Junta de Castilla y León grant PAPIJCL VA074/03.

6.1 Introduction

It is easy to see certain relations between Forgy's k-means algorithm (Forgy, 1965) and the fast-MCD algorithm (Rousseeuw and Van Driessen, 1999). These two widely applied algorithms play a very important role in Cluster Analysis and in Robust Statistics, respectively. The connection between

Trimmed k-means	**Fast-MCD**
...	...
− Randomly draw k centers.	− Randomly draw a center and a scatter matrix.
...	...
− Trim a proportion α of the most remote observations to these k centers, considering Euclidean distances.	− Trim a proportion α of the most remote observations to the center, considering Mahalanobis distances.
− Compute k new centers using the non-trimmed observations.	− Compute a new center and scatter matrix using the non-trimmed observations.
...	...
− Return the k centers leading to the "best" value of the target function.	− Return the center and scatter matrix leading to the "best" value of the target function.

Table 6.1: Schematic description of the differences between the trimmed k-means and fast-MCD algorithms.

these methods mainly refers to the application of so called"concentration" steps which will be explained later in Section 6.3.
This relation gets clearer when comparing the fast-MCD algorithm to the trimmed k-means algorithm (García-Escudero *et al.*, 2003), since trimming (outlying) data is an important characteristic of both methods. Notice, that the trimmed k-means algorithm simplifies to Forgy's k-means algorithm when the trimming level α is set to 0. More information on the trimmed k-means procedure can be found in Cuesta-Albertos *et al.* (1997) and García-Escudero and Gordaliza (1999). A very simplified comparison of the concentration steps for trimmed k-means and fast-MCD is given in Table 6.1.
The main drawback of using k-means and trimmed k-means is that they ideally search for spherically scattered groups and for clusters with equal size, whereas in many clustering problems the clusters we are looking for do not necessarily follow these assumptions. Thus, in this work, we focus on general "heterogeneous" clustering problems where elliptically contoured clusters can also be expected. Further we expect the data to contain a certain fraction α of outlying observations which would negatively affect classical clustering procedures (see García-Escudero *et al.*, 2010). In this setup, it seems logical to combine the clustering capabilities of k-means with the ability to robustly estimate covariance structures provided by the fast-MCD algorithm. Thus, we can think of applying the trimmed k-means algorithm, but considering

Mahalanobis distances when identifying the closest cluster center to each observation. The centers and scatter matrices are updated by computing the sample means and sample covariance matrices of the observations assigned to each cluster. Unfortunately, this "naive" combination of both algorithms does not provide sensible clustering results, since large groups sometimes tend to "eat" smaller ones, and the algorithm ends up finding spurious groups with few, almost collinear observations. This problem has already been described in Maronna and Jacovkis (1974).
A sensible way to address this issue is to impose constraints which control the relative difference among cluster scatters. In fact, many well-know clustering methods implement (implicitly and explicitly) such constraints on the relative cluster sizes, as for example the k-means method assumes spherical clusters with similar scatter. With this idea in mind, García-Escudero *et al.* (2008) introduced the TCLUST method, which is based on a relative size constraint on the eigenvalues of the scatter matrices defining the shape of the elliptically contoured groups. The idea of using restrictions of this type goes back to Hathaway (1985) where related constraints were proposed in a mixture fitting framework.
From a computational point of view, solving the TCLUST problem is not an easy task. One of the most critical issues in this algorithm is how to enforce the relative size constraints. Unfortunately, this is the computational bottleneck of the algorithm, because a complex optimization problem must be solved in each concentration step. In this work, we present a computational efficient algorithm for such "constrained concentration" steps, which clearly speeds up the TCLUST algorithm and makes it computationally feasible for practical applications.
It is also important to note that the idea of such constrained concentration steps can be easily extended to other constrained clustering methods like Gallegos (2002) and Gallegos and Ritter (2005).
The methodology of the discussed approach is explained in Section 6.2, whereas in Section 6.3 the corresponding algorithm is presented. Section 6.4 contains a simulation study, investigating the performance of the algorithm and Section 6.5 concludes.

6.2 Constrained robust clustering and TCLUST

Given a sample of observations $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ in $\mathbb{R}^p$ and $\phi(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ the probability density function of a p-variate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, we consider the following general *robust clustering problem*:

> Search for a partition $R_0, R_1, ..., R_k$ of the indices $\{1, ..., n\}$ with $\#R_0 = \lceil n\alpha \rceil$, centers $\boldsymbol{m}_1, ..., \boldsymbol{m}_k$, symmetric positive semidefinite scatter matrices $\boldsymbol{S}_1, ..., \boldsymbol{S}_k$ and weights $p_1, ..., p_k$ with $p_j \in$

$[0,1]$ and $\sum_{j=1}^{k} p_j = 1$, which maximizes

$$\sum_{j=1}^{k} \sum_{i \in R_j} \log\left(p_j \phi(\boldsymbol{x}_i; \boldsymbol{m}_j, \boldsymbol{S}_j)\right). \tag{6.2.1}$$

Depending on the constraints imposed on the weights p_j and scatter matrices $\boldsymbol{S}_j$, the maximization of (6.2.1) for $\alpha = 0$ leads to well established clustering procedures. For instance, a very constrained setup, assuming equal weights $p_1 = ... = p_k$ and scatter matrices $\boldsymbol{S}_1 = ... = \boldsymbol{S}_k = \sigma^2 \boldsymbol{I}$ with $\boldsymbol{I}$ being the identity matrix and $\sigma > 0$, yields the k-means method. The determinantal criterion introduced by Friedman and Rubin (1967) is obtained when assuming $p_1 = ... = p_k$ and $\boldsymbol{S}_1 = ... = \boldsymbol{S}_k = \boldsymbol{S}$ with $\boldsymbol{S}$ being a positive definite matrix. In general, the "log-likelihood" in (6.2.1) when $\alpha = 0$ and $p_1 = ... = p_k$ corresponds to the Classification-Likelihood (see e.g. Scott and Symons, 1971). The use of (6.2.1) assuming different weights p_j goes back to Symons (1981) and Bryant (1991) and is also known as the Complete-Data-Likelihood approach to Cluster Analysis.

Trimmed alternatives to the previously commented approaches can be constructed by introducing a trimming level $\alpha > 0$ to (6.2.1), which yields "trimmed log-likelihoods". This way, the trimmed k-means method in Cuesta-Albertos *et al.* (1997) extends k-means and the trimmed determinantal criterion in Gallegos and Ritter (2005) extends the determinantal criterion. Notice that $\lceil n\alpha \rceil$ observations (R_0) are not taken into account when computing (6.2.1), and thus the harmful effect of outlying observations, up to a contamination α, can be avoided. Gallegos and Ritter (2005) introduce the so-called "spurious outlier model" that theoretically justifies the use of trimmed log-likelihoods like in (6.2.1).

It is also important to note that the robust clustering problem reduces to the fast-MCD method when assuming $k = 1$ (i.e. only partitioning the data into $\lceil n\alpha \rceil$ outliers and $\lfloor n\,(1-\alpha) \rfloor$ regular observations). The fact that the same target function defines both problems emphasizes the relation between robust clustering methods and the MCD estimator.

It is straightforward to see, that the direct maximization of (6.2.1) without any constraint on the scatter matrices is not a well defined problem, since a single cluster scatter matrix $\boldsymbol{S}_j$ with $\det(\boldsymbol{S}_j) \to 0$ causes (6.2.1) to tend to infinity. Thus partitions containing spurious clusters are quite likely and preferred to more sensible solutions. This explains why the previously described "naive" algorithm (combining the two algorithms from Table 6.1) does not work appropriately.

In order to make the maximization of (6.2.1) a well defined problem, García-Escudero *et al.* (2008) propose to additionally consider the following eigenvalue-ratio constraint on the scatter matrices $\boldsymbol{S}_1, ..., \boldsymbol{S}_k$:

$$\frac{\max_{j,l} \lambda_{j,l}}{\min_{j,l} \lambda_{jl}} \leq c, \tag{6.2.2}$$

with $\lambda_{j,l}$ as the eigenvalues of the corresponding scatter matrices $\boldsymbol{S}_j$ (for $j = 1, ..., k$ and $l = 1, ..., p$) and $c \geq 1$ as a constant which controls the strength of the constraint (6.2.2). The maximization of (6.2.1) under the eigenvalue-ratio constraint (6.2.2) leads to the TCLUST problem introduced by García-Escudero *et al.* (2008). The smaller the value of c, the stronger is the restriction imposed on the solution, yielding the strongest constraint $c = 1$, which corresponds to the k-means procedure with different cluster weights.
The TCLUST method has good theoretical and robustness properties but no practically applicable algorithm is available yet when $k \cdot p$ is moderately large. With this in mind, a feasible algorithm for efficiently implementing this method will be described in the following section.

6.3 Algorithm

An algorithm for approximately maximizing (6.2.1) under the constraint (6.2.2) has been presented in García-Escudero *et al.* (2008), whereas a significantly faster approach will be presented here. Further, an inaccuracy in the presentation of the algorithm in García-Escudero *et al.* (2008) will be corrected here.
Both algorithms can be seen as a trimmed version of the Classification Expectation-Maximization (EM) algorithms proposed in Schroeder (1976) and Celeux and Govaert (1992).
In the *E-step*, at a given iteration, each observation $\boldsymbol{x}_i$ is assigned to the cluster with closest center. Since we are considering different weights and scatter matrices, the distance of an observation $\boldsymbol{x}_i$ to the center of cluster j is proposed to be quantified by a so-called "discriminant function":

$$D_j(\boldsymbol{x}_i; \theta) = p_j \phi(\boldsymbol{x}_i; \boldsymbol{m}_j, \boldsymbol{S}_j).$$

with $\theta = (p_1, ..., p_k, \boldsymbol{m}_1, ..., \boldsymbol{m}_k, \boldsymbol{S}_1, ..., \boldsymbol{S}_k)$ as the set of cluster parameters in the current iteration of the algorithm. The smaller $D_j(\boldsymbol{x}_i; \theta)$, the larger is the distance of observation $\boldsymbol{x}_i$ to a center $\boldsymbol{m}_j$. Further,

$$D(\boldsymbol{x}_i; \theta) = \max\{D_1(\boldsymbol{x}_i; \theta), ..., D_k(\boldsymbol{x}_i; \theta)\} \tag{6.3.1}$$

defines an overall measure for outlyingness.
Notice that if $k = 1$, observations with largest (6.3.1) are those with smallest Mahalanobis distances

$$(\boldsymbol{x}_i - \boldsymbol{m}_1)' \boldsymbol{S}_1^{-1} (\boldsymbol{x}_i - \boldsymbol{m}_j). \tag{6.3.2}$$

These observations are taken into account in the concentration steps of the fast-MCD algorithm.

Further, when assuming $p_1 = ... = p_k$ and $\boldsymbol{S}_1 = ... = \boldsymbol{S}_k = \sigma^2 \boldsymbol{I}$, observations with largest (6.3.1) are those with smallest values of

$$\min_{j=1,...,k} \|\boldsymbol{x}_i - \boldsymbol{m}_j\|^2,$$

as considered in the concentration steps of the (trimmed) k-means algorithm. With this notation, the $\lceil n\alpha \rceil$ observations $\boldsymbol{x}_i$ with smallest values of $D(\boldsymbol{x}_i; \theta)$ can be discarded as possible outliers (trimmed), whereas $D_j(\boldsymbol{x}_i; \theta)$ is used to assign the remaining observations to one of the j groups. Notice, that in contrast to mixture clustering approaches, this approach fully assigns each (non-trimmed) observation to a cluster and thus is a "crisp" clustering method.
In a second step, the *M-step*, the cluster parameters are updated, based on the non-trimmed observations and the corresponding cluster assignments. At this point it is crucial to constrain the cluster scatter matrices for avoiding spurious clusters.

A more detailed presentation of the proposed algorithm is given as follows:

1. *Initialization:* The procedure is initialized `nstart` times by selecting different $\theta^0 = (p_1^0, ..., p_k^0, \boldsymbol{m}_1^0, ..., \boldsymbol{m}_k^0, \boldsymbol{S}_1^0, ..., \boldsymbol{S}_k^0)$. For this purpose we propose to randomly select $k(p+1)$ observations and to accordingly compute k cluster centers $\boldsymbol{m}_j^0$ and scatter matrices $\boldsymbol{S}_j^0$ from the chosen data points. Afterwards the cluster scatter matrix constraints are applied to these $\boldsymbol{S}_j^0$, as described in Step 2.2. Weights $p_1^0, ..., p_k^0$ in the interval $(0, 1)$ and summing up to 1 are also randomly chosen.

2. *Concentration step:* The following steps are executed until convergence (i.e. $\theta^l = \theta^{l-1}$) or a maximum number of iterations `iter.max` is reached.

   2.1. *Trimming and cluster assignment (E-step):* Based on the current parameter set $\theta^l = (p_1^l, ..., p_k^l, \boldsymbol{m}_1^l, ..., \boldsymbol{m}_k^l, \boldsymbol{S}_1^l, ..., \boldsymbol{S}_k^l)$ the $\lceil n\alpha \rceil$ observations with smallest values of $D_j(\boldsymbol{x}_i, \theta^l)$ are trimmed. Each remaining observation x_i is then assigned to a cluster j, such that $D_j(\boldsymbol{x}_i, \theta^l) = D(\boldsymbol{x}_i, \theta^l)$. This yields a partition $R_0, R_1, ..., R_k$ of the indices $\{1, ..., n\}$ holding the trimmed observations in R_0 and all observations belonging to cluster j in R_j for $j = 1, ..., k$.

   2.2. *Update parameters (M-step):* Given $n_j = \#R_j$, the weights are updated by

   $$p_j^{l+1} = n_j / [n(1-\alpha)]$$

   and the centers by the sample means

   $$\boldsymbol{m}_j^{l+1} = \frac{1}{n_j} \sum_{i \in R_j} \boldsymbol{x}_i.$$

Updating the scatter estimates is more difficult, as the sample covariance matrices

$$\boldsymbol{T}_j = \frac{1}{n_j} \sum_{i \in R_j} (\boldsymbol{x}_i - \boldsymbol{m}_j^{l+1})(\boldsymbol{x}_i - \boldsymbol{m}_j^{l+1})',$$

may not satisfy the specified eigenvalue-ratio constraint. In this case, the singular-value decomposition of $\boldsymbol{T}_j = \boldsymbol{U}_j' \boldsymbol{D}_j \boldsymbol{U}_j$ is considered, with $\boldsymbol{U}_j$ being an orthogonal matrix and $\boldsymbol{D}_j = \text{diag}(d_{j1}, d_{j2}, ..., d_{jp})$ a diagonal matrix. Let us define the truncated eigenvalues as

$$d_{jl}^m = \begin{cases} d_{jl} & \text{if } d_{jl} \in [m, cm] \\ m & \text{if } d_{jl} < m \\ cm & \text{if } d_{jl} > cm \end{cases} \tag{6.3.3}$$

with m as some threshold value. The scatter matrices are updated as

$$\boldsymbol{S}_j^{l+1} = \boldsymbol{U}_j' \boldsymbol{D}_j^* \boldsymbol{U}_j,$$

with $\boldsymbol{D}_j^* = \text{diag}\left(d_{j1}^{m_{\text{opt}}}, d_{j2}^{m_{\text{opt}}}, ..., d_{jp}^{m_{\text{opt}}}\right)$ and m_{opt} minimizing

$$m \mapsto \sum_{j=1}^{k} n_j \sum_{l=1}^{p} \left(\log\left(d_{jl}\right) + \frac{d_{ij}^m}{d_{ij}} \right). \tag{6.3.4}$$

As shown in Remark 3, this expression has to be evaluated only $2kp + 1$ times for exactly finding this minimum.

3. *Evaluate target function:* After the concentration steps the value of the target function (6.2.1) is computed. The set of parameters yielding the highest value of this target function is returned as the algorithm's output.

The proposed algorithm can be used to solve the maximization of (6.2.1) when assuming equal weights $p_1 = ... = p_k$, by simply setting all weights constantly to $p_j^l = 1/k$ within each iteration.

Remark 1 *The number of random starts* `nstart` *and the maximum number of constrained-concentration steps* `iter.max` *depends on the complexity of the processed data set. Experience shows that not excessively large values of* `nstart` *and* `iter.max` *are needed to obtain a proper solution if, apart from outliers, the cluster structure is easy to be discovered (see also Section 6.4). García-Escudero* et al. *(2011) provides some graphical tools which help to make appropriate choices for the number of groups k and the trimming level* α.
If the constraints on the eigenvalues are not considered, the algorithm essentially coincides with the method proposed in Neykov et al. *(2007), which*

is also based on trimmed likelihoods. However, as already mentioned, explicitly stating relative cluster scatter constraints and providing a computational procedure for solving them is very important in this approach to robust clustering.

Remark 2 *The main novelty of this algorithm compared to García-Escudero* et al. *(2008) is how the constraint on the eigenvalue ratio is imposed. Equation (3.4) in García-Escudero* et al. *(2008) constrains eigenvalues by solving the minimization problem*

$$(d_{11}^*, d_{12}^*, ..., d_{jl}^*, ..., d_{kp}^*) \mapsto \sum_{j=1}^{k} n_j \sum_{l=1}^{p} \left(\log\left(d_{jl}\right) + \frac{d_{ij}^*}{d_{ij}} \right), \tag{6.3.5}$$

under the restriction

$$(d_{11}^*, d_{12}^*, ..., d_{jl}^*, ..., d_{kp}^*) \in \Lambda, \tag{6.3.6}$$

with Λ *as the cone*

$$\Lambda = \left\{ d_{jl}^* : d_{jl}^* \leq c \cdot d_{rs}^* \text{ for every } j, r \in \{1, ..., k\} \text{ and } l, s \in \{1, ..., p\} \right\}. \tag{6.3.7}$$

This is clearly a more complex problem than minimizing (6.3.4) because its complexity tremendously increases with the number of groups k *and the dimension* p*. In García-Escudero* et al. *(2008), the problem of minimizing (6.3.5) in* Λ *was translated into a quadratic programming problem which can be approximately solved by recursive projections onto cones (Dykstra, 1983). However, as this computationally intensive problem must be solved in each concentration step, the algorithm becomes extremely slow and even unfeasible for high values of* $k \cdot p$*. Moreover, there was a mistake in García-Escudero* et al. *(2008), as the term* n_j *in (6.3.5) was omitted, and thus the algorithm proposed there can only be applied onto similarly sized clusters.*

Remark 3 *There is a closed form for obtaining* m_{opt} *(and thus, the constrained eigenvalues) just by evaluating function (6.3.4)* $2pk + 1$ *times. Let us consider* $e_1 \leq e_2 \leq ... \leq e_{2kp}$ *obtained by ordering the following* $2pk$ *values:*

$$d_{11}, d_{12}, ..., d_{jl}, ..., d_{kp}, d_{11}/c, d_{12}/c, ..., d_{jl}/c, ..., d_{kp}/c.$$

After that, let us consider any $2pk + 1$ *values* $f_1, ..., f_{2kp+1}$ *satisfying:*

$$f_1 < e_1 \leq f_2 \leq e_2 \leq ... \leq f_{2kp} \leq e_{2kp} < f_{2kp+1},$$

and, compute

$$m_i = \frac{\sum_{j=1}^{k} n_j \left(\sum_{l=1}^{p} d_{jl}(d_{jl} < f_i) + \frac{1}{c} \sum_{l=1}^{p} d_{jl}(d_{jl} > c f_i) \right)}{\sum_{j=1}^{k} n_j \left(\sum_{l=1}^{p} ((d_{jl} < f_i) + (d_{jl} > c f_i)) \right)},$$

for $i = 1, ..., 2kp + 1$*. Finally, choose* m_{opt} *as the value of* m_i *which yields the minimum value of (6.3.4).*

Remark 4 *An implementation of the algorithm described in this work has been made available through the* ***R*** *package* `tclust` *at* `http://CRAN.R-project.org/package=tclust`*. Further, little changes to this algorithm yield a generalized version of a robust clustering method introduced by Gallegos (2002), who constrains the scatter matrices' determinants instead of their eigenvalues.*

6.4 Simulation Study

As the discussed algorithm has already been compared to other robust clustering approaches on simulated and real data sets (see Fritz *et al.*, 2011), this work concludes with a simulation study investigating the effect of the choice of parameters `iter.max` (number of concentration steps) and `nstart` (number of random initializations) on the performance of the algorithm.
In this simulation study, a so-called M5 type data set is considered, which is based on the "M5 scheme" as introduced in García-Escudero *et al.* (2008). These simulated $p \geq 2$ dimensional data sets consist of three partly overlapping clusters generated from three p-variate normal distributions with means

$$\boldsymbol{\mu}_1 = (0, \beta, 0, \ldots, 0)\,, \boldsymbol{\mu}_2 = (\beta, 0, \ldots, 0) \text{ and } \boldsymbol{\mu}_3 = (-\beta, -\beta, 0, \ldots, 0)\,,$$

with $\beta \in \mathbb{R}^+$ and covariance matrices

$$\boldsymbol{\Sigma}_1 = \operatorname{diag}(1, \ldots, 1)\,, \boldsymbol{\Sigma}_2 = \operatorname{diag}(45, 30, 1, \ldots, 1) \text{ and}$$

$$\boldsymbol{\Sigma}_3 = \begin{pmatrix} 15 & -10 & 0 & \ldots & 0 \\ -10 & 15 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The parameter β specifies how strong the clusters overlap, i.e. smaller values (e.g. 6) yield heavily overlapping clusters, whereas larger values (e.g. 10) yield a better separation of the clusters and thus a problem which is easier to solve. Theoretical cluster weights are fixed as $(0.2, 0.4, 0.4)$, implying that the first cluster size is half of the size of clusters two and three. Further two different types of outliers are considered which are added to the data:

- Type 1: Uniformly distributed outliers in the bounding box of the data.
- Type 2: Uniformly distributed outliers restricted to a random hyperplane of dimension $p - 1$.

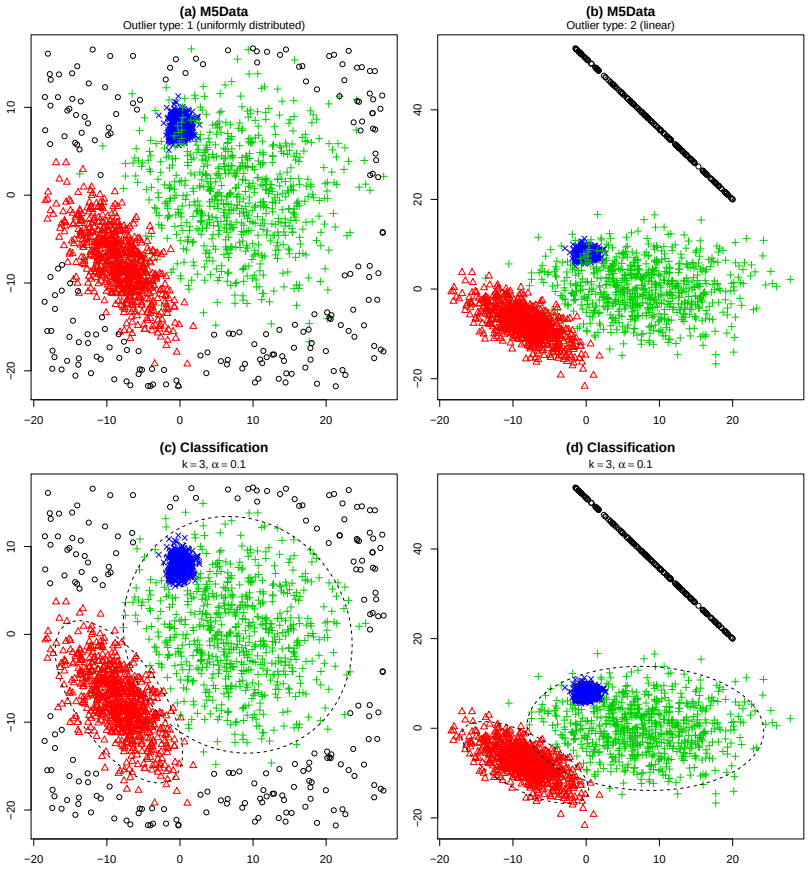

Figure 6.1: An M5 type data set in two dimensions with uniformly distributed outliers (a) and outliers restricted to a line (b). Plots (c) and (d) show the corresponding clustering results obtained by `tclust`.

All outliers are drawn under the restriction that the squared Mahalanobis distance (see Equation 6.3.2) of each outlier with respect to all three clusters must be larger than the 0.975 quantile of the chi-squared distribution with p degrees of freedom.
Choosing a number of observations $n = 2000$, parameters $p = 2$ and $\beta = 8$ and a 10% outlier portion results in data sets as shown in Figure 6.1 (a) and (b) with outlier types 1 and 2 respectively. Considering outlier type 2 in a two dimensional data set reduces the space of the outliers to a line as seen in the mentioned figure. Panels (c) and (d) in the same figure show the corresponding cluster results computed with an R implementation of the described algorithm from package `tclust`. Apparently the cluster structure is captured nicely by the algorithm, only at the boundaries and overlapping regions of the clusters some differences between the theoretical and the computed cluster assignment can be noticed.

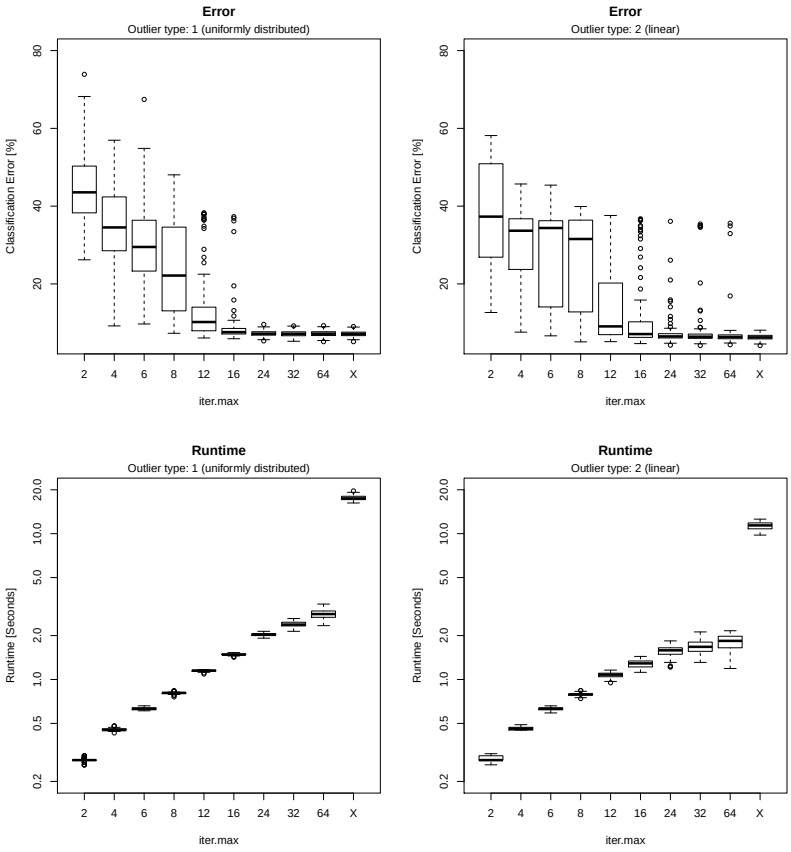

Figure 6.2: Classification errors and runtimes of the `tclust` algorithm applied to simulated M5 type data sets for different values of `iter.max` and `nstart` $= 32$ when $p = 10$ and $\beta = 6$ are fixed.

For the simulation study the algorithm has been applied on data sets of dimension $p = (2, 6, 10)$, with separation of the cluster determined by $\beta = (6, 8, 10)$ and the two described outlier types on a data set with $n = 2000$, split into three clusters of sizes 360, 720 and 720 and a 10% outlier portion yielding 200 contaminated observations. For each possible combination of these parameters 100 samples have been drawn. Further the `tclust` algorithm has been applied on each of these samples with values $(2, 4, 6, 8, 12, 16, 24, 32, 64)$ for parameters `iter.max` and `nstart`. Moreover, for each of these settings, a very precise "reference result" has been computed with parameters `iter.max` $= 10000$ and `nstart` $= 200$. All simulations were run on an AMD Phenom II X6 1055T at 2.8GHz.
Figure 6.2 shows the box plots of the classification errors in percent and runtimes for different values of `iter.max` and the two outlier types, using `nstart` $= 32$, $p = 10$ and $\mu = 6$. The label "X" at the very right of each plot

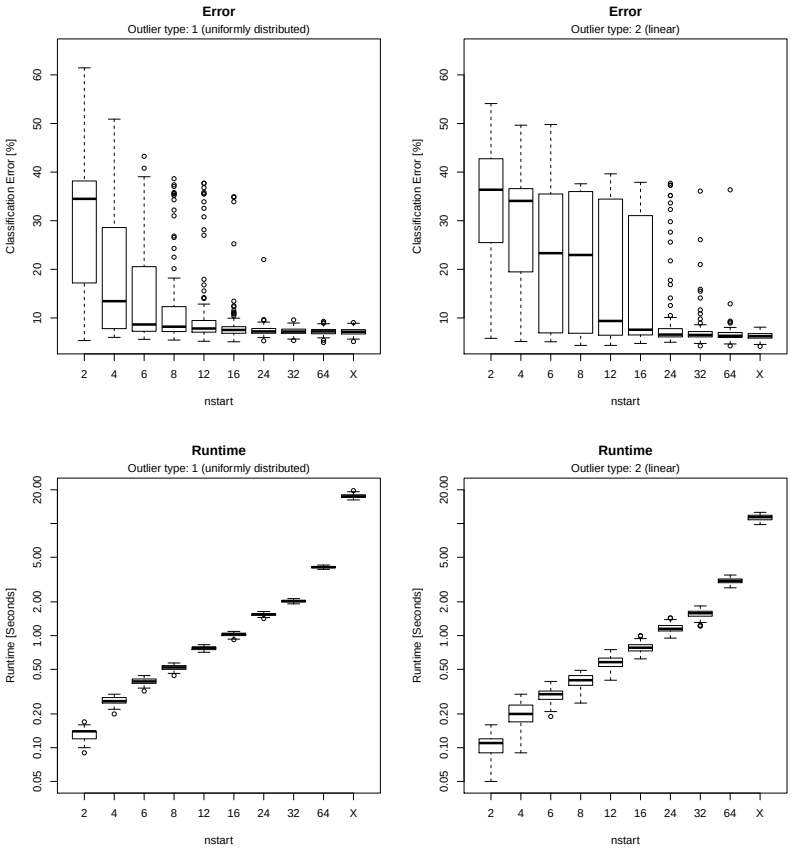

Figure 6.3: Classification errors and runtimes of the `tclust` algorithm applied to simulated M5 type data sets for different values of `nstart` and `iter.max` $= 24$ when $p = 10$ and $\beta = 6$ are fixed.

represents the "reference result", which is assumed as to be very close to the theoretically optimal solution. Differences between the outlier types can be seen, as in panel (a) a value of `iter.max` $= 24$ already gives a result very similar to the reference. On the other hand in panel (b), with the outliers restricted to a hyperplane of dimension $p - 1$, even a value `iter.max` $= 64$ yields three out of 100 solutions, which apparently differ from the results in the reference solution "X".
When considering the runtimes in panels (c) and (d) a general pattern can be observed, as at a certain point the runtimes do not increase linearly with the parameter `iter.max` anymore. This is apparently caused by the convergence criterion in Step 2 of the algorithm, which stops the iterations earlier than specified by the chosen value of `iter.max` as soon as the same parameters are obtained within two consecutive concentration steps. The runtimes are quite similar for the different outlier types, however for values

`iter.max` larger than 16 the algorithm applied to data contaminated by the second outlier type seems to converge slightly faster. This can be explained, as for the majority of the samples, the second outlier type is easier to grasp. As soon as the cluster structure has been found approximately, the outliers can be identified easily, as most of them do not overlap with the actual clusters. This is not the case with the first outlier scheme. Although the cluster structure can be found quickly, and most of the observations are assigned correctly, the outliers located in the outer regions of the clusters make it more difficult for the algorithm to converge.
Figure 6.3 shows a similar scenario, but here the parameter `nstart` is varied and the parameters `iter.max` $= 24$, $p = 10$ and $\mu = 6$ are fixed. When applying the algorithm on data contaminated with the first outlier type, results computed with `nstart` $= 24$ are almost equal to the reference solution "X" as shown in panel (a). However, when the second outlier type is considered, even `nstart` $= 64$ is not sufficient for obtaining a completely converged solution. The corresponding runtimes as shown in Figure 6.3 (c) and (d) depend linearly on the parameter `nstart`, as expected. Due to the earlier convergence of the algorithm, when contamination of the second type is present (as commented before), runtimes in panel (d) are slightly lower than in panel (a).
Figure 6.4 gives classification errors (a) and runtimes (b) for different values of β and p, the first outlier type and values `iter.max` $= 64$ and `nstart` $= 64$ fixed. As with increasing β the clusters are better separable, a larger value of β yields smaller classification errors. Due to the better separation of the clusters the algorithm converges faster when β is large, resulting in lower runtimes.
Also larger values of p decrease the classification error, as in higher dimensional space the clusters are separated more clearly. Further an increase of the number of dimensions clearly increases the runtimes, which is expected due to the algorithms's structure.

6.5 Conclusions

A feasible algorithm for robust heterogeneous clustering has been presented. The keystone of the algorithm are the constrained concentration steps which successfully combine the concentration steps of the fast-MCD algorithm with Forgy's k-means algorithm. The discussed algorithm computes these constrained concentration steps, only by additionally evaluating an explicit function at $2kp+1$ values within each iteration. A complete implementation is available in the **R** package `tclust` which is available through the `CRAN` repository.

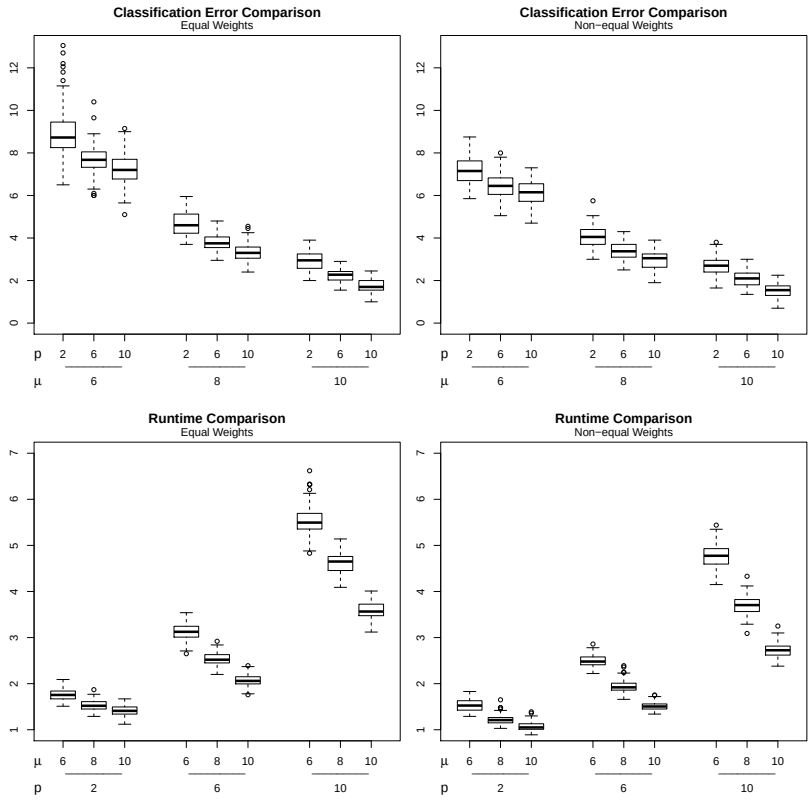

Figure 6.4: Classification errors and runtimes of the `tclust` algorithm applied to simulated M5 type data sets for different values of p and β and when values `nstart` $= 64$ and `iter.max` $= 64$ are fixed.

Appendix: Justification of the algorithm

E-step: Assuming the optimal set of parameters θ to be known, Equation (6.2.1) implies that the optimal cluster assignments of observations $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ can be obtained by using the discriminant functions $D_j(\boldsymbol{x}_i, \theta)$ as described in Step 2.1 of the algorithm.

M-step: Further, it is clear that depending on the cluster assignments (i.e. given $R_0, R_1, ..., R_k$), the values of p_j and m_j maximizing (6.2.1) are given by

$$p_j = n_j / \lfloor n(1-\alpha) \rfloor \text{ with } n_j = \#R_j \tag{6.5.1}$$

and

$$\boldsymbol{m}_j = \sum_{i \in R_j} \boldsymbol{x}_i / n_j. \tag{6.5.2}$$

Let us consider the singular-value decomposition of the sample covariance

matrices of observations $\boldsymbol{x}_i$ in group j, given by

$$\boldsymbol{U}_j'\boldsymbol{D}_j\boldsymbol{U}_j = \frac{1}{n_j}\sum_{i\in R_j}(\boldsymbol{x}_i - \boldsymbol{m}_j)(\boldsymbol{x}_i - \boldsymbol{m}_j)', \tag{6.5.3}$$

where $\boldsymbol{U}_j$ are orthogonal matrices and $\boldsymbol{D}_j = \text{diag}(d_{j1}, d_{j2}, ..., d_{jp})$ are diagonal matrices.
Let $\boldsymbol{S}_j$ be the optimally constrained scatter matrices maximizing (2.1) under restriction (6.2.2) when $R_0, R_1, ..., R_k$ are known and parameters $\boldsymbol{m}_j$ and p_j are given by (6.5.1) and (6.5.2). Analogously to the previous decomposition of the sample covariance matrices, matrices $\boldsymbol{S}_j$ can be split up into $\boldsymbol{S}_j = \boldsymbol{V}_j'\boldsymbol{D}_j^*\boldsymbol{V}_j$, with $\boldsymbol{V}_j$ orthogonal matrices and $\boldsymbol{D}_j^* = \text{diag}(d_{j1}^*, d_{j2}^*, ..., d_{jp}^*)$ diagonal matrices. It can be shown (see García-Escudero *et al.*, 2008) that the eigenvectors of the optimal constrained matrices $\boldsymbol{S}_j$ must be exactly the same as the eigenvectors of the unrestricted sample covariance matrices in (6.5.3) (i.e., we can set $\boldsymbol{U}_j = \boldsymbol{V}_j$). Thus, we just need to search for the optimal eigenvalues $\{d_{j,l}^*\}$ to obtain the optimal constrained scatter matrices $\boldsymbol{S}_j = \boldsymbol{U}_j'\boldsymbol{D}_j^*\boldsymbol{U}_j$.
Given the eigenvalues $\{d_{j,l}\}$ of the sample covariance matrices in (6.5.3), the optimal $\{d_{j,l}^*\}$ are obtained by minimizing expression (6.3.5) when $\{d_{j,l}^*\} \in \Lambda$ with Λ as defined in (6.3.7). The proof of this claim is almost identical to the proof of Proposition 4 in García-Escudero *et al.* (2008), with the only difference that expression (6.3.5) in the present work contains the cluster sizes n_j, whereas Equation (3.4) in the mentioned article wrongly did not.
Moreover, notice that Λ can be written as

$$\Lambda = \bigcup_{m\geq 0}\Lambda_m \text{ with } \Lambda_m = \bigcup_{m\geq 0}\left\{d_{jl}^* : m \leq d_{jl}^* \leq cm\right\}.$$

Thus, for globally minimizing expression (6.3.5) in Λ, we need to be able to minimize (6.3.5) when $\{d_{j,l}^*\} \in \Lambda_m$ for every possible value $m > 0$. However, the minimization (for a fixed value of m) can be simplified significantly by considering truncated eigenvalues $d_{jl}^* = d_{jl}^m$ like those in (6.3.3) which leads us to the minimization of the following target function:

$$\begin{aligned} f : m \;\mapsto\; \sum_{j=1}^{k} n_j \Bigg[& \sum_{l=1}^{p}(\log(m) + d_{jl}/m)(d_{jl} < m) \\ & + \sum_{l=1}^{p}(\log(d_{jl}) + 1)(m \leq d_{jl} < cm) \\ & + \sum_{l=1}^{p}(\log(cm) + d_{jl}/cm)(d_{jl} > cm)\Bigg], \end{aligned} \tag{6.5.4}$$

which coincides with the target function in (6.3.4).

Further, f is a continuous differentiable function minimizing in one of its critical values, which satisfy the following fixed point equation:

$$m^* = \frac{\sum_{j=1}^{k}(s_j(m^*) + t_j(m^*)/c)}{\sum_{j=1}^{k} n_j r_j(m^*)}$$

with

$$r_j(m) = \sum_{l=1}^{p}((d_{jl} < m) + (d_{jl} > cm)),$$

$$s_j(m) = \sum_{l=1}^{p} d_{jl}(d_{jl} < m) \text{ and } t_j(m) = \sum_{l=1}^{p} d_{jl}(d_{jl} > cm).$$

Functions r_j,s_j and t_j take constant values in the intervals $(-\infty, e_1], (e_1, e_2]$, ..., (e_{2k}, ∞). Therefore, we only need to evaluate (6.5.4) at the $2kp+1$ values $m_1, ..., m_{2kp+1}$ given in Remark 3.
If m_{opt} is the value of m minimizing function f, we finally set the optimal eigenvalues as $d_{jl}^* = d_{jl}^{m_{\text{opt}}}$ to obtain the optimally constrained scatter matrices $\boldsymbol{S}_j$.

Chapter 7

Implementational Details

7.1 Introduction

This chapter discusses an architecture for implementing mathematical modules for *environment applications* as **R** (**R** Development Core Team, 2010a) or **Matlab** (MATLAB, 2010) in **C++**. This concept is still under development, and originates from the implementation of the **R** packages tclust and pcaPP, which are the first modules implemented based on this architecture. The main objective is to provide a framework for implementing external modules, for mathematical purposes, which are on the one hand platform-independent, as any **R** package for instance, and moreover *environment independent*. This means, that such a module shall work together with any environment application as **R**, **Matlab**, or even user built standalone applications, only by modifying some interface layers of the module, but without changing a single line of code in its core implementation.
The motivation for environment independence is to make mathematical methods as available as possible, beyond the scope of a particular environment application and its developer community. This should help to avoid redundant and separate implementations of algorithms for different environments by different developers. Thus this concept enables developers to work together, which until now were implementing modules for their specific environment separately from each other. The reason for not simply using a numerical library as GSL (Galassi *et al.*, 2010) or Boost (Siek *et al.*, 2001) is, that the architecture shall provide a *meta interface* to any of these libraries, and thus a decision for using a specific library becomes totally independent from the module's core implementation, and can be revised and changed at any time. Thus when compiling a module for **R**, the built-in numerical routines of the environment are used by default, but it is also possible to link to other routines, as highly tuned implementations of BLAS (National Science Foundation and Department of Energy, 2010) or LAPACK (Anderson *et al.*, 1999), as available for various CPU types, but also for GPUs

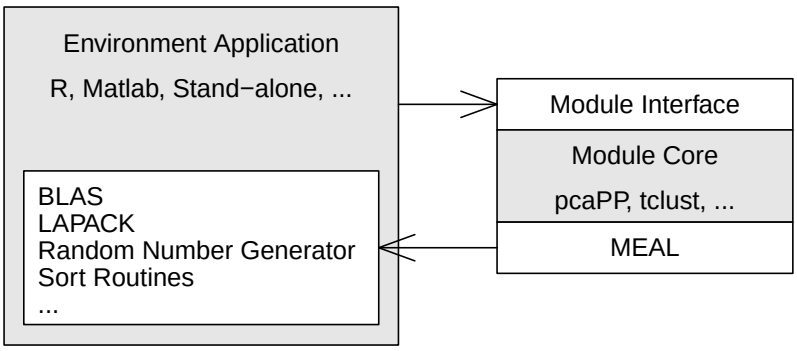

Figure 7.1: Schematic illustration of the connection between an environment application and an external module. The application calls routines in the module, which again refer to functions in the environment application itself.

and APUs, which use the power of modern computers more efficiently than conventional implementations do.
Environment independence is achieved by implementing the core functionality of a module in **C++**, and functions written in environment specific code (e.g. **R**-code) only form wrappers to the core functions.
As shown in Figure 7.1, the module core is implemented without directly calling routines in the environment application, as e.g. to **R**'s built-in random number generator. Such calls are made to an interface layer called *MEAL* (Mathematical Environment Abstraction Layer) and are then automatically routed to the corresponding routine in the used environment. This interface layer is implemented once for each considered type of environment application and can then be used for any module which is based on this architecture. A second interface layer, the *Module Interface* is an abstraction layer for calls from the environment application to the module itself, which for instance is responsible for data conversion. When compiling a module for different types of environment applications, only these interface layers have to be exchanged, without touching the implementation of the module's core functions.
Note, that only parts of the packages which implement the central algorithms can be provided in an environment-independent manner. Specialized methods as e.g. plotting functions have to be adapted and re-implemented completely for each considered environment, as for this purpose there is no independent standard availble yet.

7.2 Architecture

A complete overview of all layers of this architecture is given in Figure 7.2. The illustration refers to some possible environment applications as **R**, **Matlab** and user-built stand-alone applications in general. The model consists of 7 layers, whereas Layer 1 represents the calling environment application, and Layers 2-7 the actual module.
Layers 2 and 3 define an interface for calls from the environment to the module, whereas the MEAL interface (Layer 7) routes calls from the module back to different components in the calling environment (e.g. the BLAS or LAPACK routines which come along with R or Matlab).
Thus, when the module shall be compiled for a new environment, only the definition of these two interfaces (Layers 2, 3 and 7) has to be updated, but no changes to the code of the module core have to be considered.
Note, that each layer is only aware of the existence of the next lower layer, and thus does only call functions of its direct successor. The only exception is the bottom Layer 7, which does not have a direct successor, and only calls functions of the environment application, the uppermost Layer 1. Moreover, when compiling a module for a stand-alone application, not all layers may be implemented, as e.g. layers for data conversions may not be needed in such a setting.

7.2.1 Application Layer

The application layer represents any program which intends to use modules based on this architecture, as pcaPP or tclust. This could either be **R**, **Matlab**, any other statistics program and also a user-build application serving any purpose. The only requirement to the application is, that it is able to call functions in an external module (e.g. a dll in Windows).
Considering **R** as environment application, it is important to keep in mind, that this application provides some numerical routines as BLAS, LAPACK, some sort routines and random number generators, which might be of interest in a module. Thus this architecture provides a unified interface for accessing these routines (see Section 7.2.6).
Matlab only provides an interface to BLAS and LAPACK, but external modules are not able to call the internal sort routines or the random number generator.
Any other application environment may optionally specify any of these routines. However, if they are not provided in the application Layer 1, the bottom layer must either implement them or provide access to an appropriate library (see Section 7.2.7).

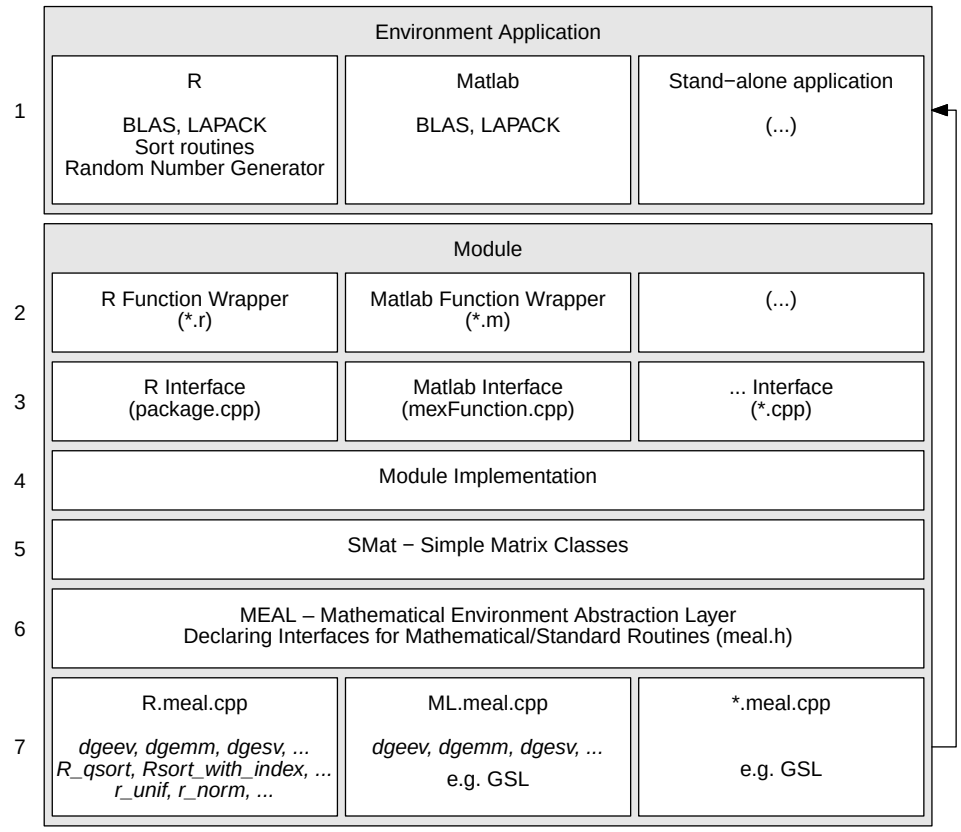

Figure 7.2: The seven layers, the packages tclust and pcaPP consist of. Layer 1 represents the environment application, Layers 2-7 an external module. The interfaces to the module are implemented by Layers 2, 3 and 7. Each layer exclusively accesses functions in its direct successor, whereas the bottom Layer 7 is the only layer which is able to directly call routines in Layer 1.

7.2.2 Module Interface (Environment)

Environments like **R** or **Matlab**, usually provide their particular scripting language, which is used for writing local functions, but also for accessing external modules. For accessing such modules, which are usually written in a different programming language (as **C**, **C++** or **Fortran**) a wrapping function must be implemented, which checks the user input for consistency (this might be easier in the environment's language), converts the data structures from the environment's representation to the representation needed in the external module, and then actually calls the modules functions.
R provides several functions for calling routines in external modules. However, in the context of the considered platform- and environment-independent architecture, the proper choice is the function `.C` (see **R** Development Core

Team, 2010b).
For accessing external modules from **Matlab**, only a simple `.m` file (also called mex - file) has to be specified, which points to the external module.
When considering a stand-alone application, this layer may serve for data conversion and would be implemented in the same programming language as the application itself. However, if no data conversion is necessary, this layer may be obsolete, and the routines in Layer 3 can be called directly.

7.2.3 Module Interface (C++)

This layer defines the entry points to the module's **C++** functions which are called from the environment. In combination with the **R**-function `.C`, this might be any exported function with return type `void` taking pointers to `int`, `double` or `char` as arguments (see **R** Development Core Team, 2010b).
Matlab uses a so called `mexFunction` as the only entry-point to external modules. This *gateway* function calls the corresponding functions in the module's implementation (Layer 4) and also passes along the provided arguments.
Depending on the design of the code of a stand-alone application, this layer might not be necessary either, as the module's functions in Layer 4 can be called directly. However, for any sort of necessary argument conversions the possibility of implementing this layer still exists.

7.2.4 Module Implementation

This layer defines and implements the module's actual core functionality. As the preceding layers are already designed according to the used environment, the module's functionality itself is implemented completely independently, without considering the particular application environment. Further, as each layer is only aware of its direct successor, the functions in this layer cannot directly refer to routines which are provided by a particular environment, but must refer to the subsequent layer (e.g. for calling LAPACK, BLAS, ...).
In the following Layer 7 takes care of routing the corresponding calls to the according routine in the environment layer.

7.2.5 SMat - Simple Matrix Classes

This level represents a set of proprietary matrix classes, which were implemented in the context of the tclust package, and then also introduced to the pcaPP package, resulting in an identical architecture of both packages.
In comparison to fully-grown matrix classes as the Blitz++ library (Veldhuizen, 1998), uBLAS as a part of the Boost library (Siek *et al.*, 2001) or the MTL library (Siek and Lumsdaine, 1999), which additionally offer numerical routines, the SMat classes reduce to the following functionality:

- Memory management for n-dimensional vectors (matrix representation is reduced to dense matrices in column major at this time).

- The implementation of a thin interface layer which provides access to other libraries as BLAS/LAPACK implementations, sort routines, random number generators, etc., either as part of an environment application as **R**, **Matlab**, etc., or as stand alone libraries (see 7.2.7). Thus, these classes are not intended to implement any numerical routines, by themselves, but provide a uniform interface to a broad variety of highly efficient numerical libraries.

- *Iterator functions*, which combine vectors, matrices and tensors by element-wise applying arbitrary user-defined functions, which cannot be easily represented by BLAS/LAPACK.

  These functions are designed as a link to future technologies as **C++** AMP (see Sutter, 2011), which is intended to become a cross-platform **C++** language extension for multithreaded heterogeneous computing at any level of complexity, beyond the functionality offered by BLAS and LAPACK routines.

  Thus programmers with average skills will also be able to execute any matrix or vector operation simultaneously on different types of processing units, as CPUs, GPUs and APUs, using the resources of modern computers more efficiently, which until now has been considered as a task of *gurus* - very few, highly specialized programmers which were able to utilize a computer's resources to its full extent.

Note, that the performance of the current setting, which is based on the proprietary SMat classes is still being evaluated. In the future this layer may be implemented differently, depending on how other promising matrix libraries as Blitz++, uBLAS or MTL perform in the context of concepts as **C++** AMP.

7.2.6 MEAL - Declaration

This layer consists of a set of definitions, specifying an interface for enabling the preceding layer to access various routines in the environment application. Currently the most important BLAS and LAPACK routines, sort routines and random number generators are accessible via this interface. Note, that the functionality of this layer has not been defined completely yet, and thus it is still expanding. This, for instance, refers to the support of the full BLAS and LAPACK functionality, which is being implemented at the moment.

7.2.7 MEAL - Implementation

Here the previously declared functions are implemented. Thus, a call to e.g. a BLAS routine has to be routed to the corresponding function in the environment application.
If the environment application does not provide the full functionality as required in Layer 6, this layer has to take care of implementing the corresponding functions.
This is the case with **Matlab**, as in contrast to **R**, neither its random number generator, nor its internal sort routines are accessible from external modules. Thus, an implementation of random number generators and sort routines can be taken from the Gnu Scientific Library (Galassi *et al.*, 2010) or the Boost library (Siek *et al.*, 2001).
Further, a stand-alone environment application may not contain any of the mentioned routines at all, and thus this layer is responsible for including and linking an according BLAS and LAPACK library.

7.3 Conclusions

An overview of an architecture for mathematical modules has been given, which has been developed jointly with the **R** packages tclust and pcaPP. Admittedly, this concept was developed in the scope of these packages, and thus it is not implemented to its complete extent yet. However the **R**-package pcaPP (available on CRAN, `http://CRAN.R-project.org/package=pcaPP`) proves this architecture's functionality, as the included vignette "Compiling pcaPP for Matlab" explains how to compile, and shows how to use the package in **Matlab** (version $\geq$ 2010a).
A complete MEAL implementation for **R** is currently being developed, and also a full implementation for **Matlab** will be available shortly. Further, an implementation for stand-alone applications based on GSL or the Boost library is planned and will be published on CRAN (`http://cran.r-project.org/`) via the packages tclust and pcaPP. Thus, the application of these modules will not be restricted to a single environment, but will be available to a wide range of users and developers, which hopefully also consider to implement future software projects in environment-independent manner.

Bibliography

Anaya-Izquierdo K, Critchley F, Vines K (2011). "Orthogonal simple component analysis: a new, exploratory approach." *Annals of Applied Statistics*, **5**(1), 486–522.

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J, Croz JD, Greenbaum A, Hammarling S, McKenney A, Sorensen D (1999). *LAPACK Users' Guide*. Third edition. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Banfield JD, Raftery AE (1993). "Model-based Gaussian and non-Gaussian Clustering." *Biometrics*, **49**(3), 803–821.

Bedall F, Zimmermann H (1979). "As 143: The mediancentre." *Applied Statistics*, **28**, 325–328.

Bryant P (1991). "Large-sample Results for Optimization-based Clustering Methods." *Journal of Classification*, **8**(1), 31–44.

Campbell N (1980). "Robust Procedures in Multivariate Analysis I: Robust Covariance Estimation." *Applied Statistics*, **29**, 231–237.

Cattell R (1966). "The scree test for the number of factors." *Multivariate Behaviour Research*, **1**, 245–276.

Celeux G, Govaert G (1992). "A Classification EM algorithm for clustering and two stochastic versions." *Computational Statistics & Data Analysis*, **14**(3), 315–332.

Chaudhuri P (1996). "On a geometric notion of quantiles for multivariate data." *J Amer Statist Assoc*, **91**, 862–872.

Cook RD (1999). "Graphical detection of regression outliers and mixtures." In *Proceedings of the International Statistical Institute 1999*. ISI, Finland.

Croux C, Filzmoser P, Oliveira M (2007). "Algorithms for Projection-Pursuit Robust Principal Component Analysis." *Chemometrics and Intelligent Laboratory Systems*, **87**, 218–225.

Croux C, Gelper S, Haesbroeck G (2010). "Robust scatter regularization." In G Saporta, Y Lechevallier (eds.), *Compstat 2010, Book of Abstracts*, p. 138. Conservatoire National des Arts et Métiers (CNAM) and the French National Institute for Research in Computer Science and Control (INRIA), Paris.

Croux C, Haesbroeck G (2000). "Principal Components Analysis Based on Robust Estimators of the Covariance or Correlation Matrix: Influence Functions and Efficiencies." *Biometrika*, **87**, 603–618.

Croux C, Ruiz-Gazen A (1996). "A fast algorithm for robust principal components based on projection pursuit." In A Prat (ed.), *Compstat: Proceedings in Computational Statistics*, pp. 211–216. Physica-Verlag, Heidelberg.

Croux C, Ruiz-Gazen A (2005). "High Breakdown Estimators for Principal Components: The Projection-pursuit Approach Revisited." *Journal of Multivariate Analysis*, **95**, 206–226.

Cuesta-Albertos J, Gordaliza A, Matrán C (1997). "Trimmed k-means: an Attempt to Robustify Quantizers." *Annals of Statistics*, **25**(2), 553–576.

Davies L (1987). "Asymptotic Behaviour of S-Estimates of Multivariate Location Parameters and Dispersion Matrices." *The Annals of Statistics*, **15**, 1269–1292.

Debruyne M, Hubert M, Van Horebeek J (2010). "Detecting influential observations in Kernel PCA." *Computational Statistics & Data Analysis*, **54**(12), 3007–3019.

Dennis J, Schnabel R (1983). "Numerical Methods for Unconstrained Optimization and Nonlinear Equations." *Prentice Hall, New Jersey*.

Development Core Team R (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, `http://www.R-project.org`.

Donoho DL (1982). *Breakdown properties of multivariate location estimators.* Ph.D. thesis, Harvard University, Boston. Ph.D. qualifying paper.

Donoho DL, Huber P (1983). "The notion of breakdown-point." In PJ Bickel, KA Doksum, e J L Hodges Jr (eds.), *In A Festschrift for Erich L. Lehmann*, pp. 157–184. Wadsworth, Belmont, CA.

Dykstra R (1983). "An Algorithm for Restricted Least Squares Regression." *Journal of the American Statistical Association*, **78**(384), 837–842.

Farcomeni A (2009). "An Exact Approach to Sparse Principal Component Analysis." *Computational Statistics*, **24**(4), 583–604.

Filzmoser P (1999). "Robust principal components and factor analysis in the geostatistical treatment of environmental data." *Environmetrics*, **10**, 363–375.

Filzmoser P, Fritz H, Kalcher K (2010). *pcaPP: Robust PCA by Projection Pursuit*. R package version 1.8-3.

Fletcher R, Reeves C (1964). "Function minimization by conjugate gradients." *Computer Journal*, **7**, 148–154.

Flury B, Riedwyl H (1988). *Multivariate Statistics. A Practical Approach*. Chapman and Hall, London.

Forgy E (1965). "Cluster analysis of multivariate data: efficiency versus interpretability of classifications." *Biometrics*, **21**, 768–780.

Fraley C, Raftery AE (1998). "How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis." *The Computer Journal*, **41**(8), 578–588.

Friedman H, Rubin J (1967). "On Some Invariant Criterion for Grouping Data." *Journal of the American Statistical Association*, **63**(320), 1159–1178.

Fritz H, García-Escudero L, Mayo-Iscar A (2011). "A fast algorithm for robust constrained clustering." Unpublished manuscript, `http://www.eio.uva.es/infor/personas/tclust_algorithm.pdf`.

Gabriel KR (1971). "The biplot graphic display of matrices with application to principal component analysis." *Biometrika*, **58**(3), 453–467.

Galassi M, Davies J, Theiler J, Gough B, Jungman G, Alken P, Booth M, Rossi F (2010). *GNU Scientific Library Reference Manual*. 1.14 edition. Network Theory Ltd.

Gallegos MT (2002). "Maximum likelihood clustering with outliers." In K Jajuga, A Sokolowski, H Bock (eds.), *Classification, Clustering and Data Analysis: Recent advances and applications.*, pp. 247–255. Springer-Verlag.

Gallegos MT, Ritter G (2005). "A Robust Method for Cluster Analysis." *Annals of Statistics*, **33**(1), 347–380.

Gallegos MT, Ritter G (2009). "Trimming Algorithms for Clustering Contaminated Grouped Data and their Robustness." *Advances in Data Analysis and Classification*, **3**(2), 135–167.

Gallegos MT, Ritter G (2010). "Using Combinatorial Optimization in Model-based Trimmed Clustering with Cardinality Constraints." *Computational Statistics & Data Analysis*, **54**(3), 637–654.

García-Escudero LA, Gordaliza A (1999). "Robustness properties of k-means and trimmed k-means." *Journal of the American Statistical Association*, **94**(447), 956–969.

García-Escudero LA, Gordaliza A, Matrán C (2003). "Trimming Tools in Exploratory Data Analysis." *Journal of Computational and Graphical Statistics*, **12**(2), 434–449.

García-Escudero LA, Gordaliza A, Matrán C, Mayo-Iscar A (2008). "A General Trimming Approach to Robust Cluster Analysis." *Annals of Statistics*, **36**(3), 1324–1345.

García-Escudero LA, Gordaliza A, Matrán C, Mayo-Iscar A (2010). "A Review of Robust Clustering Methods." *Advances in Data Analysis and Classification*, **4**(2-3), 89–109.

García-Escudero LA, Gordaliza A, Matrán C, Mayo-Iscar A (2011). "Exploring the number of groups in robust model-based clustering." *Statistics and Computing*, **Forthcoming**. Preprint available at `http://www.eio.uva.es/infor/personas/langel.html`.

Gervini D (2008). "Robust functional estimation using the median and spherical principal components." *Biometrika*, **95**, 587–600.

Gnanadesikan R, Kettenring JR (1972). "Robust estimates, residuals, and outlier detection with multiresponse data." *Biometrics*, **28**, 81–124.

Gower J (1974). "As 78: The mediancentre." *Journal of the Royal Statistical Society*, **23**, 466–470.

Gower J, Hand DJ (1996). *Biplots.* Chapman & Hall, London.

Guo J, James G, Levina E, Michailidis G, J Z (2010). "Principal Component Analysis with Sparse Fused Loadings." *Journal of Computational and Graphical Statistics*. To appear.

Hardin J, Rocke DM (2004). "Outlier Detection in the Multiple Cluster Setting Using the Minimum Covariance Determinant Estimator." *Computational Statistics & Data Analysis*, **44**(4), 625 – 638.

Hathaway RJ (1985). "A Constrained Formulation of Maximum Likelihood Estimation for Normal Mixture Distributions." *Annals of Statistics*, **13**(2), 795–800.

Hennig C (2004). "Breakdown points for maximum likelihood-estimators of location-scale mixtures." *Annals of Statistics*, **32**(4), 1313–1340.

Hennig C (2010). ***fpc**: Flexible procedures for clustering. Version 2.0-2.* `http://CRAN.R-project.org/package=fpc`.

Hössjer O, Croux C (1995). "Generalizing univariate signed rank statistics for testing and estimating a multivariate location parameter." *Nonparametric Statistics*, **4**, 293–308.

Hubert M, Rousseeuw PJ, Van Aelst S (2008). "High-breakdown robust multivariate methods." *Statistical Science*, **23**(1), 92–119.

Hubert M, Rousseeuw PJ, Vanden Branden K (2005). "ROBPCA: A New Approach to Robust Principal Component Analysis." *Technometrics*, **47**, 64–79.

Hubert M, Rousseeuw PJ, Verboven S (2002). "A fast method for principal components with application to chemometrics." *Chemometrics and Intelligent Laboratory Systems*, **60**, 101–111.

Jolliffe IT (1986). *Principal Component Analysis.* Springer Verlag, New York.

Jolliffe IT (1995). "Rotation of principal components: choice of normalization constraints." *Journal of Applied Statistics*, **22**, 29–35.

Jolliffe IT (2002). *Principal Component Analysis.* Second edition. Springer-Verlag, New York.

Jolliffe IT, Trendafilov NT, Uddin M (2003). "A Modified Principal Component Technique Based on the LASSO." *Journal of Computational and Graphical Statistics*, **12**, 531–547.

Journée M, Nesterov Y, Richtárik P, Sepulchre R (2010). "Generalized Power Method for Sparse Principal Component Analysis." *Journal of Machine Learning Research*, **11**, 517–553.

Kaufman L, Rousseeuw PJ (1990). *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley and Sons, New York.

Kibler D, Aha D, Albert M (1989). "Instance-based prediction of real-valued attributes." *Computational Intelligence*, **5**, 51–57.

Leisch F (2004). "FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R." *Journal of Statistical Software*, **11**(8), 1–18.

Leng C, Wang H (2009). "On General Adaptive Sparse Principal Component Analysis." *Journal of Computational and Graphical Statistics*, **18**(1), 201–215.

Li G, Chen Z (1985). "Projection-Pursuit Approach to Robust Dispersion Matrices and Principal Components: Primary Theory and Monte Carlo." *Journal of the American Statistical Association*, **80**(391), 759–766.

Lopuhaä H, Rousseeuw PJ (1991). "Breakdown points of affine equivariant estimators of multivariate location and covariance matrices." *The Annals of Statistics*, **19**(1), 229–248.

Lykou A, Whittaker J (2010). "Sparse CCA using a Lasso with positivity constraints." *Computational Statistics & Data Analysis*, **54**(12), 3144–3157.

MacQueen JB (1967). "Some Methods for Classification and Analysis of MultiVariate Observations." In LM Le Cam, J Neyman (eds.), *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297. University of California Press.

Maronna R (1976). "Robust M-Estimators of Multivariate Location and Scatter." *The Annals of Statistics*, **4**, 51–67.

Maronna R (2005). "Principal components and orthogonal regression based on robust scales." *Technometrics*, **47**(3), 264–273.

Maronna R (2011). "Robust ridge regression for high-dimensional data." *Technometrics*, **53**(1), 44–53.

Maronna R, Jacovkis PM (1974). "Multivariate Clustering Procedures with Variable Metrics." *Biometrics*, **30**(3), 499–505.

Maronna R, Martin R, Yohai V (2006). *Robust Statistics: Theory and Methods*. John Wiley and Sons.

Maronna R, Yohai V (2008). "Robust low-rank approximation of data matrices with elementwise contamination." *Technometrics*, **50**(3), 295–304.

Maronna R, Zamar R (2002). "Robust Estimation of Location and Dispersion for High-dimensional Data Sets." *Technometrics*, **44**(4), 307–317.

Maronna RA, Yohai VJ (1995). "The Behavior of the Stahel-Donoho Robust Multivariate Estimator." *Journal of the American Statistical Association*, **90**(429), 330–341.

MATLAB (2010). *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts.

McLachlan G, Peel D (2000). *Finite Mixture Models.* John Wiley & Sons, New York.

National Science Foundation and Department of Energy (2010). *BLAS (Basic Linear Algebra Subprograms).* Netlib Repository at UTK and ORNL. `http://www.netlib.org/blas/`.

Nelder J, Mead R (1965). "A simplex algorithm for function minimization." *Computer Journal*, **7**, 308–313.

Neykov N, Filzmoser P, Dimova R, Neytchev P (2007). "Robust Fitting of Mixtures Using the Trimmed Likelihood Estimator." *Computational Statistics & Data Analysis*, **52**(1), 299–308.

Nocedal J, Wright S (2006). *Numerical Optimization, 2nd edn.* Springer-Verlag.

R Development Core Team (2010a). ***R**: A Language and Environment for Statistical Computing.* **R** Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, `http://www.R-project.org/`.

R Development Core Team (2010b). *Writing R extensions.* **R** Foundation for Statistical Computing, Vienna, Austria. `http://www.R-project.org/`.

Reimann C, Filzmoser P, R G, Dutter R (2008). *Statistical data analysis explained.* John Wiley, Chichester.

Rocke DM, Woodruff DL (2002). "Computational Connections Between Robust Multivariate Analysis and Clustering." In W Härdle, R B (eds.), *COMPSTAT 2002 Proceedings in Computational Statistics*, pp. 255–260.

Rousseeuw P, Croux C (1993). "Alternatives to the Median Absolute Deviation." *Journal of the American Statistical Association*, **88**(424), 1273–1283.

Rousseeuw PJ (1984). "Least Median of Squares Regression." *Journal of the American Statistical Association*, **79**(388), 871–880.

Rousseeuw PJ (1985). "Multivariate Estimation with High Breakdown Point." In W Grossmann, G Pflug, I Vincze, W Wertz (eds.), *Mathematical Statistics and Applications*, volume B, pp. 283–297. Reidel, Dordrecht.

Rousseeuw PJ (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis." *Journal of Computational and Applied Mathematics*, **20**(1), 53–65.

Rousseeuw PJ, Leroy AM (1987). *Robust Regression and Outlier Detection.* John Wiley & Sons, Inc., New York. ISBN 0-471-85233-3.

Rousseeuw PJ, Van Driessen K (1999). "A Fast Algorithm for the Minimum Covariance Determinant Estimator." *Technometrics*, **41**, 212–223.

Schnabel R, Koontz J, Weiss B (1985). "A modular system of algorithms for unconstrained minimization." *ACM Trans Math Software*, **11**, 419–440.

Schroeder A (1976). "Analyse d'un mélange de distributions de probabilités de même type." *Rev. Statist. Appl.*, **24**, 39–62.

Scott AJ, Symons MJ (1971). "Clustering Methods Based on Likelihood Ratio Criteria." *Biometrics*, **27**(2), 387–397.

Serneels S, Croux C, Filzmoser P, Van Espen P (2005). "Partial robust M-regression." *Chemometrics and Intelligent Laboratory Systems*, **79**, 55–64.

Siek JG, Lee LQ, Lumsdaine A (2001). *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley. ISBN 978-0201729146.

Siek JG, Lumsdaine A (1999). "The Matrix Template Library: Generic Components for High-Performance Scientific Computing." *Computing in Science and Engineering*, **1**(6).

Stahel WA (1981). "Breakdown of Covariance Estimators." *Technical report*, Eidgenössische technische Hochschule, Zürich. Research Report No. 31.

Stanimirova I, Walczak B, Massart D, Simenov V (2004). "A Comparison between two Robust PCA Algorithms." *Chemometrics and Intelligent Laboratory Systems*, **71**, 83–95.

Steinhaus H (1956). "Sur la division des corp materiels en parties." *Bull. Acad. Polon. Sci.*, **1**, 801–804.

Sutter H (2011). "Heterogeneous Computing and C++ AMP." In *AMD Fusion Developer Summit*.

Swierenga H, de Weijer AP, van Wijk RJ, Buydens LMC (1999). "Strategy for constructing robust multivariate calibration models." *Chemometrics and Intelligent Laboratoryy Systems*, **49**, 1–17.

Symons M (1981). "Clustering criteria and multivariate normal mixtures." *Biometrics*, **37**, 35–43.

Tibshirani R (1996). "Regression shrinkage and selection via the Lasso." *Journal of the Royal Statistical Society, Series B*, **58**, 267–288.

Trendafilov NT, Jolliffe IT (2006). "Projected Gradient Approach to the Numerical Solution of the SCoTLASS." *Computational Statistics & Data Analysis*, **50**(1), 242–253.

Van Aelst S, Wang X, Zamar RH, Zhu R (2006). "Linear Grouping Using Orthogonal Regression." *Computational Statistics & Data Analysis*, **50**(5), 1287–1312.

Vardi Y, Zhang C (2000). "The multivariate l_1-median and associated data depth." *Proc National Academy of Science*, **97**(4), 1423–1426.

Veldhuizen TL (1998). "Arrays in Blitz++." In D Caromel, RR Oldehoeft, M Tholburn (eds.), *ISCOPE*, volume 1505 of *Lecture Notes in Computer Science*, pp. 223–230. Springer. ISBN 3-540-65387-2.

Vines S (2000). "Simple principal components." *Applied Statistics*, **49**, 441–451.

Ward JH (1963). "Hierarchical Grouping to Optimize an Objective Function." *Journal of the American Statistical Association*, **58**(301), 236–244.

Weber A (1909). "Über den Standort der Industrien." *Mohr and Tübingen.*

Weiszfeld E (1937). "Sur le point pour lequel la somme des distances de n points donnés est minimum." *Tôhoku Mathematical Journal*, **43**, 355–386.

Witten D, Tibshirani R (2011). "Penalized classification using Fisher's linear discriminant." *Journal of the Royal Statistical Society, Series B.* In press.

Witten D, Tibshirani R, Hastie T (2009). "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis." *Biostatistics*, **10**(3), 515–534.

Woodruff D, Reiners T (2004). "Experiments with, and on, Algorithms for Maximum Likelihood Clustering." *Computational Statistics & Data Analysis*, **47**(2), 237–253.

Zou H, Hastie T (2005). "Regularization and variable selection via the elastic net." *Journal of the Royal Statistical Society, Series B*, **67**, 301–320.

Zou H, Hastie T, Tibshirani R (2006). "Sparse Principal Component Analysis." *Journal of Computational and Graphical Statistics*, **15**(2), 265–286.

Curriculum Vitae

Personal Details

Name:	Heinrich Fritz
Date of birth:	27.04.1983
Place of birth:	Vienna, Austria
Citizenship:	Austria
Marital Status:	Single

Education

2008	Participated in the ERASMUS program at the Universidad Politécnica de Valencia, Spain
2007-present	PhD in technical sciences, Mathematics at the Vienna University of Technology
2006-2007	Masters in Computer Science Management (graduated with distinction) at the Vienna University of Technology
2006-2007	Masters in Business Engineering and Computer Science (graduated with distinction) at the Vienna University of Technology
2003-2006	Bachelor in Data Engineering and Statistics (graduated with distinction) at the Vienna University of Technology
2002-2003	Military service
2002	A-levels (with distinction) at Höhere technische Bundeslehranstalt, Schulzentrum Ungargasse, Vienna, Austria

Conferences

2010	3rd International Conference on Computing & Statistics (ERCIM), University of London, UK
2007	International Workshop on Robust Statistics and R, BIRS Banff, Canada
2006	2nd International R User Conference (useR), Vienna University of Economics and Business, Austria