Use R!

Ron Wehrens

Chemometrics with R

Multivariate Data Analysis in the Natural Sciences and Life Sciences

Springer

Use R!

Series Editors:
Robert Gentleman Kurt Hornik Giovanni Parmigiani

For other titles published in this series, go to
http://www.springer.com/series/6991

Ron Wehrens

Chemometrics with R

Multivariate Data Analysis in the Natural Sciences and Life Sciences

Springer

Ron Wehrens
Fondazione Edmund Mach
Research and Innovation Centre
Via E. Mach 1
38010 San Michele all'Adige
Italy
ron.wehrens@iasma.it

ISBN 978-3-642-17840-5 e-ISBN 978-3-642-17841-2
DOI 10.1007/978-3-642-17841-2
Springer Heidelberg Dordrecht London New York

Cover design: deblik, Berlin

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

For Odilia, Chris and Luc

Preface

The natural sciences, and the life sciences in particular, have seen a huge increase in the amount and complexity of data being generated with every experiment. It is only some decades ago that scientists were typically measuring single numbers – weights, extinctions, absorbances – usually directly related to compound concentrations. Data analysis came down to estimating univariate regression lines, uncertainties and reproducibilities. Later, more sophisticated equipment generated complete spectra, where the response of the system is wavelength-dependent. Scientists were confronted with the question how to turn these spectra into useable results such as concentrations. Things became more complex after that: chromatographic techniques for separating mixtures were coupled to high-resolution (mass) spectrometers, yielding a data matrix for every sample, often with large numbers of variables in both chromatographic and spectroscopic directions. A set of such samples corresponds to a data cube rather than a matrix. In parallel, rapid developments in biology saw a massive increase in the ratio of variables to objects in that area as well.

As a result, scientists today are faced with the increasingly difficult task to make sense of it all. Although most will have had a basic course in statistics, such a course is unlikely to have covered much multivariate material. In addition, many of the classical concepts have a rough time when applied to the types of data encountered nowadays – the multiple-testing problem is a vivid illustration. Nevertheless, even though data analysis has become a field in itself (or rather: a large number of specialized fields), scientists generating experimental data should know at least some of the ways to interpret their data, if only to be able to ascertain the quality of what they have generated. Cookbook approaches, involving blindly pushing a sequence of buttons in a software package, should be avoided. Sometimes the things that deviate from expected behaviour are the most interesting in a data set, rather than unfortunate measurement errors. These deviations can show up at any time point during data analysis, during data preprocessing, modelling, interpretation... Every phase in this pipeline should be carefully executed and results, also

at an intermediate stage, should be checked using common sense and prior knowledge.

This also puts restrictions on the software that is being used. It should be sufficiently powerful and flexible to fit complicated models and handle large and complex data sets, and on the other hand should allow the user to exactly follow what is being calculated – black-box software should be avoided if possible. Moreover, the software should allow for reproducible results, something that is hard to achieve with many point-and-click programs: even with a reasonably detailed prescription, different users can often obtain quite different results. R [1], with its rapidly expanding user community, nicely fits the bill. It is quickly becoming the most important tool in statistical bioinformatics and related fields. The base system already provides a large array of useful procedures; in particular, the high-quality graphics system should be mentioned. The most important feature, however, is the package system, allowing users to contribute software for their own fields, containing manual pages and examples that are directly executable. The result is that many packages have been contributed by users for specific applications; the examples and the manual pages make it easy to see what is happening.

Purpose of this book.

Something of this philosophy also can be found in the way this book is set up. The aim is to present a broad field of science in an accessible way, mainly using illustrative examples that can be reproduced immediately by the reader. It is written with several goals in mind:

- **An introduction to multivariate analysis.** On an abstract level, this book presents the route from raw data to information. All steps, starting from the data preprocessing and exploratory analysis to the (statistical) validation of the results, are considered. For students or scientists with little experience in handling real data, this provides a general overview that is sometimes hard to get from classical textbooks. The theory is presented as far as necessary to understand the principles of the methods and the focus is on immediate application on data sets, either from real scientific examples, or specifically suited to illustrate characteristics of the analyses.
- **An introduction to R.** For those scientists already working in the fields of bioinformatics, biostatistics and chemometrics but using other software, the book provides an accessible overview on how to perform the most common analyses in R [1]. Many packages are available on the standard repositories, CRAN[1] and BIOCONDUCTOR[2], but for people unfamiliar with the basics of R the learning curve can be pretty steep – for software, power and complexity are usually correlated. This book is an attempt to provide a more gentle way up.

[1] `http://cran.r-project.org`
[2] `http://www.bioconductor.org`

- **Combinding multivariate data analysis and R.** The combination of the previous two goals is especially geared towards university students, at the beginning of their specialization: it is of prime importance to obtain hands-on experience on real data sets. It does take some help to start reading R code – once a certain level has been reached, it becomes more easy. The focus therefore is not just on the use of the many packages that are available, but also on showing how the methods are implemented. In many cases, simplified versions of the algorithms are given explicitly in the text, so that the reader is able to follow step-by-step what is happening. It is this insight in (at least the basics of) the techniques that is essential for fruitful application.

The book has been explicitly set up for self-study. The user is encouraged to try out the examples, and to substitute his or her own data as well. If used in a university course, it is possible to keep the classical "teaching" of theory to a minimum; during the lessons, teachers can concentrate on the analysis of real data. There is no substitute for practice.

Prior knowledge.

Some material is assumed to be familiar. Basic statistics, for example, including hypothesis tests, the construction of confidence intervals, analysis of variance and least-squares regression are referred to, but not explained. The same goes for basic matrix algebra. The reader should have some experience in programming in general (variables, variable types, functions, program control, etcetera). It is assumed the reader has installed R, and has a basic working knowledge of R, roughly corresponding to having worked through the excellent "Introduction to R" [2], which can be found on the CRAN website. In some cases, less mundane functions will receive a bit more attention in the text; examples are the `apply` and `sweep` functions. We will only focus on the comman-line interface: Windows users may find it easier to perform actions using point-and-click.

The R package **ChemometricsWithR**.

With the book comes a package, too: **ChemometricsWithR** contains all data sets and functions used in this book. Installing the package will cause all other packages used in the book to be available as well – an overview of these packages can be found in Appendix A. In the examples it is always assumed that the **ChemometricsWithR** package is loaded; where functions or data sets from other packages are used for the first time, this is explicitly mentioned in the text.

More information about the data sets used in the book can be found in the references – no details will be given about the background or interpretation of the measurement techniques.

Acknowledgements.

This book has its origins in a reader for the Chemometrics course at the Radboud University Nijmegen covering exploratory analysis (PCA), clustering (hierarchical methods and k-means), discriminant analysis (LDA, QDA) and multivariate regression (PCR, PLS). Also material from a later course in Pattern Recognition has been included. I am grateful for all the feedback from the students, and especially for the remarks, suggestions and criticisms from my colleagues at the Department of Analytical Chemistry of the Radboud University Nijmegen. I am indebted to Patrick Krooshof and Tom Bloemberg, who have contributed in a major way in developing the material for the courses. Finally, I would like to thank all who have read (parts of) the manuscript and with their suggestions have helped improving it, in particular Tom Bloemberg, Karl Molt, Lionel Blanchet, Pietro Franceschi, and Jan Gerretzen.

Trento, *Ron Wehrens*
September 2010

Contents

1

Introduction

In the life sciences, molecular biology in particular, the amount of data has exploded in the last decade. Sequencing a whole genome is becoming routine work, and shortly the amount of money needed to do so will be less than the cost of a medium-sized television set. Rather than focussing on measuring specific predefined characteristics of the sample[1] modern techniques aim at generating a holistic view, sometimes called a "fingerprint". As a result, one analysis of one single sample can easily yield megabytes of data. These physical samples typically are complex mixtures and may, e.g., correspond to body fluids of patients and controls, measured with possibly several different spectroscopic techniques; environmental samples (air, water, soil); measurements on different cell cultures or one cell culture under different treatments; industrial samples from process industry, pharmaceutical industry or food industry; samples of competitor products; quality control samples, and many others. The types of data we will concentrate on are generated by analytical chemical measurement techniques, and are in almost all cases directly related to concentrations or amounts of specific classes of chemicals such as metabolites or proteins. The corresponding research fields are called metabolomics and proteomics, and a host of other -omics sciences with similar characteristics exist. A well-known example from molecular biology is transcriptomics, focussing on the levels of mRNA obtained by transcription from DNA strains. Although we do not include any transcriptomics data, many of the techniques treated in this book are directly applicable – in that sense, the characteristics of data of completely different origins can still be comparable.

These data can be analysed at different levels. The most direct approach is to analyse them as raw data (intensities, spectra, ...), without any prior interpretation other than a suitable pretreatment. Although this has the advantage

[1] The word "sample" will be used both for the physical objects on which measurements are performed (the chemical use of the word) and for the current realization of all possible measurements (the statistical use). Which one is meant should be clear from the context.

R. Wehrens, *Chemometrics with R: Multivariate Data Analysis in the Natural Sciences and Life Sciences*, Use R!, DOI 10.1007/978-3-642-17841-2_1,

that it is completely objective, it is usually also more difficult: typically, the number of variables is huge and the interpretability of the statistical models that are generated to describe the data often is low. A more often used strategy is to apply domain knowledge to convert the raw data into more abstract variables such as concentrations, for example by quantifying a set of compounds in a mixture based on a library of pure spectra. The advantage is that the statistical analysis can be performed on the quantities that really matter, and that the models are simpler and easier to validate and interpret. The obvious disadvantage is the dependence on the interpretation step: not always it is easy to decide which compounds are present and in what amounts. Any error at this stage cannot be corrected in later analysis stages.

The extremely rapid development of analytical techniques in biology and chemistry has left data analysis far behind, and as a result the statistical analysis and interpretation of the data has become a major bottleneck in the pipeline from measurement to information. Academic training in multivariate statistics in the life sciences is lagging. Bioinformatics departments are the primary source of scientists with such a background, but bioinformatics is a very broad field covering many other topics as well. Statistics and machine learning departments are usually too far away from the life sciences to establish joint educational programmes. As a result, scientists doing the data analysis very often have a background in biology or chemistry, and have acquired their statistical skills by training-on-the-job. This can be an advantage, since it makes it easier to interpret results and assess the relevance of certain findings. At the same time, there is a need for easily accessible background material and opportunities for self-study: books like the excellent "The Elements of Statistical Learning" [3] form an invaluable source of information but can also be a somewhat daunting read for scientists without much statistical background.

This book aims to fill the gap, at least to some extent. It is important to combine the sometimes rather abstract descriptions of the statistical techniques with hands-on experience behind a computer screen. In many ways R [1] is the ideal software platform to achieve this – it is extremely powerful, the many add-on packages provide a huge range of functionalities in different areas, and it is freely accessible. As in the other books in this series, the examples can be followed step-by-step by typing or cutting-and-pasting the code, and it is easy to plug in one's own data. To date, there is only one other book specifically focused on the use of R in a similar field of science: "Introduction to Multivariate Statistical Analysis in Chemometrics" [4] which to some extent complements the current volume, in particular in its treatment of robust statistics.

Here, the concepts behind the most important data analysis techniques will be explained using a minimum of mathematics, but in such a way that the book still can be used as a student's text. Its structure more or less follows the steps made in a "classical" data analysis, starting with the *data pretreatment* in Part I. This step is hugely important, yet is often treated only cursorily. An unfortunate choice here can destroy any hope of achieving good results:

background knowledge of the system under study as well as the nature of the measurements should be used in making decisions. This is where science meets art: there are no clear-cut rules, and only by experience we will learn what the best solution is.

The next phase, subject of Part II, consists of *exploratory analysis*. What structure is visible? Are there any outliers? Which samples are very similar, which are different? Which variables are correlated? Questions like these are most easily assessed by eye – the human capacity for pattern recognition in two dimensions is far superior to any statistical method. The methods at this stage all feature strong visualization capabilities. Usually, they are model-free; no model is fitted, and the assumptions about the data are kept to a minimum.

Once we are at the *modelling* phase, described in Part III, we very often do make assumptions: some models work optimally with normally distributed data, for example. The purpose of modelling can be twofold. The first is prediction. Given a set of analytical data, we want to be able to predict properties of the samples that cannot be measured easily. An example is the assessment of whether a specific treatment will be useful for a patient with particular characteristics. Such an application is known as *classification* – one is interested in modelling class membership (will or will not respond). The other major field is *regression*, where the aim is to model continuous real variables (blood pressure, protein content, ...). Such predictive models can mean a big improvement in quality of life, and save large amounts of money. The prediction error is usually taken as a quality measure: a model that is able to predict with high accuracy must have captured some real information about the system under study. Unfortunately, in most cases no analytical expressions can be derived for prediction accuracy, and other ways of estimating prediction accuracy are required in a process called *validation*. A popular example is crossvalidation.

The second aim of statistical modelling is *interpretation*, one of the topics in Part IV. Who cares if the model is able to tell me that this is a Golden Delicious apple rather than a Granny Smith? The label in the supermarket already told me so; but the question of course is why they taste different, feel different and look different. Fitting a predictive model in such a case may still be informative: when we are able to find out why the model makes a particular prediction, we may be able to learn something about the underlying physical, chemical or biological processes. If we know that a particular gene is associated with the process that we are studying, and both this gene and another one show up as important variables in our statistical model, then we may deduce that the second gene is also involved. This may lead to several new hypotheses that should be tested in the lab. Obviously, when a model has little or no predictive ability it does not make too much sense to try and extract this type of information.

Our knowledge of the system can also serve as a tool to assess the quality of our model. A model that fits the data and seems to be able to predict well is not going to be very popular when its parameters contradict what we know about the underlying process. Often, prior knowledge is available (we

expect a peak at a certain position; we know that model coefficients should not be negative; this coefficient should be larger than the other), and we can use that knowledge to assess the relevance of the fitted model. Alternatively, we can constrain the model in the training phase to take prior knowledge into account, which is often done with constraints. In other cases, the model is hard to interpret because of the sheer number of coefficients that have been fitted, and graphical summaries may fail to show what variables contribute in what way. In such cases, *variable selection* can come to the rescue: by discarding the majority of the variables, hopefully without compromising the model quality, one can often improve predictions *and* make the model much more easy to interpret. Unfortunately, variable selection is an NP-complete problem (which in practice means that even for moderate-sized systems it may be impossible to assess all possible solutions) and one never can be sure that the optimal solution has been found. But then again, any improvement over the original, full, model is a bonus.

For each of the stages in this "classical" data analysis pipeline, a plethora of methods is available. It can be hard to assess which techniques should be considered in a particular problem, and perhaps even more importantly, which should not. The view taken here is that the simplest possibilities should be considered first; only when the results are unsatisfactory, one should turn to more complex solutions. Of course, this is only a very crude first approach, and experienced scientists will have devised many shortcuts and alternatives that work better for their types of data. In this book, I have been forced to make choices. It is impossible to treat all methods, or even a large subset, in detail. Therefore the focus is on an ensemble of methods that will give the reader a broad range of possibilities, with enough background information to acquaint oneself with other methods, not mentioned in this book, if needed. In some cases, methods deserve a mention because of the popularity within the bioinformatics or chemometrics communities. Such methods, together with some typical applications, are treated in the final part of the book.

Given the huge number of packages available on CRAN and the speed with which new ones appear, it is impossible to mention all that are relevant to the material in this book. Where possible, I have limited myself to the recommended packages, and those coming with a default R installation. Of course, alternative, perhaps even much simpler, solutions may be available in the packages that this book does not consider. It pays to periodically scan the CRAN and Bioconductor repositories, or, e.g., check the Task Views that provide an overview of all packages available in certain areas – there is one on Physics and Chemistry, too.

Part I

Preliminaries

2

Data

In this chapter, some typical data sets are presented, several of which will occur throughout the book. All data sets are accessible, either through one of the packages mentioned in the text, or in the **ChemometricsWithR** package. Chemical data sets nowadays are often characterized by a relatively low number of samples and a large number of variables, a result of the predominant spectroscopic measuring techniques enabling the chemist to rapidly acquire a complete spectrum for one sample. Depending on the actual technique employed, the number of variables can vary from several hundreds (typical in infrared measurements) to tens of thousands (e.g., in Nuclear Magnetic Resonance, NMR). A second characteristic is the high correlation between variables: neighbouring spectral variables usually convey very similar information. An example is shown in Figure 2.1, depicting the gasoline data set, one of several data sets that will be used throughout this book. It shows near-infrared (NIR) spectra of sixty gasolines at wavelengths from 900 to 1700 nm in 2 nm intervals [5], and is available in the **pls** package. The plot is made using the following piece of code:

```
> data(gasoline, package = "pls")
> wavelengths <- seq(900, 1700, by = 2)
> matplot(wavelengths, t(gasoline$NIR), type = "l",
+         lty = 1, xlab = "Wavelength (nm)", ylab = "1/R")
```

The `matplot` function is used to plot all columns of matrix `t(gasoline$NIR)` (or, equivalently, all rows of matrix `gasoline$NIR`) against the specified wavelengths. Clearly, all samples have very similar features – it is impossible to distinguish individual samples in the plot. NIR spectra are notoriously hard to interpret: they consist of a large number of heavily overlapping peaks which leads to more or less smooth spectra. Nevertheless, the technique has proven to be of immense value in industry: it is a rapid, non-destructive method of analysis requiring almost no sample preprocessing, and it can be used for quantitative predictions of sample properties. The data used here can be used to quantitatively assess the octane number of the gasoline samples, for instance.

R. Wehrens, *Chemometrics with R: Multivariate Data Analysis in the Natural Sciences and Life Sciences*, Use R!, DOI 10.1007/978-3-642-17841-2_2,

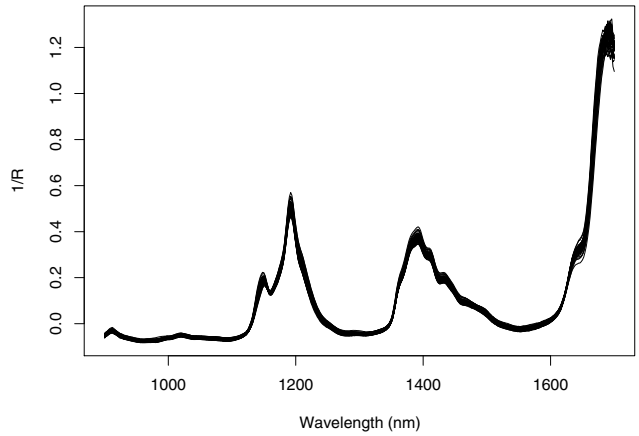

Fig. 2.1. Near-infrared spectra of sixty gasoline samples, consisting of 401 reflectance values measured at equally spaced wavelengths between 900 and 1700 nm.

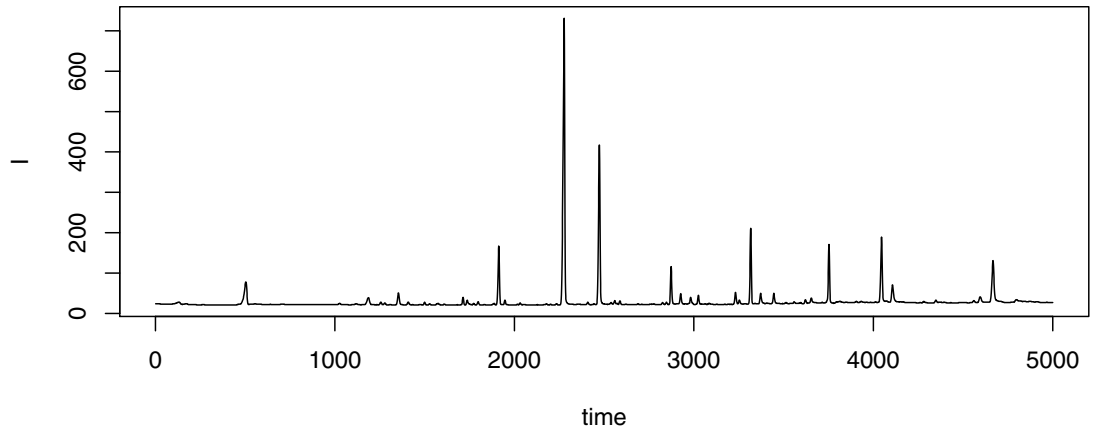

Fig. 2.2. The first gas chromatogram of data set `gaschrom` from the **ptw** package.

In other cases, specific variables can be directly related to absolute or relative concentrations. An example is the `gaschrom` data set from the **ptw** package, containing gas chromatograms measured for calibration purposes. The first sample is shown in Figure 2.2. Each feature, or peak, corresponds to the elution of a compound, or in more complex cases, a number of overlapping compounds. These peaks can be easily quantified, usually by measuring peak area, but sometimes also by peak height. Since the number of features usually is orders of magnitude smaller than the number of variables in the original data, summarising the chromatograms with a peak table containing position and intensity information can lead to significant data compression.

An example in which most of the variables correspond to concentrations is the wine data set, used throughout the book. It is a set consisting of 177 wine samples, with thirteen measured variables [6]:

```
> data(wines, package = "kohonen")
> colnames(wines)
```

```
 [1] "alcohol"           "malic acid"        "ash"
 [4] "ash alkalinity"    "magnesium"         "tot. phenols"
 [7] "flavonoids"        "non-flav. phenols" "proanth"
[10] "col. int."         "col. hue"          "OD ratio"
[13] "proline"
```

Variables are reported in different units. All variables apart from `"col. int."`, `"col. hue"` and `"OD ratio"` are concentrations. The meaning of the variables color intensity and color hue is obvious; the OD ratio is the ratio between the absorbance at wavelengths 280 and 315 nm. All wines are from the Piedmont region in Italy. Three different classes of wines are present: Barolo, Grignolino and Barberas. Barolo wine is made from Nebbiolo grapes; the other two wines have the name of the grapes from which they are made. Production areas are partly overlapping [6].

```
> table(vintages)
```

```
vintages
   Barbera     Barolo Grignolino
        48         58         71
```

The obvious aim in the analysis of such a data set is to see whether there is any structure that can be related to the three cultivars. Possible questions are: "which varieties are most similar?", "which variables are indicative of the variety?", "can we discern subclasses within varieties?", etcetera.

A quick overview of the first few variables can be obtained with a so-called pairs plot[1]:

```
> pairs(wines[,1:3], pch = wine.classes, col = wine.classes)
```

This leads to the plot shown in Figure 2.3. It is clear that the three classes can be separated quite easily – consider the plot of alcohol against malic acid, for example.

A further data set comes from mass spectrometry. It contains 327 samples from three groups: patients with prostate cancer, benign prostatic hyperplasia, and normal controls [7,8]. The data have already been preprocessed (binned, baseline-corrected, normalized – see Chapter 3). The m/z values range from 200 to 2000 Dalton. The data set is available in the R package **msProstate**:

[1] Gray-scale figures such as shown throughout the book are obtained by, e.g., `col = gray(0:2/4)[wine.classes]`.
In the text and the code we will in almost all cases use the default R colour palette.

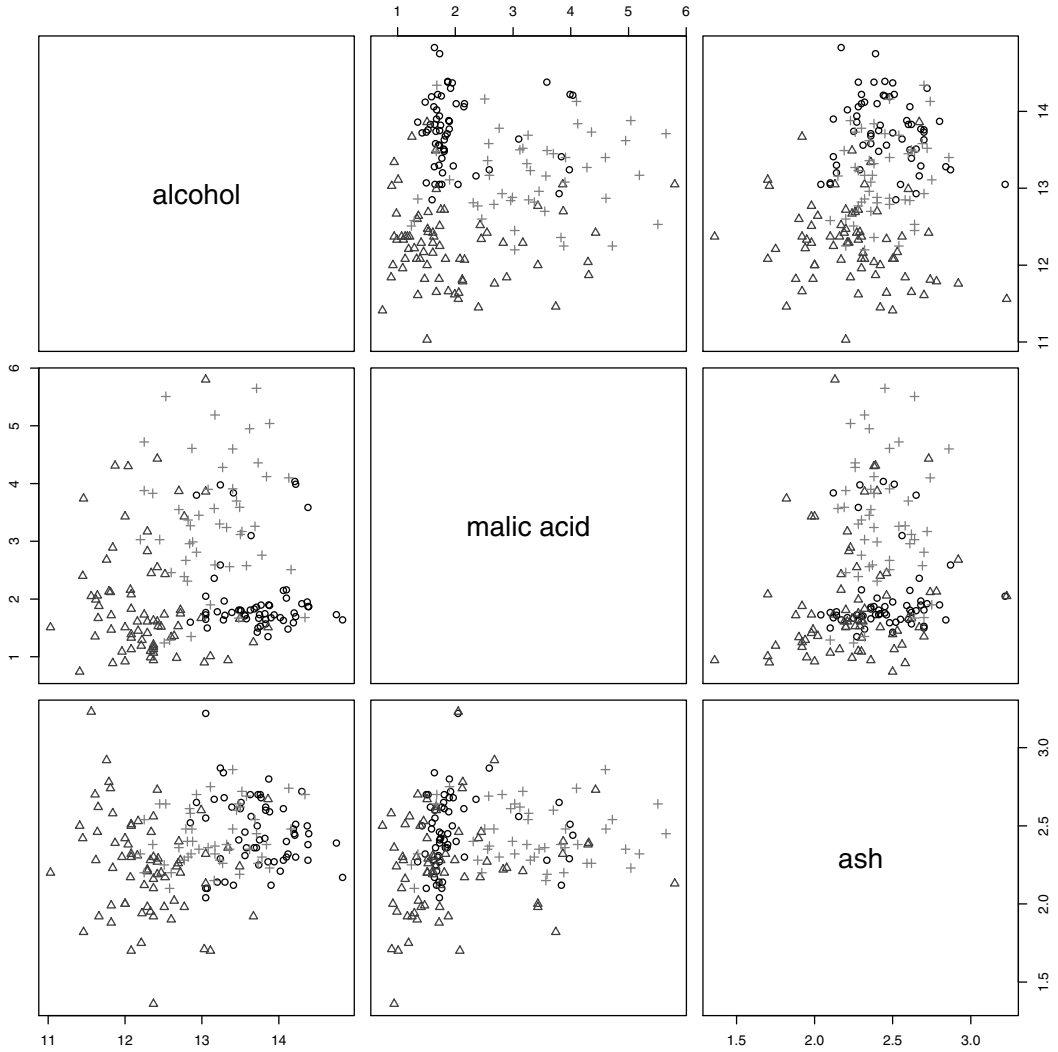

Fig. 2.3. A pairs plot of the first three variables of the wine data. The three vintages are indicated with different shades of gray and plotting symbols: Barbera wines are indicated with black circles, Barolos with dark gray triangles and Grignolinos with gray plusses.

```
> data(Prostate2000Raw, package = "msProstate")
> plot(Prostate2000Raw$mz, Prostate2000Raw$intensity[,1],
+      type = "h", xlab = "m/z", ylab = "Intensity",
+      main = "Prostate data")
```

Figure 2.4 shows the first mass spectrum, that of a healthy control sample. In total, there are 168 tumour samples, 81 controls, and 78 cases of benign prostate enlargement: all samples have been measured in duplicate.

```
> table(Prostate2000Raw$type)

   bph control     pca
   156     162     336
```

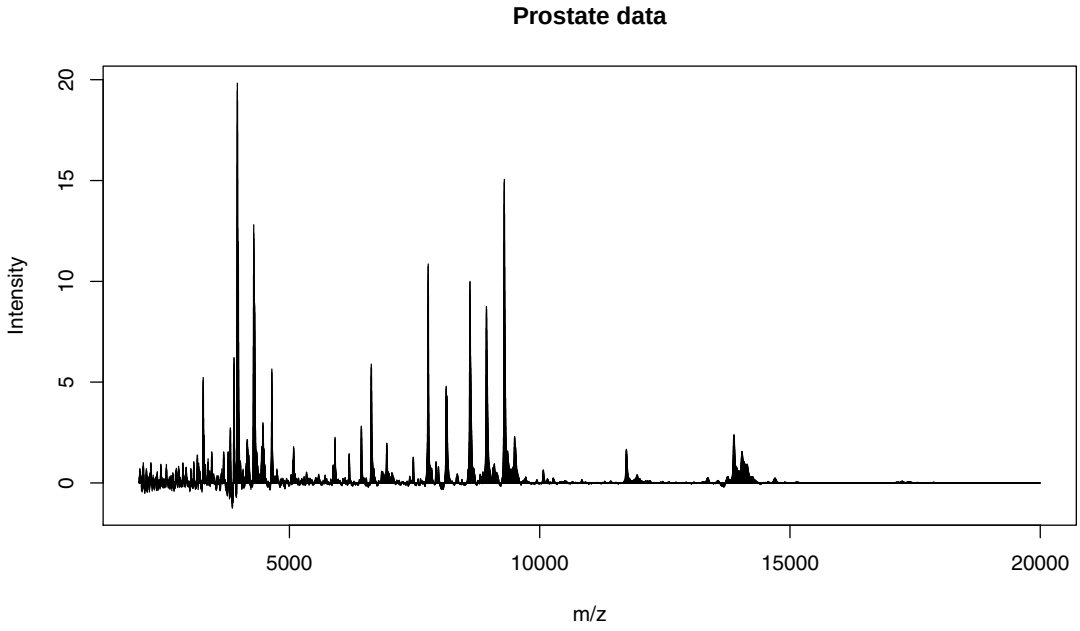

Fig. 2.4. The first mass spectrum in the prostate MS data set.

Such data can serve as diagnostic tools to distinguish between healthy and diseased tissue, or to differentiate between several disease states. The number of samples is almost always very low – for rare diseases, patients are scarce, and stratification to obtain relatively homogeneous groups (age, sex, smoking habits, ...) usually does the rest; and in cases where the measurement is unpleasant or dangerous it may be difficult or even unethical to get data from healthy controls. On the other hand, the number of variables per sample is often huge. This puts severe restrictions on the kind of analysis that can be performed and makes thorough validation even more important.

The final data set in this chapter comes from LC-MS, the combination of liquid chromatography and mass spectrometry. The chromatography step serves to separate the components of a mixture on the basis of properties like polarity, size, or affinity. At specific time points a mass spectrum is recorded, containing the counts of particles with specific mass-to-charge (m/z) ratios. Measuring several samples therefore leads to a data cube of dimensions `ntime`, `nmz`, and `nsample`; the number of timepoints is typically in the order or thousands, whereas the number of samples rarely exceeds one hundred. Mass spectra can be recorded at a very high resolution and to enable statistical analysis, m/z values are typically *binned* (or "bucketed"). Even then, thousands of variables are no exception. Package **ptw** provides a data set, `lcms`, containing data on three tryptic digests of E. coli proteins [9]. Figure 2.5 shows a top view of the first sample, with projections to the top and right of the main plot. The top projection leads to the "Total Ion Current" (TIC) chromatogram, and would be obtained if there would be no separation along the m/z axis;

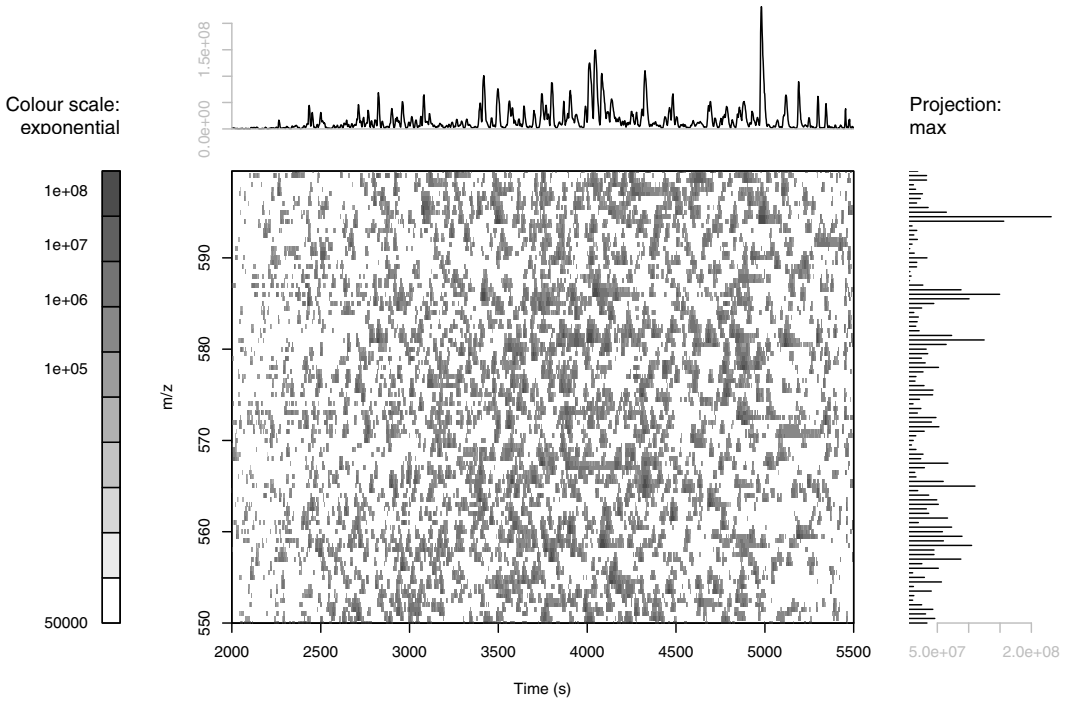

Fig. 2.5. Top view of the first sample in data set `lcms`. The TIC chromatogram is shown on the top, and the direct infusion mass spectrum on the right.

similarly, if the chromatographic dimension would be absent, the mass spectrum of the whole sample would be very close to the projection on the right (a "direct infusion" spectrum). The whole data set consists of three of such planes, leading to a data cube of size $100 \times 2000 \times 3$.

3

Preprocessing

Textbook examples typically use clean, perfect data, allowing the techniques of interest to be explained and illustrated. However, in real life data are messy, noisy, incomplete, downright faulty, or a combination of these. The first step in any data analysis often consists of preprocessing to assess and possibly improve data quality. This step may actually take more time than the analysis itself, and more often than not the process consists of an iterative procedure where data preprocessing steps are alternated with data analysis steps.

Some problems can immediately be recognized, such as measurement noise, spikes, and unrealistic values. In these cases, taking appropriate action is rarely a problem. More difficult are the cases where it is not obvious which characteristics of the data contain information, and which do not. There are many examples where chance correlations lead to statistical models that are perfectly able to model the training data but have no predictive abilities whatsoever.

This chapter will focus on standard preprocessing techniques used in the natural sciences and the life sciences. Data are typically spectra or chromatograms, and topics include noise reduction, baseline removal, peak alignment, peak picking, and scaling. Only the basic general techniques are mentioned here; some more specific ways to improve the quality of the data will be treated in later chapters. Examples include Orthogonal Partial Least Squares for removing uncorrelated variation (Section 11.2) and variable selection (Chapter 10).

3.1 Dealing with Noise

Physico-chemical data always contain noise, where the term "noise" is usually reserved for small, fast, random fluctuations of the response. The first aim of any scientific experiment is to generate data of the highest quality, and much effort is usually put into decreasing noise levels. The simplest experimental way is to perform n repeated measurements, and average the individual spectra, leading to a noise reduction with a factor $\sqrt{n}$. In NMR spectroscopy, for

R. Wehrens, *Chemometrics with R: Multivariate Data Analysis in the Natural Sciences and Life Sciences*, Use R!, DOI 10.1007/978-3-642-17841-2_3,

example, a relatively insensitive analytical method, signal averaging is routine practice, where one has to strike a balance between measurement time and data quality.

As an example, we consider the prostate data, where each sample has been measured in duplicate. The replicate measurements of the prostate data cover consecutive rows in the data matrix. Averaging can be done using the following steps:

```
> prostate.array <- array(t(Prostate2000Raw$intensity),
+                         c(2, 327, 10523))
> prostate <- apply(prostate.array, c(2,3), mean)
> dim(prostate)
```

```
[1]   327 10523
```

The idea is to convert the matrix into an array where the first dimension contains the two replicates for every sample – each element in the first dimension thus contains one complete set of measurements. The final data matrix is obtained by averaging the replicates. The function `apply` is useful here: it (indeed) applies the function given by the third argument to the data indicated by the first argument while keeping the dimensions indicated by the second – got that? In this code snippet the outcome of `apply` is a matrix having dimensions equal to the second and third dimensions of the input array. The first dimension is averaged out. As we will see later, `apply` is extremely handy in many situations. There is, however, also another much faster possibility using `rowsum`:

```
> prostate <- rowsum(t(Prostate2000Raw$intensity),
                     group = rep(1:327, each = 2),
                     reorder = FALSE) / 2
> dim(prostate)
```

```
[1]   327 10523
```

This function sums all the rows for which the grouping variable (the second argument) is equal. Since there are two replicates for every sample, the result is divided by two to get the average values, and is stored in variable `prostate`. For this new variable we should also keep track of the corresponding class labels:

```
> prostate.type <- Prostate2000Raw$type[seq(1, 654, by = 2)]
```

To plot the result, we combine the original data with the averaged data in a three-column matrix `x`:

```
> x <- cbind(prostate[1,1:500],
+            Prostate2000Raw$intensity[1:500, 1:2])
```

The result of the signal averaging is visualized in Figure 3.1, again using the function `matplot`:

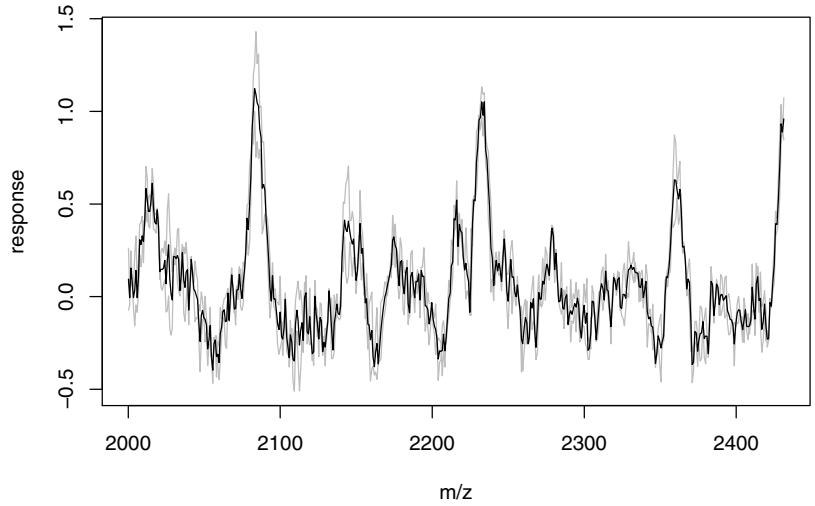

Fig. 3.1. The first averaged mass spectrum in the Prostate data set; only the first 500 m/z values are shown. Original data are in gray.

```
> matplot(Prostate2000Raw$mz[1:500], x, type = "l",
+         col = c(1, "gray", "gray"), lty = c(1,2,2),
+         xlab = "m/z", ylab = "response")
```

Clearly, despite the averaging, the noise is appreciable; reducing the noise level while taking care not to destroy the data structure would make subsequent analysis much easier.

The simplest approach is to apply a *running mean*, i.e. to replace every single value by the average of the k points around it. The value of k needs to be optimized; large values lead to a high degree of smoothing, but also to peak distortion, and low values of k can only make small changes to the signal. Running means can be easily calculated using the function `embed`, providing is a matrix containing as rows successive chunks of the original data vector; using the function `rowMeans` one then can obtain the desired running means.

```
> rmeans <- rowMeans(embed(prostate[1,1:500], 5))
> plot(Prostate2000Raw$mz[1:500], prostate[1,1:500],
+      type = "l", xlab = "m/z", ylab = "response",
+      main = "running means", col = "gray")
> lines(Prostate2000Raw$mz[3:498], rmeans, type = "l")
```

As can be seen in the left plot in Figure 3.2, the smoothing effectively reduces the noise level. Note that the points at the extremes need to be treated separately in this implementation. The price to be paid is that peak heights are decreased, and especially with larger spans one will see appreciable peak broadening. These effects can sometimes be countered by using running medi-

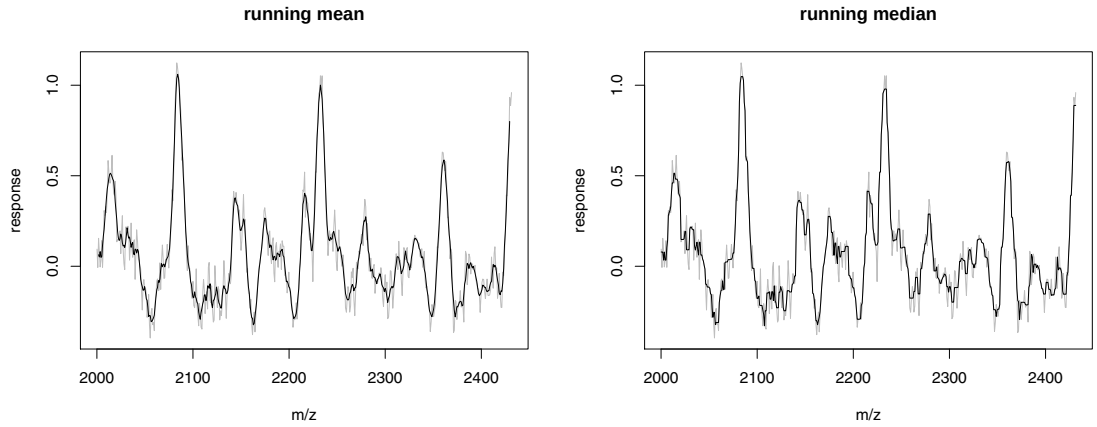

Fig. 3.2. Smoothing of the averaged mass spectrum of Figure 3.1: a running mean (left plot) and a running median (right plot), both with a window size of five.

ans instead of running means. The function `runmed`, part of the **stats** package, is available for this:

```
> plot(Prostate2000Raw$mz[1:500], prostate[1,1:500],
+      type = "l", xlab = "m/z", ylab = "response",
+      main = "running median", col = "gray")
> lines(Prostate2000Raw$mz[1:500],
+       runmed(prostate[1,1:500], k = 5), type = "l")
```

The result is shown in the right plot in Figure 3.2. Its appearance is less smooth than the running mean with the same window size; in particular, peak shapes seem less natural. Note that the function `runmed` does return a vector with the same length as the input – the points at the extremes are left unchanged. The plots of the residuals in Figure 3.3 show that both smoothing techniques do quite a good job in removing high-frequency noise components without distorting the signal too much.

Many other smoothing functions are available – only a few will be mentioned here briefly. In signal processing, Savitsky-Golay filters are a popular choice [10]; every point is replaced by a smoothed estimate obtained from a local polynomial regression. An added advantage is that derivatives can simultaneously be calculated (see below). In statistics, robust versions of locally weighted regression [11] are often used; `loess` and its predecessor `lowess` are available in R as simple-to-use implementations. The fact that the fluctuations in the noise usually are much faster than the data has led to a whole class of frequency-based smoothing methods, of which wavelets [12] are perhaps the most popular ones. The idea is to set the coefficients for the high-frequency components to zero, which should leave only the signal component.

A special kind of smoothing is formed by *binning*, also called bucketing, which not only averages consecutive values but also decreases the number of variables. To replace five data points with their average, one can use:

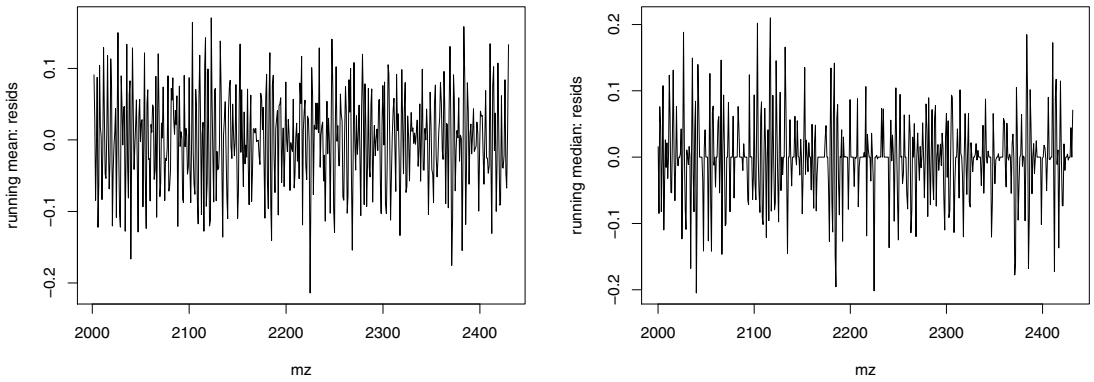

Fig. 3.3. Residuals of smoothing the first spectrum of the prostate data with a running mean (left plot) or running median (right).

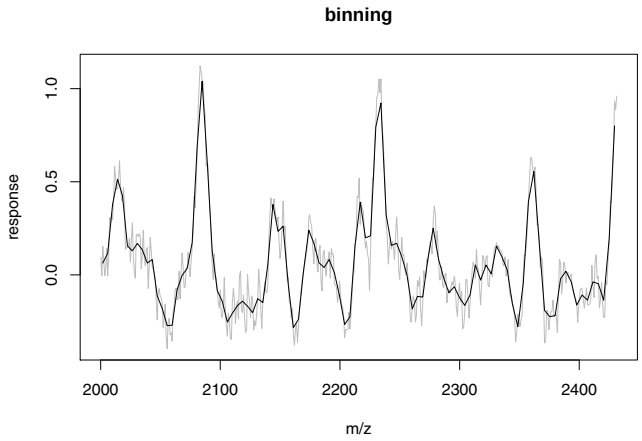

Fig. 3.4. Binned version of the mass spectrometry data from Figure 3.1. Five data points constitute one bin.

```
> mznew <- colMeans(matrix(Prostate2000Raw$mz[1:500], nrow = 5))
> xnew <- colMeans(matrix(prostate[1, 1:500], nrow = 5))
> plot(Prostate2000Raw$mz[1:500], prostate[1, 1:500],
+      type = "l", xlab = "m/z", ylab = "response",
+      main = "binning", col = "gray")
> lines(mznew, xnew)
```

We have seen the idea before: in this case we fill a matrix with five rows column-wise with the data, and then average over the rows[1]. This leads to the plot in Figure 3.4. Obviously, the binned representation gives a cruder description of the data, but still is able to follow the main features. Again, determining the optimal bin size is a matter of trial and error. Binning has several major advantages over running means and medians. First, it can be

[1] What would the code look like using `rowsum`?

applied when data are not equidistant and even when the data are given as positions and intensities of features, as is often the case with mass-spectrometric data. Second, the effect of peak shifts (see below) is decreased: even when a peak is slightly shifted, it will probably be still within the same bin. And finally, more often than not the variable-to-object ratio is extremely large in data sets from the life sciences. Summarising the information in fewer variables in many cases makes the subsequent statistical modelling more easy.

Although smoothing leads to data that are much better looking, one should also be aware of the dangers. Too much smoothing will remove features, and even when applied prudently, the noise characteristics of the data will be different. This may significantly affect statistical modelling.

3.2 Baseline Removal

In some forms of spectroscopy one can encounter a baseline, or "background signal" that is far away from the zero level. Since this influences measures like peak height and peak area, it is of utmost importance to correct for such phenomena.

Infrared spectroscopy, for instance, can lead to scatter effects – the surface of the sample influences the measurement. As a result, one often observes spectral offsets: two spectra of the same material may show a constant difference over the whole wavelength range. This may be easily removed by taking first derivatives (i.e., looking at the *differences* between intensities at sequential wavelengths, rather than the intensities themselves). Take a look at the gasoline data:

```
> nir.diff <- t(apply(gasoline$NIR, 1, diff))
> matplot(wavelengths[-1] + 1, t(nir.diff),
+         type = "l", xlab = "Wavelength (nm)",
+         ylab = "1/R (1st deriv.)", lty = 1, col = 1)
```

Note that the number of variables decreases by one. The result is shown in Figure 3.5. Comparison with the original data (Figure 2.1) shows more detailed structure; the price is an increase in noise. A better way to obtain first-derivative spectra is given by the Savitsky-Golay filter, which is not only a smoother but can also be used to calculate derivatives:

```
> nir.deriv <- apply(gasoline$NIR, 1, sgolayfilt, m = 1)
```

In this particular case, the differences between the two methods are very small.

Another way to remove scatter effects in infrared spectroscopy is Multiplicative Scatter Correction (MSC, [13,14]). One effectively models the signal of a query spectrum as a linear function of the reference spectrum:

$$y_q = a + by_r$$

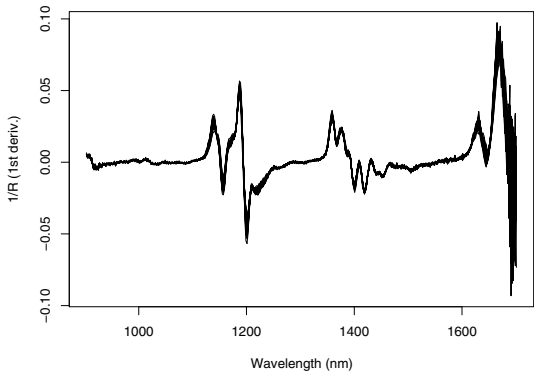

Fig. 3.5. First-derivative representation of the gasoline NIR data.

An obvious reference spectrum may not be available, and then often a mean spectrum is used. This is also the approach in the `msc` function of the **pls** package:

```
> nir.msc <- msc(gasoline$NIR)
```

For the gasoline data, the differences are quite small.

In more difficult cases, a non-constant baseline drift can be observed. First derivatives are not enough to counter such effects, and one has to resort to techniques that actually estimate the shape of the baseline. The exact function is usually not important – the baseline will be subtracted and that is it. To illustrate this point, consider the first chromatogram in the `gaschrom` data. One very simple solution is to connect local minima, obtained from, e.g., 200-point sections:

```
> x <- gaschrom[1,]
> lsection <- 200
> xmat <- matrix(x, nrow=lsection)
> ymin <- apply(xmat, 2, min)
> plot(x, type = "l", col = "gray", ylim = c(20, 50),
+      xlab = "Time", ylab = "I")
> lines(rep(ymin, each = lsection))
```

We have used the by now familiar trick to convert a vector to a matrix and calculate minimal values for every column to obtain the intensity levels of the horizontal line segments. The result is shown in the left plot of Figure 3.6. Obviously, a more smooth baseline estimate would be better. One function that can be used is `loess`, fitting local polynomials through the minimal values:

```
> minlocs <- seq(lsection/2 + 1, length(x), len = length(ymin))
> bsln.loess <- loess(ymin ~ minlocs)
> lines(predict(bsln.loess, 1:length(x)), lwd = 2)
```

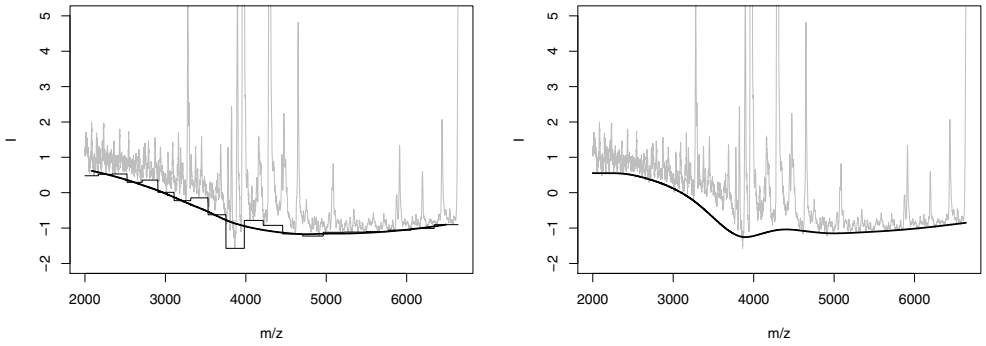

Fig. 3.6. Simple baseline correction for the first chromatogram in the `gaschrom` data: in the left plot the baseline is estimated by a series of twenty local minima, the connected horizontal segments. The thick line indicates the loess smooth (using default settings) of these minima. Right plot: asymmetric least squares estimate of the baseline.

This leads to the right plot in Figure 3.6. The `loess` solution clearly is much smoother and does not follow the data that much. For the current distortion, that is all right, but in some cases one may want a different behaviour. Twiddling with the settings (the length of the segments to find local minima, and `loess` smoother settings) can lead to even better results.

Another alternative is to use asymmetric least squares, where deviations above the fitted curve are not taken into account (or only with a very small weight). This is implemented in function `baseline.corr` in the **ptw** package, which returns a baseline-corrected signal. Internally, it uses the function `asysm` to estimate the baseline:

```
> plot(x, col = "gray", type = "l", ylim = c(20, 50),
+      xlab = "Time", ylab = "I")
> lines(asysm(x), lwd = 2)
```

The result is shown in the right plot of Figure 3.6. Again, the parameters of the `asysm` function may be tweaked to get optimal results.

3.3 Aligning Peaks – Warping

Many analytical data suffer from small shifts in peak positions. In NMR spectroscopy, for example, the position of peaks may be influenced by the pH. What complicates matters is that in NMR, these shifts are by no means uniform over the data; rather, only very few peaks shift whereas the majority will remain at their original locations. The peaks may even move in different directions. In mass spectrometry, the shift is more uniform over the m/z axis and is more easy to account for – if one aims to analyse the data in matrix form, binning is required, and in many cases a suitable choice of bins will

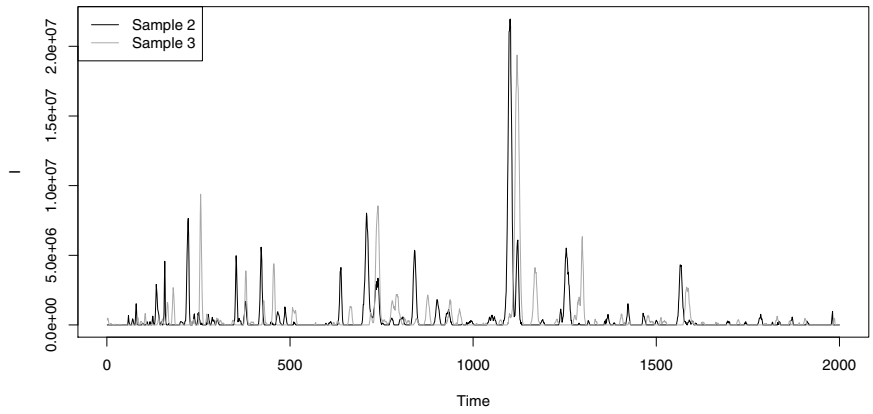

Fig. 3.7. Comparison of two mass chromatograms of the `lcms` data set. Clearly, corresponding features are not in the same positions.

already remove most if not all of the effects of shifts. Moreover, peak shifts are usually small, and may be easily corrected for by the use of standards.

The biggest shifts, however, are encountered in liquid chromatography. Two different chromatographic columns almost never give identical elution profiles, up to the extent that peaks may even swap positions. The situation is worse than in gas chromatography, since retention mechanisms are more complex in the liquid phase than in the gas phase. Moreover, ageing is an important factor in column quality, and a column that has been used for some time almost certainly will show different chromatograms than when freshly installed.

Peak shifts pose significant problems in modelling. In Figure 3.7 the first mass chromatograms in two of the samples of the `lcms` data are shown:

```
> plot(lcms[1,,2], type = "l", xlab = "Time", ylab = "I")
> lines(lcms[1,,3], type = "l", col = "gray")
```

Clearly, both chromatograms contain the same features, although at different locations – the shift is, equally clearly, not constant over the whole range. Comparing these chromatograms with a distance-based similarity function based on Euclidean distance, comparing signals at the same time points, will lead to the conclusion that there are huge differences, whereas the chromatograms in reality are very similar.

Correction of such shifts is known as "time warping", one of the more catchy names in data analysis. The technique originates in speech processing [15, 16], and nowadays many forms exist. The most popular in the literature for natural sciences and life sciences are Dynamic Time Warping (DTW, [17]), Correlation-Optmized Warping (COW, [18]) and Parametric Time Warping (PTW, [19]). Often, the squared differences between the two signals are used as optimization criterion; this is the case for DTW and the original version of PTW [19]. The R package **ptw** [9] also provides a measure called the *weighted*

cross correlation (WCC, [20]) to assess the similarity of two patterns – note that in this context the WCC is used as a distance measure so that a value of zero indicates perfect alignment [9]. COW maximizes the correlation between patterns, where the signals are cut into several segments which are treated separately. Both DTW and PTW currently are available as R packages on CRAN, as well as a penalized form of dynamic time warping (package **VPdtw**, [21]).

3.3.1 Parametric Time Warping

In PTW, one approximates the time axis of the reference signal by applying a polynomial transformation of the time axis of the sample [19]:

$$\hat{S}(t_k) = S(w(t_k)) \tag{3.1}$$

where $\hat{S}(t_k)$ is the value of the warped signal at time point t_k, where k is an index. The warping function, w, is given by:

$$w(t) = \sum_{j=0}^{J} a_j t^j \tag{3.2}$$

with J the maximal order of the polynomial. Only low-order polynomials are used in general. Since neighbouring points on the time axis will be warped with almost the same amount, peak shape distortions are limited. The optimization then finds the set of coefficients $a_0, ...a_J$ that minimizes the difference between the sample S and reference R, using whatever difference measure is desired.

This procedure is very suitable for modelling the gradual deterioration of chromatographic columns, so that measurements taken days or weeks apart can still be made comparable. For situations where a few individual peak shifts have to be corrected (e.g. pH-dependent shifting of patterns in NMR spectra), the technique is less ideal.

We will illustrate the use of PTW by aligning the mass chromatograms of Figure 3.7. We will use sample number 2 as a reference, and warp the third sample so that the peak positions show maximal overlap:

```
> sref <- lcms[1,,2]
> ssamp <- lcms[1,,3]
> lcms.warp <- ptw(sref, ssamp, init.coef = c(0, 1, 0))
> summary(lcms.warp)

PTW object: single alignment of 1 sample on 1 reference.

Warping coefficients:
          [,1]     [,2]          [,3]
[1,] 41.83585 0.974073 5.479969e-06

Warping criterion: WCC
Value: 0.08781744
```

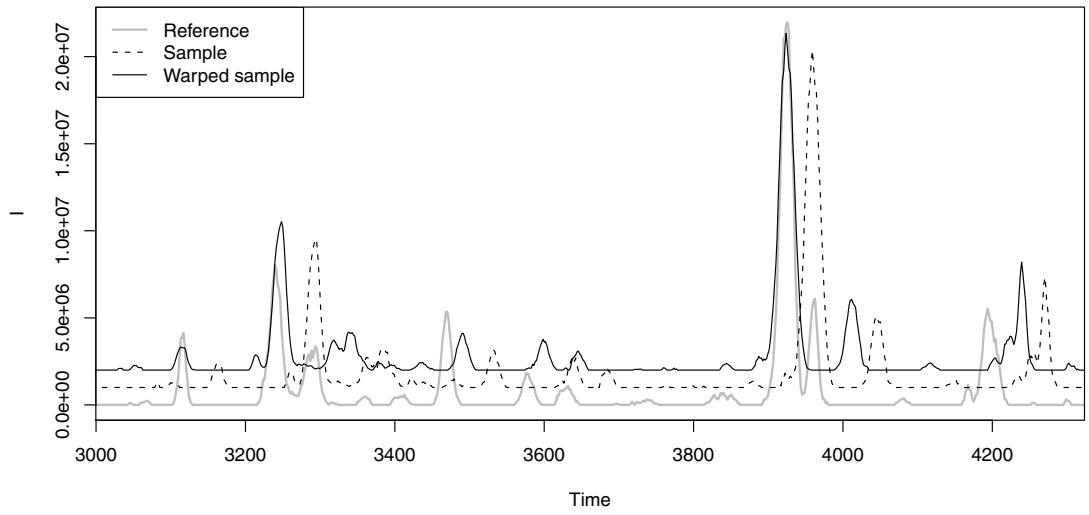

Fig. 3.8. PTW of the data shown in Figure 3.7, using a quadratic warping function. Small offsets have been added to the sample and warped sample spectra.

Using the default quadratic warping function with initial values `init.coef = c(0, 1, 0)`, corresponding to the unit warp (no shift, unit stretch, no quadratic warping), we arrive at a warping where the sample is shifted almost 42 points to the right, is compressed almost 3%, and experiences also a quadratic warping (with a small coefficient). The result is an agreement of just under 0.09, according to the default WCC criterion. A visual check confirms that the peak alignment is much improved:

```
> plot(time, sref, type = "l", lwd = 2, col = "gray",
+      xlim = c(time[600], time[1300]),
+      xlab = "Time", ylab = "I")
> lines(time, ssamp + 1e6, lty = 2)
> lines(time, lcms.warp$warped.sample + 2e6)
> legend("topleft", lty = c(1,2,1), col = c("gray", 1, 1),
+        legend = c("Reference", "Sample", "Warped sample"),
+        lwd = c(2, 1, 1))
```

The result is shown in Figure 3.8. To show the individual traces more clearly, a small vertical offset has been applied to both the unwarped and warped sample. Obviously, the biggest gains can be made at the largest peaks, and in the warped sample the features around 3250 and 3900 seconds are aligned really well. Nevertheless, some other features, such as the peaks between 3450 and 3650 seconds, and the peaks at 3950 and 4200 seconds still show shifts. This simple quadratic warping function apparently is not flexible enough to iron out these differences. Note that RMS-based warping in this case leads to very similar results:

```
> lcms.warpRMS <- ptw(sref, ssamp, optim.crit = "RMS")
> lcms.warpRMS$warp.coef

          [,1]      [,2]          [,3]
[1,] 40.84908 0.9719185 9.619694e-06
```

More complex warping functions, fitting polynomials of degrees three to five, can be tried:

```
> lcms.warp2 <- ptw(sref, ssamp, init = c(0, 1, 0, 0))
> lcms.warp3 <- ptw(sref, ssamp, init = c(0, 1, 0, 0, 0))
> lcms.warp4 <- ptw(sref, ssamp, init = c(0, 1, 0, 0, 0, 0))
```

To visualize these warping functions, we first gather all warpings in one list, and obtain the qualities of each element using a close relative of the `apply` function, `sapply`:

```
> allwarps <- list(lcms.warp, lcms.warp2, lcms.warp3, lcms.warp4)
> wccs <- sapply(allwarps, function(x) x$crit.value)
> wccs <- round(wccs*1000) / 1000
```

Where `apply` operates on rows or columns of a matrix, `sapply` performs actions on list elements, and returns the result in a simple way, in this case a matrix. We use the `round` trick to avoid having too many digits in the plot later on.

Because we are interested in the deviations from the identity warp (i.e. no change), we subtract that from the warping functions:

```
> allwarp.funs <- sapply(allwarps, function(x) x$warp.fun)
> warpings <- allwarp.funs - 1:length(sref)
```

Finally, we can plot the columns of the resulting matrix using the `matplot` function:

```
> matplot(warpings, type = "l", lty = rep(c(1,2), 2),
+         col = rep(c(1,"gray"), each = 2))
> legend("topleft", lty = rep(c(1,2), 2),
+        col = rep(c(1,"gray"), each = 2),
+        legend = paste("Degree", 2:5, " - WCC =", wccs))
```

This leads to Figure 3.9. A horizontal line here would indicate the identity warp. The fourth-degree warping shows the biggest deviation from the unwarped signal, especially in the right part of the chromatogram, but is rewarded by the lowest distance between sample and reference. The warping functions for degrees three and five are almost identical. This shows, in fact, that it is possible to end up in a local minimum: the fifth-degree warping function should be at least as good as the fourth-degree function. It is usually a good idea to use several different starting values in these situations.

One very important characteristic of LC-MS data is that it contains multiple m/z traces. If the main reason for differences in retention time is the

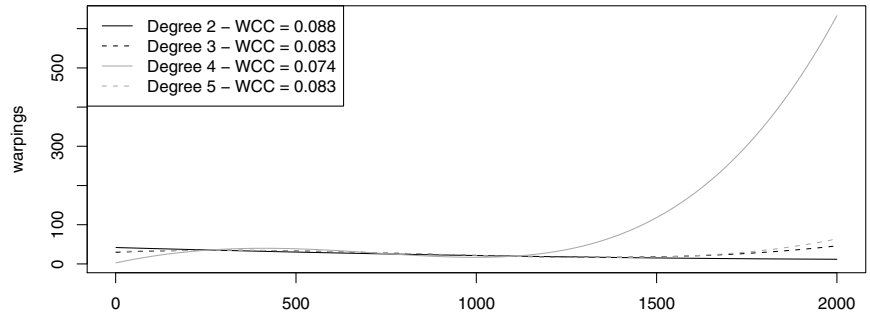

Fig. 3.9. PTW warping functions for different degrees. The fourth degree warping is the most agressive but also achieves the best agreement.

state of the column, all traces should be warped with the same warping function. When using multiple traces, one should have less trouble identifying the optimal warping since ambiguities, that may exist in single chromatograms, are resolved easily [9]. The `ptw` package also has provisions for such so-called global alignment – we will come back to this later. One can also select traces on the basis of certain criteria, such as maximal intensity. A popular criterion is Windig's Component Detection Algorithm (CODA, [22]), which is one of the selection criteriain function `select.traces` in the **ptw** package. CODA basically selects high-variance traces after smoothing, and a unit-length scaling step – the variance of a trace with one high-intensity feature will always be larger than that of a trace with the same length but many low-intensity features.

The following example chooses the ten traces which show the highest values for the CODA criterion, and uses these for constructing a warping function:

```
> sref <- lcms[,,2]
> ssamp <- lcms[,,3]
> traces <- select.traces(sref, criterion = "var")
> lcms.warpglobal <-
+   ptw(sref, ssamp, warp.type = "global",
+       selected.traces = traces$trace.nrs[1:10])
> summary(lcms.warpglobal)

PTW object: global alignment of 10 samples on 10 references.

Warping coefficients:
         [,1]      [,2]         [,3]
[1,] 91.37956 0.8727316 5.969587e-05

Warping criterion: WCC
Value: 0.1309732
```

Note that the ten selected traces are seen as separate samples for which one global warping function should be found. This is a situation that will occur often in practice – using a couple of information-rich mass traces the quality of the alignment is probably much higher than when using all traces simultaneously. If necessary, small, individual alignments can be made to correct for any remaining shifts. Let us compare an eight-degree individual warping with a global warping of degree two, followed by a four-degree individual warping. The latter strategy also generates a net warping of degree eight [9]:

```
> sample2.indiv.warp <-
+   ptw(lcms[,,3], lcms[,,2],
+       init.coef = c(0, 1, 0, 0, 0, 0, 0, 0, 0))
> sample2.global.warp <-
+   ptw(lcms[,,3], lcms[,,2], init.coef = c(0, 1, 0),
+       warp.type = "multiple")
> sample2.final.warp <-
+   ptw(lcms[,,3], lcms[,,2],
+       init.coef = c(sample2.global.warp$warp.coef, 0, 0))
```

The individual warping is initialized using estimates for the lower degree coefficients found in the global warping. We evaluate the results by looking at the total ion currents, given by the column sums of the warped samples:

```
> plot(time, colSums(lcms[,,3]), col = "gray", lwd = 3,
+      type = "l", main = "PTW (indiv.)", ylab = "I")
> lines(time, colSums(sample2.indiv.warp$warped.sample))
> legend("topleft", legend = c("Sample", "Reference"),
+        lty = 1, col = c("black", "gray"), lwd = c(1,3))
```

This gives the top panel in Figure 3.10, showing the result of the individual eight-degree warping. The bottom plot is produced with similar code, using `sample2.final.warp` instead of `sample2.indiv.warp`. Clearly, already the individual alignments lead to an overall result that is not bad at all, with very good agreement in the middle of the signal. At longer times, however, the compression is much too strong and the dominant feature just after 5000 seconds is placed too early in the warped signal. The combined strategy fares much better with a more or less perfect warping. Note the absence of the aligned signal at the extremes in both cases, the result of an overall compression.

3.3.2 Dynamic Time Warping

Also in Dynamic Time Warping (DTW), implemented in package **dtw** [23], the first step is to construct a warping function. This provides a mapping from the indices in the query signal to the points in the reference signal[2]:

[2] Note that the order of sample and reference signals is reversed compared to the `ptw` function.

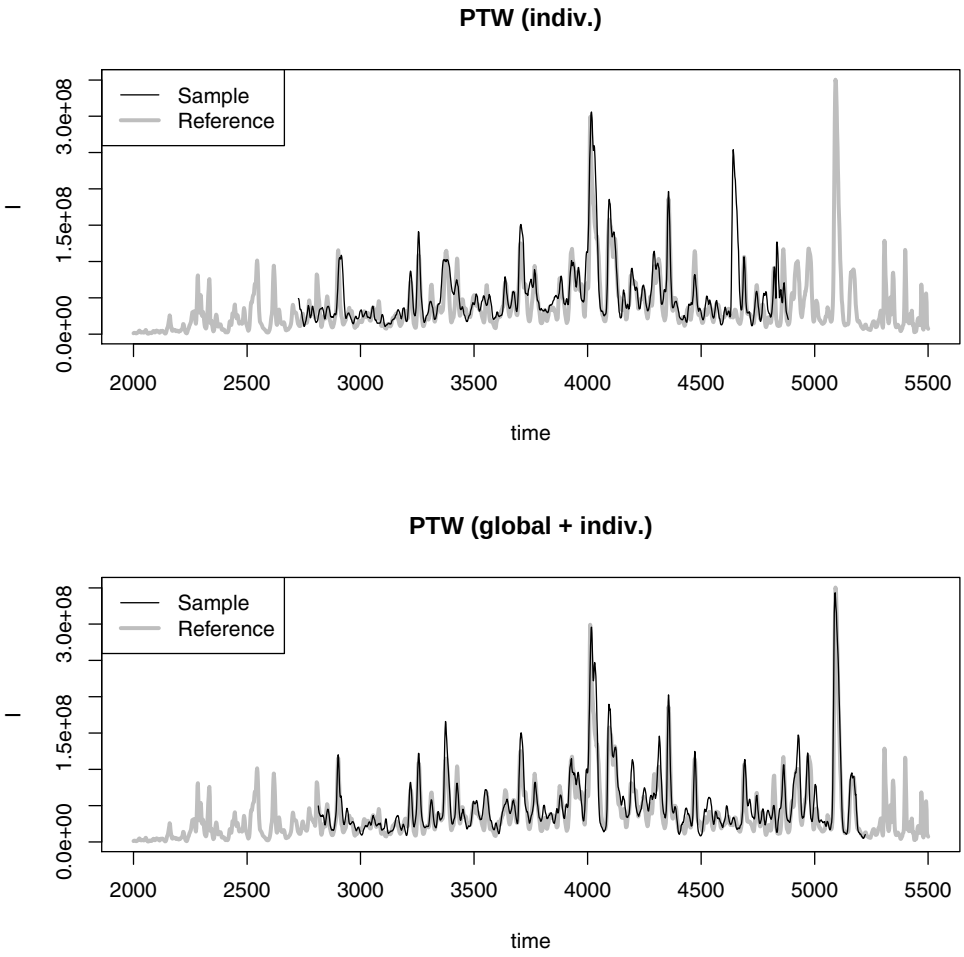

Fig. 3.10. Comparison of individual and global parametric time warping for samples two and three of the `lcms` data: the total ion current (TIC) of the aligned samples is shown. In the top panel, all mass traces have been warped individually using a warping function of degree eight – the bottom panel shows a two-stage warping using a global warping of degree two, followed by an individual warping of degree four.

```
> warpfun <- dtw(ssamp, sref)
> plot(warpfun)
> abline(-20, 1, col = "gray", lty = 2)
```

The result is shown in the left plot of Figure 3.11. A horizontal segment indicates that several points in the query signal are mapped to the same point in the reference signal; the axis of the query signal is compressed by elimination (or rather, averaging) of points. Similarly, vertical segments indicate a stretching of the query signal axis by the duplication of points. Note that these horizontal and vertical regions in the warping function of Figure 3.11 may also lead to peak shape distortions.

DTW chooses the warping function that minimizes the (weighted) distance between the warped signals:

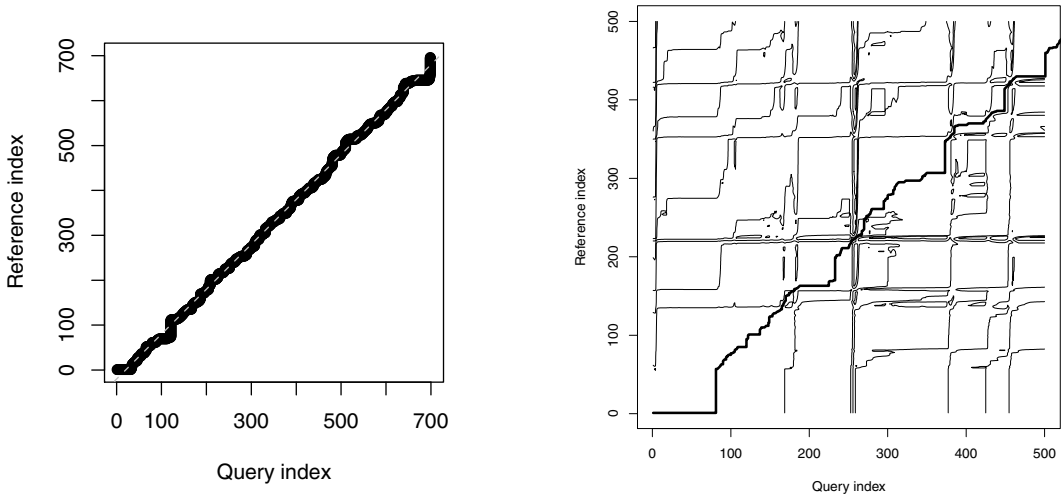

Fig. 3.11. Left plot: warping function of the data from Figure 3.7. The identity warp is indicated with a dashed gray line. Right plot: lower left corner of the warping function plot, showing the contour lines of the cost function, as well as the final warping function (fat line).

$$\sum_k \{q(m(k)) - r(n(k))\}^2 w(k) / \sum_k w(k)$$

where k is the common axis to which both the query and the reference are mapped, $m(k)$ is the warped query signal and $n(k)$ is the warped reference. Note that in this symmetric formulation there is no difference in treatment of query and reference signals: reversing the roles would lead to the same mapping. The weights are used to remove the tendency to select the shortest warping path, but should be chosen with care. The weighting scheme in the original publication [15] is for point $k + 1$:

$$w(k + 1) = m(k + 1) - m(k) + n(k + 1) - n(k)$$

That is, if both indices advance to the next point, the weight is 2; if only one of the indices advances to the next point, the weight is 1. A part of the cumulative distance from the start of both signals is shown in the right plot of Figure 3.11: the warping function finds the minimum through the (often very noisy) surface.

Obviously, such a procedure is very flexible, and indeed, one can define warping functions that put any two signals on top of each other, no matter how different they are. This is of course not what is desired, and usually several constraints are employed to keep the warping function from extreme distortions. One can, e.g., limit the maximal warping, or limit the size of individual warping steps. The `dtw` package implements these constraints and also provides the possibility to align signals of different length.

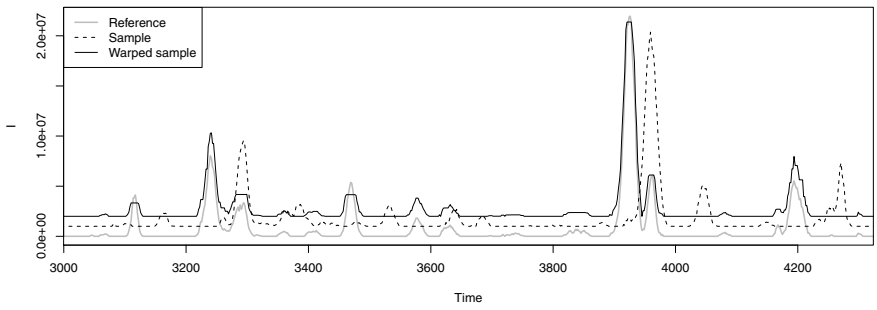

Fig. 3.12. DTW-corrected mass spectrometry data.

Once the warping function is calculated, we can use it to actually map the points in the second signal to positions corresponding to the first. For this, the `warp` function should be used, which internally performs a linear interpolation of the common axis to the original axes:

```
> wx2 <- warp(warpfun)
> plot(mz, x[,1], type = "l", xlab = "m/z", ylab = "I")
> points(mz, x[wx2,2], col = "gray")
```

The warped signal can directly be compared to the reference. The result is shown in Figure 3.12. Immediately one can see that the warping is perfect: peaks are in exactly the same positions. The only differences between the two signals are now found in areas where the peaks in the reference signal are higher than in the warped signal (e.g. m/z values at 3,100 and just below 3,300) – these peak distortions can not be corrected for.

These data are ideally suited for DTW: individual mass traces contain not too many, nicely separated peaks, so that it is clear what features should be aligned. The quality of the warping becomes clear when we align all traces individually and then compare the TIC of the warped sample with the TIC of the reference:

```
> sample2.dtw <- matrix(0, 100, 2000)
> for (i in 1:100) {
+   warpfun.dtw <- dtw(lcms[i,,3], lcms[i,,2])
+   new.indices <- warp(warpfun.dtw, index.reference = FALSE)
+   sample2.dtw[i,] <- lcms[i,new.indices,3]
+ }
```

The result is shown in the top plot of Figure 3.13 – this should be compared with the top plot in Figure 3.10. Clearly, the DTW result is much better. Note that since there is no overall compression, the length of the warped sample equals the original length. Global alignment, using one warping function for all traces simultaneously, is available using the following code:

```
> warp.dtw.gl <- dtw(t(lcms[,,3]), t(lcms[,,2]))
> samp.aligned <- lcms[,warp(warp.dtw.gl),3]
```

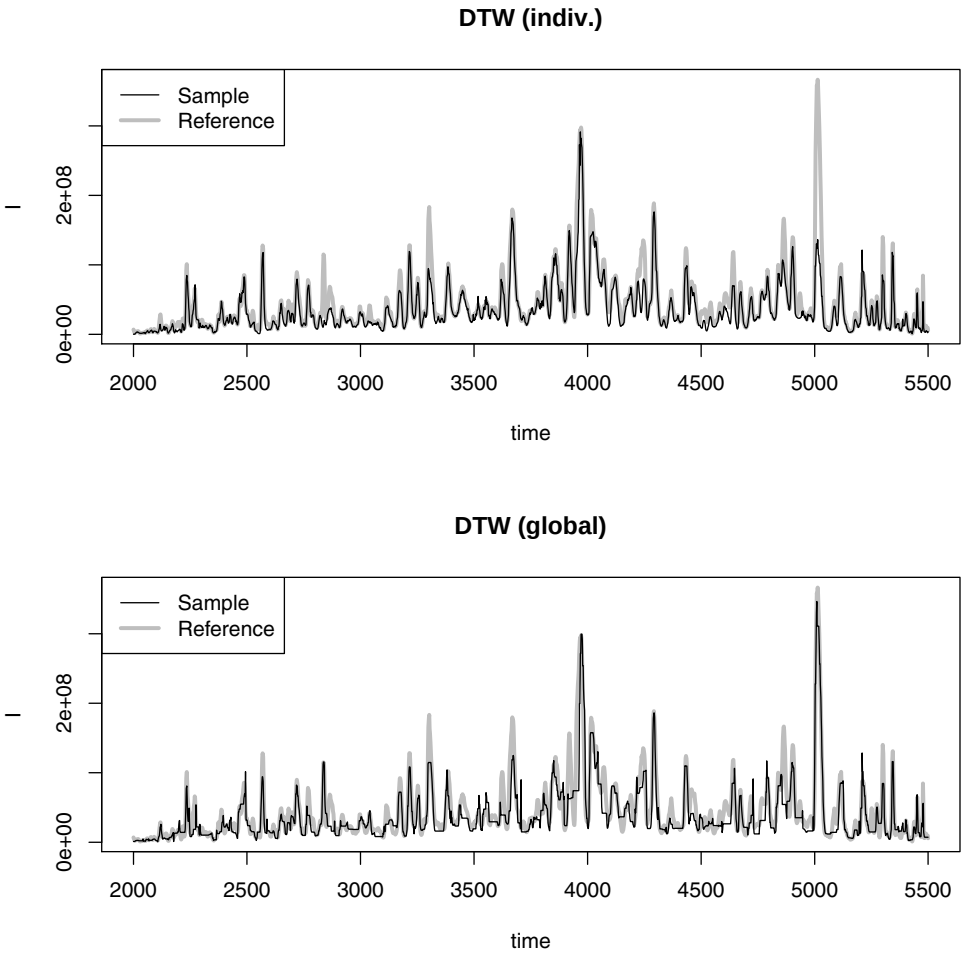

Fig. 3.13. TIC profiles of samples two and three of the `lcms` data. Top plot: DTW alignment using individual traces. Bottom plot: global DTW alignment.

The result, although still very good, is less convincing than the sum of individual DTW alignments: although the features are still aligned correctly, there are many examples of peak deformations. Global alignment with DTW is more constrained and therefore is forced to make compromises. One should be careful, however, when applying extremely flexible warping methods such as DTW to data sets in which not all peaks can be matched: in this particular example we have aligned replicate measurements. In practice, one very often will want to compare samples from different classes, and probably will want to identify those peaks that are discriminating between the classes. Alignment methods that are too flexible may be led astray by the presence of extra peaks, especially when these are of high intensity. More constrained versions of DTW, or PTW, then would probably be more appropriate.

3.3.3 Practicalities

In almost all cases, a set of signals should be aligned in such a way that all features of interest are at the same positions in every trace. One strategy is to use the column means of the data matrix as a reference. This is only possible with very small shifts and will lead to peak broadening. Simply taking a random sample from the set as a reference is better but still may be improved upon – it usually pays to perform some experiments to see which reference would lead to the smallest distortion of the other signals, while still leading to good alignment. If the number of samples is not too large, one can perform all possible combinations and see which one comes out best.

Careful data pretreatment is essential – baselines may severely influence the results and should be removed before alignment. In fact, one of the motivations of the CODA algorithm is to select traces that do not contain a baseline [22]. Another point of attention is the fact that features can have intensities differing several orders in magnitude. Often, the biggest gain in the alignment optimization is achieved by getting the prominent features in the right location. Sometimes, this dominance leads to suboptimal alignments. Also differences in intensity between sample and reference signals can distort the results. Methods to cope with these phenomena will be treated in Section 3.5. Finally, it has been shown that in some cases results can be improved when the signals are divided into segments which are aligned individually [17]. Especially with more constrained warping methods like PTW this adds flexibility, but again, there is a danger of warping too much and mapping features onto the wrong locations. Especially in cases where there may be differences between the samples (control versus diseased, for instance) there is a risk that a biomarker peak, present only in one of the two classes, is incorrectly aligned. This, again, is all the more probable when that particular peak has a high intensity.

Packages **dtw** and **ptw** are by no means alone in tackling alignment. We already mentioned the **VPdtw** package: in addition, several Bioconductor packages, such as **PROcess** and **xcms**, implement both general and more specific alignment procedures, in most cases for mass spectrometry data or hyphenated techniques like LC-MS.

3.4 Peak Picking

Several of the problems associated with misalignment can be avoided if the spectra can be transformed into lists of features, a process that is also known as peak picking. The first question of course is: what is a peak, exactly? This depends on the spectroscopic technique – usually it is a local maximum in a more or less smooth curve. In NMR, for instance, peaks usually have a specific shape (a Lorentz line shape). This knowledge can be used to fit the peaks to the data, and also to give quality assessments of the features

that are identified. In chromatography, peaks can be described by a modified normal distribution, where the modification is allowing for peak tailing and other experimental imperfections. In cases where we do not want to make assumptions about peak shape, we are forced to more crude methods, e.g., finding a list of local maxima. One simple way to do this is again to make use of the `embed` function that splits up the spectrum in many overlapping segments. For each segment, we can calculate the location of the local maximum, and eliminate those segments where the local maximum is at the beginning or at the end. A function implementing this strategy is given in the next piece of code:

```
> pick.peaks <- function(x, span) {
+   span.width <- span * 2 + 1
+   loc.max <- span.width + 1 -
+     apply(embed(x, span.width), 1, which.max)
+   loc.max[loc.max == 1 | loc.max == span.width] <- NA
+
+   pks <- loc.max + 0:(length(loc.max)-1)
+   unique(pks[!is.na(pks)])
+ }
```

The `span` parameter determines the width of the segments: wider segments will cause fewer peaks to be found. Let us investigate the effect using the prostate data from Figure 3.2:

```
> pks10 <- pick.peaks(rmeans, 10)
> plot(prostate.mz[3:498], rmeans, type = "l",
+      xlab = "m/z", ylab = "Response")
> abline(v = prostate.mz[pks10 + 2], col = "gray")
> pks40 <- pick.peaks(rmeans, 40)
> plot(prostate.mz[3:498], rmeans, type = "l",
+      xlab = "m/z", ylab = "Response", main = "span = 40")
> abline(v = prostate.mz[pks40 + 2], col = "gray")
```

This leads to the plots in Figure 3.14; with the wider span, many of the smaller features are not detected. At the same time, the many noisy features that are found with the smaller span, e.g., around m/z value 2040, probably do not consitute valid features. Clearly, the results of peak picking depend crucially on the degree and quality of the smoothing – method-specific peak picking methods will lead to superior results.

Once the positions of the features have been identified, one should quantify the signals. If an explicit peak model has been fitted, the normal approach would be to use the peak area, obtained by integrating between certain limits; if not, very often the peak height is taken. Under the assumption that peak widths are relatively constant, the two measures lead to similar results. If possible, one should then identify the signals: in, e.g., mass spectrometry this would mean the identification of the corresponding fragment ion. Having these

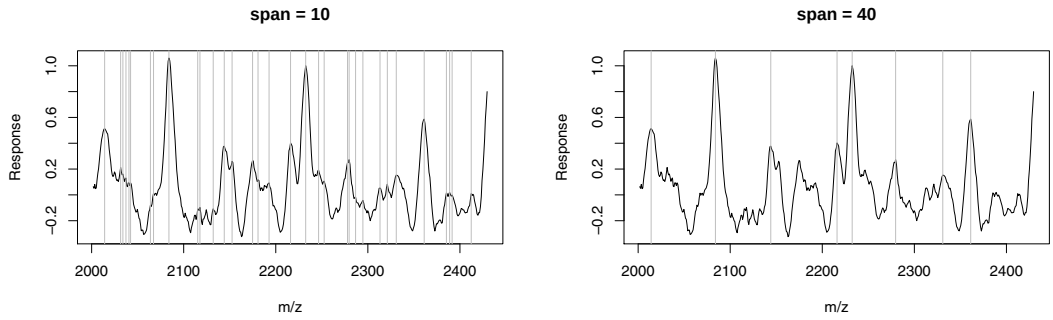

Fig. 3.14. Peak picking by identifying local maxima (prostate data): in the left figure the span is ten points, in the right forty.

assignments makes it much easier to compare spectra of different samples: even if the features are not exactly at the same position, it still is clear which signals to compare. Thus, the need for peak alignment is obviated. In practice, however, it is rare to have complete assignments of spectral data from complex samples, and an alignment step remains necessary.

3.5 Scaling

The scaling method that is employed can totally change the result of an analysis. One should therefore carefully consider what scaling method (if any) is appropriate. Scaling can serve several purposes. Many analytical methods provide data that are not on an absolute scale; the raw data in such a case are cannot be used directly when comparing different samples. If some kind of internal standard is present, it can be used to calibrate the intensities. In NMR, for instance, the TMS (tetramethylsilane, added to indicate the position of the origin on the x-axis) peak can be used for this if its concentration is known. Peak heights can then be compared directly. However, even in that situation it may be necessary to further scale intensities, since samples may contain different concentrations. A good example is the analysis of a set of urine samples by NMR. These samples will show appreciable global differences in concentrations, perhaps due to the amount of liquid the individuals have been consuming. This usually is not of interest – rather, one is interested in finding one or perhaps a couple of metabolites with concentrations that deviate from the general pattern. As an example, consider the first ten spectra of the `prostate` data:

```
> range(apply(prostate[1:10,], 1, max))

[1] 16.35960 68.89841

> range(rowSums(prostate[1:10,]))

[1]  2531.694 15207.851
```

The intensity differences within these first ten spectra are already a factor five for both statistics. If these differences are not related to the phenomenon we are interested in but are caused, e.g., by the nature of the measurements, then it is important to remove them. As stated earlier, also in cases where alignment is necessary, this type of differences between samples can hamper the analysis.

Several options exist to make peak intensities comparable over a series of spectra. The most often-used are *range scaling*, *length scaling* and *variance scaling*. In range scaling, one makes sure that the data have the same minimal and maximal values. Often, only the maximal value is considered important since for many forms of spectroscopy zero is the natural lower bound. Length scaling sets the length of each spectrum to one; variance scaling sets the variance to one. The implementation in R is easy. Here, these three methods are shown for the first ten spectra of the prostate data. Range scaling can be performed by

```
> prost10.rangesc <- sweep(prostate[1:10,], MARGIN = 1,
+                          apply(prostate[1:10,], 1, max),
+                          FUN = "/")
> apply(prost10.rangesc, 1, max)

 [1] 1 1 1 1 1 1 1 1 1 1

> range(rowSums(prost10.rangesc))

[1] 103.3278 220.7286
```

The `sweep` function is very similar to `apply` – it performs an action for every row or column of a data matrix. The `MARGIN` argument states which dimension is affected. In this case the `MARGIN = 1` indicates the rows; column-wise sweeping would be achieved with `MARGIN = 2`. The third argument is the statistic that is to be swept out, here the vector of the per-row maximal values. The final argument states how the sweeping is to be done. The default is to use subtraction; here we use division. Clearly, the differences between the spectra have decreased.

Length scaling is done by dividing each row by the square root of the sum of its squared elements:

```
> prost10.lengthsc <- sweep(prostate[1:10,], MARGIN = 1,
+                           apply(prostate[1:10,], 1,
+                                 function(x) sqrt(sum(x^2))),
+                           FUN = "/")
> range(apply(prost10.lengthsc, 1, max))

[1] 0.1107480 0.2058059

> range(rowSums(prost10.lengthsc))

[1] 18.93725 30.23589
```

The difference between the smallest and largest values is now less than a factor of two. Variance scaling has a similar effect:

```
> prost10.varsc <- sweep(prostate[1:10,], MARGIN = 1,
+                        sd(t(prostate[1:10,])),
+                        FUN = "/")
> range(apply(prost10.varsc, 1, max))

[1] 11.69716 21.65927

> range(rowSums(prost10.varsc))

[1] 1976.494 3245.692
```

In this case we use the function `sd` which returns the standard deviations of the columns of a matrix – hence the transpose.

The underlying hypothesis in these scaling methods is that the maximal intensities, or the vector lengths, or the variances, should be equal in all objects. However, there is no real way of assessing whether these assumptions are correct, and it is therefore always advisable to assess different options.

Often in statistical modelling, especially in a regression context, we are more interested in deviations from a mean value than in the values per se. These deviations can be obtained by *mean-centering*, where one subtracts the mean value from every column in the data matrix, for example with the gasoline data:

```
> NIR.mc <- t(sweep(gasoline$NIR, 2, colMeans(gasoline$NIR)))
```

Subtraction is the default operation in `sweep`, but one can also use `sweep` to perform other functions using the `FUN` argument. An even easier way to achieve the same effect is to use the `scale` function:

```
> NIR.mc <- scale(gasoline$NIR, scale = FALSE)
> matplot(wavelengths, t(NIR.mc),
+         type = "l", xlab = "Wavelength (nm)",
+         ylab = "1/R (mean-centered)", lty = 1, col = 1)
```

The result is shown in Figure 3.15; note the differences with the raw data shown in Figure 2.1 and the first derivatives in Figure 3.5.

When variables have been measured in different units or have widely different scales, we should take this into account. Obviously, one does not want the scale in which a variable is measured to have a large influence on the model: just switching to other units would lead to different results. One popular way of removing this dependence on units is called *autoscaling*, where every column $\boldsymbol{x}_i$ is replaced by

$$(\boldsymbol{x}_i - \hat{\mu}_i)/\hat{\sigma}_i$$

In statistics, this is often termed *standardization* of data; the effect is that all variables are considered equally important. This type of scaling is appropriate

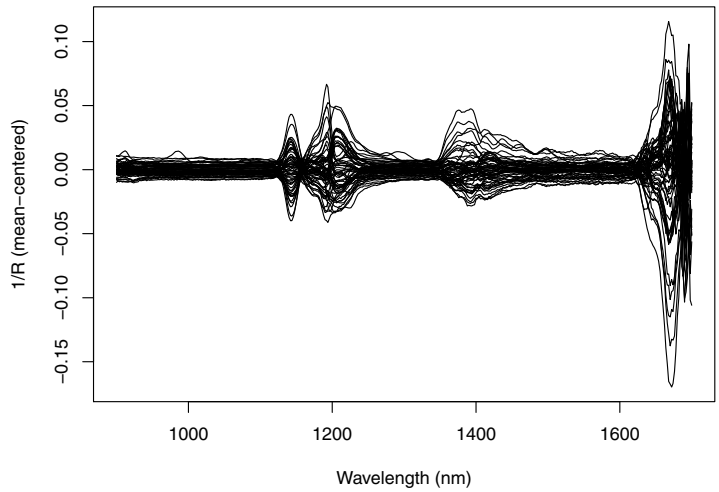

Fig. 3.15. Mean-centered gasoline NIR data.

for the wine data, since the variables have different units and very different ranges:

```
> apply(wines, 2, range)

     alcohol malic acid  ash ash alkalinity magnesium
[1,]   11.03       0.74 1.36           10.6        70
[2,]   14.83       5.80 3.23           30.0       162
     tot. phenols flavonoids non-flav. phenols proanth
[1,]         0.98       0.34              0.13    0.41
[2,]         3.88       5.08              0.66    3.58
     col. int. col. hue OD ratio proline
[1,]      1.28     0.48     1.27     278
[2,]     13.00     1.71     4.00    1680
```

The `apply` function in this case returns the range of every column. Clearly, the last variable (proline) has values that are quite a lot bigger than the other variables. The `scale` function, already mentioned, also does autoscaling: simply use the argument `scale = TRUE`, or do not mention it at all (it is the default):

```
> wines.sc <- scale(wines)
> boxplot(wines.mc ~ col(wines.mc),
+         main = "Mean-centered wine data")
> boxplot(wines.sc ~ col(wines.sc),
+         main = "Autoscaled wine data")
```

The result is shown in Figure 3.16. In the left plot, showing the mean-centered data, the dominance of proline is clearly visible, and any structure that may

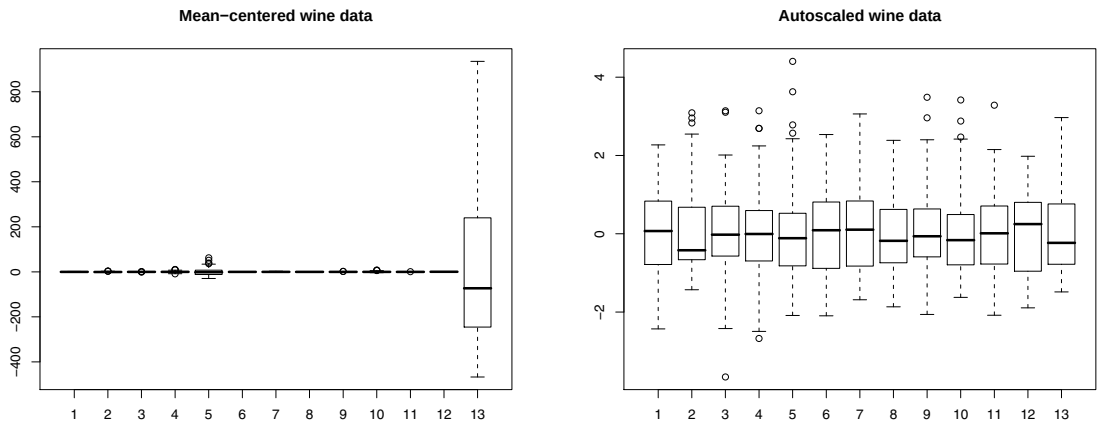

Fig. 3.16. Boxplots for the thirteen mean-centered variables (left) and auto-scaled variables (right) in the wine data set.

be present in the other variables is hard to detect. The right plot is much more informative. In almost all cases where variables indicate concentrations or amounts of chemical compounds, or represent measurements in unrelated units, autoscaling is a good idea. For the wine data, we will use always autoscaled data in examples, even in cases where scaling does not matter.

For spectral data, prevalent in the natural sciences and the life sciences, on the other hand, autoscaling is usually not recommended. Very often, the data consist of areas of high information content, *viz.* containing peaks of different intensities, and areas containing only noise. When every spectral variable is set to the same standard deviation, the noise is blown up to the same size as the signal that contains the actual information. Clearly, this is not a desirable situation, and in such cases mean-centering is much preferred.

Specialized preprocessing methods are used a lot in spectroscopy. When the total intensity in the spectra is sample-dependent, spectra should be scaled in such a way that intensities can be compared. A typical example is given by the analysis of urine spectra (of whatever type): depending on how much a person has been drinking, urine samples can be more or less diluted, and therefore there is no direct relation between peak intensities in spectra from different subjects. Apart from the range scaling method that we saw earlier, one other form of scaling is often used, especially in NIR applications: Standard Normal Variate scaling (SNV). This method essentially does autoscaling on the rows instead of the columns. That is, every spectrum will after scaling have a mean of zero and a standard deviation of 1. This gets rid of arbitrary offsets and multiplication factors. Obviously, the assumption that all spectra should have the same mean and variance is not always realistic! In some cases, the fact that the overall intensity in one spectrum is consistently higher may contain important information. In most cases, SNV gives results that are very close to MSC.

When noise is multiplicative in nature rather than additive, the level of variation depends on the signal strength. Since most noise reduction methods assume additive noise, a simple solution is to perform a log-transformation of the data. This decreases the influence of the larger features, and makes the noise more constant over the whole range. Next, regular noise reduction methods can be applied. Log transformation can also be used to reduce the dominance of the largest features, which can be disturbing the analysis, e.g., in alignment applications. A similar effect can be obtained by using *Pareto scaling*, which is the same as autoscaling with the exception that the square root of the standard deviation is used in rescaling, rather than the standard deviation itself. This form of scaling has become popular in biomarker identification applications, e.g., in metabolomics: one would like to find variables that behave differently in two populations, and one is mostly interested in those variables that have high intensities.

3.6 Missing Data

Missing data are measurements that for some reason have not led to a valid result. In spectroscopic measurements, missing data are not usually encountered, but in many other areas of science they occur frequently. The main question to be answered is: are the data missing at random? If yes, then we can probably get around the problem, provided there are not too many missing data. If no, then it means that there is some reason behind the distribution of NAs in the data set – and that means trouble. We may never be able to tell whether the missingness of some data points is not related to the process that we are studying. Therefore, in many practical applications the missing-at-random hypothesis is taken for granted.

Then, there are several strategies: one may use only those samples for which there are no missing values; one may leave out all variables containing missing values, or one can try to estimate the missing values from the other data points, a process that is known as *imputation*. Given the fact that missing values are not playing a prominent part in the analysis of spectral data in the life sciences, the main focus of this book, we will leave it at the observation that R provides ample mechanisms for dealing with missing values, which are usually represented by `NA`. Many functions have built-in capabilities of handling them. The simplest is just to remove the missing values (`na.rm = TRUE`, e.g. one of the arguments of functions like `mean`). Matrix functions like `cov` and `cor` have the `use` argument, that for instance can choose to consider only complete observations. Consult the manual pages for more information and examples.

3.7 Conclusion

Data preprocessing is an art, in most cases requiring substantial background knowledge. Because several steps are taken sequentially, the number of possible schemes is often huge. Should one scale first and then remove noise or the other way around? Individual steps will influence each other: noise removal may make it more easy to correct for a sloping baseline, but the presence of a baseline may also influence your estimate of what is noise. General recipes are hard to give, but some problems are more serious than others. The presence of peak shifts, for instance, will make any multivariate analysis very hard to interpret.

Finally, one should realise that bad data preprocessing can never be compensated for in the subsequent analysis. One should *always* inspect the data before and after preprocessing and assess whether the relevant information has been kept while disturbing signals have been removed. Of course, that is easier said than done – and probably one will go through a series of modelling cycles before one is completely satisfied with the result.

Part II

Exploratory Analysis

4

Principal Component Analysis

Principal Component Analysis (PCA, [24, 25]) is a technique which, quite literally, takes a different viewpoint of multivariate data. In fact, PCA defines new variables, consisting of linear combinations of the original ones, in such a way that the first axis is in the direction containing most variation. Every subsequent new variable is orthogonal to previous variables, but again in the direction containing most of the remaining variation. The new variables are examples of what often is called *latent variables* (LVs), and in the context of PCA they are also termed *principal components* (PCs).

The central idea is that more often than not high-dimensional data are not of full rank, implying that many of the variables are superfluous. If we look at high-resolution spectra, for example, it is immediately obvious that neighbouring wavelengths are highly correlated and contain similar information. Of course, one can try to pick only those wavelengths that appear to be informative, or at least differ from the other wavelengths in the selected set. This could, e.g., be based on clustering the variables, and selecting for each cluster one "representative". However, this approach is quite elaborate and will lead to different results when using different clustering methods and cutting criteria. Another approach is to use variable selection, given some criterion – one example is to select the limited set of variables leading to a matrix with maximal rank. Variable selection is notoriously diffucult, especially in high-dimensional cases. In practice, many more or less equivalent solutions exist, which makes the interpretation quite difficult. We will come back to variable selection methods in Chapter 10.

PCA is an alternative. It provides a direct mapping of high-dimensional data into a lower-dimensional space containing most of the information in the original data. The coordinates of the samples in the new space are called *scores*, often indicated with the symbol $\boldsymbol{T}$. The new dimensions are linear combinations of the original variables, and are called *loadings* (symbol $\boldsymbol{P}$). The term Principal Component (PC) can refer to both scores and loadings; which is meant is usually clear from the context. Thus, one can speak of

R. Wehrens, *Chemometrics with R: Multivariate Data Analysis in the Natural Sciences and Life Sciences*, Use R!, DOI 10.1007/978-3-642-17841-2_4,

sample coordinates in the space spanned by PC 1 and 2, but also of variables contributing greatly to PC 1.

The matrix multiplication of scores and loadings leads to an approximation $\tilde{\boldsymbol{X}}$ of the original data $\boldsymbol{X}$:

$$\tilde{\boldsymbol{X}} = \boldsymbol{T}_a \boldsymbol{P}_a^T \tag{4.1}$$

Superscript ^{T}, as usual, indicates the transpose of a matrix. The subscript a indicates how many components are taken into account: the largest possible number of PCs is the minimum of the number of rows and columns of the matrix:

$$a_{\max} = \min(n, p) \tag{4.2}$$

If $a = a_{\max}$, the approximation is perfect and $\tilde{\boldsymbol{X}} = \boldsymbol{X}$.

The PCs are orthogonal combinations of variables defined in such a way that [25]:

- the variances of the scores are maximal;
- the sum of the Euclidean distances between the scores is maximal;
- the reconstruction of $\boldsymbol{X}$ is as close as possible to the original: $||\boldsymbol{X} - \tilde{\boldsymbol{X}}||$ is minimal.

These three criteria are equivalent [24]; the next section will show how to find the PCs.

PCA has many advantages: it is simple, has a unique analytical solution optimizing a clear and unambiguous criterion, and often leads to a more easily interpretable data representation. The price we have to pay is that we do not have a small set of wavelengths carrying the information but a small set of principal components, in which *all* wavelengths are represented. Note that the underlying assumption is that variation equals information. Intuitively, this makes sense: one can not learn much from a constant number.

Once PCA has defined the latent variables, one can plot all samples in the data set while ignoring all higher-order PCs. Usually, only a few PCs are needed to capture a large fraction of the variance in the data set (although this is highly dependent on the type of data). That means that a plot of (the scores of) PC 1 versus PC 2 can already be highly informative. Equally useful is a plot of the contributions of the (original) variables to the important PCs. These visualizations of high-dimensional data perhaps form the most important application of PCA. Later, we will see that the scores can also be used in regression and classification problems.

4.1 The Machinery

Currently, PCA is implemented even in low-level numerical software such as spreadsheets. Nevertheless, it is good to know the basics behind the computations. In almost all cases, the algorithm used to calculate the PCs is *Singular*

Value Decomposition (SVD)[1]. It decomposes an $n \times p$ mean-centered data matrix $\boldsymbol{X}$ into three parts:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T \tag{4.3}$$

where $\boldsymbol{U}$ is a $n \times a$ orthonormal matrix containing the left singular vectors, $\boldsymbol{D}$ is a diagonal matrix $(a \times a)$ containing the singular values, and $\boldsymbol{V}$ is a $p \times a$ orthonormal matrix containing the right singular vectors. The latter are what in PCA terminology is called the loadings – the product of the first two matrices forms the scores:

$$\boldsymbol{X} = (\boldsymbol{U}\boldsymbol{D})\boldsymbol{V}^T = \boldsymbol{T}\boldsymbol{P}^T \tag{4.4}$$

The interpretation of matrices $\boldsymbol{T}$, $\boldsymbol{P}$, $\boldsymbol{U}$, $\boldsymbol{D}$ and $\boldsymbol{V}$ is straightforward. The loadings, columns in matrix $\boldsymbol{P}$ (or equivalently, the right singular vectors, columns in matrix $\boldsymbol{V}$) give the weights of the original variables in the PCs. Variables that have very low values in a specific column of $\boldsymbol{V}$ contribute only very little to that particular latent variable. The scores, columns in $\boldsymbol{T}$, constitute the coordinates in the space of the latent variables. Put differently: these are the coordinates of the samples as we see them from our new PCA viewpoint. The columns in $\boldsymbol{U}$ give the same coordinates in a normalized form – they have unit variances, whereas the columns in $\boldsymbol{T}$ have variances corresponding to the variances of each particular PC. These variances λ_i are proportional to the squares of the diagonal elements in matrix $\boldsymbol{D}$:

$$\lambda_i = d_i^2/(n-1)$$

The fraction of variance explained by PC i can therefore be expressed as

$$FV(i) = \lambda_i / \sum_{j=1}^{a} \lambda_j \tag{4.5}$$

One main problem in the application of PCA is the decision on how many PCs to retain; we will come back to this in Section 4.3.

One final remark needs to be made about the unique solution given by the SVD algorithm: it is only unique up to the sign. As is clear from, e.g., Equation 4.4, one can obtain exactly the same solution by reversing the sign of *both* scores and loadings simultaneously. There are no conventions, so one should always keep in mind that this possibility exists. For the interpretation of the data, it make no difference whatsoever.

Although SVD is a fast algorithm in some cases it can be efficient not to apply it to the data matrix directly, especially in cases where there is a large difference in the numbers of rows and columns. In such a case, it is faster

[1] One alternative for SVD is the application of the Eigen decomposition on the covariance or correlation matrix of the data. SVD is numerically more stable and is therefore preferred in most cases.

to apply SVD to either $\boldsymbol{X}^T\boldsymbol{X}$ or $\boldsymbol{X}\boldsymbol{X}^T$, whichever is the smaller one. If the number of columns is much smaller than the number of rows, one would obtain

$$\boldsymbol{X}^T\boldsymbol{X} = (\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T)^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}^T$$

Applying `svd`[2] directly yields loadings and sums of squares. Matrix $\boldsymbol{T}$, the score matrix, is easily found by right-multiplying both sides of Equation 4.4 with $\boldsymbol{P}$:

$$\boldsymbol{X}\boldsymbol{P} = \boldsymbol{T}\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{T} \tag{4.6}$$

because of the orthonormality of $\boldsymbol{P}$. Similarly, we can find the left singular vectors and singular values when applying SVD to $\boldsymbol{X}\boldsymbol{X}^T$ – see the example in the next section.

4.2 Doing It Yourself

Calculating scores and loadings is easy: consider the wine data first. We perform PCA on the autoscaled data to remove the effects of the different scales of the variables using the `svd` function provided by R

```
> wines.svd <- svd(wines.sc)
> wines.scores <- wines.svd$u %*% diag(wines.svd$d)
> wines.loadings <- wines.svd$v
```

The first two PCs represent the plane that contains most of the variance; how much exactly is given by the squares of the values on the diagonal of $\boldsymbol{D}$. The importance of individual PCs is usually given by the percentage of the overall variance that is explained:

```
> wines.vars <- wines.svd$d^2 / (nrow(wines) - 1)
> wines.totalvar <- sum(wines.vars)
> wines.relvars <- wines.vars / wines.totalvar
> variances <- 100 * round(wines.relvars, digits = 3)
> variances[1:5]

[1] 36.0 19.2 11.2  7.1  6.6
```

The first PC covers more than one third of the total variance; for the fifth PC this amount is down to one fifteenth.

The scores show the positions of the individual wine samples in the coordinate system of the PCs. A score plot can be produced as follows:

```
> plot(wines.scores[,1:2], type = "n",
+      xlab = paste("PC 1 (", variances[1], "%)", sep = ""),
+      ylab = paste("PC 2 (", variances[2], "%)", sep = ""))
> abline(h = 0, v = 0, col = "gray")
> points(wines.scores[,1:2], pch = wine.classes)
```

[2] Or `eigen`, which returns eigenvectors and eigenvalues.

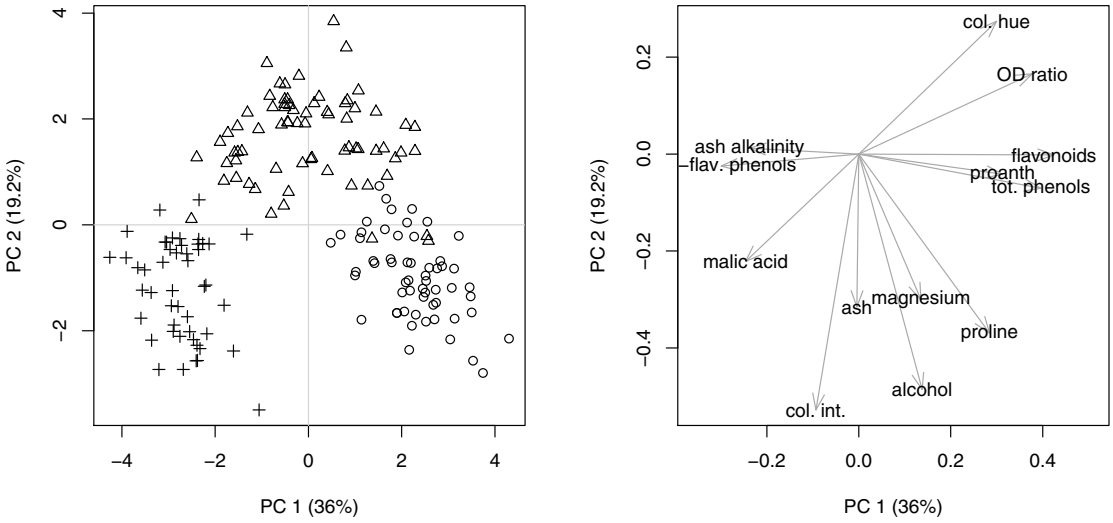

Fig. 4.1. Left plot: scores on PCs 1 and 2 for the autoscaled wine data. Different symbols correspond to the three cultivars. Right plot: loadings on PCs 1 and 2.

The result is depicted in the left plot of Figure 4.1. It is good practice to indicate the amount of variance associated with each PC on the axis labels. The three cultivars can be clearly distinguished: class 1, Barbera, indicated with open circles, has the smallest scores on PC 1 and class 2 (Barolo – plusses in the figure) the largest. PC 2, corresponding to 19% of the variance, improves the separation by separating the Grignolinos, the middle class on PC 1, from the other two. Note that this is a happy coincidence: PCA does not explicitly look to discriminate between classes. In this case, the three cultivars clearly have different characteristics. What characteristics these are can be seen in the loading plot, shown on the right in Figure4.1. It shows the contribution of the original variables to the PCs. Loadings are traditionally shown as arrows from the origin:

```
> plot(wines.loadings[,1] * 1.2, wines.loadings[,2], type = "n",
+      xlab = paste("PC 1 (", variances[1], "%)", sep = ""),
+      ylab = paste("PC 2 (", variances[2], "%)", sep = ""))
> arrows(0, 0, wines.loadings[,1], wines.loadings[,2],
+      col = "darkgray", length = .15, angle = 20)
> text(wines.loadings[,1:2], labels = colnames(wines))
```

The factor of 1.2 in the plot command is used to create space for the text labels. Clearly, the wines of class 3 are distinguished by lower values of alcohol and a lower color intensity. Wines of class 1 have high flavonoid and phenol content and are low in non-flavonoid phenols; the reverse is true for wines of class 2. All of these conclusions could probably have been drawn also by looking at class-specific boxplots for all variables – however, the combination of one

score plot and one loading plot shows this in a much simpler way, and even presents direct information on correlations between variables and objects. We will come back on this point later, when treating biplots.

As an example of the kind of speed improvement one can expect when applying SVD on the crossproduct matrices rather than the original data, consider the `prostate` data. Timings can be obtained by wrapping the code within a `system.time` call:

```
> system.time({
+   prost.svd <- svd(prostate)
+   prost.scores <- prost.svd$u %*% diag(prost.svd$d)
+   prost.variances <- prost.svd$d^2 / (nrow(prostate) - 1)
+   prost.loadings <- prost.svd$v
+   })

   user  system elapsed
 30.706   0.264  30.988
```

Here, the number of variables is much larger than the number of objects (which, by the way, is not extremely small either), so we perform SVD on the matrix $\boldsymbol{X}\boldsymbol{X}^T$:

```
> system.time({
+   prost.tcp <- tcrossprod(prostate)
+   prost.svd <- svd(prost.tcp)
+   prost.scores <- prost.svd$u %*% diag(sqrt(prost.svd$d))
+   prost.variances <- prost.svd$d / (nrow(prostate) - 1)
+   prost.loadings <- solve(prost.scores, prostate)
+   })

   user  system elapsed
 16.085   0.132  16.216
```

The second option is twice as fast.

4.3 Choosing the Number of PCs

The question how many PCs to consider, or put differently: where the information stops and the noise begins, is difficult to answer. Many methods consider the amount of variance explained, and use statistical tests or graphical methods to define which PCs to include. In this section we briefly review some of the more popular methods.

The amount of variance per PC is usually depicted in a scree plot: either the variances themselves or the logarithms of the variances are shown as bars. Often, one also considers the fraction of the total variance explained by every single PC. The last few PCs usually contain no information and, especially on

a log scale, tend to make the scree plot less interpretable, so they are usually not taken into account in the plot.

```
> par(mfrow = c(2,2))
> barplot(wines.vars[1:10], main = "Variances",
>         names.arg = paste("PC", 1:10))
> barplot(log(wines.variances[1:10]), main = "log(Variances)",
>         names.arg = paste("PC", 1:10))
> barplot(wines.relvars[1:10], main = "Relative variances",
>         names.arg = paste("PC", 1:10))
> barplot(cumsum(100 * wines.relvars[1:10]),
>         main = "Cumulative variances (%)",
>         names.arg = paste("PC", 1:10), ylim = c(0, 100))
```

This leads to the plots in Figure 4.2. Clearly, PCs 1 and 2 explain much more variance than the others: together they cover 55% of the variance. The scree plots show no clear cut-off, which in real life is the rule rather than the exception. Depending on the goal of the investigation, for these data one could consider three or five PCs. Choosing four PCs would not make much sense in this case, since the fifth PC would explain almost the same amount of variance: if the fourth is included, the fifth should be, too.

4.3.1 Statistical Tests

One can show that the explained variance for the i-th component, $\lambda_i = d_i^2/(n-1)$ is asymptotically normally distributed [26, 27], which leads to confidence intervals of the form

$$\ln(\lambda_i) \pm z_\alpha \sqrt{\frac{2}{n-1}}$$

For the wine example, 95% confidence intervals can therefore be obtained as

```
> llambdas <- log(wines.vars)
> CIwidth <- qnorm(.975) * sqrt(2 / (nrow(wines) - 1))
> CIs <- cbind(exp(llambdas - CIwidth),
+              wines.vars,
+              exp(llambdas + CIwidth))
> colnames(CIs) <- c("CI 0.025", "  Estimate", "  CI 0.975")
> CIs[1:5,]
```

```
      CI (0.025)   Estimate   CI (0.975)
PC 1      3.7958      4.678        5.765
PC 2      2.0297      2.501        3.083
PC 3      1.1793      1.453        1.791
PC 4      0.7501      0.924        1.139
PC 5      0.6993      0.862        1.062
```

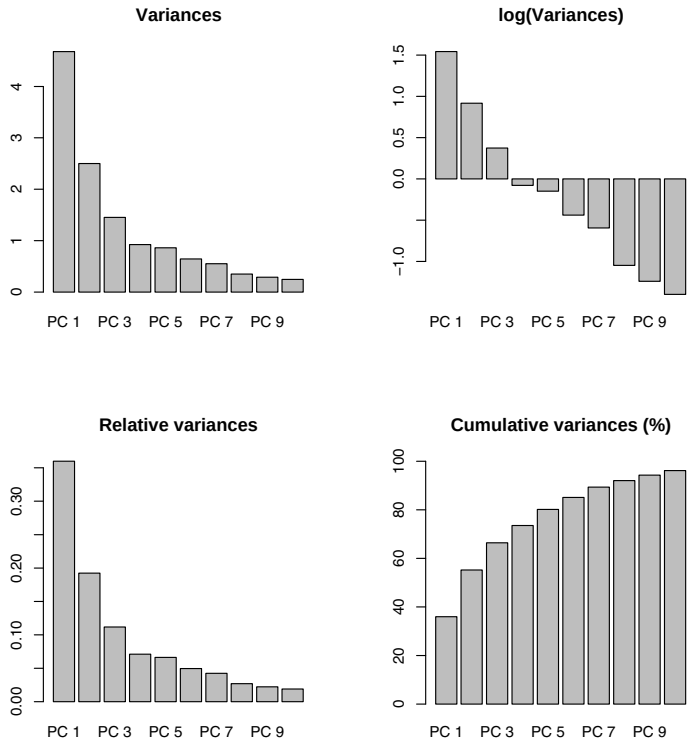

Fig. 4.2. Scree plots for the assessment of the amount of variance explained by each PC. From left to right, top to bottom: variances, logarithms of variances, fractions of the total variance and cumulative percentage of total variance.

Mardia *et al.* present an approach testing the equality of variances for individual PCs [26]. For the autoscaled wine data, one could test whether the last $p-k$ PCs are equally important, i.e. have equal values of λ. The quantity

$$(n-1)(p-k)\log(a_0/g_0)$$

is distributed approximately as a χ^2-statistic with $(p-k+2)(p-k-1)/2$ degrees of freedom [26, p. 236]. In this formula, a_0 and g_0 indicate arithmetic and geometric means of the $p-k$ smallest variances, respectively. We can use this test to assess whether the last three PCs are useful or not:

```
> small.ones <- wines.vars[11:13]
> n <- nrow(wines)
> nsmall <- length(small.ones)
> geo.mean <- prod(small.ones)^{1/nsmall}
> mychisq <- (n - 1) * nsmall * log(mean(small.ones) / geo.mean)
> ndf <- (nsmall + 2) * (nsmall - 1) / 2
> 1 - pchisq(mychisq, ndf)

[1] 9.91e-05
```

This test finds that after PC 10 there is still a difference in the variances. In fact, the test finds a difference after any other cutoff, too: apparently all PCs are significant.

The use of statistical tests for determining the optimal number of PCs has never really caught on. Most scientists are prepared to accept a certain loss of information (variance) provided that the results, the score plots and loading plots, help to answer scientific questions. Most often, one uses informal graphical methods: if an elbow shows up in the scree plot that can be used to decide on the number of PCs. In other applications, notably with spectral data, one can sometimes check the loadings for structure. If there is no more structure – in the form of peak-like shapes – present in the loadings of higher PCs, they can safely be discarded.

4.4 Projections

Once we have our low-dimensional view of the original high-dimensional data, we may be interested how other data are positioned. This may be data from new samples, measured a different instrument, or at a different day. The key point is that the low-dimensional representation allows us to look at the data and in one glance assess whether there are patterns. Obtaining scores for new data is pretty easy. Given a new data matrix $\boldsymbol{X}$, the projections in the space defined by loadings $\boldsymbol{P}$ can be obtained by simple right-multiplication:

$$\boldsymbol{X}\boldsymbol{P} = \boldsymbol{T}\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{T}$$

The wine data, for example, are more or less ordered according to vintage. If we would perform PCA on the first half of the data, the third class, the Grignolino wines, would not play a part in defining the PCs, and the first class, Barbera, would dominate. To see the effect of this, we can project the second half of the data matrix into the PCA space defined by the first half. We start by constructing scores and loadings:

```
> X1 <- scale(wines[1:88,])
> X1.svd <- svd(X1)
> X1.pca <- list(scores = X1.svd$u %*% diag(X1.svd$d),
+                loadings = X1.svd$v)
```

Then, we scale the second half of the data using the means and standard deviations of the first half. This is a very important detail that sometimes is missed – obviously, both halves should have the same point of origin. Using the scaled second half of the data, we calculated the corresponding scores, and show them in a plot, together with the scores of the first half:

```
> X2 <- scale(wines[89:177,],
+             center = attr(X1, "scaled:center"),
+             scale = attr(X1, "scaled:scale"))
```

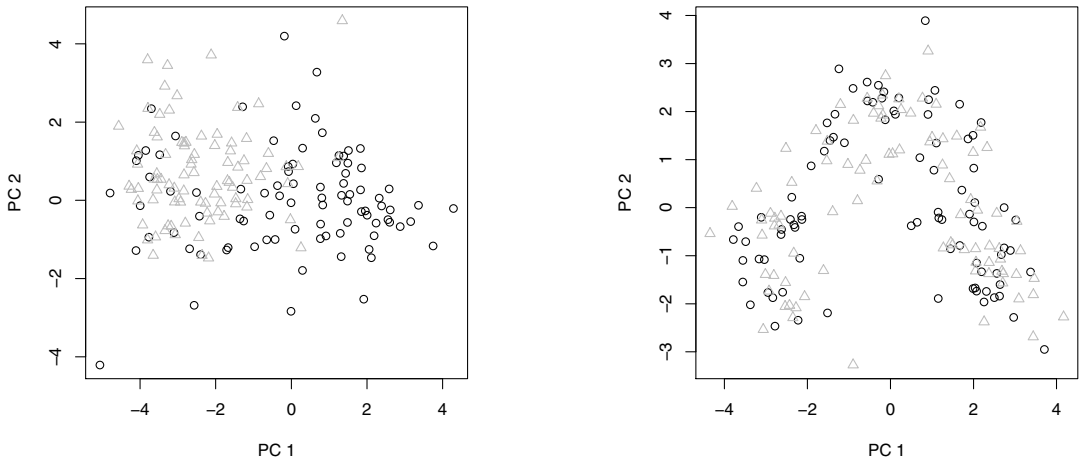

Fig. 4.3. Projections in PCA space. Left plot: second half of the wine data (triangles) projected into the PCA space defined by the first half (circles). Right plot: PCA model based on the odd rows (circles). The even rows (triangles) are projected in this space. The result is very similar to a PCA on the complete data matrix.

```
> X2.scores <- X2 %*% X1.pca$loadings
> plot(rbind(X1.pca$scores, X2.scores),
+      pch = rep(c(1,2), c(88, 89)),
+      col = rep(c(1,2), c(88, 89)),
+      xlab = "PC 1", ylab = "PC 2")
```

This leads to the left plot in Figure 4.3. Clearly, the familiar shape of the PC 1 vs. PC 2 plot has been destroyed, and what is more: the triangles, corresponding to the second half of the data, are generally in a different location than the first half (circles). Since Grignolino wines are only present in the second half, and Barbera wines only in the first half, this comes as no surprise. The right plot in the same figure shows a completely different picture. It has been obtained by defining `X1` and `X2` as follows:

```
> odd <- seq(1, nrow(wines), by = 2)
> even <- seq(2, nrow(wines), by = 2)
> X1 <- scale(wines[odd,])
> X2 <- scale(wines[even,],
+             center = attr(X1, "scaled:center"),
+             scale = attr(X1, "scaled:scale"))
```

Now both halves have similar compositions, and the data clouds neatly overlap. In fact, this is a very important way to check whether a division in training and test set, a topic that we will talk about extensively in later chapters, is a good one.

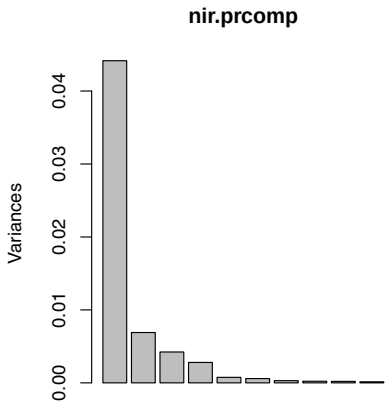

Fig. 4.4. Scree plot for the NIR data. By default, the scree plot shows not more than ten PCs.

4.5 R Functions for PCA

The standard R function for PCA is `prcomp`. By default, the data are mean-centered (but not scaled!). We will show its use on the gasoline data:

```
> nir.prcomp <- prcomp(gasoline$NIR)
> summary(nir.prcomp, digits = 2)

Importance of components:
                          PC1   PC2    PC3    PC4    PC5     PC6
Standard deviation     0.210 0.083 0.0651 0.0529 0.0275 0.02426
Proportion of Variance 0.726 0.113 0.0695 0.0460 0.0124 0.00967
Cumulative Proportion  0.726 0.839 0.9086 0.9546 0.9670 0.97664
```

The output of the `summary` command is limited here to only six PCs. One can see that they explain almost 98% of the variance. The default `plot` command is to show a scree plot (in fact, the `screeplot` function is used):

```
> plot(nir.prcomp)
```

The result is depicted in Figure 4.4. Probably, most people would select four components to be included: although the first is much larger than the others, components two to four still contribute a substantial amount, whereas higher components are much less important.

Plotting the loadings of the first four PCs shows some interesting structure. The are available as the `rotation` element of the `prcomp` object:

```
> nir.loadings <- nir.prcomp$rotation[,1:4]
```

The original variables are, of course, highly correlated, and therefore connecting subsequent variables in the loading space forms a trajectory. In the following code, producing the plots in Figure 4.5, the variables with the most extreme loadings have been indicated – these can easily be found using the `identify` function (see the manual page for details).

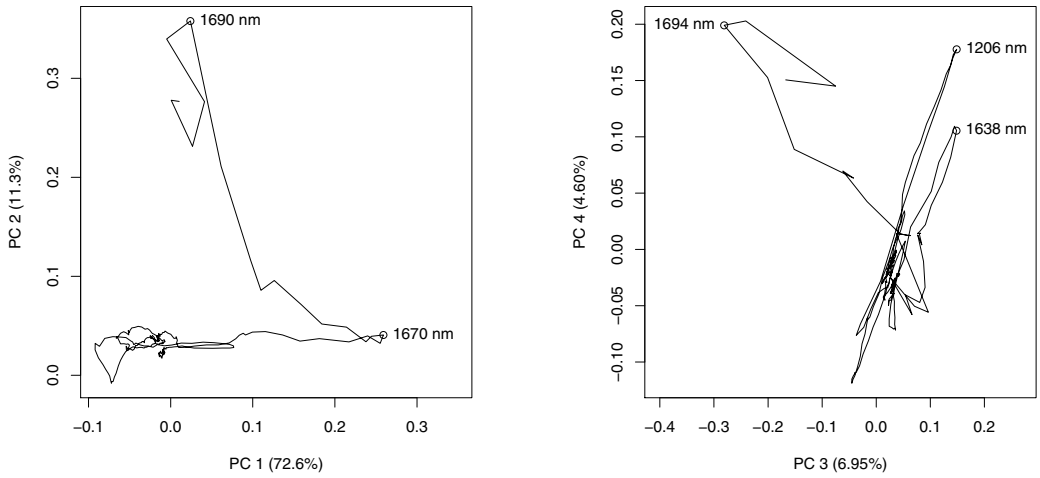

Fig. 4.5. Loading plots for the mean-centered gasoline data; the left plot shows PC 1 versus PC 2, and the right plot PC 3 versus PC 4. Some extreme variable weights are indicated with the corresponding wavelengths.

```
> par(mfrow = c(1,2), pty = "s")
> offset <- c(0, 0.09) # to create space for labels
> plot(nir.loadings[,1:2], type = "l",
+      xlim = range(nir.loadings[,1]) + offset,
+      xlab = "PC 1 (72.6%)", ylab = "PC 2 (11.3%)")
> points(nir.loadings[c(386, 396), 1:2])
> text(nir.loadings[c(386, 396), 1:2], pos = 4,
+      labels = paste(c(1670, 1690), "nm"))
```

The procedure for PCs three and four is completely analogous:

```
> offset <- c(-0.12, 0.12) # to create space for labels
> plot(nir.loadings[,3:4], type = "l",
+      xlim = range(nir.loadings[,3]) + offset,
+      xlab = "PC 3 (6.95%)", ylab = "PC 4 (4.60%)")
> points(nir.loadings[c(154, 370, 398), 3:4])
> text(nir.loadings[c(154, 370), 3:4], pos = 4,
+      labels = paste(c(1206, 1638), "nm"))
> text(nir.loadings[398, 3:4, drop = FALSE],
+      labels = "1694 nm", pos = 2)
```

We can see that the variation on PC 1 is mainly attributable to the intensities around 1670 nm, whereas the wavelengths around 1690 nm are contributing most to PC 2. PC 3 shows the largest loadings at 1638, 1694 and 1206 nm; the latter two are also extreme loadings on PC 4. These wavelengths correspond with areas of significant variation (see Figure 2.1).

An even more interesting visualization is the *biplot* [28, 29]. This shows scores and loadings in one and the same plot, which can makes interpretation

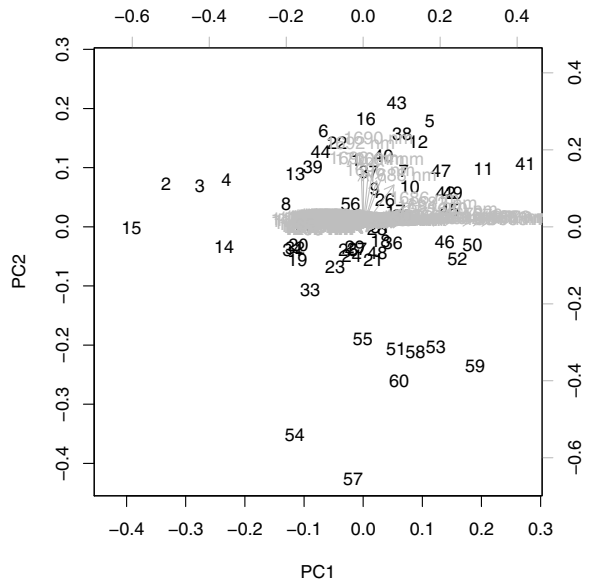

Fig. 4.6. Biplot for the mean-centered gasoline data showing PC 1 versus PC 2.

easier. The origins for the score and loading plots are overlayed, and the two sets of points are plotted in separate axis systems. An example is shown in Figure 4.6:

```
> biplot(nir.prcomp)
```

Again, scores are plotted as individual points, and loadings as arrows. The axes corresponding to the scores are shown on the bottom and the left of the picture; the axis to the right and no top are for the loadings. Since there are many correlated variables, the loading plot looks quite crowded.

We can see in the biplot that wavelength 1690 nm is one of the arrows pointing upward, in the direction of PC 2. Samples 15 and 41 are most different in PC 1, and samples 57 and 43 differ most in PC 2. Let us plot the mean-centered data for these four samples (since the mean-centered data are what is fed to PCA):

```
> extremes <- c(15,41,45,57)
> Xextr <- scale(gasoline$NIR, scale = FALSE)[extremes,]
> wavelengths <- seq(900, 1700, by = 2)
> matplot(wavelengths, t(Xextr),
+         type = "l", xlab = "Wavelength (nm)",
+         ylab = "Intensity (mean-scaled)", lty = c(1,1,2,2),
+         col = c(1,"gray",1,"gray"))
> abline(v = wavelengths[extreme.vars], col = "gray", lty = 3)
> legend("bottomleft", legend = paste("sample", extremes),
+        lty = c(1,1,2,2), col = c(1,2,1,2), bty = "n")
```

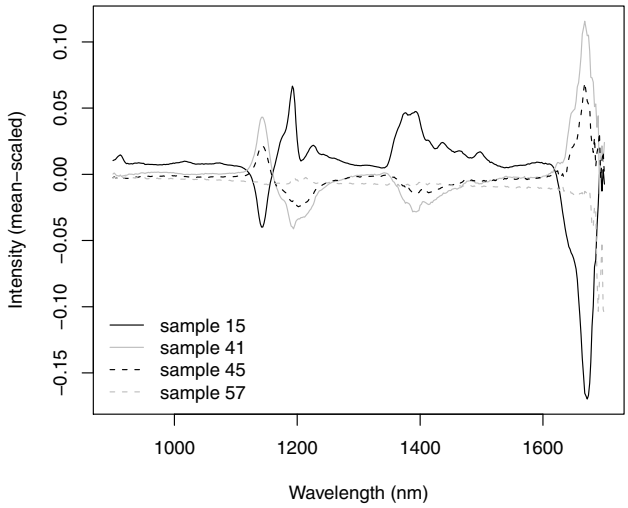

Fig. 4.7. Mean-centered spectra of samples that are extreme in either component.

This results in Figure 4.7. The largest difference in intensity, at 1670 nm, corresponds with the most important variables in PC 1 – and although it is not directly recognizable from the loadings in this figure, this is exactly the wavelength with the largest loading on PC 1. The largest difference in the samples that are extreme in PC 2 is just above that, at 1690 nm, in agreement with our earlier observation. Another major feature just below 1200 nm is represented in the bottom left corner of the loading plot of PC 1 vs PC 2.

Another R function, `princomp`, also does PCA, using the `eigen` decomposition on the covariance matrix instead of directly performing `svd` on the data themselves. Since the `svd`-based calculations are more stable, these are to be preferred for regular applications; `princomp` does allow you to provide a specific covariance matrix that will be used for the decomposition, which can be useful in some situations (see Section 11.1), but is retained mainly for compatibility reasons.

The package **ChemometricsWithR** comes with a set of PCA functions, too, based on the code presented in this chapter. The basic function is `PCA`, and the usual generic functions `print`, `plot`, and `summary` are available, as well as some auxiliary functions such as `screeplot`, `project`, and the extraction functions `variances`, `loadings` and `scores`. To produce, e.g., plots very similar to the ones shown in Figure 4.1, for example, one can issue:

```
> wines.PCA <- PCA(scale(wines))
> scoreplot(wines.PCA, pch = wine.classes, col = wine.classes)
> loadingplot(wines.PCA, show.names = TRUE)
```

One useful feature of these functions is that the percentage of explained variance is automatically shown at the axis labels.

4.6 Related Methods

PCA is not alone in its aim to find low-dimensional representations of high-dimensional data sets. Several other methods try to do the same thing, but rather than finding the projection that maximizes the explained variance, they choose other criteria. In Principal Coordinate Analysis and the related multidimensional scaling methods, the aim is to find a low-dimensional projection that reproduces the experimentally found *distances* between the data points. When these distances are Euclidean, the results are the same or very similar to PCA results; however, other distances can be used as well. Independent Component Analysis maximizes deviations from normality rather than variance, and Factor Analysis concentrates on reproducing covariances. We will briefly review both in the next paragraphs.

4.6.1 Multidimensional Scaling

In some cases, applying PCA to the raw data matrix is not appropriate, for example in situations where regular Euclidean distances do not apply – similarities between chemical structures, e.g., can be expressed easily in several different ways, but it is not at all clear how to represent molecules into fixed-length structure descriptors [30], something that is required by distance measures such as the Euclidean distance. Even when comparing spectra or chromatograms, the Euclidean distance can be inappropriate, for instance in the presence of peak shifts [20,9]. In other cases, raw data are simply not available and the only information one has consists of similarities. Based on the sample similarities, the goal of methods like Multidimensional Scaling (MDS, [31,32]) is to reconstruct a low-dimensional map of samples that leads to the same similarity matrix as the original data (or a very close approximation). Since visualization usually is one of the main aims, the number of dimensions usually is set to two, but in principle one could find an optimal configuration with other dimensionalities as well.

The problem is something like making a topographical map, given only the distances between the cities in the country. In this case, an exact solution is possible in two dimensions since the original distance matrix was calculated from two-dimensional coordinates. Note that although distances can be reproduced exactly, the map still has rotational and translational freedom – in practice this does not pose any problems, however. An amusing example is given by maps not based on kilometers but rather on travel time – the main cities will be moved to the center of the plot since they usually are connected by high-speed trains, whereas smaller villages will appear to be further away. In such a case, and in virtually all practical applications, a two-dimensional plot will not be able to reproduce all similarities exactly.

In MDS, there are several ways to indicate the agreement between the two distance matrices, and these lead to different methods. The simplest approach

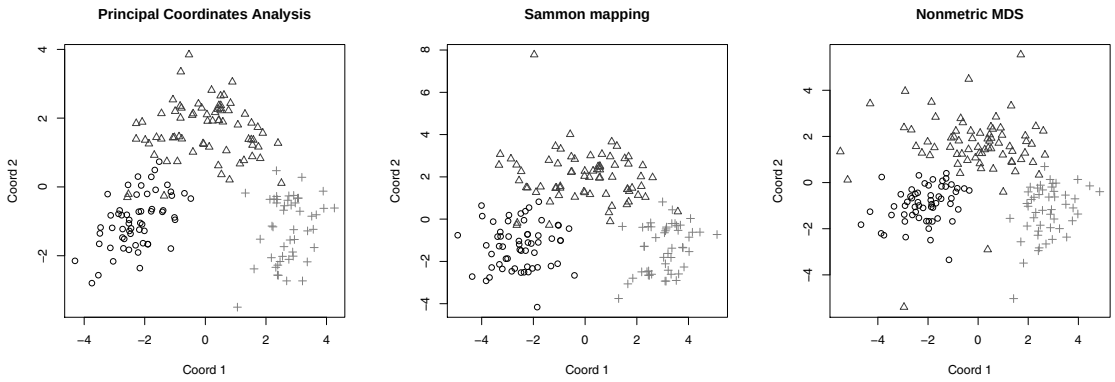

Fig. 4.8. Multidimensional scaling approaches on the wine data: classical MDS (left), Sammon mapping (middle) and non-metric MDS (right).

is to perform PCA on the double-centered distance matrix[3], an approach that is known as Principal Coordinate Analysis, or Classical MDS [33]. The criterion to be minimized is called the stress, and is given by

$$S = \sum_{j<i}(||x_i - x_j|| - e_{ij})^2 = \sum_{j<i}(d_{ij} - e_{ij})^2$$

where e_{ij} corresponds with the true, given, distances, and d_{ij} are the distances between objects x_i and x_j in the low-dimensional space.

In R, this is available as the function `cmdscale`:

```
> wines.dist <- dist(scale(wines))
> wines.cmdscale <- cmdscale(wines.dist)
> plot(wines.cmdscale$points,
+      pch = wine.classes, col = wine.classes,
+      main = "Principal Coordinate Analysis",
+      xlab = "Coord 1", ylab = "Coord 2")
```

This leads to the left plot in Figure 4.8, which up to the reversal of the sign in the first component is exactly equal to the scores plot of Figure 4.1.

Other approaches optimize slightly different criteria: two well-known examples are Sammon mapping and Kruskal-Wallis mapping [34] – both are available in the **MASS** package as functions `sammon` and isoMDS, respectively. Sammon mapping decreases the influence of large distances, which can dominate the map completely. It minimizes the following stress criterion:

$$S = \frac{1}{\sum_i \sum_{j<i} d_{ij}} \sum_i \sum_{j<i} \frac{d_{ij} - e_{ij}}{d_{ij}}$$

[3] Double centering is performed by mean-centering in both row and column dimensions, and subsequently adding the grand mean of the original matrix to center the data around the origin.

Since no analytical solution is available, gradient descent optimization is employed to find the optimum. The starting point usually is the classical solution, but one can also provide another configuration – indeed, one approach to try to avoid local optima is to repeat the mapping starting from many different starting sets, or to use different sets of search parameters. The wine data are a case in point: whereas the default initial step size of .2 generates exactly the same solution as the PCA, tinkering with this parameter leads to a substantial improvement:

```
> wines.sammon <- sammon(wines.dist)

Initial stress        : 0.14753
stress after   0 iters: 0.14753

> wines.sammon <- sammon(wines.dist, magic = .00003)

Initial stress        : 0.14753
stress after  10 iters: 0.14347, magic = 0.002
stress after  19 iters: 0.11250

> plot(wines.sammon$points, main = "Sammon mapping",
+      col = wine.classes, pch = wine.classes,
+      xlab = "Coord 1", ylab = "Coord 2")
```

Note that the function `sammon` returns a list rather than a coordinate matrix: the coordinates in low-dimensional space can be accessed in list element `points`.

The non-metric scaling implemented in `isoMDS` uses a two-step optimization that alternatively finds a good configuration in low-dimensional space, and an appropriate non-monotone transformation. In effect, one finds a set of points that leads to the same *order* of the distances in the low-dimensional approximation and in the real data, rather than resulting in approximately the same distances.

```
> wines.isoMDS <- isoMDS(wines.dist)

initial  value 25.980817
iter   5 value 19.977649
iter  10 value 17.798597
iter  15 value 17.327216
iter  20 value 17.128918
iter  20 value 17.116929
iter  20 value 17.111706
final  value 17.111706
converged

> plot(wines.isoMDS$points, main = "Non-metric MDS",
+      col = wine.classes, pch = wine.classes,
+      xlab = "Coord 1", ylab = "Coord 2")
```

The results of Sammon mapping and IsoMDS are shown in the middle and right plots of Figure 4.8, respectively. Here we again see the familiar horse-shoe shape, although it is somewhat different in nature in the non-metric version in the plot on the right. There, the Grignolino class is much more spread out than the other two.

MDS is popular in the social sciences, but much less so in the life sciences: maybe there its disadvantages are more important. The first drawback is that MDS requires a full distance matrix. For data sets with many thousands of samples this can be prohibitive in terms of computer memory. The other side of the coin is that the (nowadays more common) data sets with many more samples than variables do not present any problem; they can be analysed by MDS easily. The second and probably more important disadvantage is that MDS does not provide an explicit mapping operator. That means that new objects cannot be projected into the lower-dimensional point configuration as we did before with PCA; either one redoes the MDS mapping, or one positions the new samples as good as possible within the space of the mapped ones and takes several iterative steps to obtain a new, complete, set of low-dimensional coordinates. Finally, the fact that the techniques rely on optimization, rather than an analytical solution, is a disadvantage: not only does it take more time, especially with larger data sets, but also the optimization settings may need to be tweaked for optimal performance.

4.6.2 Independent Component Analysis and Projection Pursuit

Variation in many cases equals information, one of the reasons behind the widespread application of PCA. Or, to put it the other way around, a variable that has a constant value does not provide much information. However, there are many examples where the *relevant* information is hidden in small differences, and is easily overwhelmed by other sources of variation that are of no interest. The technique of *Projection Pursuit* [35,36,37] is a generalization of PCA where a number of different criteria can be optimized. One can for instance choose a viewpoint that maximises some grouping in the data. In general, however, there is no analytical solution for any of these criteria, except for the variance criterium used in PCA. A special case of Projection Pursuit is *Independent Component Analysis* (ICA) [38], where the view is taken to maximise deviation from multivariate normality, given by the negentropy J. This is the difference of the entropy of a normally distributed random variable $H(x_{\mathrm{G}})$ and the entropy of the variable under consideration $H(x)$

$$J(x) = H(x_{\mathrm{G}}) - H(x)$$

where the entropy itself is given by

$$H(x) = -\int f(x) \log f(x) dx$$

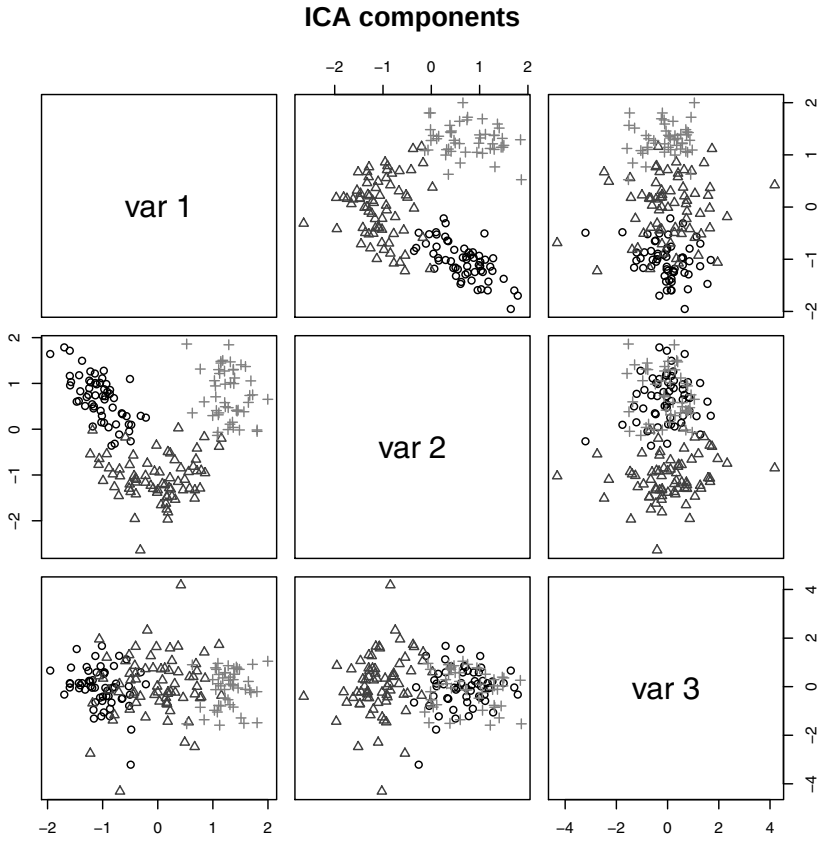

Fig. 4.9. Fast ICA applied to the wine data, based on a three-component model.

Since the entropy of a normally distributed variable is maximal, the negentropy is always positive [39]. Unfortunately, this quantity is hard to calculate, and in practice approximations, such as kurtosis and the fourth moment are used. Package **fastICA** provides the fastICA procedure of Hyvarinen and Oja [40], employing other approximations that are more robust and faster. The algorithm can be executed either fully in R, or using C code for greater speed.

The fastICA algorithm first mean-centers the data, and then performs a "whitening", *viz.* a principal component analysis. These principal components are then rotated in order to optimize the non-gaussianity criterion. Applied to the wine data, this gives the result shown in Figure 4.9:

```
> wines.ica <- fastICA(wines.sc, 3)
> pairs(wines.ica$S, main = "ICA components",
+       col = wine.classes, pch = wine.classes)
```

It is interesting to note that the first ICA component in this case is not related to the difference in variety. Instead, it is the plot of IC 2 versus IC 3 that shows most discrimination, and is most similar to the PCA score plot shown in Figure 4.1. One characteristic that should be noted is that ICA

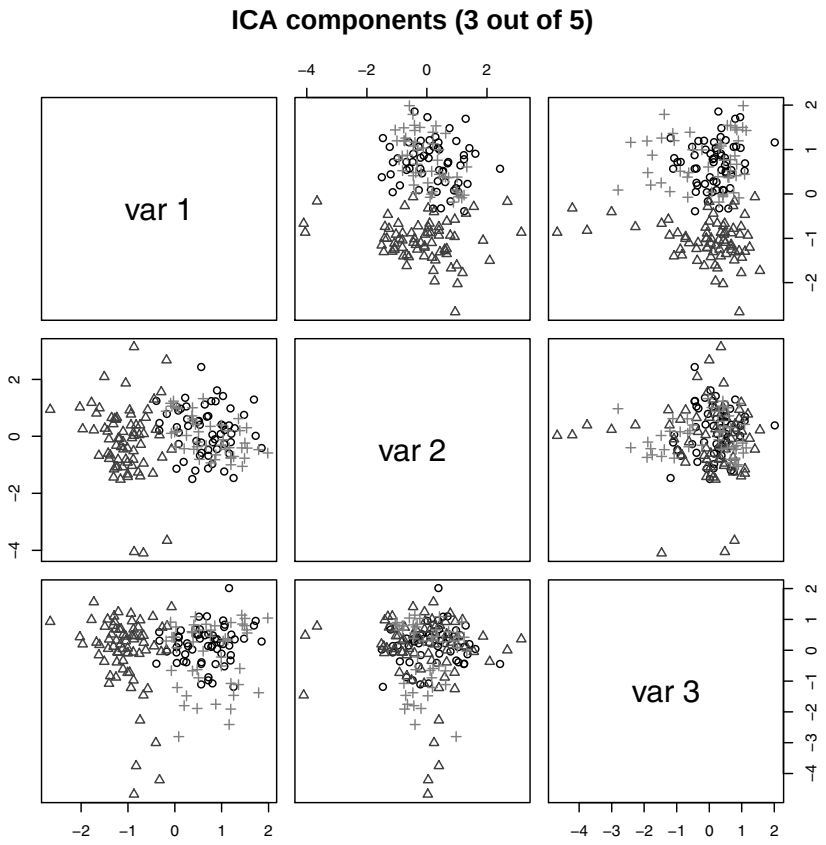

Fig. 4.10. Fast ICA applied to the wine data using five components; only the first three components are shown.

components can change, depending on their number: where in PCA the first component remains the same, no matter how many other components are included, in ICA this is not the case. The pairs plot will therefore be different when taking, e.g., a five-component ICA model:

```
> wines.ica5 <- fastICA(wines.sc, 5)
> pairs(wines.ica5$S[,1:3],
+       main = "ICA components (3 out of 5)",
+       col = wine.classes, pch = wine.classes)
```

In Figure 4.10 we can see that the scores are quite different from the ones in Figure 4.9: now, only component two shows discrimination between the three varieties. In fact, for this model the familiar horse shoe appears in the plot of IC 2 versus IC 5! One should be very cautious in the inspection of ICA score plots. Of course, class separation is not the only criterion we can apply – just because we happen to know in this case what the grouping is does not mean that projections not showing this grouping should be considered "not useful".

4.6.3 Factor Analysis

Another procedure closely related to PCA is Factor Analysis (FA), developed some eighty years ago by the psychologist Charles Spearman, who hypothesized that a large number of abilities (mathematical, artistic, verbal) could be summarized in one underlying factor "intelligence" [41]. Although this view is no longer mainstream, the idea caught on, and FA can be summarized as trying to describe a set of observed variables with a small number of abstract latent factors.

The technique is very similar to PCA, but there is a fundamental difference. PCA aims at finding a series of rotations in such a way that the first axis corresponds with the direction of most variance, and each subsequent orthogonal axis explains the most of the remaining variance. In other words, PCA does not fit an explicit model. FA, on the other hand, does. For a mean-centered matrix $\boldsymbol{X}$, the FA model is

$$\boldsymbol{X} = \boldsymbol{LF} + \boldsymbol{U}$$

where $\boldsymbol{L}$ is the matrix of loadings on the *common factors* $\boldsymbol{F}$, and $\boldsymbol{U}$ is a matrix of *specific factors*, also called *uniquenesses*. The common factors again are linear combinations of the original variables, and the scores present the positions of the samples in the new coordinate system. The result is a set of latent factors that capture as much variation, shared between variables, as possible. Variation that is unique to one specific variable will end up in the specific factors. Especially in fields like psychology it is customary to try and interpret the common factors like in the original approach by Spearman. Summarizing, it can be stated that PCA tries to represent as much as possible of the diagonal elements of the covariance matrix, whereas FA aims at reproducing the off-diagonal elements [24].

There is considerable confusion between PCA and FA, and many examples can be found where PCA models are actually called factor analysis models: one reason is that the simplest way (in a first approximation) to estimate the FA model of Equation 4.6.3 is to perform a PCA – this method of estimation is called Principal Factor Analysis. However, other methods exist, e.g., based on Maximum Likelihood, that provide more accurate models. The second source of confusion is that for spectroscopic data in particular, scientists are often trying to interpret the PCs of PCA. In that sense, they are more interested in the FA model than in the model-free transformation given by PCA.

Factor analysis is available in R as the function `factanal`. Application to the wine data is straightforward:

```
> wines.fa <- factanal(wines.sc, 3, scores = "regression")
> wines.fa

Call:
factanal(x = wines.sc, factors = 3)
```

```
Uniquenesses:
     alcohol     malic acid            ash    ash alkalinity
       0.393          0.729          0.524             0.068
   magnesium   tot. phenols     flavonoids non-flav. phenols
       0.843          0.198          0.070             0.660
     proanth      col. int.       col. hue          OD ratio
       0.559          0.243          0.503             0.248
     proline
       0.388

Loadings:
                 Factor1 Factor2 Factor3
alcohol                    0.776
malic acid       -0.467            0.211
ash                        0.287   0.626
ash alkalinity   -0.297  -0.313   0.864
magnesium         0.119   0.367
tot. phenols      0.825   0.346
flavonoids        0.928   0.262
non-flav. phenols -0.533  -0.140   0.192
proanth           0.621   0.226
col. int.        -0.412   0.751   0.153
col. hue          0.653  -0.206  -0.170
OD ratio          0.865
proline           0.355   0.684  -0.134

               Factor1 Factor2 Factor3
SS loadings      4.005   2.261   1.310
Proportion Var   0.308   0.174   0.101
Cumulative Var   0.308   0.482   0.583

Test of the hypothesis that 3 factors are sufficient.
The chi square statistic is 159.14 on 42 degrees of freedom.
The p-value is 1.57e-15
```

The default `print` method for a factor analysis object shows the uniquenesses, i.e., those parts that cannot be explained by linear combinations of other variables. The uniquenesses of the variables `ash alkalinity` and `flavonoids`, e.g., are very low, indicating that they may be explained well by the other variables. The loadings are printed in such a way as to draw attention to patterns: only three digits are shown after the decimal point, and smaller loadings are not printed.

Scores can be calculated in several different ways, indicated by the `scores` argument of the `factanal` function. Differences between the methods are usually not very large; consult the manual pages of `factanal` to get more infor-

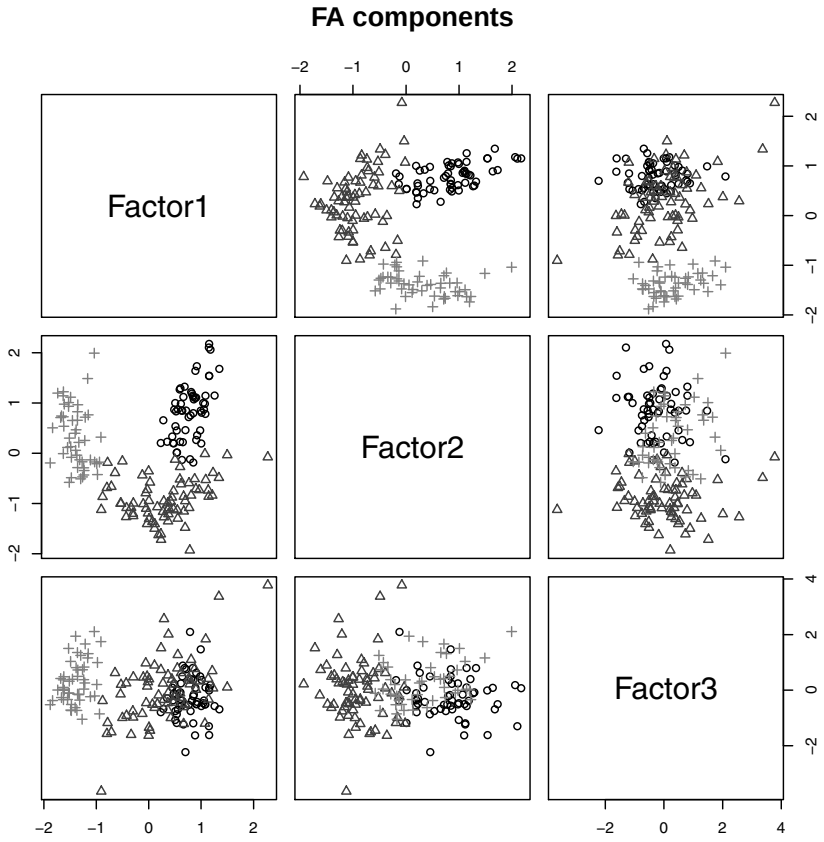

Fig. 4.11. Scores for the FA model of the wine data using three components.

mation about the exact implementation. The result for the regression scores, shown in Figure 4.11, is only slightly different from what we have seen earlier with PCA; the familiar horse shoe is again visible in the first two components.

In Factor Analysis it is usual to rotate the components in such a way that the interpretability is enhanced. One of the ways to do this is to require that as few loadings as possible have large values, something that can be achieved by a so-called *varimax* rotation. This rotation is applied by default in `factanal`, and is the reason why the horseshoe is rotated compared to the PCA score plot in Figure 4.1.

4.6.4 Discussion

Although there are some publications where ICA and FA are applied to data sets in the life sciences, their number is limited, and certainly much lower than the number of applications of PCA. There are several of reasons for this. Both ICA and FA do not have analytical solutions and require optimization to achieve their objectives, which takes more computing time, and can lead to different results, depending on the optimization settings. Moreover, several algorithms are available, each having slightly different definitions, which makes

the results harder to compare and to interpret. PCA, on the other hand, always gives the same result (apart from the sign). In particular, PCA scores and loadings do not change when the number of component is altered. Since choosing the "right" number of components can be quite difficult, this is a real advantage over applying ICA. In a typical application, there are so many choices to make with respect to preprocessing, scaling, outlier detection and others, that there is a healthy tendency to choose methods that have as few tweaking possibilities as possible – if not, one can spend forever investigating the effects of small differences in analysis parameters. Nevertheless, there are cases where ICA, FA, or other dimension reduction methods can have definite advantages over PCA, and it can pay to check that.

Part VI

Appendices

R. Wehrens, *Chemometrics with R: Multivariate Data Analysis in the Natural Sciences and Life Sciences*, Use R!, DOI 10.1007/978-3-642-17841-2,

A

R Packages Used in this Book

The table below contains an overview of R packages mentioned (but not necessarily treated in the same detail) in the book.

Table A.1. R packages used in this book

ada	**elasticnet**	**mclust**	**rda**
ALS	**fastICA**	**meboot**	**relaxo**
AMORE	**fingerprint**	**msProstate**	**robustbase**
boost	**FRB**	**neuralnet**	**rpart**
boot	**glmnet**	**nnet**	**rrcov**
BootPR	**gpls**	**paltran**	**sfsmisc**
bootstrap	**gtools**	**VPdtw**	**som**
caMassClass	**ipred**	**pls**	**spls**
ChemometricsWithR	**kohonen**	**plsgenomics**	**stats**
class	**lars**	**plspm**	**subselect**
cluster	**lasso2**	**ppls**	**TIMP**
DAIM	**leaps**	**PROcess**	**tree**
dtw	**lpc**	**ptw**	**wccsom**
e1071	**lspls**	**randomForest**	**xcms**
EffectiveDose	**MASS**		

References

1. R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010. ISBN 3-900051-07-0.
2. W.N. Venables, D.M. Smith, and the R development Core Team. An introduction to R, December 2009. Version 2.10.1.
3. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, New York, 2001.
4. K. Varmuza and P. Filzmoser. *Introduction to Multivariate Statistical Analysis in Chemometrics*. Taylor & Francis - CRC Press, Boca Raton, FL, USA, 2009.
5. J. Kalivas. Two data sets of near infrared spectra. *Chemom. Intell. Lab. Syst*, 37:255–259, 1997.
6. M. Forina, C. Armanino, M. Castino, and M. Ubigli. Multivariate data analysis as a discriminating method of the origin of wines. *Vitis*, 25:189–201, 1986.
7. B.-L. Adam, Y. Qu, J.W. Davis, M.D. Ward, M.A. Clements, L.H. Cazares, O.J. Semmes, P.F. Schellhammer, Y. Yasui, Z. Feng, and G.L. Wright. Serum protein fingerprinting coupled with a pattern-matching algorithm distinguishes prostate cancer from benign prostate hyperplasia and healthy men. *Cancer Res.*, 62(13):3609–3614, July 2002.
8. Y. Qu, B.-L. Adam, Y. Yasui, M.D. Ward, L.H. Cazares, P.F. Schellhammer, Z. Feng, O.J. Semmes, and G.L. Wright. Boosted decision tree analysis of surface-enhanced laser desorption/ionization mass spectral serum profiles discriminates prostate cancer from noncancer patients. *Clin Chem*, 48(10):1835–43, October 2002.
9. T.G. Bloemberg, J. Gerretzen, H.J.P. Wouters, J. Gloerich, M. van Dael, H.J.C.T. Wessels, L.P. van den Heuvel, P.H.C. Eilers, L.M.C. Buydens, and R. Wehrens. Improved parametric time warping for proteomics. *Chemom. Intell. Lab. Systems*, 2010.
10. A. Savitsky and M.J.E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Anal. Chem.*, 36:1627–1639, 1964.
11. W.S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *J. Am. Stat. Assoc.*, 74:829–836, 1979.
12. G.P. Nason. *Wavelet methods in statistics with R*. Springer, New York, 2008.
13. P. Geladi, D. MacDougall, and H. Martens. Linearization and scatter-correction for NIR reflectance spectra of meat. *Appl. Spectr.*, 39:491–500, 1985.

14. T. Næs, T. Isaksson, and B.R. Kowalski. Locally weighted regression and scatter correction for near-infrared reflectance data. *Anal. Chem.*, 62:664–673, 1990.
15. H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust., Speech, Signal Process.*, 26:43–49, 1978.
16. L.R. Rabiner, A.E. Rosenberg, and S.E. Levinson. Considerations in dynamic time warping algorithms for discrete word recognition. *IEEE Trans. Acoust., Speech, Signal Process.*, 26:575–582, 1978.
17. C.P. Wang and T.L. Isenhour. Time-warping algorithm applied to chromatographic peak matching gas chromatography / Fourier Transform infrared / Mass Spectrometry. *Anal. Chem.*, 59:649–654, 1987.
18. N.P.V. Nielsen, J.M. Carstensen, and J. Smedsgaard. Aligning of single and multiple wavelength chromatographic profiles for chemometric data analysis using correlation optimized warping. *J. Chrom. A*, 805:17–35, 1998.
19. P.H.C. Eilers. Parametric time warping. *Anal. Chem.*, 76:404–411, 2004.
20. R. de Gelder, R. Wehrens, and J.A. Hageman. A generalized expression for the similarity spectra: application to powder diffraction pattern classification. *J. Comput. Chem.*, 22(3):273–289, 2001.
21. D. Clifford, G. Stone, I. Montoliu, S. Rezzi, F.-P. Martin, P. Guy, S. Bruce, and S. Kochhar. Alignment using variable penalty dynamic time warping. *Anal. Chem.*, 81:1000–1007, 2009.
22. W. Windig, J. Phalp, and A. Payna. A noise and background reduction method for component detection in liquid chromatography/mass spectrometry. *Anal. Chem.*, 68:3602–3606, 1996.
23. T. Giorgino. Computing and visualizing dynamic time warping alignments in R: the dtw package. *J. Stat. Softw.*, 31(7), 2009.
24. J.E. Jackson. *A User's Guide to Principal Components*. Wiley, Chichester, 1991.
25. I.T. Jolliffe. *Principal Component Analysis*. Springer, New York, 1986.
26. K. Mardia, J. Kent, and J. Bibby. *Multivariate Analysis*. Academic Press, 1979.
27. W. Härdle and L. Simar. *Applied Multivariate Statistical Analysis*. Springer, Berlin, 2nd edition, 2007.
28. K.R. Gabriel. The biplot graphic display of matrices with application to principal component analysis. *Biometrika*, 58:453–467, 1971.
29. J.C. Gower and D.J. Hand. *Biplots*. Number 54 in Monographs on Statistics and Applied Probability. Chapman and Hall, London, UK, 1996.
30. K. Baumann. Uniform-length molecular descriptors for quantitative structure-property relationships (QSPR) and quantitative structure-activity relationships (QSAR): classification studies and similarity searching. *Trends Anal. Chem.*, 18(1):36–46, 1999.
31. T.F. Cox and M.A.A. Cox. *Multidimensional Scaling*. Chapman and Hall, 2001.
32. I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling*. Springer, 2nd edition, 2005.
33. J.C. Gower. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53:325–328, 1966.
34. B.D. Ripley. *Pattern recognition and neural networks*. Cambridge University Press, 1996.

35. J.H. Friedman and J.W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, C23:881–889, 1974.
36. P.J. Huber. Projection pursuit. *The Annals of Statistics*, 13:435–475, 1985.
37. J.H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82:249–266, 1987.
38. A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley, Chichester, 2001.
39. T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, Chichester, 1991.
40. A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13:411–430, 2000.
41. C. Spearman. "General intelligence", objectively determined and measured. *Am. J. Psychol.*, 15:201–293, 1904.
42. T. Kohonen. *Self-Organizing Maps*. Number 30 in Springer Series in Information Sciences. Springer, Berlin, 3 edition, 2001.
43. R. Wehrens and L.M.C. Buydens. Self- and super-organising maps in R: the kohonen package. *Journal of Statistical Software*, 21(5), 9 2007.
44. A. Ultsch. Self-organizing neural networks for visualization and classification. In O. Opitz, B. Lausen, and R. Klar, editors, *Information and Classification – Concepts, Methods and Applications*, pages 307–313. Springer Verlag, 1993.
45. W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.
46. R. Wehrens and E. Willighagen. Mapping databases of x-ray powder patterns. *R News*, 6(3):24–28, August 2006.
47. M. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, 95:14863–14868, December 1998.
48. L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data – An Introduction to Cluster Analysis*. John Wiley & Sons, New York, 1990.
49. G.J. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley & Sons, New York, 2000.
50. C. Fraley and A.E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *J. Am. Stat. Assoc.*, 97:611–631, 2002.
51. A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B*, 39(1):1–38, 1977.
52. G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, 1997.
53. H. Akaike. A new look at the statistical model identification. *IEEE Trans. Automatic Control*, 19:716–723, 1974.
54. G. Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6:461–464, 1978.
55. C. Fraley and A.E. Raftery. Enhanced software for model-based clustering, discriminant analysis, and density estimation: MCLUST. *J. Classif.*, 20:263–286, 2003.
56. C. Fraley. Algorithms for model-based gaussian hierarchical clustering. *SIAM J. Scient. Comput.*, 20:270–281, 1998.
57. J.D. Banfield and A.E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803–821, 1993.
58. L. Hubert. Comparing partitions. *J. Classif.*, 2:193–218, 1985.

59. W.M. Rand. Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.*, 66:846–850, 1971.
60. J. Vesanto and E. Alhoniemi. Clustering of the self-organising map. *IEEE Trans. Neural Netw.*, 11:586–600, 2000.
61. E.B. Fowlkes and C.L. Mallows. A method for comparing two hierarchical clusterings. *J. Am. Stat. Assoc.*, 78:553–584, 1983. Including discussion.
62. L.A. Goodman and W.H. Kruskal. Measures of association for cross classifications. *J. Am. Statist. Assoc.*, 49:732–764, 1954.
63. M. Meila. Comparing clusterings – an information-based distance. *J. Multivar. Anal.*, 98(5):873–895, 2007.
64. G.J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley-Interscience, 2004.
65. M. Stone. Cross-validatory choice and assessment of statistical predictions. *J. R. Statist. Soc. B*, 36:111–147, 1974. Including discussion.
66. B. Efron and R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, New York, 1993.
67. R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.
68. J. Friedman. Regularized discriminant analysis. *J. Am. Stat. Assoc.*, 84:165–175, 1989.
69. S. Dudoit, J. Fridlyand, and T.P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *J. Am. Stat. Assoc.*, 97:77–87, 2002.
70. D. Hand and K. Yu. Idiot's Bayes – not so stupid after all? *Int. Statist. Rev.*, 69:385–398, 2001.
71. R. Tibshirani, T. Hastie, B. Narashimhan, and G. Chu. Class prediction by nearest shrunken centroids with applications to dna microarrays. *Statistical Science*, 18:104–117, 2003.
72. Y. Guo, T. Hastie, and R. Tibshirani. Regularized linear discriminant analysis and its application in microarrays. *Biostatistics*, 8(1):86–100, 2007.
73. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.
74. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
75. J.R. Quinlan. The induction of decision trees. *Mach. Learn.*, 1(1):81–106, 1986.
76. T.M. Therneau and E.J. Atkinson. An introduction to recursive partitioning using the RPART routines. Technical Report 61, Mayo Foundation, September 1997.
77. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.
78. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based Learning Methods*. Cambridge University Press, 2000.
79. B. Schölkopf and A.J. Smola. *Learning with kernels*. MIT Press, Cambridge, MA, 2002.
80. F. Rosenblatt. *Principles of neurodynamics*. Spartan Books, Washington DC, 1962.
81. D.E. Rumelhard and J.L. McClelland, editors. *Parallel distributed processing: explorations in the microstructure of cognition. Volume 1: Foundations*. MIT Press, Cambridge MA, 1986.
82. Frauke Günther and Stefan Fritsch. neuralnet: Training of neural networks. *The R Journal*, 2(1):30–38, June 2010.

83. B.-H. Mevik and R. Wehrens. The pls package: principal component and partial least squares regression in R. *J. Stat. Soft.*, 18(2), 2007.
84. B.-H. Mevik and H.R. Cederkvist. Mean squared error of prediction (MSEP) estimates for principal component regression (PCR) and partial least squares regression (PLSR). *J. Chemom.*, 18:422–429, 2004.
85. A.S. Barros and D.N. Rutledge. Genetic algorithms applied to the selection of principal components. *Chemom. Intell. Lab. Syst.*, 40:65–81, 1998.
86. B.S. Dayal and J.F. MacGregor. Improved PLS algorithms. *J. Chemom.*, 11:73–85, 1997.
87. H. Martens and T. Næs. *Multivariate Calibration.* Wiley, Chichester, 1989.
88. S. Rännar, F. Lindgren, P. Geladi, and S. Wold. The PLS Kernel algorithm for data sets with many variables and fewer objects. Part 1: Theory and algorithm. *J. Chemom.*, 8:111, 1994.
89. S. de Jong. SIMPLS: an alternative approach to partial least squares regression. *Chemom. Intell. Lab. Syst.*, 18:251–263, 1993.
90. I.E. Frank and J.H. Friedman. A statistical view of some chemometrics regression tools. *Technometrics*, 35:109–135, 1993.
91. S. Wold, N. Kettaneh-Wold, and B. Skagerberg. Nonlinear PLS modeling. *Chemom. Intell. Lab. Syst.*, 7:53–65, 1989.
92. K. Hasegawa, T. Kimura, Y. Miyashita, and K. Funatsu. Nonlinear partial least squares modeling of phenyl alkylamines with the monoamine oxidase inhibitory activities. *J. Chem. Inf. Comput. Sci-*, 36:1025–1029, 1996.
93. K. Jorgensen, V. H. Segtnan, K. Thyholt, and T. Næs. A comparison of methods for analysing regression models with both spectral and designed variables. *J. Chemometr.*, 18:451–464, 2004.
94. B. Ding and R. Gentleman. Classification using penalized partial least squares. *J. Comput. Graph. Stat.*, 14:280–298, 2005.
95. B.D. Marx. Iteratively reweighted partial least squares estimation for generalized linear regression. *Technometrics*, 38:374–381, 1996.
96. C.J.F. ter Braak and S. Juggins. Weighted averaging partial least squares regression WAPLS: an improved method for reconstructing environmental variables from species assemblages. *Hydrobiologia*, 269:485–502, 1993.
97. M. Tenenhaus, V. Esposito Vinzi, Y.M. Chatelin, and C. Lauro. PLS path modelling. *Comput. Stat. Data Anal.*, 48:159–2005, 2005.
98. N. Krämer, A.-L. Boulesteix, and G. Tutz. Penalized partial least squares with applications to B-spline transformations and functional data. *Chemom. Intell. Lab. Syst.*, 94:60–69, 2008.
99. H. Chun and S. Keles. Sparse partial least squares for simultaneous dimension reduction and variable selection. *J. Royal Stat. Soc. – Series B*, 72:3–25, 2010.
100. A.E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58:54–59, 1962.
101. A.E. Hoerl and R.W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 8:27–51, 1970.
102. A.E. Hoerl, R.W. Kennard, and K.F. Baldwin. Ridge regression: some simulations. *Commun. Stat. – Simul. Comput.*, 4:105–123, 1975.
103. J.F. Lawless and P. Wang. A simulation study of ridge and other regression estimators. *Commun. Stat. – Theory and Methods*, 5:303–323, 1976.
104. M. Stone and R.J. Brooks. Continuum regression: cross-validated sequentially constructed prediction embracing ordinary least squares, partial least squares

and principal components regression (with discussion). *J. R. Statist. Soc.*, 52:237–269, 1990.
105. S. de Jong and H.A.L. Kiers. Principal covariates regression: Part I. Theory. *Chemom. Intell. Lab. Syst.*, 14:155–164, 1992.
106. R.W. Kennard and L. Stone. Computer aided design of experiments. *Technometrics*, 11:137–148, 1969.
107. C.D. Brown and H.T. Davis. Receiver operating characteristic curves and related decision measures: a tutorial. *Chemom. Intell. Lab. Syst.*, 80:24–38, 2006.
108. C.L. Mallows. Some comments on Cp. *Technometrics*, 15:661–675, 1973.
109. P. Craven and G. Wahba. Smoothing noisy data with spline functions. *Numer. Math.*, 31:377–403, 1979.
110. S. Smit, M.J. van Breemen, H.C.J. Hoefsloot, A.K. Smilde, J.M.F.G. Aerts, and C.G. de Koster. Assessing the statistical validity of proteomics based biomarkers. *Anal. Chim. Acta*, 592:210–217, 2007.
111. A.C. Davison and D.V. Hinkley. *Bootstrap Methods and their Applications*. Cambridge University Press, Cambridge, 1997.
112. B. Efron and R. Tibshirani. Improvements on cross-validation: the .632+ bootstrap method. *J. Am. Stat. Assoc.*, 92:548–560, 1997.
113. B. Efron. Bootstrap methods: another look at the jackknife. *Ann. Stat.*, 7:1–26, 1979.
114. L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
115. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
116. Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.
117. V. Svetnik, A. Liaw, C. Tong, J.C. Culberson, R.P. Sheridan, and B.P. Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *J. Chem. Inf. Comput. Sci.*, 43(6):1947–58, 2003.
118. G. Michailides, K. Johnson, and M. Culp. ada: an R package for stochastic boosting. *J. Stat. Softw.*, 17(2), 2006.
119. J.H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Ann. Stat.*, 28:337–374, 2000.
120. R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Stat.*, 26:1651–1686, 1998.
121. R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B*, 58:267–288, 1996.
122. R. Wehrens and W.E. van der Linden. Bootstrapping principal-component regression models. *J. Chemom.*, 11(2):157–171, 1997.
123. A.H. Land and A.G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28:497–520, 1960.
124. G.M. Furnival and G.M. Wilson. Regression by leaps and bounds. *Technometrics*, 16:499–511, 1974.
125. B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.
126. H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *J. Royal. Stat. Soc. B*, 67:301–320, 2005.
127. S. Kirkpatrick, C.D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

128. V. Cerny. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985.
129. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.
130. V. Granville, M. Krivanek, and J.-P. Rasson. Simulated annealing: a proof of convergence. *IEEE Trans. Patt. Anal. Machine Intell.*, 16:652–656, 1994.
131. A.P. Duarte Silva. Efficient variable screening for multivariate analysis. *J. Mult. Anal.*, 76:35–62, 2001.
132. D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, Boston, MA., 1989.
133. R. Leardi. Genetic algorithms in chemometrics and chemistry: a review. *J. Chemom.*, 15:559–569, 2001.
134. J. Shao. Linear model selection by cross-validation. *J. Am. Statist. Assoc.*, 88:486–494, 2003.
135. K. Baumann, H. Albert, and M. von Korff. A systematic evaluation of the benefits and hazards of variable selection in latent variable regression. Part I. Search algorithm, theory and simulations. *J. Chemometr.*, 16:339–350, 2002.
136. K. Baumann, H. Albert, and M. von Korff. A systematic evaluation of the benefits and hazards of variable selection in latent variable regression. Part II. Practical applications. *J. Chemometr.*, 16:351–360, 2002.
137. M. Hubert. Robust calibration. In S.D. Brown, R. Tauler, and B. Walczak, editors, *Comprehensive Chemometrics – Chemical and Biochemical Data Analysis*, chapter 3.07, pages 315–343. Elsevier, 2009.
138. C. Croux and G. Haesbroeck. Principal components analysis based on robust estimators of the covariance or correlation matrix. *Biometrika*, 87:603–618, 2000.
139. P. Rousseeuw. Least median of squares regression. *J. Am. Stat. Assoc.*, 79:871–880, 1984.
140. P.J. Rousseeuw and K. van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41:212–223, 1999.
141. M. Hubert, P.J. Rousseeuw, and K. Vanden Branden. ROBPCA: a new approach to robust principal component analysis. *Technometrics*, 47:64–79, 2005.
142. V. Todorov and P. Filzmoser. An object oriented framework for robust multivariate analysis. *J. Stat. Softw.*, 32(3):1–47, 2009.
143. M. Hubert and K. Vanden Branden. Robust methods for partial least squares regression. *J. Chemom.*, 17:537–549, 2003.
144. B. Liebmann, P. Filzmoser, and K. Varmuza. Robust and classical PLS regression compared. *J. Chemom.*, 24:111–120, 2009.
145. S. Wold, H. Antti, F. Lindgren, and J. Ohman. Orthogonal signal correction of near-infrared spectra. *Chemom. Intell. Lab. Syst.*, 44:175–185, 1998.
146. O. Svensson, T. Kourti, and J.F. MacGregor. A comparison of orthogonal signal correction algorithms and characteristics. *J. Chemom.*, 16:176–188, 2002.
147. J. Trygg and S. Wold. Orthogonal projections to latent structures (O-PLS). *J. Chemom.*, 16:119–128, 2002.
148. M. Barker and W. Rayens. Partial least squares for discrimination. *J. Chemom.*, 17:166–173, 2003.
149. D.V. Nguyen and D. Rocke. Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics*, 18:39–50, 2002.

150. A.L. Boulesteix. PLS dimension reduction for classification with high-dimensional microarray data. *Stat. Appl. Genet. Mol. Biol.*, 3, 2004. Article 33.
151. O.E. de Noord. Multivariate calibration standardization. *Chemom. Intell. Lab. Syst.*, 25:85–97, 1994.
152. Y. Wang, D.J. Veltkamp, and B.R. Kowalski. Multivariate instrument standardization. *Anal. Chem.*, 63:2750–2756, 1994.
153. Z.Y. Wang, T. Dean, and B.R. Kowalski. Additive background correction in multivariate instrument standardization. *Anal. Chem.*, 67:2379–2385, 1995.
154. E. Bouveresse, D.L. Massart, and P. Dardenne. Modified algorithm for standardization of near-infrared spectrometric instruments. *Anal. Chem.*, 67:1381–1389, 1995.
155. Y. Wang, M.J. Lysaght, and B.R. Kowalski. Improvement of multivariate calibration through instrument standardization. *Anal. Chem.*, 64:764–771, 1992.
156. R. Tauler, S. Lacorte, and D. Barceló. Application of multivariate self-modeling curve resolution to the quantitation of trace levels of organophosphorus pesticides in natural waters from interlaboratory studies. *J. Chromatogr. A*, 730:177–183, 1996.
157. W.H. Lawton and E.A. Sylvestre. Self-modeling curve resolution. *Technometrics*, 13:617–633, 1971.
158. R. Tauler. Multivariate curve resolution applied to second-order data. *Chemom. Intell. Lab. Syst.*, 30:133–146, 1995.
159. A. de Juan, M. Maeder, M. Martinez, and R. Tauler. Combining hard- and soft-modelling techniques to solve kinetic problems. *Chemom. Intell. Lab. Syst.*, 54:49–67, 2000.
160. M. Maeder. Evolving factor analysis for the resolution of overlapping chromatographic peaks. *Anal. Chem.*, 59:527–530, 1987.
161. H.R. Keller and D.L. Massart. Peak purity control in liquid chromatography with photodiode-array detection by a fixed size moving window evolving factor analysis. *Anal. Chim. Acta*, 246:379–390, 1991.
162. F. Questa Sanchez, M.S. Khots, D.L. Massart, and J.O. de Beer. Algorithm for the assessment of peak purity in liquid chromatography with photodiode-array detection. *Anal. Chim. Acta*, 285:181–192, 1994.
163. W. Windig and J. Guilment. Interactive self-modeling mixture analysis. *Anal. Chem.*, 63:1425–1432, 1991.
164. F. Cuesta Sanchez, B. van de Bogaert, S.C. Rutan, and D.L. Massart. Multivariate peak purity approaches. *Chemom. Intell. Lab. Syst.*, 36:153–164, 1996.
165. P. Stilbs. Molecular self-diffusion coefficients in Fourier transform nuclear magnetic resonance spectrometric analysis of complex mixtures. *Anal. Chem.*, 53:2135–2137, 1981.
166. P. Stilbs. Fourier-transform pulsed-gradient spin-echo studies of molecular diffusion. *Progr. NMR Spectrosc.*, 19:1–45, 1987.
167. R. Huo, R. Wehrens, J. van Duynhoven, and L.M.C. Buydens. Assessment of techniques for DOSY NMR data processing. *Anal. Chim. Acta*, 490:231–251, 2003.
168. R. Huo, R. Wehrens, and L.M.C. Buydens. Improved DOSY NMR data processing by data enhancement and combination of multivariate curve resolution with non-linear least squares fitting. *J. Magn. Reson.*, 169:257–269, 2004.

169. K.M. Mullen, I.H.M. van Stokkum, and V.V. Mihaleva. Global analysis of multiple gas chromatography-mass spectrometry (GC/MS) data sets: a method for resolution of co-eluting components with comparison to MCR-ALS. *Chemom. Intell. Lab. Syst.*, 95:150–163, 2009.
170. G. Munoz and A. de Juan. pH- and time-dependent hemoglobin transitions: a case study for process modelling. *Anal. Chim. Acta*, 595:198–208, 2007.

Index