# CHEMOMETRICS I

Study Guide

Original version: Ron Wehrens
Adapted by: Jan Gerretzen and Geert Postma
Institute for Molecules and Materials
Dept. of Analytical Chemistry/Chemometrics
Radboud University Nijmegen

Version 1.04 (January 4, 2015)

# Contents


1. Introduction ..... 4
   1.1 Objectives ..... 4
   1.2 Contents of the course ..... 4
   1.3 Course organization ..... 5
   1.4 Examination ..... 6
   1.5 Required background ..... 6
   1.6 Course materials ..... 6
   1.7 Lecturers ..... 7
   1.8 Data ..... 7
2. Principal Component Analysis ..... 10
   2.1 Multivariate data ..... 10
   2.2 PCA ..... 11
   2.3 PCA: Algorithms ..... 12
      2.3.1 SVD ..... 12
      2.3.2 Other approaches ..... 13
      2.3.3 Example: wine data ..... 14
   2.4 Selection of the number of PCs ..... 14
   2.5 Visualization ..... 15
      2.5.1 Score plots ..... 15
      2.5.2 Loading plots ..... 16
      2.5.3 Biplots ..... 17
   2.6 Projections ..... 19
   2.7 Scaling ..... 19
   2.8 Concluding remarks ..... 20
3. Cluster analysis ..... 22
   3.1 Introduction ..... 22
   3.2 Hierarchical methods ..... 22
      3.2.1 Algorithms ..... 22
      3.2.2 Choosing the optimal number of clusters ..... 24
      3.2.3 Discussion ..... 25
   3.3 Partitional methods: $k$-means clustering ..... 26
      3.3.1 Algorithm ..... 26
      3.3.2 Discussion ..... 28

3.4 Concluding remarks ..... 28
4. Classification ..... 30
4.1 Introduction ..... 30
4.2 Discriminant analysis ..... 30
4.2.1 Linear discriminant analysis ..... 31
4.2.2 Quadratic discriminant analysis ..... 35
4.3 Nearest-neighbor approaches ..... 36
4.4 Error estimation ..... 37
4.4.1 Using a training set and a test set ..... 37
4.4.2 Cross-validation ..... 38
4.5 Classification: the correct way ..... 39
4.6 Discussion ..... 40
5. Multivariate regression ..... 42
5.1 Introduction ..... 42
5.2 Principal Components Regression (PCR) ..... 43
5.2.1 The algorithm ..... 43
5.2.2 Selecting the optimal number of components ..... 44
5.2.3 Discussion ..... 46
5.3 Partial Least Squares Regression (PLS) ..... 47
5.3.1 The algorithm(s) ..... 47
5.3.2 Interpretation ..... 49
5.3.3 Discussion ..... 49
5.4 Variable selection ..... 50
5.5 Example ..... 50
5.6 Discussion ..... 52
Bibliography ..... 54
Version history ..... 55
Acknowledgements ..... 56

# 1. Introduction

## 1.1 Objectives

The course Chemometrics I gives an overview of the field of chemometrics, which is sometimes described as “data analysis in chemistry”. The methods originate in most cases from (multivariate) statistics and computer science, but have an added twist because of the nature of chemical data in general. The field originated in analytical chemistry, which is not surprising given the stress on measurements, and in particular multivariate measurements in that area. Indeed, the prime contribution of chemometrics to science in general is often taken to be PLS (Partial Least Squares), a method that was developed to be able to build predictive models out of spectra containing hundreds or even thousands of wavelengths.

Now, PLS is used in almost every field where multivariate calibration is important, from psychology and sociology to economics to mathematical statistics. In general, the methods in chemometrics are known in other fields as well, but the application requires thought and knowledge about the chemical nature of the problem.

Given the multivariate nature of the data that nowadays are produced, in fact one cannot do without techniques that can handle this kind of data.

Therefore, this course focuses not only on the characteristics of the statistical modelling techniques, but also on the application of these methods in situations with typical chemical characteristics. At the end of the course, the student knows the principles behind the most important chemometrical methods, when and how to apply them, and will have practical experience in applying these methods to several data sets. The course is obligatory for students doing an internship at the Department of Analytical Chemistry.

## 1.2 Contents of the course

The course starts with the visualization of medium and high-dimensional data. Quickly the limitations of the usual plots (although useful) will become apparent. At that point Principal Component Analysis (PCA) will be introduced as the prime workhorse for the exploratory analysis of large data sets.

Next, the analysis of grouped data is tackled. In many cases, one is interested in finding a hitherto unknown grouping. A plethora of clustering methods is available for this case, more than can - and probably ever should - be taught in one course. The basics of the most common hierarchical and partitional clustering methods are given, with an emphasis on interpretation. It should be noted that different methods tend to use different (distributional) assumptions and more often than not yield different results.

A similar but distinct situation arises when one, again, is interested in a grouping, but now already knows which groups are present. The purpose then is to generate predictive models, i.e. models that are able to infer the correct class for new data. This process is called classification, and again many methods are available. We show two extremes: on the one hand Linear Discriminant Analysis (LDA), where one assumes that classes are normally distributed with identical covariance matrices, and on the other hand the nonparametric k-nearest-neighbour (KNN) method, which assigns new objects to the closest object in the training set. Validation is a topic that will receive ample attention in this

respect: in cases like this it is imperative that some unbiased measure of (expected) success can be derived.

Finally, we enter the world where we are not so much interested in finding or reproducing a grouping —a discrete variable—but more in predicting continuous variables (often properties). Multiple linear regression fails horribly if independent variables are correlated, as is almost always the case in chemical data. Two related multivariate regression models are treated in this course, Principal Component Regression (PCR) and Partial Least Squares regression (PLS). Again, validation is of crucial importance.

These four subjects are summarized in Table 1.1.

***Table 1.1*** *– The contents of Chemometrics I*

| *Subject* | *Chapter in Study Guide* | *Chapter (in references [1, 2])* |
|---|---|---|
| Principal Component Analysis | 2 | A14, B31.1-31.5 |
| Clustering | 3 | B30 |
| Classification | 4 | B33 |
| Multivariate regression | 5 | A10.1-10.5, B35.6-35.7, B36 |

## 1.3 Course organization

The course makes use of a wide variety of instruction media: lectures, self-study using this reader, the course website and background literature, and last but not least, exercises. During the exercises parts of the theory are implemented by writing functions. Different people have different preferences and abilities, and by offering many forms for transferring knowledge we hope to address as many students as possible. This also implies some degree of freedom for the students: they are free to, e.g., do the computer exercises at home at any time they want.

However, the course has a tight schedule, and it is hard or even impossible to catch up in case of a delay. Therefore, there are several constraints. Every subject is introduced in a two-hour lecture. The students then have one week (on average) to acquaint themselves with the theory and work on the computer exercises. The results of these exercises (being programmed functions, which eventually will form a data analysis toolbox, and answers to the questions) shall then be checked by the student during the first problem session (seminar) following each lecture. An active participation is required. Failing to do so will result in exclusion from the course. Students work in pairs; in this way, they can help each other. If an odd number of students register for the course, one group will consist of three students. Doing the course alone is actively discouraged! Finally, for each of the four topics, students will receive assignments. After having studied the computer exercises and having reworked the examples in this reader and in the handbook, students should be able to write the reports on the assignments relatively easy. A second problem session (scheduled during the second week following each lecture) is available for finalizing the assignments and asking questions to the lecturers. During this problem session the outcome of the previous assignment can be discussed, as well. However, one should not underestimate the amount of work needed for a thorough understanding. Also note that the outcome of the assignments depend on the correctness of the functions and scripts that you have made.

## 1.4 Examination

During the course, four assignments will be handed out. For each assignment, students will write a report, answering the questions in the assignment, and giving sufficient explanation (text, figures) to motivate the answers. So, not only answer the questions of the assignments, but also include an introduction, mention the methods that are applied (and why and how), the results that are obtained, show illustrative figures, discuss the results, and at the end some conclusion. The first assignment (on PCA) will probably be detailed and already frequently will describe which method the student should apply and how. But later on the assignments will be more general, not specifying exactly all steps that have to be executed: then it is up to the student to make choices and motivate them. The scripts used to obtain the answers should be included as well. The grading of each report will be based on the presence and correctness of all these items. One week after receiving the assignment the report (in pdf format), as well as the scripts, should be sent by email to the appropriate lecturer. **The last assignment and associated report will have to be made by each student individually.** For the final mark the first 3 reports each will count for $1/6^{th}$; the last report will count for 50%, with the added constraint that all assignment reports should be marked with at least a 6. There will be no separate exam.

If an assignment is not accepted, the students get one chance to improve the assignment so that it passes the requirements; if this does not lead to a successful conclusion, the students can try again the next time the course is given. This enables them to re-visit the lectures and gives them more time to study the material; any assignments that have been done successfully will remain valid for one course, so these do not have to be redone. *Note: since the scripts/functions to be programmed by the students are to be used for the assignment and as such constitute an essential part of the assignment and its evaluation, copying of the scripts/functions (from elsewhere or others) will be seen as fraud, and corresponding measures will be taken.*

## 1.5 Required background

To be able to do this course, it is necessary to have passed the Statistics course in the second year of Chemistry, Molecular Life Sciences and Natural Sciences, or an equivalent course. An elementary knowledge of linear algebra is required. Also basic knowledge of Matlab and some experience of programming in Matlab is required. More specifically, you should know about and have some experience with the construction and use of own made functions, if-then-else statements, for-loops and while loops, addressing vectors and matrices, and be able to plot figures. This because some experience with programming will make life during this course a lot easier. Although the software side of the computer exercises is not a goal in itself, one can only truly understand the algorithms and techniques presented in this course from hands-on experience. That means that students are, in several cases, required to program their own functions. The exercises associated with the first lecture will give a gentle introduction to the way Matlab is used in the course; several documents are available on the web giving more information. A quick reference guide containing the most important commands and some examples are given in the appendix of the exercise guide.

## 1.6 Course materials

This reader contains a short summary of the theory. More details can be found in the Handbook of Chemometrics and Qualimetrics, parts A and B by Massart, Vandeginste and others [1, 2], which are available in the library; they can be consulted in the reading room of the department of Analytical Chemistry, too.

An important part of the course consists of computer exercises, described in a separate manual. These exercises can be done using Matlab. It really pays to invest some time in learning Matlab beyond the simplest commands: in the long run, you will profit. All information necessary for the course, including this reader, data sets, and the computer exercises, can be found on the website dedicated to Chemometrics I: http://webchem2.science.ru.nl/chemometrics-i/

## 1.7 Lecturers

The course lecturers are Dr. J. Jansen and Dr. G. Postma (room HG01.721 and HG01.712, emails: jj.jansen@science.ru.nl and g.postma@science.ru.nl). The computer exercises will be supervised by PhD students and postdocs of the department of Analytical Chemistry. During the first problem session following each lecture these will be available for answering technical and theoretical questions on the computer exercises; questions on the theory can also be posed to the lecturers. A complete schedule of lectures and subjects is available on the course website.

## 1.8 Data

Several data sets, also available from the course website, will be used in this reader as well as in the computer exercises. Many concepts will be introduced and illustrated using a data set of 177 Italian wines. The wines are from the same region, but are made of three different grape cultivars. Of each wine sample, thirteen characteristics have been measured (see Table 1.2).

***Table 1.2*** *– The thirteen variables of the wine data set. OD280/OD315 is the ratio of optical densities at 280nm and 315nm for a diluted wine.*

| Variable no. | Description |
|---|---|
| **1** | Alcohol concentration |
| **2** | Malic acid concentration |
| **3** | Ash |
| **4** | Alkalinity of ash |
| **5** | Magnesium concentration |
| **6** | Total phenols concentration |
| **7** | Flavonoids concentration |
| **8** | Non-flavonoid phenols concentration |
| **9** | Proanthocyanins concentration |
| **10** | Color intensity |
| **11** | Hue |
| **12** | OD280/OD315 of diluted wines |
| **13** | Proline concentration |

The second data set is the famous Iris data set (see the first computer exercises). It consists of length measurements on flower leaves of 150 irises of three varieties: *setosa*, *versicolor*, and *virginica*. The data set therefore is a matrix consisting of 150 rows (the irises) and four columns: the variables are sepal length, sepal width, petal length, and petal width, respectively[1]. Class information - the iris varieties - is available, too. The set-up of the computer practical generally is to try out a new technique on the Iris data (with hints and examples), and next to analyze a more complex, biological,

[1] Petals and sepals are the colored leaves and green leaves of a flower, respectively.

physical or chemical, data set. In this way, acquired knowledge can be applied in practice - and it forms a real test to see whether everything was clear in the lectures and introductory exercises.

A real-life data set, used in the exercises, has been measured by analysing soil samples from the Afferdensche and Deestsche Waarden, floodplains along the Waal river close to Nijmegen. Nine transects (straight line segments) of 47.5 meters have been sampled, see Figure 1.1. In each transect, 96 reflection spectra are recorded (intervals of 0.5 meters), and every 10 meters a soil sample has been taken. In total, there are 54 soil samples. Several soil parameters have been measured for each sample, see Table 1.3.
Finally, for the multivariate regression part the stackloss data set, used in Chapter 9 of the Handbook [1] is available from the website; again, you should try to reproduce the examples given.

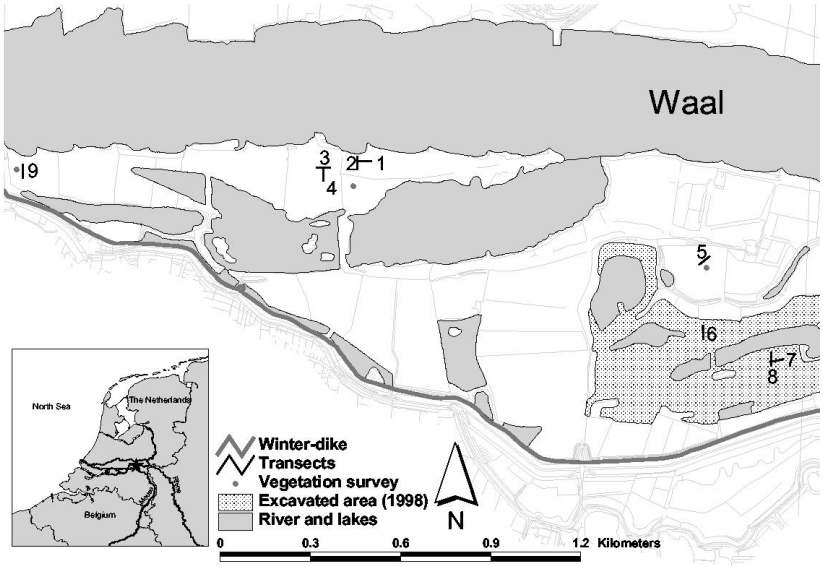


***Figure 1.1*** *- Location of transects in the Afferdensche and Deestsche Waarden.*

***Table 1.3*** *- Soil data from several locations in the Afferdensche and Deestsche Waarden. The Zn and Zn1 parameters both measure the same zinc concentration, but with two different analytical techniques.*

| Variable no. | Description |
|---|---|
| **1** | Transect number (Tr) |
| **2** | Distance on transect |
| **3** | Sample id. |
| **4** | Ni concentration (mg/kg) |
| **5** | Cd concentration (mg/kg) |
| **6** | Cu concentration (mg/kg) |
| **7** | Zn concentration (mg/kg) |
| **8** | Zn1 concentration (mg/kg) |
| **9** | Pb concentration (mg/kg) |
| **10** | Se concentration (mg/kg) |
| **11** | Organic matter content |
| **12** | Texture class (1: sand, 2: sandy loam, 3: loam, 4: clay loam) |
| **13** | Moisture content initial |
| **14** | Moisture content after 5 month storage |
| **15** | Vegetation type minus half metre |
| **16** | Vegetation type at measured point |
| **17** | Vegetation type plus half metre |
| **18** | Vegetation type (1: grass species, 2: other vegetation) |

# 2. Principal Component Analysis

Massart et al. [1, 2]: Chapters 17 and 31.1-31.5

## 2.1 Multivariate data

The term "multivariate data" is used when for every sample "many" variables have been measured. How many is many, in this respect, differs per application, but in general people have severe difficulties in thinking in more than three dimensions, and multivariate techniques (see below) will therefore be useful in as few as four dimensions. The simplest form of multivariate data - also the form we will concentrate on in this course - is the one where the same variables have been measured for every sample. This allows the ordering of data in a matrix, where, by convention, the samples constitute the rows, and the variables are the columns. An example is given in Figure 2.1. Note that it is not necessary that all variables are in the same units, nor that they have the same range. The proline concentration in the wine samples, for example, is given in units that causes the numbers in the table to be much larger than is the case for the other variables. The data matrix can be seen as a representation of all samples in high-dimensional space; every wine sample shown in Figure 2.1 is a point in thirteen-dimensional space, and every variable that has been measured constitutes one "axis". This concept may be unfamiliar; in most cases, axes are chosen to be orthogonal, and therefore, independent of each other. In this example, several of the variables show appreciable correlation, and the "true" dimensionality of the space of the wine data therefore is smaller than thirteen. One important aim of multivariate analysis is to discover this "true" dimensionality, i.e. the number of variables that is actually needed to describe the information in the data.

| | Alcohol | Malic acid | Ash | … | Proline |
|---|---|---|---|---|---|
| Sample 1 | 13.20 | 1.78 | 2.14 | … | 1050.00 |
| Sample 2 | 13.16 | 2.36 | 2.67 | … | 1185.00 |
| Sample 3 | 14.37 | 1.95 | 2.50 | … | 1480.00 |
| Sample 4 | 13.24 | 2.59 | 2.87 | … | 735.00 |
| … | … | … | … | … | … |
| Sample 177 | 14.13 | 4.10 | 2.74 | … | 560.00 |

***Figure 2.1*** *- The matrix arrangement of the wine data; the wine samples—constituting the rows of the matrix—each are characterized by thirteen variables (the columns).*

There is no compelling reason to represent samples by rows and variables by columns, and one may equally well analyze the transpose of the matrix (i.e. with columns and rows interchanged). In that case, we can also say that the variables in this data set are defined by the samples: every variable then is a point in 177-dimensional space, one dimension for each sample. Both ways of looking at the data are valid and can be useful.

Further interesting information on a data matrix $\boldsymbol{X}$ can be obtained by studying the sample variance-covariance matrix (or covariance matrix, in short), given by

$$\boldsymbol{S} = \mathrm{cov}(\boldsymbol{X}) = \frac{1}{n-1}\langle \boldsymbol{X} \rangle^T \langle \boldsymbol{X} \rangle$$

where $\langle \boldsymbol{X} \rangle$ is a mean-centered matrix, i.e. a matrix where for each column the column mean has been subtracted[2]. The symmetric and square sample covariance matrix (indicated by $\boldsymbol{S}$) contains the variances of the individual variables on the diagonal, and thus immediately shows which variables show the largest variation. The off-diagonal elements show which variables change similarly (in the data set at hand, of course) and which show reverse behavior - in the latter case the covariance is negative. One difficulty with this matrix is that it depends on the units of measurement; if different variables have different units, it can be hard to interpret the matrix. The correlation matrix is another representation of information the data that does not suffer from measurement-unit effects. It only contains numbers between 1 and -1, where 1 indicates total correlation and -1 total anti-correlation. The price we pay is that we lose information on the individual variance contributions; we can no longer see which variable has the largest variation. The correlation matrix can be obtained by dividing every row and every column of the covariance matrix by the standard deviation of the corresponding column in the original data matrix:

$$\mathrm{cor}(\boldsymbol{X}) = \boldsymbol{GSG}$$

where $\boldsymbol{G}$ is the diagonal matrix containing the inverse standard deviations of the columns in $\langle \boldsymbol{X} \rangle$.

The main aims of multivariate analysis, as described in this chapter, are the summary and visualization of multivariate data. The human eye as yet is the ultimate tool for pattern recognition, and visualization therefore is of prime importance in the exploratory analysis of data. We can easily see groups in the data, outliers, and correlations, all of which can severely distort statistical analyses when not taken into account. The most important multivariate analysis tool, used in almost all branches of science where multivariate data arise, is Principal Component Analysis.

## 2.2 PCA

Principal Component Analysis (PCA) 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, and in the context of PCA they are also termed principal components (PCs).

The central idea is that many high-dimensional data sets contain a limited amount of real information, and that many variables are in fact not needed. If we look at high resolution spectra, for example, it is immediately obvious that neighboring wavelengths are highly correlated, and therefore 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. However, variable selection (as this is called) performs often unsatisfactory, and moreover is notoriously difficult: since there is no analytical solution it requires optimization techniques. In practice, many equivalent solutions exist, which makes the interpretation quite difficult.

PCA, on the other hand, provides an analytical solution optimizing a clear and unambiguous criterion. 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. Obviously, some

[2] Such a mean-centered matrix represents variation around a mean instead of variation around the origin; in many cases (but not all!) the variation around a mean value is more meaningful.

wavelengths have more weight than others, and this can help in the interpretation as well. Note that the underlying assumption is that variation equals information. Intuitively, this makes sense: one cannot learn much from a constant number.

Once PCA has defined the new latent variables, one can plot all samples in the data set while ignoring all higher-order latent variables. Usually, only a few latent variables are needed to capture more than ninety percent of the variance in the data set (although this is highly dependent on the type of data). That means that a plot of PC 1 versus PC 2 can already be highly informative. One can also investigate the contributions of the (original) variables to the important PCs.

Perhaps the most important use of PCA is in the visualization, or graphical summary, of high-dimensional data. In a later stage of the course we will see that the scores can also be used in regression and classification problems.

## 2.3 PCA: Algorithms

### 2.3.1 SVD

Currently, PCA is implemented in even 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 principal components is Singular Value Decomposition (SVD), which is fast and numerically stable. It decomposes an $n \times p$ mean-centered[3] data matrix $\boldsymbol{X}$ into three parts:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T \tag{2.1}$$

where $\boldsymbol{U}$ is a $n \times a$ orthonormal4 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[4] matrix containing the right singular vectors. Superscript $T$ indicates the transpose of a matrix. The number $a$ equals the minimum of $n$ and $p$. In PCA, the singular values and left singular vectors are usually joined to obtain matrices $\boldsymbol{T}$, the "scores". Matrix $\boldsymbol{V}$, when used in the sense of "loadings" is often indicated with the symbol $\boldsymbol{P}$:

$$\boldsymbol{X} = (\boldsymbol{U}\boldsymbol{D})\boldsymbol{V}^T = \boldsymbol{T}\boldsymbol{P}^T \tag{2.2}$$

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 new latent variables. Variables that have very low values in a specific column of $\boldsymbol{V}$ contribute only very little to that particular latent variable. The diagonal of matrix $\boldsymbol{D}$ contains the singular values (remember, the off-diagonal elements are zero). A basic result from linear algebra states that the variance of a matrix $\boldsymbol{X}$ equals the sum of the squared singular values; the squared value of a diagonal element is therefore a direct measure for how much of the variance in $\boldsymbol{X}$ is explained by that PC. 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, but now normalized. The orientation of the points in space is equal, only the numbers at the axes have changed.

[3] PCA on data that are not mean-centered can lead to some problems; see the exercises.
[4] Orthonormal means that the angle between the column vectors is ninety degrees (orthogonal), and that the sum of the squared elements in each column equals one (normalized).

### 2.3.2 Other approaches

Other algorithms can be used to obtain scores and loadings. In the early days of chemometrics, before computers became commonplace, the NIPALS algorithm was often used. In contrast to SVD, this algorithm calculates principal components one at a time; this makes it possible to stop after a certain number of PCs. With current hardware (and specialized numerical software libraries) there is no reason anymore not to use SVD. Even very large matrices can be decomposed in the blink of an eye on commonplace hardware.

In some cases, however, there is a useful trick that can be applied to speed up calculations. This can be the case when one has to perform hundreds of SVDs, something that can happen in simulations. The trick is to apply SVD not to the original data matrix $\boldsymbol{X}$ but to either $\boldsymbol{X}^T\boldsymbol{X}$ or $\boldsymbol{X}\boldsymbol{X}^T$. These matrices are symmetrical, which implies that the left and right singular vectors are equal. Depending on which form is chosen, the singular vectors of the cross product matrices are equal to either the left or right singular vectors of matrix $\boldsymbol{X}$, and the singular values are the squares of the singular values of $\boldsymbol{X}$. Instead of SVD, sometimes the eigen-decomposition is used. However, this is less stable numerically.

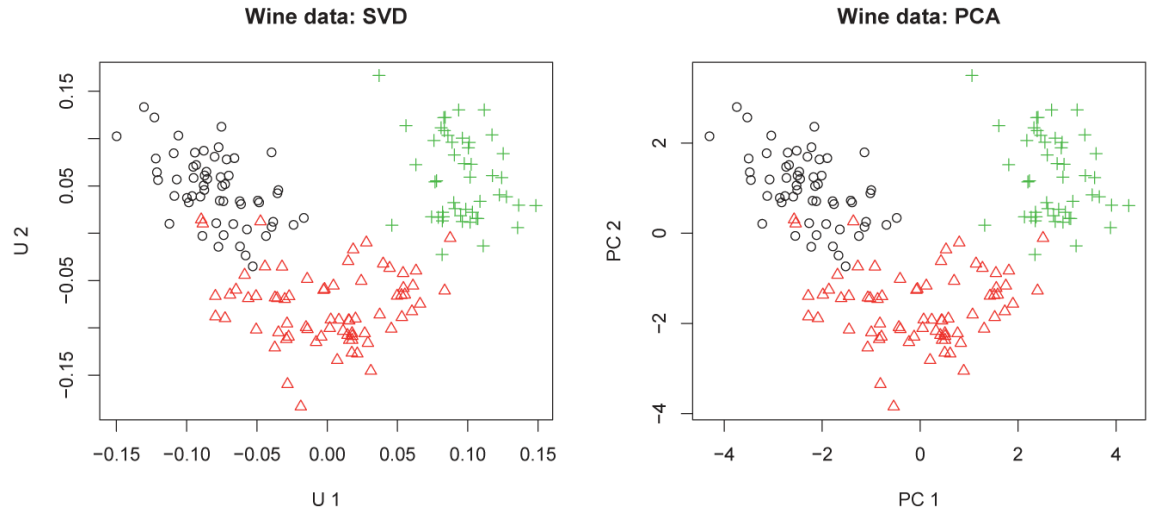


***Figure 2.2*** *- First two left singular vectors (left plot) and first two score vectors (right plot) for the wine data. They only differ in scale.*

After obtaining the singular values and the left or right singular vectors, one can easily obtain the other set of singular vectors by projection. Suppose one has found $\boldsymbol{P}$ in this way, then matrix $\boldsymbol{T}$ is easily found by right-multiplying both sides of Equation 2.2 with $\boldsymbol{P}$:

$$\boldsymbol{X}\boldsymbol{P} = \boldsymbol{T}\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{T}\boldsymbol{P}^{-1}\boldsymbol{P} = \boldsymbol{T} \tag{2.3}$$

because of the orthonormality of $\boldsymbol{P}$. In case the number of variables is much larger than the number of samples, it is faster to calculate the left singular vectors; however, to calculate the scores from these, one has to remember how to handle the singular values properly. This will be clarified in one of the exercises.

### 2.3.3 Example: wine data

As an example we here show the SVD and PCA results of the wine data set (see Table 1.2)[5]. The wines originate from the same region in Italy, but consist of three different cultivars. It may be that these lead to different "typical" values for the thirteen variables, but that is not at all certain without an investigation. Figure 2.2 shows the plots of the first versus the second column in matrices $\boldsymbol{U}$ and $\boldsymbol{T}$, respectively.

In the left plot, we see the first column of matrix $\boldsymbol{U}$ on the $x$-axis, and the second column on the $y$-axis. The symbols indicate the three different cultivars; clearly, the three classes can be distinguished from each other. In the right plot, we show the first two PCs[6]; the plot is identical to the left plot with the exception of the axes. The difference is caused by the singular values: the first value is 28.7, and the second 21.0.

## 2.4 Selection of the number of PCs

If we take $a = a_{\max} = \min(n, p)$—i.e., we look at all PCs—we get the same information as in the original data matrix $\boldsymbol{X}$: equation 2.1 holds exactly. This means we do not lose any information by doing a PCA. However, there is not much to gain either; we are still left with a large number of variables to inspect. The only advantage is that the latent variables are ordered: the first "explains" most variation, followed by the second, etcetera. At some point, however, there is no real information in further PCs. The question now is: how can we distinguish between real information and noise, or, more to the point: can we represent the information in the matrix with a smaller number of PCs? And how many PCs do we really need?

The answer is almost never unambiguous, unfortunately. Many formal statistical tests exist to determine which PCs correspond to signal and which to noise, but none of these has any claim to generality. Therefore, we advocate a less formal approach based on a graphical assessment of the variance explained by the individual PCs. For the $i$th PC, the fraction of the total variance is given by

$$\mathrm{Var}(PC_i) = \frac{d_i^2}{\sum_{j=1} d_j^2} \tag{2.4}$$

where $d_i$ is the $i$th singular value obtained from SVD.

One can plot the cumulative fraction of variance explained, that is, up to and including the current PC, and determine an arbitrary cut-off value. Often, 80 or 90 percent is used. One then retains that number of PCs that explains at least this amount of variance. Plotting the logarithm of the variance explained against the PC number leads to a so-called Scree-plot; in many cases it shows a characteristic elbow, after which the explained variance does not decrease very much. For the wine data, the cumulative variance explained and the Scree plot are shown in Figure 2.3.

After having selected the appropriate number of components, say $a$ ($< a_{\max}$), one can build a reconstruction $\widetilde{\boldsymbol{X}}$ of the original data matrix $\boldsymbol{X}$:

[5] The data have been autoscaled to nullify the effect of the different measurement scales of the variables; see later in this chapter.

[6] In practice, the term "PC" can indicate scores or loadings; the context usually makes it clear which one is meant. In this case, we show score vectors.

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^T + \boldsymbol{E} = \widetilde{\boldsymbol{X}} + \boldsymbol{E} \tag{2.5}$$

where $\boldsymbol{T}$ and $\boldsymbol{P}$ indicate score and loading matrices with fewer than $a_{\max}$ components. Matrix $\boldsymbol{E}$ contains the residuals; if $a$ is well-chosen, $\boldsymbol{E}$ should contain little or no relevant structure. However, this is hard to assess since $\boldsymbol{E}$ has the same dimensions as $\boldsymbol{X}$, and this matrix was judged too large to inspect visually... A final way to check the appropriateness of the chosen number of PCs, especially suited to spectral data, is to plot the loadings (see below) of the components that have not been selected: if these show "random" behavior then they probably have been rightly ignored. If, on the other hand, clear structure is visible, there is a chance that they contain important information.

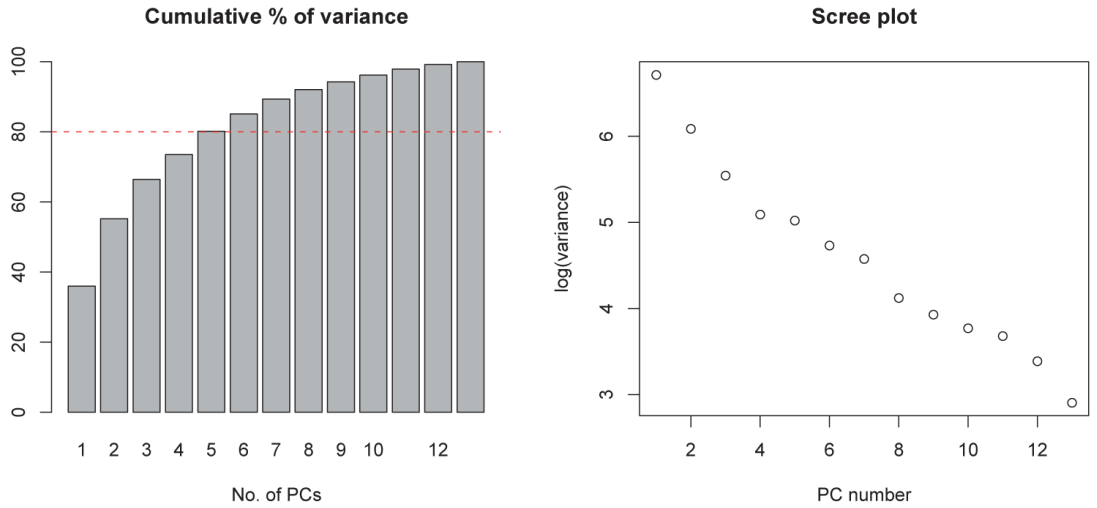


***Figure 2.3*** *- Determining the number of significant PCs for the wine data, based on cumulative percentage of variance explained (left) and a Scree plot (right). If one aims to have at least 80% variance explained, one would choose five PCs; both plots are not very conclusive for these data.*

## 2.5 Visualization

As already stated earlier, one of the main applications of PCA is the visualization of high dimensional data. The human brain still is the most powerful pattern recognition tool, so presenting plots that show all the relevant information and leave out the uninteresting stuff clearly is a worthwhile strategy. PCA comes with basically two types of plots: score plots, that show the samples in the data set in relation to each other, and loading plots, showing the contributions of the original variables to the principal components. A combination of the two, a so-called biplot is also possible, and this can be a really powerful tool. It does take a bit of thought, however, to fully appreciate all possibilities.

### 2.5.1 Score plots

In score plots, one shows the samples of the data set in the coordinate system of the principal components. An example is already given in Figure 2.2. Every symbol in that plot corresponds to one sample (i.e. one wine sample); different plotting symbols are used for wines from different cultivars. In other cases such a classification may not be available, but even in that case one might suspect a three-class grouping, just from looking at the score plot. In the pairs plot shown in Figure 2.4, all combinations of score plots from PC 1 to 4 are given. In this figure, the score plot shown on the right in Figure 2.2 can be found as the left plot in the second row (PC 1 is on the $x$-axis, PC 2 on the $y$-axis). In Figure 2.4, one can see that the three-class grouping is only visible in the plot of PCs 1 versus 2.

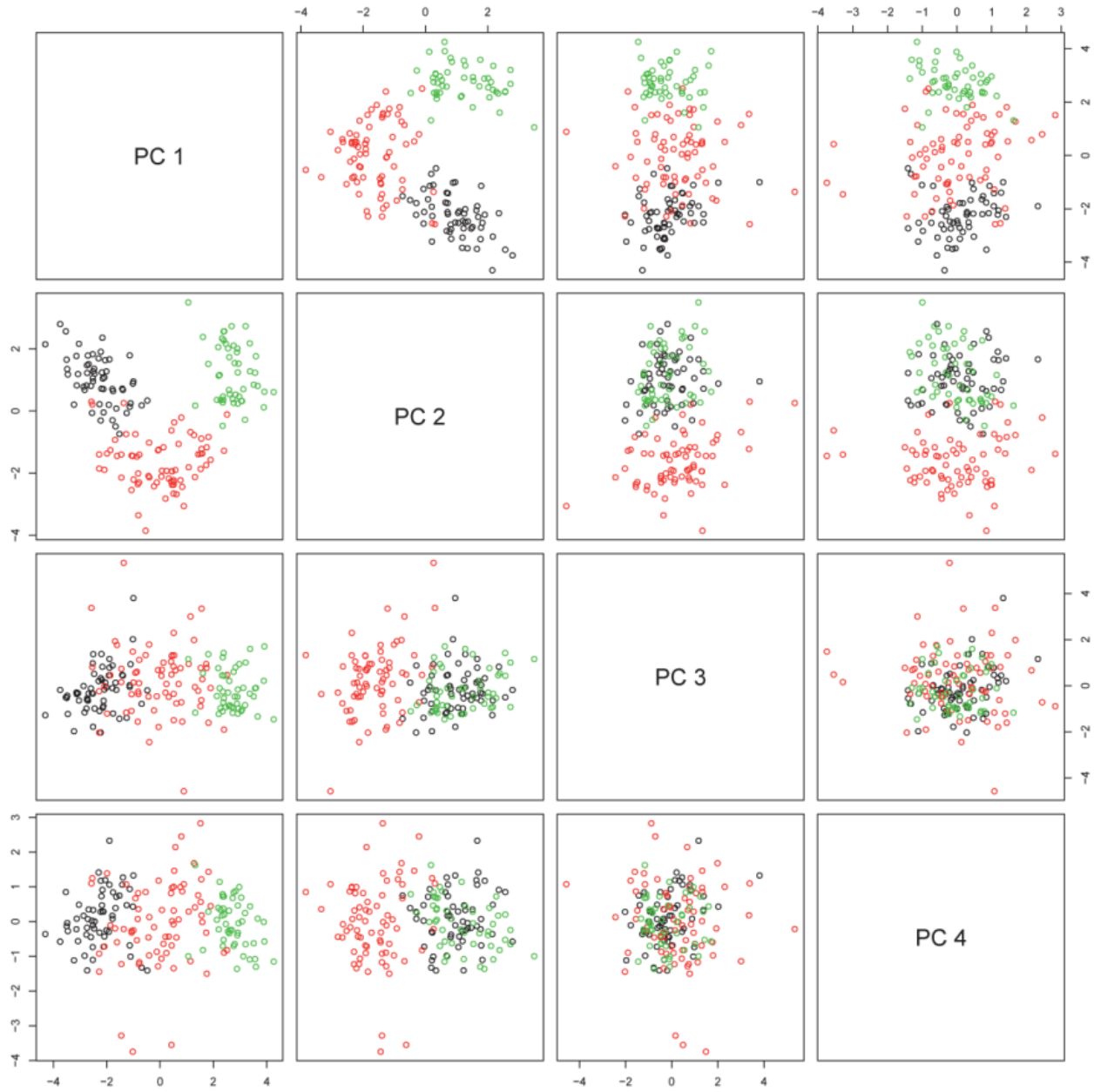


***Figure 2.4*** *- All possible score plots for PCs 1-4 (wine data). The three-class grouping is especially visible in the plots of PC 1 (separating in fact all three classes) against 2 (separating one against the two others); PCs 3 and 4 contain no class information.*

### 2.5.2 Loading plots

Loadings indicate how much the original variables contribute to every single one of the principal components. For instance, because the wine data set contains 13 original variables, the loading vector for every PC will have 13 numbers. Small numbers (i.e., close to zero) indicate a small influence of that particular variable to that PC; large numbers indicate a large influence. The first two loading vectors for the wine data are plotted in Figure 2.5.

The extreme values for PC 1 are found in variables 6, 7 and 12. These correspond to the concentrations of the total of phenols, flavonoids (both very important for the taste), and the

OD280/OD315 of diluted wines, respectively (see Table 1.2). PC 2, on the other hand, is more influenced by variables 1 (alcohol content), 10 (color intensity) and 13 (proline concentration).

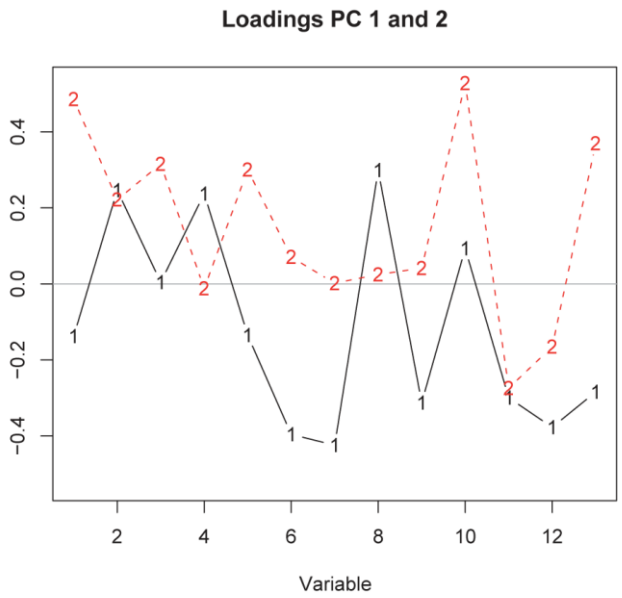


***Figure 2.5*** *- The first two loading vectors of the wine data.*

### 2.5.3 Biplots

One can plot scores and loadings in the same plot: this is called a biplot. Because scores and loadings usually are on different scales, their ranges have to be reconciled. In the example below, the axes on the bottom and left sides of the plot relate to scores (i.e. coordinates of samples on the first two PCs), and axes on the top and right sides relate to loadings (i.e. the coordinates of the original variables on the first two PCs). There are more subtleties involved: one can "distribute" the singular values evenly over scores and loadings (multiply both by the square root of the singular values) or join singular values with loadings rather than with scores. In all cases, the resulting plots look quite similar. Here, we will stick with the convention from Equation 2.2. The consequence of this is that distances between samples are preserved, while distances between variables become distorted.

Leaving that issue as it is, we will now briefly discuss the basic points on how to read these plots. The biplot for the wine data is shown in Figure 2.6. First of all, the regular score plot of PC 1 versus PC 2, (the right plot in Figure 2.2) is visible. The three class structure is not indicated here. On top of that, the biplot shows an alternative for the loading plot of Figure 2.5: the arrows from the origin. This particular visualization provides information about the correlation between variables: all arrows pointing in roughly the same direction are correlated. Conversely, arrows pointing in perpendicular directions are not correlated. In Figure 2.6 we can see that e.g. variables 6, 7 and 9 are highly correlated with each other (correlations in the reconstructed matrix $\tilde{\boldsymbol{X}}$ larger than 0.995), anticorrelated with variables 4 and 8 ($r$-values smaller than -0.985), while variable 10 (a.o.) is more or less uncorrelated with these ($|r| < 0.25$). Furthermore, we can immediately see which original variables are important in the individual PCs. The variables with the highest positive loadings on PC 1 are 2, 4 and 8; the ones with the largest negative loadings are 6 and 7. This is in agreement with Figure 2.5 (check this!).

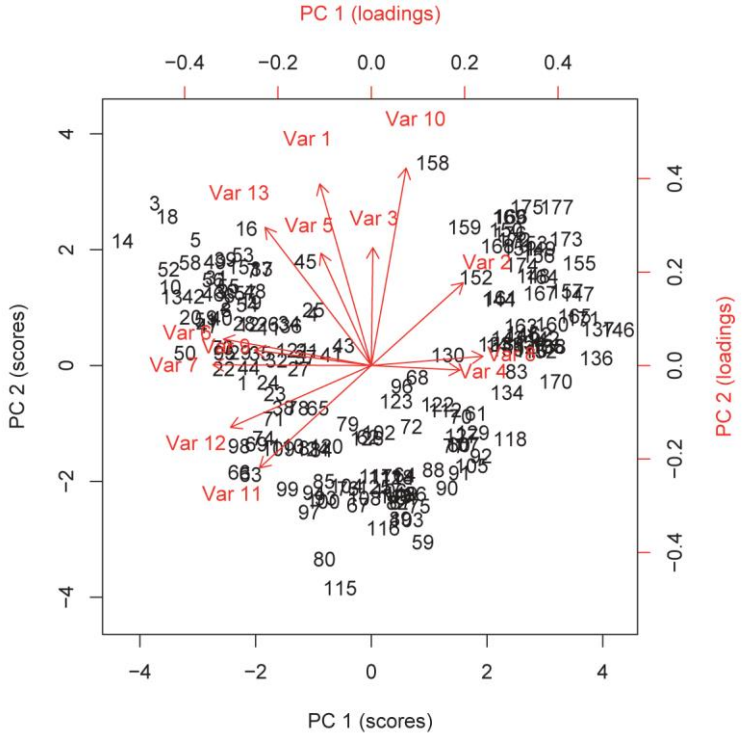


***Figure 2.6*** *- Biplot for the wine data, PCs 1 and 2. Loadings are indicated in red arrows; the corresponding scale is to the top and to the right. The scale on the bottom and to the left corresponds to scores, indicated with circles.*

Finally, one can immediately see which samples have high values for particular variables. If they are far away from the origin, and lying in the same direction as the arrow for a particular variable, the corresponding element in the reconstructed matrix $\widetilde{\boldsymbol{X}}$ will be large.

From Figure 2.6 we can easily see that the samples with the highest values for variable 10 are 158, 175 and 177, respectively, while the samples with the largest negative values for that variable are 115 and 80. The exact value in the reconstructed matrix can be found by multiplying the length of the loading arrow with the length of the projection of a sample to that arrow. From the above discussion it should be clear that one should be very careful in drawing conclusions when the residuals $\boldsymbol{E}$ are large, or put differently, when two principal components are not enough to describe the data.

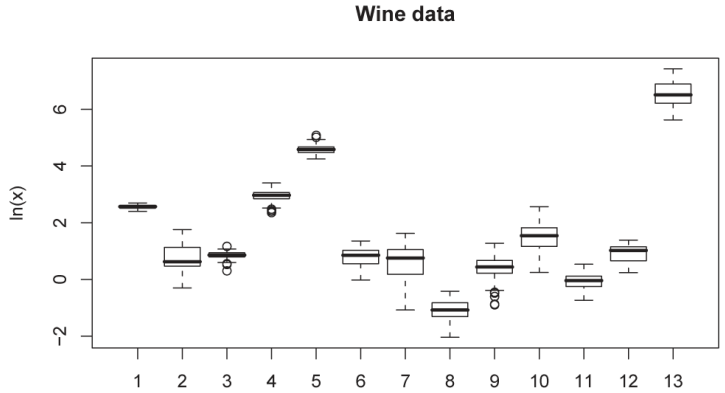


***Figure 2.7*** *- Boxplots for the thirteen variables in the wine data set (logarithmic scale).*

## 2.6 Projections

Recall that the PC loadings, linear combinations of original variables, themselves are variables spanning a space in which all points can be projected. The projections are given by the scores. However, using Equation 2.3 one can also project data into that space that have not been used to define the PCs, provided that the same variables are present in both the new and original data. Suppose that after the original data have been measured and analyzed using PCA, another, similar data set is measured (e.g., the same form of spectroscopy on new samples), and one is interested in the relationship between the two data sets. Because of day-to-day variation, instrument shifts or other influences on the measurement process, there may be appreciable differences between the two sets. By plotting the PCs (either concentrating on scores and loadings, or looking at biplots) one can easily assess whether these differences are important or not. We will see another important use of projection into PC space later in the course, in the chapters on classification and regression.

## 2.7 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. The most-often used scaling methods are mean-centering, mentioned earlier, where one replaces every column by the column minus the mean of that column, and autoscaling, where one standardizes by first performing mean-centering, and then dividing by the standard deviation (again, per-column). The latter method is appropriate in cases where the variables are measured in very different units, or have very different ranges. This, e.g., is the case with the wine data. Boxplots for all thirteen variables are shown in Figure 2.7 in logarithmic units: if we would have shown the original values, only proline (variable 13) would have been visible! Now, one can see that the proline values are in the hundreds (mean value equals 745) whereas variable 8 (non-flavonoid phenols concentration) has values smaller than one (mean equals 0.36). As a contrast, Figure 2.8 shows the same plots for mean-centered and auto-scaled data, respectively. Logarithmic scaling, in these plots, cannot be used because the data contain negative values (because of the mean-centering step).

The PCA results shown earlier have been obtained with auto-scaled data. Just for comparison, in Figure 2.8 we show the score plots for PC 1 and 2 with the raw data (no scaling) and mean-centering, respectively. Clearly, doing a PCA on the unscaled data would have PC 1 (the loadings, that is) consist almost exclusively of proline concentration[7].

For spectral data, on the other hand, column-by-column autoscaling can be lethal, especially in spectra that show clear peaks, such as IR or NMR spectra. Every spectral variable is set to the same standard deviation, whether that variable corresponds to a real peak or just noise (baseline). In those cases, mean-centering is much preferred. Specialized pre-processing methods are also used a lot in spectroscopy. In some forms of spectroscopy (such as IR) one may encounter scatter effects (the surface of the sample influences the measurement), which lead to 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 wavelength rather than the intensities themselves). Another scaling method that can be used to tackle those situations is SNV-scaling (Standard Normal Variate): this method essentially does autoscaling on the rows instead of the columns. That is, every spectrum will after scaling have a

[7] Why not try this at home?

mean of zero and a standard deviation of 1. The main idea is to get rid of arbitrary offsets. Especially in infrared spectroscopy this form of scaling has become very popular. 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.

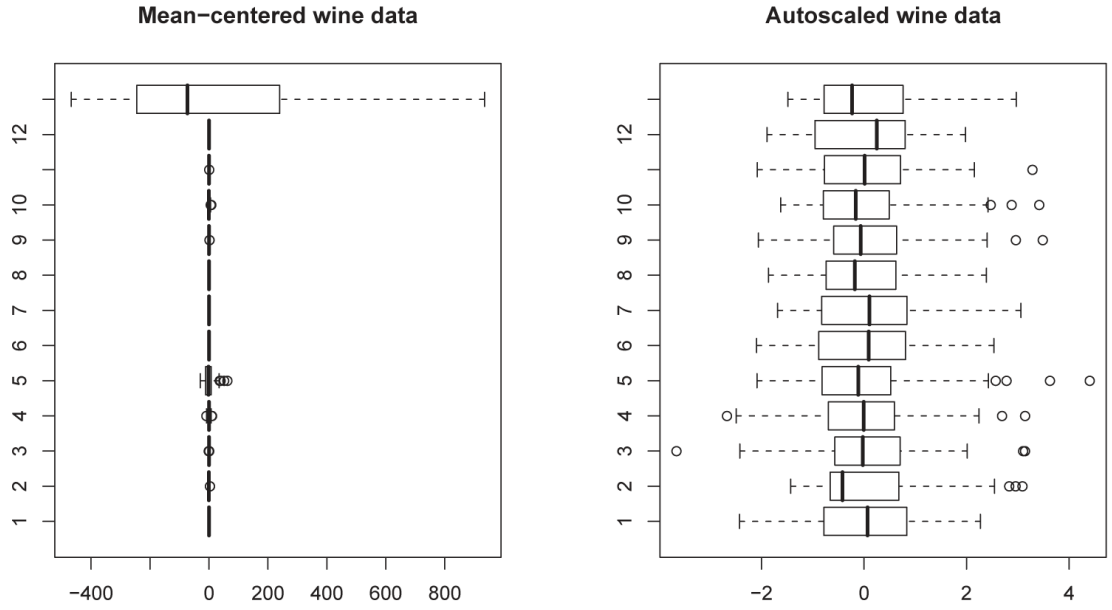


***Figure 2.8*** *- Boxplots for the thirteen mean-centered variables (left) and auto-scaled variables (right) in the wine data set.*

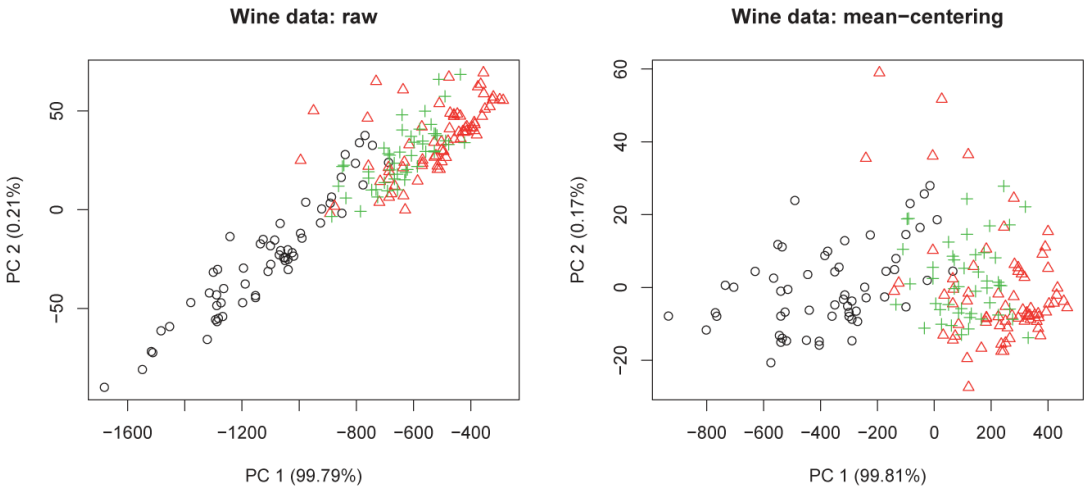


***Figure 2.9*** *- Score plots for PC 1 and 2 using raw data (left) and mean-centered data (right) for the wine data set. Compare these plots with Figure 2.2, where the first two PCs of the auto-scaled data are shown.*

## 2.8 Concluding remarks

PCA gives directions along which one has the “best” view of the data, according to the criterion of maximal variance. Obviously, one can reverse the sign of the axes without sacrificing anything: therefore one is free to choose the sign of e.g. the scores. Since the product of scores and loadings should approximate the original data, reversing the sign of the scores will automatically lead to a

reversal in sign of the loadings as well. Most computer packages have some rule of thumb, in order to achieve some sort of consistency in the sign of (e.g.) the loadings of the first principal component. However, different packages may use different conventions. As long as one remembers that one may mirror score and loading plots, there is no problem. As an example, in the loading plots for PC 1 and 2 of the wine data (Figure 2.5), one should not draw conclusions from the fact that the loadings on PC 1 for variables two and eight are positive. What is important is that they have the same sign: malic acid (variable 2) and non-flavonoid phenols (8) have a similar effect, which is opposite to the effect of proanthocyanins (9). One may arbitrarily flip the sign of the loadings, because this will be accompanied by a corresponding sign change in the scores.

A final remark is that the criterion of maximal variance is not the only property one can think of. Variation to some extent equals information, and indeed, a variable that has a constant value does not say anything about differences between samples - and perhaps is constant for all conceivable samples. 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 is a generalization of PCA where a number of different criteria can be optimized. One can for instance choose a viewpoint that maximizes some grouping in the data. In general, however, there is no analytical solution for any of these criteria, except for the variance criterion. A special case of Projection Pursuit is Independent Component Analysis (ICA), where the view is taken to maximize deviation from multivariate normality; to make things complicated, there are again several ways to achieve this. In this course, we will not pursue these directions any further.

# 3. Cluster analysis

Massart et al. [2]: Chapter 30

## 3.1 Introduction

To get insight in the structure of high dimensional data sets, one can try to group or cluster the objects. Usually, this is based on object-wise similarities or distances. The processes called cluster analysis, or clustering, and since the late nineties it has become hugely popular because of the advent of high-throughput measurement techniques in biology, such as DNA microarrays. There, the activities of tens of thousands of genes are measured, often as a function of a specific treatment, or as a time series. Of course, the question is which genes show the same activity pattern: if an unknown gene has much the same behavior as another gene of which it is known that it is involved in, e.g., cell differentiation, one can hypothesize that the unknown gene has a similar function.

Unfortunately, with a slight exaggeration one could say that there are about as many clustering algorithms as there are scientists and by no means do they always give the same results. In other words, one can always get the result one wants, if one only tries enough methods.

One of the reasons for this is that most clustering methods are not backed by statistical theory, so that it is difficult to perform statistical tests. Moreover, assessing the quality of the clustering, or validation, is a problem: although one can easily perform every single clustering method on a particular data set, it is very difficult to choose the best result. There is no obvious criterion that can be used, since different methods "optimize" different things. Users often do not have a good idea of the assumptions behind the clustering methods, and the particular characteristics associated with them.

In this chapter, we concentrate on two popular classes of methods. The hierarchical methods are represented by single, average and complete linkage, respectively, while $k$-means is an example of partitional methods. We will also consider the effect of pre-processing of the data.

## 3.2 Hierarchical methods

Quite often, data have a hierarchical structure in the sense that groups consist of mutually exclusive sub-groups. This is often visualized in a tree-like structure, called a dendrogram. An example showing the distances between several mammalian species, as calculated from the alignment of the DNA sequences of the interphotoreceptor retinoid binding protein (532 nucleotides) [3], is depicted in Figure 3.1. The dendrogram presents an intuitive and appealing way for visualizing the hierarchical structure: the y-axis indicates the "distance" between different groups, whereas the connections show where successive splits (or joins) take place. Obviously, whales and dolphins are closely related. This results in a merge of the two classes at a very small distance. The elephant is more related to the sea cow than to the horse. The marsupial mole and the wombat, two animals living in Australia, are more similar to each other than to the other species in this plot.

### 3.2.1 Algorithms

Hierarchical clustering starts with a square matrix containing distances or (dis)similarities; in the following we will assume we have the data in the form of distances. It is almost always performed in a bottom-up fashion: starting with all objects in separate clusters, one looks for the two most similar clusters and joins them. Then, the distance matrix is updated. It is here that there are several

possible choices. How do you determine the distance between two clusters containing more than one member?

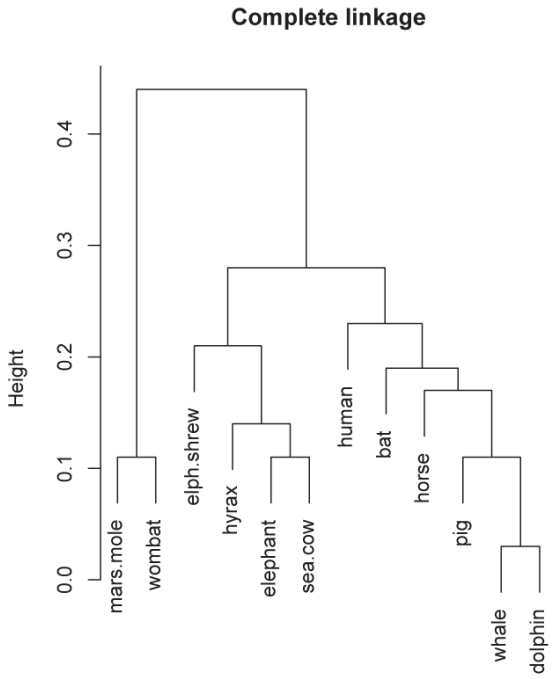


***Figure 3.1*** *- Dendrogram from complete linkage hierarchical clustering of the similarities between twelve mammalian DNA sequences.*

One option is to take the shortest distance between clusters. This leads to the single-linkage algorithm. It joins two groups if any members of both groups are close together, a strategy that is sometimes also referred to as "friends-of-friends". This means that a friend of a friend, even though you may not even know him/her, is automatically a friend, too. Typically, this clustering leads to dendrograms with one large cluster and a few small clusters, maybe even individual objects.

The opposite strategy is complete linkage clustering: there, the distance between clusters is determined by the objects that are furthest apart. In other words: to belong to the same cluster, the distances to all cluster members must be small[8]. This strategy leads to much more compact and rounded clusters. In Figure 3.2 some of the different distances between clusters are visualised.

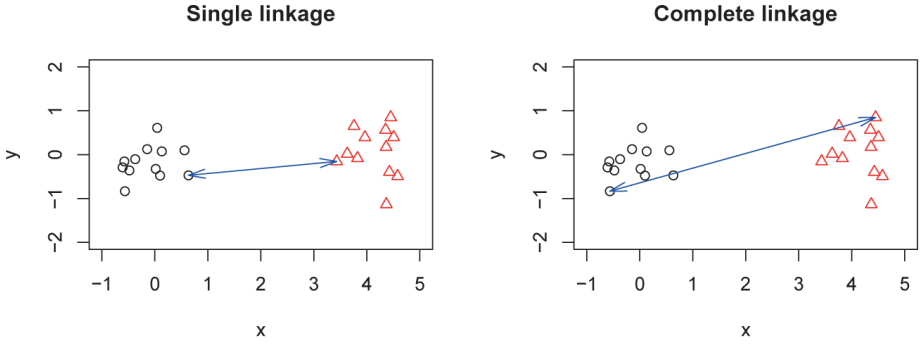


***Figure 3.2*** *- Distances between clusters: single linkage (left) takes the smallest possible distance, whereas complete linkage (right) the largest possible distance.*

[8] We can only be friends if all our friends are friends of both of us.

Of course, intermediate strategies are possible, too. Taking the average distance between cluster members leads to average linkage, whereas Ward's method, explicitly taking into account the number of objects in a cluster, leads to clusters that are very similar to complete linkage.

### 3.2.2 Choosing the optimal number of clusters

In principle, a dendrogram from a hierarchical clustering method in itself is not yet a clustering, since it does not give a grouping as such. However, one can "cut" the diagram at a certain height, and all objects that are connected are supposed to be in one and the same cluster. In the example of the twelve species, cutting the dendrogram at a height of 0.25 leads to three clusters. One is the "Ozzie" cluster containing the wombat and the marsupial mole, the second is the cluster of the elephant and relatives, and the third is a cluster containing humans, pigs and bats. Note that lowering the cut-off to a value of 0.22 would lead to four clusters, where humans are split off from the pig cluster.

So, which is "correct"? Are we really in the same group as pigs and horses[9] , or are we a separate of our own? The answer, of course, is neither: we should not forget we are forcing a continuous scale of differences into a discrete class structure. This can be very interesting and even helpful, but we should take care not to place too much confidence in the results. There are a number of cases where the results of clustering can be misleading, even. The first is the cases where in reality there is no class structure. Cutting a dendrogram will always give you clusters: there is no warning light flashing when you investigate a data set with no class structure, unfortunately. Furthermore, even when there are clusters, they may be too close to separate, or they may even overlap. In these cases it is impossible to conclude anything about individual cases (although it can still be possible to infer characteristics of the clusters as a whole).

The two keys to get out of this conundrum are formed by the use of prior information, and by visualization. If you know class structure is present, and you already have information about part of that structure, the clustering methods that fail to reproduce that knowledge obviously are not performing well, and you are more likely to trust the results of the methods that do find what you already know. In the case of the twelve species, the fact that the two Australian species form a class on their own as soon as the group is split in more than one cluster surely makes sense, as does the close resemblance between whales and dolphins.

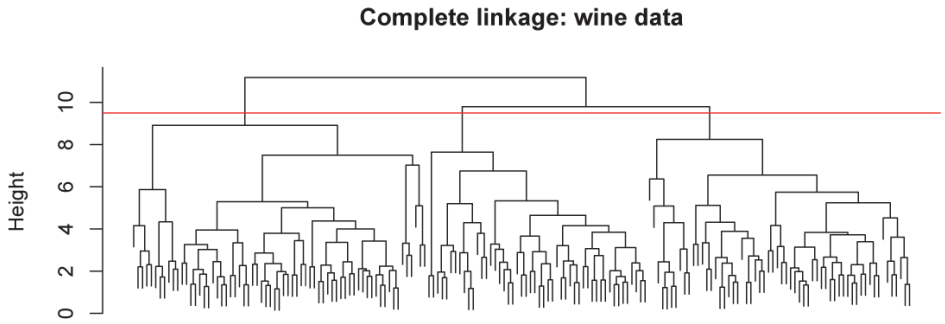


***Figure 3.3*** *- Complete linkage clustering dendrogram of the wine data.*

Another idea is to visualize the (original) data, and give every cluster a different color and plotting symbol. One can easily see if clusters are overlapping or are nicely separated. Cutting the complete-

[9] Only based on the similarity of a DNA sequence coding for some protein, of course...

linkage dendrogram of the wine data, shown in Figure 3.3, at such a height that three clusters are generated, we can visualize the result in a plot such as the one shown in Figure 3.4. Again, we use the first two PCs to visualize the data; complete linkage does not do too badly.

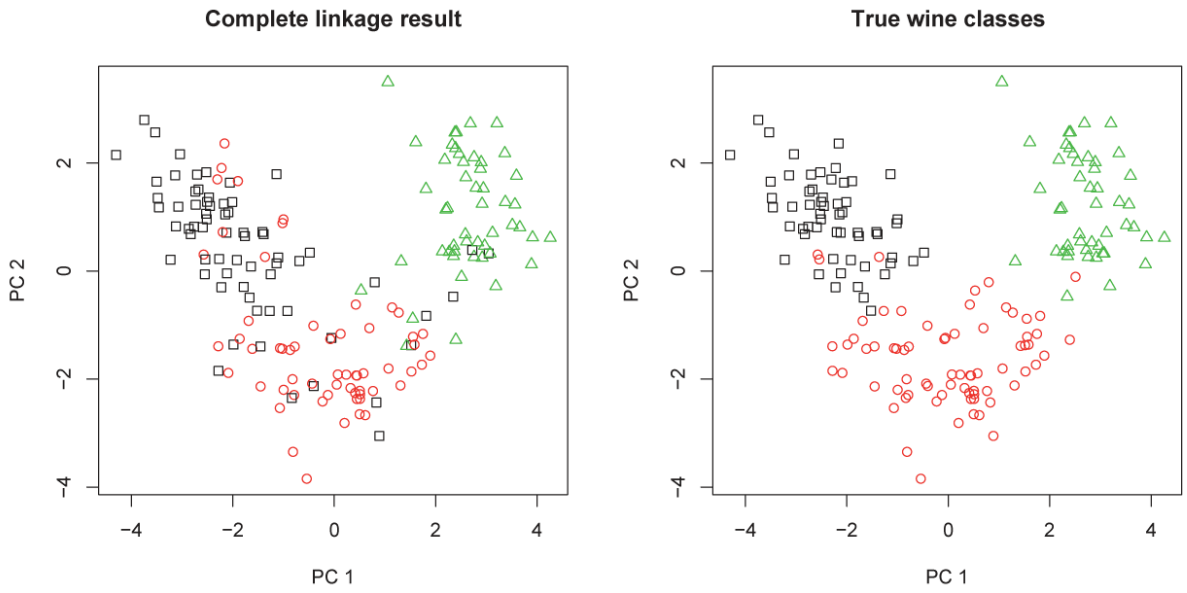


***Figure 3.4*** *- Complete linkage clustering into three clusters for the wine data set (left) and the true wine classes (right). The first two PCs are used for plotting.*

### 3.2.3 Discussion

Obviously, hierarchical clustering will work best when the data actually have a hierarchical structure: that is, when clusters contain subclusters, or when some clusters are more similar than others. In practice, this is quite often the case. In the example of the twelve mammalian species, the distances between the species are the result of the natural evolution: at some point in time species diverge because of changes in the DNA sequence and from there on follow an independent path. The idea of a common ancestor is directly reflected in the dendrogram. However, also in other applications where the hierarchical structure is not so obvious hierarchical clustering may work well.

Note that the dendrogram can be visualized in a number of equivalent ways: the ordering of the groupings from left to right is arbitrary to some extent and may depend on your software package. The tree in Figure 3.1, for example, would be exactly the same if the two Australian species are plotted on the right, or if the labels of whales and dolphins were exchanged. Any (valid) conclusion drawn from the dendrogram will not change because of this ambiguity.

If there are ties in the data, however, another form of ambiguity may be present that can change the dendrogram: if the algorithm can pick two different merges with the same distance, the result may depend on the order of the data. Usually, the implementation will pick the first available merge that has the smallest possible distance. With experimental data this will not often happen, but in cases where distances are measured by discrete numbers, or when distances are copied from a book (as is the case for the dendrogram in Figure 3.1), the number of reported figures may be quite low and ties can easily arise. It is easily checked if the ties have an effect on the dendrogram: just permute the rows and columns of the distance matrix, and recalculate the dendrogram. If after a couple of permutations no differences are found, one can be reasonably sure that the dendrogram is reliable.

Permuting the rows and columns of the similarity matrix of the evolutionary data leads—in the case of complete linkage clustering—to one distinctly different tree, shown in Figure 3.5.

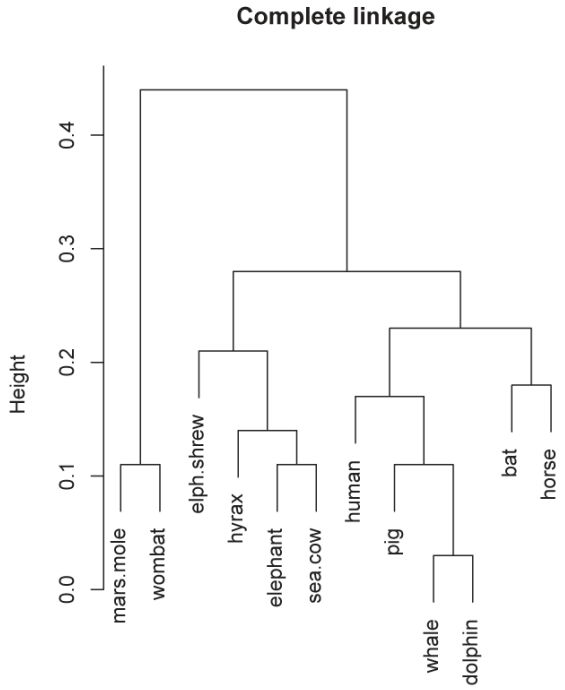


***Figure 3.5*** *- Alternative dendrogram from complete linkage hierarchical clustering of the similarities between twelve mammalian DNA sequences. This dendrogram is the result of permuting the objects.*

Obviously, there are some differences: there seem to be four "families" rather than three. Species "horse" and "bat" no longer are in the same cluster as whale and dolphin if the dendrogram is cut at a height of 0.2. Again, although in practice this ambiguity may not arise very often, one should be aware of the possibility.

## 3.3 Partitional methods: $k$-means clustering

A completely different approach is taken by the partitional methods. Instead of starting with individual objects as clusters and progressively joining adjacent clusters, they choose a set of cluster centers in such a way that the overall distance (however that is defined) of all objects to the cluster centers is minimized. The algorithm is iterative and usually starts with random cluster centers; it ends when no changes in the cluster assignment of individual objects is observed. Again, many different flavors exist, each with its own characteristics. In general, however, these algorithms are very fast and are suited for large numbers of objects. The calculation of the complete distance matrix is unnecessary—only the distances to the cluster centers need to be calculated, where the number of clusters is much smaller than the number of objects—and this saves cpu and memory resources.

### 3.3.1 Algorithm

The $k$-means algorithm is very simple and basically consists of two steps. It is initialized by a random choice of cluster centers, e.g. a random selection of objects in the data set or random values within the range for each variable. Then the following two steps are iterated:

1. Calculate the distance of an object to all cluster centers and assign the object to the closest center; do this for all objects.

2. Update the cluster centers: the new values are given by the means of all objects assigned to them.

Instead of choosing random cluster centers, one could pick a random assignment for every object and start at step two in the above algorithm. This leads to a somewhat slower convergence in general, so that almost always the strategy given above is used.

Note that, contrary to hierarchical clustering, one solution is found with a predefined number of clusters. If one repeats the algorithm with another number of clusters, there may be little or no relation between the two outcomes. This is one of the main differences with hierarchical clustering.

An example of clustering the wine data with $k$-means is given in Figure 3.6. It starts with selecting three cluster centers, in this case objects 176, 71, and 21. All objects are assigned to the nearest cluster center, using Euclidean distances. Objects belonging to clusters 1, 2 and 3 are shown in the PC plot using black squares, red circles and green triangles, respectively. The filled plotting symbols indicate the cluster centers; only in iteration 0 do they coincide with objects. Again, the clustering corresponds nicely with the true wine classes (shown in Figure 3.4).

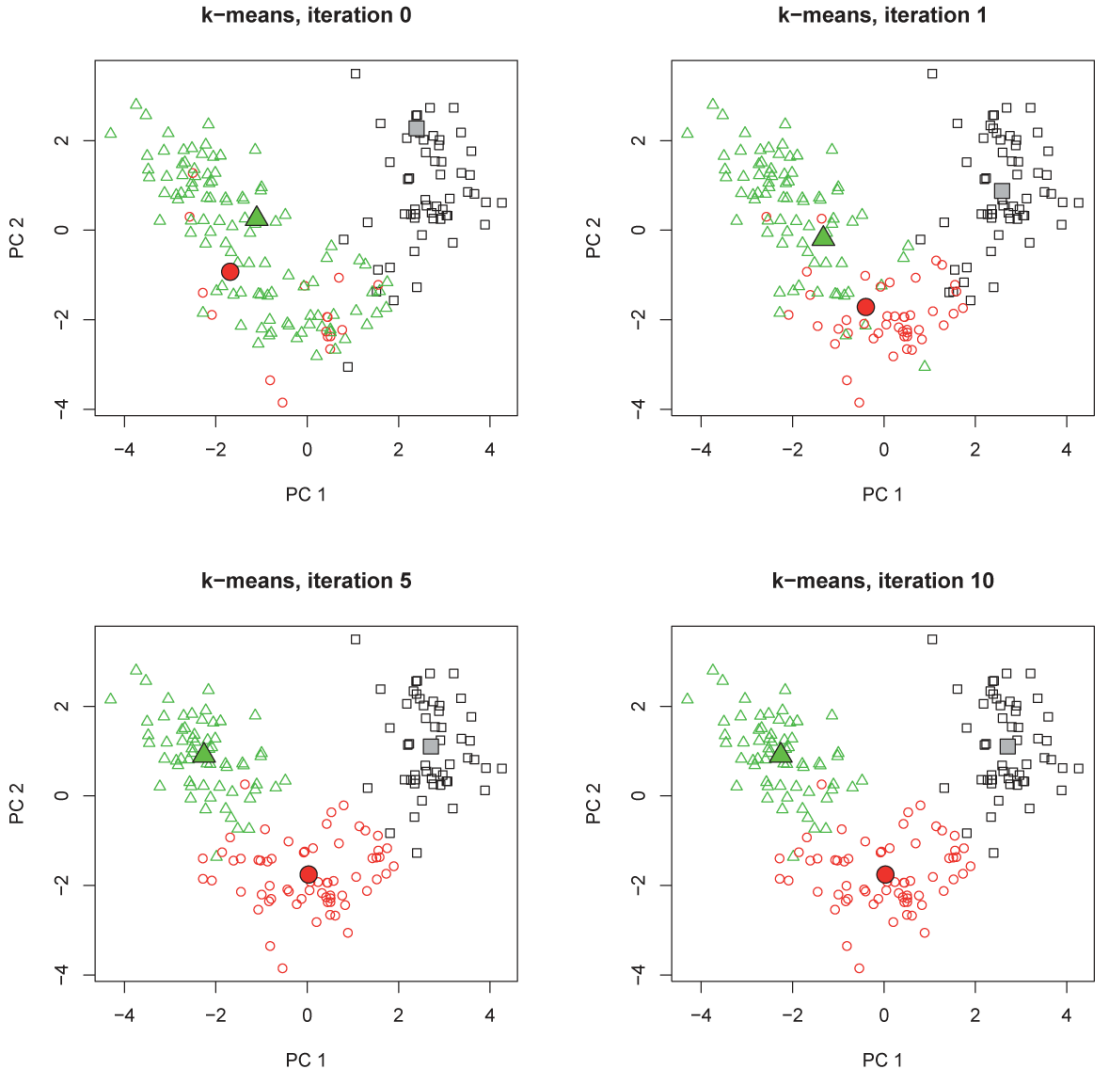


***Figure 3.6*** *- Clustering the wine data into three clusters using $k$-means; the results are shown in the first two PCs. Already in iteration 5 the final solution is reached.*

### 3.3.2 Discussion

The $k$-means algorithm enjoys great popularity through its simplicity, ease of interpretation, and speed. It does have a few drawbacks, however. We already mentioned the fact that one should pick the number of clusters in advance. In general, the correct number (if such a thing exists at all) is never known, and one will probably try several different clusterings with different numbers of clusters. Whereas hierarchical clustering delivers this in one go—the dendrogram only has to be cut at different positions—for $k$-means clustering (and partitional methods in general) one should repeat the whole clustering procedure. As already said, the results with four or five clusters may differ dramatically.

Worse, even a repeated clustering with the same number of clusters may give a completely different result! Remember that we start from a random initialization. A bad choice may get the algorithm stuck in a local minimum. One can then pick the one that leads to the smallest overall distance. Of course, several minima with comparable overall distance measure may exist, so that several different but equally good clustering solutions can be found by the algorithm. This may seem just annoying (OK, you repeat all clusterings ten times), but in practice is quite a big problem: what makes you think that clustering eleven is not much better? Or, put differently, when do you stop?

Finally, one should realize that k-means looks for spherical clusters, since in that way one minimizes the overall distance. When the Euclidean distance is used, the clusters also look spherical to our Euclidean-trained eyes; if other distance measures are used, such as the Manhattan distance (also known as the city-block distance, for obvious reasons) or the Mahalanobis distance (taking into account the covariance of every cluster), cluster shapes may seem quite different. In Table 3.1 a few examples of often used distance measures are given. Other distances can be employed to describe differences of non-real valued data, such as differences between categories.

***Table 3.1*** *– Several distance measures between vectors $x_i$ and $x_j$ .*

| *Name* | *Formula* | *Comment* |
|---|---|---|
| Squared Euclidean | $(x_i - x_j)(x_i - x_j)^T$ | Usual *squared* distance between two vectors |
| Manhattan | $\sum \lvert x_i - x_j \rvert$ | Absolute difference between two vectors |
| Maximum | $\max(x_i - x_j)$ | Maximal difference between elements of two vectors |
| Mahalanobis[10] | $(x_i - x_j)\Sigma^{-1}(x_i - x_j)^T$ | *Squared* distance using covariance matrix $\Sigma$ |

## 3.4 Concluding remarks

Hierarchical methods are suitable in cases with not too many objects, where there is a hierarchical structure, such as subclusters. Many variables are not too much of a problem. The rate-limiting step is the calculation of the distance matrix. The dendrogram provides an appealing presentation of the cluster structure, and can be used to assess clusterings with different numbers of clusters very quickly. Partitional methods can be used with large datasets: only a small distance matrix needs to be calculated. Therefore they are fast and general.

[10] Note: $\Sigma^{-1}$ means the inverse of the covariance matrix $\Sigma$.

Both types of clustering have their share of difficulties. The choice of a distance function dramatically influences the result. Hierarchical linkage comes in different flavors, and also with partitional methods one should decide on the optimal distance. Both types of clustering methods yield “crisp” clusters, that is, objects are assigned to exactly one cluster, without any doubt. For partitional methods, there are alternatives where each object gets a membership value for each of the clusters. If a crisp clustering is required, at the end of the algorithm the object is assigned to the cluster for which it has the highest membership. For example, in fuzzy $k$-means clustering one minimizes

$$\sum_i \sum_j u_{ij}^2 (x_i - m_j)^2$$

where $u_{ij}$ indicates the membership of object $i$ ($x_i$) to cluster $j$ (with center $m_j$).

Partitional methods force one to decide on the number of clusters beforehand, or perform multiple clusterings with different numbers of clusters. Moreover, there can be considerable differences upon repeated clustering, something that is less prominent in hierarchical clustering (only with ties in the distance data). The main problem with hierarchical clustering is that the bottom-up joining procedure may be too strict: once an object is placed in a certain category, it will stay there, whatever happens further on in the algorithm. Of course, there are many examples where this leads to a sub-optimal clustering. More generally, there may not be a hierarchical structure to begin with.

Finally, one should take care not to over-interpret the results. If you ask for five clusters, that is exactly what you get. Suppose one has a banana-shaped cluster. Methods like $k$-means, but also complete linkage, will typically describe such a banana with three or four spherical clusters. The question is: are you interested in the peas or the pod? It may very well be that several clusters in fact describe one and the same group, and that to find the other four clusters one should actually look for more than five clusters.

Clustering is, because of the lack of “hard” criteria, more of an art than a science. Without additional knowledge about the data or the problem, it is hard to decide which one of several different clusterings is best. This, unfortunately, in some areas has led to a practice in which all available clustering routines are applied, and the one that seems most “logical” is selected and considered to describe “reality”. One should always keep in mind that this is a gross overestimation of the powers of clustering.

# 4. Classification

Massart et al. [2]: Chapter 33

## 4.1 Introduction

Classification, also known as supervised pattern recognition, aims at building predictive models for classifying future observations in one of a set of predefined classes. It is related to clustering, but the difference is that now we know what to look for; we already know the class centers and shapes. The class labels may have been obtained from information other than the one we are going to use in the statistical model. For instance, one may have data - say, concentration levels of several hundreds of proteins in blood - from two groups of people, healthy, and not-so-healthy, and one aims at building a classification model that distinguishes between the two states on the basis of the protein levels. The diagnosis may have been based on symptoms and several medical tests; it may not even be possible to distinguish patients from healthy controls, but one can try. In classification, one describes both the healthy and diseased subjects with a kind of "average" protein level profile. New cases will be classified in the group that seems most similar.

In most cases, the classes are described on the basis of a training set that serves to define the characteristics of every group. In this chapter, two of the most common approaches will be treated in some detail: a set of techniques known under the header of discriminant analysis, and methods relying on neighbor information, i.e., ranks rather than distances. The prime representants of both types are Linear Discriminant Analysis (LDA) and $k$-Nearest-Neighbours (KNN). Both have their merits and drawbacks, but as a rule of thumb they usually perform very well over a wide range of applications. More sophisticated techniques often require more data or require the data to fulfil a series of constraints in order to work well.

## 4.2 Discriminant analysis

In discriminant analysis, one often assumes a normal distribution for the individual classes: $N_p(\mu_k, \Sigma_k)$, where the subscript $p$ indicates that the data are $p$-dimensional, and the sample mean and covariance matrix for class $k$ are given by $\hat{\boldsymbol{\mu}}_k$ and $\hat{\boldsymbol{\Sigma}}_k$, respectively. One can then classify a new object, which can be seen as a point in $p$-dimensional space, to the class that has the highest probability density ("likelihood") at that point.

Consider the following univariate example with two groups: group one is $N(0; 5)$ and group 2 is $N(1; 1)$. The probability density functions (pdfs) are shown in Figure 4.1. The pdfs are given by

$$L_i(x; \mu_i; \sigma_i) = \frac{1}{\sigma_i\sqrt{2\pi}} \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right) \tag{4.1}$$

It is not too difficult to show that $L_1 > L_2$ if

$$\frac{12}{25x^2} - x + \frac{1}{2} - \ln 5 > 0$$

in this case the regions outside the interval [-0.9, 2.9]. In more general terms, one can show [4] that for one-dimensional data $L_1 > L_2$ if

$$x^2\left(\frac{1}{\sigma_1^2}-\frac{1}{\sigma_2^2}\right)-2x\left(\frac{\mu_1}{\sigma_1^2}-\frac{\mu_2}{\sigma_2^2}\right)+\left(\frac{\mu_1^2}{\sigma_1^2}-\frac{\mu_2^2}{\sigma_2^2}\right)<2\ln\frac{\sigma_2}{\sigma_1} \tag{4.2}$$

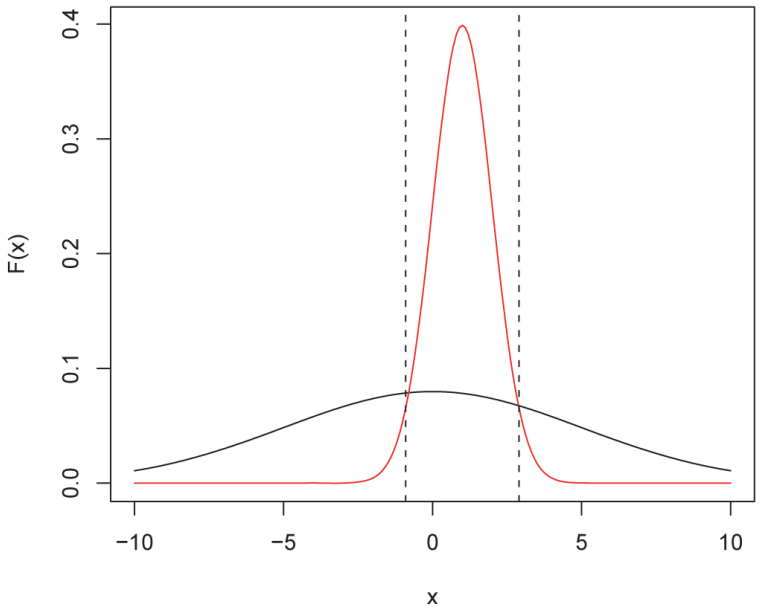


***Figure 4.1*** *- Univariate probability density functions for $N(0;5)$ (class 1) and $N(1;1)$ (class 2). The optimal discrimination is achieved by classifying everything between -0.9 and 2.9 to class 2.*

The way discriminant analysis is presented here is often called maximum likelihood discriminant analysis (ML-DA). Several other forms are possible as well; these will be mentioned—where appropriate—below.

An especially important simplification of the general DA framework is obtained when one assumes that the variances of all groups are equal. This leads to Linear Discriminant Analysis (LDA). In that case, the boundaries between classes are described by straight lines. The number of parameters that must be estimated then is relatively small: one has to estimate one covariance matrix ($p(p+1)/2$ parameters[11]), and for each group a cluster center ($p$ parameters). For $G$ groups this leads to a total of $Gp+p(p+1)/2$ estimates. On the other hand, in a situation where there are no constraints on the covariance matrix (this corresponds to Quadratic Discriminant Analysis, QDA) one has to estimate a covariance matrix for each individual class. This leads a total of $Gp(p+3)/2$ parameters, a significantly larger number, especially when the number of dimensions is large. There are even greedier forms of discriminant analysis, particularly suitable when classes are not well described by a normal distribution: one can describe each class with a mixture of normal distributions, and then assign an object to the class for which the overall mixture density is maximal. Thus, for every class one estimates several means and covariance matrices—one describes the pod by a set of peas. Obviously, this technique can only be used when the ratio of objects to variables is very large.

### 4.2.1 Linear discriminant analysis

Whole books have been written on the subject of LDA; we can only present the basics here. In the two-group one-dimensional equal-variance case, Equation 4.2, after some rearranging, becomes particularly simple:

[11] Note that the covariance matrix is symmetric!

$$(\mu_1 + \mu_2)(\mu_1 - \mu_2) < 2x(\mu_1 - \mu_2) \tag{4.3}$$

If $x > \frac{1}{2}(\mu_1 + \mu_2)$, it is classified to the largest of $\mu_1$ and $\mu_2$. A multivariate formulation of the two-group case leads to the allocation of an object $x$ to class 1 if

$$\alpha^T(x - \mu_2)^T > 0 \tag{4.4}$$

where $\alpha = \Sigma^{-1}(\mu_1 - \mu_2)^T$ and $\mu = \frac{1}{2}(\mu_1 + \mu_2)$. Note that this discriminant function is linear in $x$; hence the name Linear Discriminant Analysis (LDA). The separating hyperplane passes through the midpoint between the cluster centers, but is not necessarily perpendicular to the segment connecting the two centers. More generally, in the multi-class case an object is assigned to that class for which the (squared) Mahalanobis distance

$$(x - \mu_i)\Sigma^{-1}(x - \mu_i)^T \tag{4.5}$$

is minimal over all values of $i = 1, \ldots, G$. Again, one can show that the class boundaries are linear.

In reality, of course, one does not know the true covariances $\boldsymbol{\Sigma}$. One then uses the so-called plugin estimate $\widehat{\boldsymbol{\Sigma}} = \boldsymbol{S}$, the sample covariance matrix estimated from the data. It can be obtained by pooling the individual sample covariance matrices $\boldsymbol{S}_i$:

$$\boldsymbol{S} = \frac{1}{n - G}\sum_{i=1}^{G} n_i \boldsymbol{S}_i \tag{4.6}$$

where there are $G$ groups, $n_i$ is the number of objects in group $i$, and the total number of objects is $n$.

As an example in two dimensions with three classes, consider the wine data, and more in particular the variables flavonoids and proline. The characteristics for the three classes are given in Table 4.1, and the plot is shown in Figure 4.2. Although class 1 is less spherical than the other two, all three classes are modelled with the same covariance matrix (the numbers in Table 4.1 have been rounded to two decimal places but the exact numbers were used in the construction of the pooled matrix):

$$\begin{pmatrix} 0.28 & 0.01 \\ 0.01 & 0.31 \end{pmatrix}$$

Note that this covariance matrix is nearly diagonal, and that the numbers on the diagonal are very similar. This leads to the nearly circular contour lines in the left plot of Figure 4.2. The right plot in the same figure shows the areas assigned to the classes. For two-class problems, one can explicitly calculate the boundaries between the classes; when we do that we obtain the dashed lines that are indicated in the figure. The discriminant function for a two-group classification, given by Equation 4.4, applied pair-wise to the three classes in the example, leads to

$$x_2 = 0.48 - 0.49x_1 \text{ (1 vs 2)}$$
$$x_2 = 0.18 - 1.60x_1 \text{ (1 vs 3)}$$
$$x_2 = 1.60 - 3.60x_1 \text{ (2 vs 3)}$$

***Table 4.1*** *– Several distance measures between vectors $x_i$ and $x_j$ .*

| *Class* | *1* | *2* | *3* |
|---|---|---|---|
| $\boldsymbol{n}$ | 58 | 71 | 48 |
| Class means | $(0.96, 1.18)$ | $(0.06, -0.72)$ | $(-1.24, -0.37)$ |
| Covariance matrices | $\begin{pmatrix} 0.16 & 0.11 \\ 0.11 & 0.50 \end{pmatrix}$ | $\begin{pmatrix} 0.50 & -0.04 \\ -0.04 & 0.25 \end{pmatrix}$ | $\begin{pmatrix} 0.09 & -0.03 \\ -0.03 & 0.13 \end{pmatrix}$ |

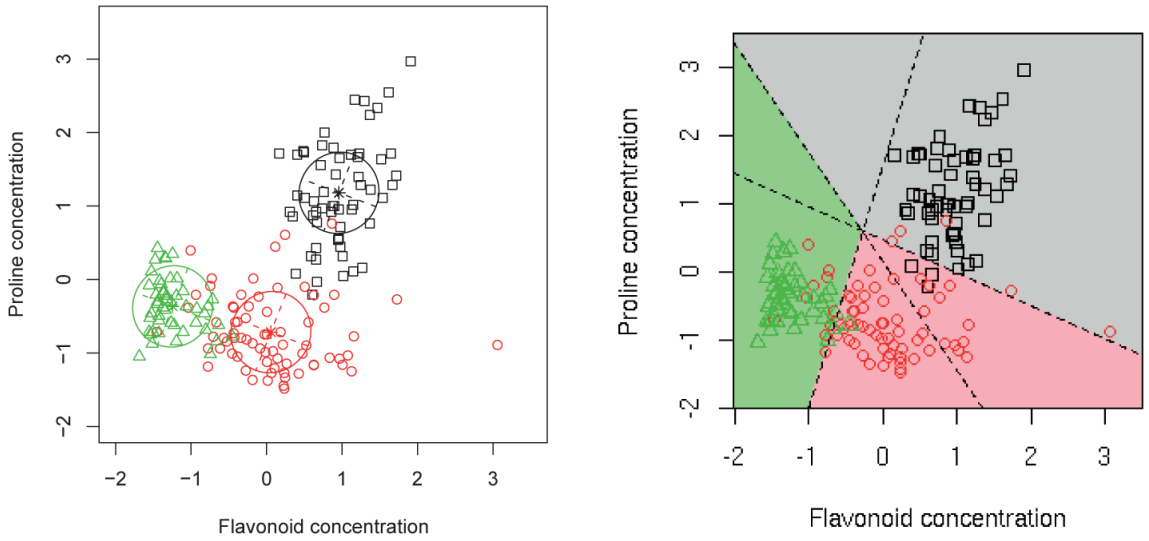


***Figure 4.2*** *- Linear discriminant analysis for the wine data using variables 7 (flavonoid concentration) and 13 (proline concentration). The left plot shows the data and the contours of constant density enclosing 95% of the probabilities; the right plot shows the areas that are classified to the respective classes.*

One can also take a more brute-force approach, and assign every point on a grid to the closest class center. This approach has been used to show the background colors in the right plot of Figure 4.2. For the more complicated forms of DA, where explicit boundaries are more difficult or even impossible to calculate, this approach will be used.

### *Fisher LDA*

Sir Ronald Aylmer Fisher (1890-1962), arguably the most influential statistician of the twentieth century, took a different approach to discriminant analysis. Rather than assuming a particular distribution for individual clusters, he devised a way to find a sensible rule to discriminate between classes. His suggestion, which is known as Fisher's LDA, is to look for a linear combination of variables a that maximizes the ratio of the within-groups sums of squares and the between-groups sums of squares:

$$\boldsymbol{a}^T \boldsymbol{B} \boldsymbol{a} / \boldsymbol{a}^T \boldsymbol{W} \boldsymbol{a} \tag{4.7}$$

where $\boldsymbol{B}$ and $\boldsymbol{W}$ are the between-groups and within-groups covariance matrices, respectively. They are calculated by

$$\boldsymbol{W} = \sum_{i=1}^{G} \tilde{\boldsymbol{X}}_i^T \tilde{\boldsymbol{X}}_i \tag{4.8}$$

$$\boldsymbol{B} = \sum_{i=1}^{G} n_i (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})^T \tag{4.9}$$

where $\tilde{\boldsymbol{X}}_i$ is the centered part of the data matrix containing objects of class $i$, and $\bar{\boldsymbol{x}}_i$ and $\bar{\boldsymbol{x}}$ are the mean vectors for class $i$ and the whole data matrix, respectively. Put differently: $\boldsymbol{W}$ is the variation around the class centers, and $\boldsymbol{B}$ is the variation of the class centers around the global mean.

Fisher's criterion is equivalent to finding a linear combination $\boldsymbol{a}$ that leads to small distances within each class, and large distances between classes. It can be shown that the maximization of Equation 4.7 leads to an eigenvalue problem, and that the solution $\boldsymbol{a}$ is given by the eigenvector of $\boldsymbol{W}^{-1}\boldsymbol{B}$ corresponding with the largest eigenvalue[12]. An object $x$ is then assigned to class $i$ if the discriminant score $\boldsymbol{ax}$ is closest to the discriminant score $\boldsymbol{a}\bar{\boldsymbol{x}}_i$ . The result for the wine data (with two variables) is shown in Figure 4.3.

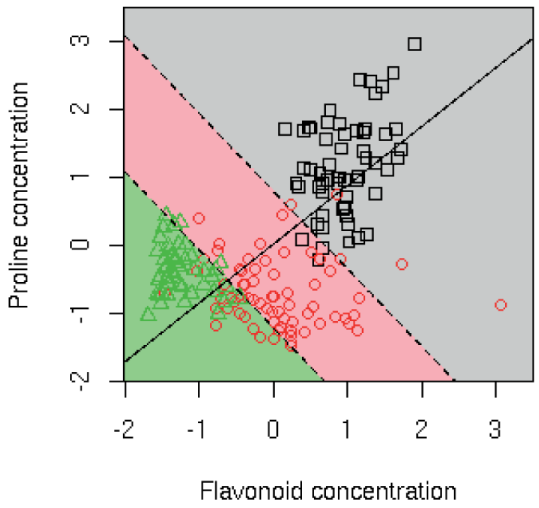


***Figure 4.3*** *- Fisher LDA for the wine data using variables 7 (flavonoid concentration) and 13 (proline concentration). The discriminant function is shown as a solid line; dashed lines indicate the boundaries between classes 1 and 2, and 2 and 3, respectively.*

Interestingly, although Fisher took a completely different starting point and did not assume normality nor equal covariances, the two-group case leads to exactly the same solution as ML-LDA. For problems with more than two groups, the results are different unless the sample means are collinear[13]. It is possible to utilize the information in the second and higher eigenvectors of $\boldsymbol{W}^{-1}\boldsymbol{B}$ as well; these are sometimes called canonical variates. The maximum number of canonical variates that can be extracted is one less than the number of groups. In this course, we will not pursue the matter further.

[12] Note that the 'eig' function in Matlab does not order the eigenvectors according to the size of the eigenvalues!

[13] Note that the text in reference [2] on page 220 presents Fisher's LDA rather than ML-LDA.

### *Taking into account prior probabilities*

In many cases it makes sense to incorporate information about prior probabilities; some classes may be more probable than others, for example. This is usually reflected in the class sizes in the training set, and therefore is taken into account when calculating the pooled covariance matrix, but it is not explicitly used in the discrimination rule. However, it is relatively simple to do so: instead of maximising $L_i$ one now maximises $\pi_i L_i$, where $\pi_i$ is an estimate of the prior probability of class $i$. In the two-group case, this has the effect of shifting the critical value of the discriminant function with an amount of $\log(\pi_2/\pi_1)$ (Equation 4.4). This approach is sometimes referred to as the Bayesian discriminant rule[14]. Obviously, when all prior probabilities are equal, the Bayesian and ML discriminant rules coincide.

One situation where this is of importance is the case where the classes have very different sizes. The training set, on which the LDA model is built, needs sufficient objects from all classes to be able to model also the smallest class; however, the larger classes are more likely to occur. One can then take equal numbers of samples from all classes so that all classes are equally well modelled, and adjust the $\pi_i$ values to reflect information on class size.

For the wine data, where the smallest group is just over half the size of the largest group, the differences between ML-LDA and the Bayesian version are shown in Figure 4.4. The background colour of the figure, and the solid lines, indicate the Bayesian result taking into account class sizes, whereas the dashed lines are the ML-LDA result. The differences are small: one can see that the largest class (red/pink) takes away a bit of space from the others, just as expected.

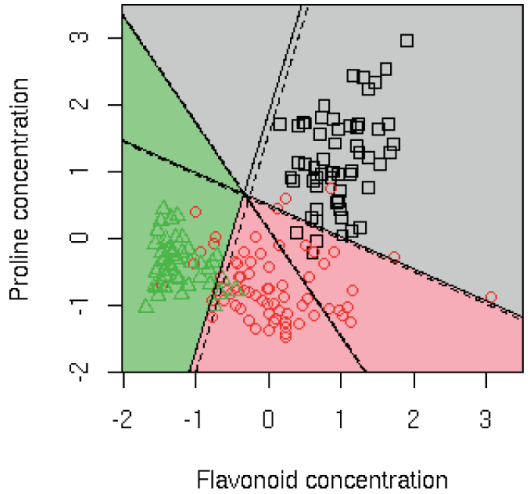


***Figure 4.4*** *- ML-LDA class separation for the wine data with two variables (dashed lines) compared to the Bayesian version taking into account class sizes (background color and solid lines).*

## 4.2.2 Quadratic discriminant analysis

Quadratic discriminant analysis (QDA) takes the same route as LDA, with the exception that every class is described by its own covariance matrix. If all covariance matrices are exactly the same, QDA and LDA lead to the same decision boundaries. The algorithm is pretty simple as well: again one calculates the Mahalanobis distances of all points to the class centers, and assigns each point to the

[14] In Bayesian statistics, it is possible to incorporate prior information in a model; the approach is quite different from classical statistics, in that the final result—posterior probabilities—depends both on the data and the prior probabilities.

closest class. The difference with LDA is that one now does not use the pooled covariance matrix, but a class-dependent covariance matrix. In QDA, one can correct for unequal class sizes (or account for prior probabilities) in exactly the same way as in LDA.

In Figure 4.5 the QDA classification is given for the wine data set (again, only two variables). The ML-LDA decision boundaries are indicated as dashed lines for comparison. The correction for unequal class size leads, as in the LDA case, to only minor differences.

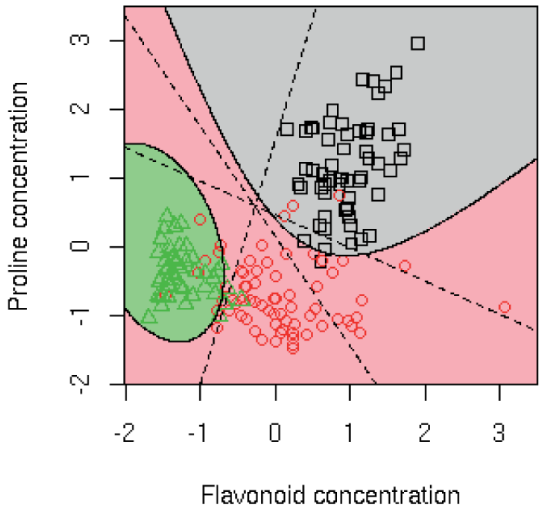


***Figure 4.5*** *- QDA class separation for the wine data with two variables (background color and solid lines) compared to the ML-LDA classification (dashed lines).*

## 4.3 Nearest-neighbor approaches

A completely different approach, not relying on any neat distribution whatsoever, is formed by a group of techniques focusing on distances between objects, and in particular on the few objects that are closest. These techniques are known under the name of $k$-nearest neighbors (KNN), where $k$ is a number to be determined. If $k = 1$, only the closest neighbor is taken into account, and any new object will be classified in the class of its closest neighbor in the training set. If $k > 1$, the classification is not so straightforward in cases where the $k$ nearest neighbors are not all of the same class. Usually, a majority vote is performed (difficult with $k = 2$!). One can even think of a classification in the category "uncertain", when there is too much discrepancy between the neighbors.

Note that the areas of the space classified to one class can be much more scattered than with LDA or QDA; in extreme cases one can even find a patch-work-like pattern. The smaller the number $k$, the more fragmented the areas will become; only one object is needed to convert its immediate surroundings to a particular class. This is illustrated for the wine data in Figure 4.6; for three different values of $k$ the boundaries between the three classes are indicated. Note the white areas with $k = 3$ and $k = 5$: in these cases, there is no preference for a class. Considering three classes, as in this example, at least two of the nearest neighbors should belong to a class if a point is to be assigned to that class with $k = 3$, and with $k = 5$, at least three of the nearest neighbors should agree in their classification. Of course, other choices can be made.

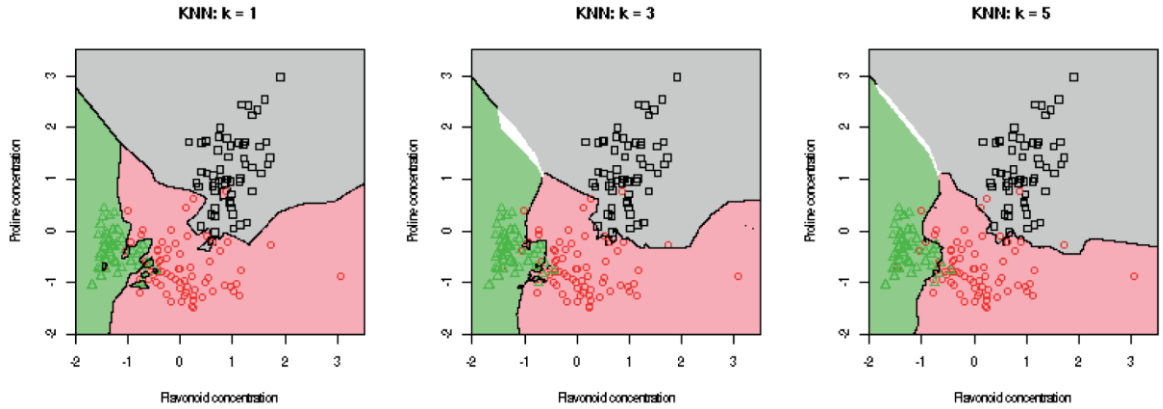


***Figure 4.6*** *- KNN classification for the two-dimensional wine data: k equals 1, 3 and 5, respectively, from left to right. White areas indicate doubt.*

One potential disadvantage of the KNN method is that in principle, the whole training set—the training set in a sense is the model! —should be saved, which can be a nuisance for large data sets. Predictions for new objects can be slow and storing really large data sets may present memory problems. However, things are not as bad as they seem, since in many cases one can safely prune away objects without sacrificing information. For the wine data, it is obvious that in large parts of the space there is no doubt: only objects from one class are present. Many of these objects can be removed and one then still will get exactly the same classification for all possible new objects.

## 4.4 Error estimation

It is important to be able to say something about the expected error when using classification techniques like LDA and KNN. The easiest and most often employed way to obtain error estimates is to split the data in training and test sets. The model is built using the training data, and the error associated with the test set is then an estimate of the error for future observations. Although the procedure is simple enough, there are a few traps to avoid. Other options to obtain error estimates are cross-validation and bootstrapping; only cross-validation will be treated here.

### 4.4.1 Using a training set and a test set

First of all, one should take care not to use any information of the test set, otherwise the error estimates are biased, i.e. too optimistic. An often-made error is to scale (autoscaling, mean-centering) the data before the split into training and test sets. Obviously, the information of the objects in the test set is being used: column means and standard deviations are influenced by data from the test set. And one further error is often made: if one uses mean-centering, for instance, for the training set, it is obvious that the test set should be centered in the same way. And “the same way” really means just that: one should use the means of the training set to scale the test set, and not the means of the test set! The same goes for methods like autoscaling. Obviously, taking logarithms or square roots is a different story: there, no information of the test set is taken into account.

Furthermore, one could say that even dividing the data set into a training and a test set should be done randomly to avoid bias; in the extreme case one could pick the test set in such a way that it is completely enclosed within the training set. Although this leads to a good prediction of the test set, it also makes the corresponding error estimate a bit smaller than it should be. Always distrust

algorithms that divide data sets into disjoint training and test sets using some non-random algorithm!

Of course, there is the possibility that an unlucky division in training and test sets leads to really unpleasant behavior. One can check whether a pathological case has been obtained by a simple visualization (e.g. using PCA), but one should be very careful not to reject a division too easily. Instead of rejecting a suspect division, it is better to generate multiple divisions in training and test sets, and average the error rates afterwards.

For the two-dimensional subset of the wine data, one can generate a test set by leaving out every third object[15]. Thus, the classification models are built using more than 66% of the data, which hopefully leaves enough information to reliably classify also the smaller class. In Table 4.2 the classification results of several of the methods presented here have been summarized.

***Table 4.2*** *– Prediction results on a test set of 59 objects for the two-dimensional wine data set. Numbers are percentages of correct classifications. The label "Bayes" with the LDA and QDA methods indicates that the group size is taken into account; with the label "ML" all groups are a priori equally likely.*

| ***LDA*** | | ***QDA*** | | ***KNN*** | | |
|---|---|---|---|---|---|---|
| ***ML*** | ***Bayes*** | ***ML*** | ***Bayes*** | ***K=1*** | ***K=2*** | ***K=3*** |
| 91.5 | 91.5 | 91.5 | 93.2 | 93.2 | 91.5 | 93.2 |

The percentages correspond to either four or five misclassifications. All classifiers misclassify objects 60, 96, and 150. All four DA approaches also misclassify object 21, while three out of four misclassify object 69. The KNN procedures on the other hand have no trouble at all classifying object 21, but instead two out of three misclassify object 81, and two out of three misclassify object 69. Clearly, different techniques have trouble with different objects! It should be noted that the data set here does not really present a difficult problem (also because the two variables were selected to present a nice separation between the three classes), but in practice the differences can be quite big.

### 4.4.2 Cross-validation

In cross-validation, the idea of multiple training and test sets is developed further. The general procedure is as follows: one leaves out a certain part of the data, trains the classifier on the remainder, and uses the left-out bit—sometimes called the out-of-bag samples—to estimate the error. Next, the two data sets are joined again, and a new test set is split off. This continues until all objects have been left out exactly once. If the size of the test set equals one, the procedure has received the name Leave-One-Out (LOO) cross-validation; it is shown to be unbiased but can have appreciable variance. That is: on average, the estimate is correct, but individual components may deviate considerably.

Better results are usually obtained by leaving out a larger fraction, e.g. 10% of the data. The largest errors cancel out to an extent so that the variance decreases; however, one pays the price of a small bias. In general, the pros outweigh the cons, so that this procedure is quite often applied. It also

[15] Danger!! If the order of the samples is not random, then we may make a systematic error here... It would be better to place the objects in random order first, and only then take every third sample. However, you would no longer be able to reproduce our results.

leads to significant speed improvements for larger data sets, although for the simple techniques presented in this chapter it is not likely to be very important. For some methods, like LDA, it is even possible to derive analytical expressions for the LOO cross-validation error. The whole cross-validation procedure is illustrated in Figure 4.7.

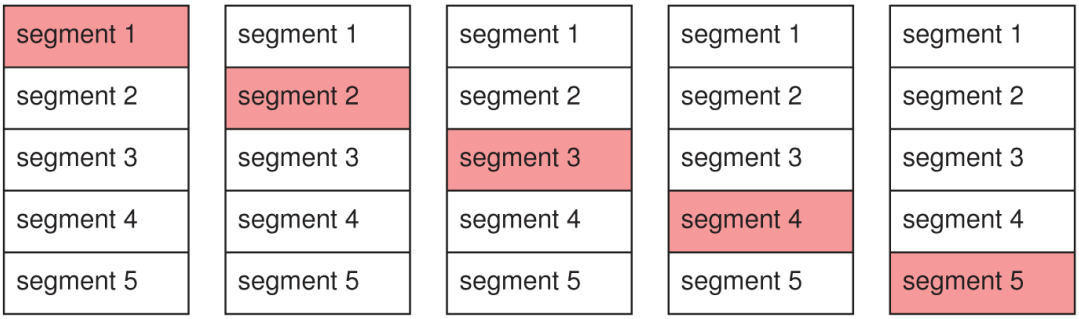


***Figure 4.7*** *- Illustration of cross-validation; in the first iteration, segment 1 of the data is left out during training and used as a test set. Every segment in turn is left out. From the prediction errors of the left-out samples the overall cross-validated error estimate is obtained.*

## 4.5 Classification: the correct way

The final goal of classification is to build a model with good predictive abilities, and an estimate of these predictive abilities in the form of an estimated prediction error. If one uses an algorithm that does not require any tuning, such as LDA or QDA, either cross validation (for small data sets) or a division in a training and test set (for large sets) will do fine. In both cases one makes a model based on a subset of the data and uses the rest of the data to estimate the prediction error. The final model will be constructed with the whole data set, and it is silently assumed that the model constructed with this bigger set will have approximately the same predictive abilities as the model(s) built using only the training data.

However, when the methods can be tweaked to improve performance, things are different. Several of the techniques discussed so far have tuneable parameters: the number $k$ in KNN is a good example. Obviously, one wants to optimize the value of $k$. This can be achieved by using one of the validation techniques mentioned above. Especially cross-validation is popular for this type of application; one performs a cross-validation for $k = 1$, $k = 3$ and $k = 5$, for example, and afterwards picks the model that leads to the smallest error estimate. The same goes for the LDA and QDA approaches if one has doubts on whether or not to include information on prior class probabilities: comparing the cross-validated error estimates will point to the optimal setting.

However, one should be aware of one major snag: when applied in this way, the error estimate from cross-validation is no longer an unbiased estimate of prediction error. It has been used to pick the optimal model, and therefore is biased downward—it is too optimistic. The solution is to combine two validation approaches. It is customary to use cross-validation for the optimization of the model (e.g. choosing the optimal value for $k$), and use an independent test set for assessing prediction error. Given enough data, the ideal strategy is then the following:

1. Divide the data in a training set and a test set. The best way to do this is to use a random division.
2. Do not even look at the test set, but put it away in a safe place, until you are satisfied with your model.

3. Using only the training data, perform a cross-validation loop, either leaving out one object at a time, or multiple objects at a time.
4. From the cross-validation, pick the parameter settings that minimize the cross-validation error, and with these settings build a model with the complete training set.
5. Use this final model to predict the test set: this gives you an unbiased estimate of prediction error.

Again, it should be emphasized that one should take care that the scaling does not use information from either the test set or the out-of-bag samples. So when autoscaling is required in the scheme above, the means and variances of the variables should be estimated from the in-bag samples, and the out-of-bag samples should be scaled using these values. The final model, built on the complete training set, can be constructed using the means and variances of the columns of the complete training set, and these values should in turn be used to scale the test set. If one fails to separate the information in the training and test phases, the result will be too optimistic.

## 4.6 Discussion

Despite their rather strict assumptions, LDA and QDA have an excellent track record in many applications. Even when data are not normally distributed and do not have equal or even similar covariances, LDA has shown to be successful in quite a lot of applications; when enough data are available, QDA has, in general, also performed quite well. How can this be? Simple models with very few parameters such as LDA enjoy one big advantage over more complex models: they are less likely to make wildly inaccurate predictions. It is an example of a bias-variance trade-off: we accept the odd error now and then if the model can be estimated with low variance [5]. Or, put differently, the data support only simple decision boundaries. For QDA, in any case, there is no hope on a good result when too few data are available; a rule of thumb is that there should be at least five times as many data points as the number of parameters that is to be estimated.

The number of classification techniques is huge; many more variants of discriminant analysis exist than can possibly be treated here. Other techniques are based on regression; it can be shown that there are intimate connections between the discriminant analysis techniques explored in this chapter and least-squares regression. Other classification methods based on regression include PLS-DA, based on Partial Least Squares regression (see the next chapter), and logistic regression, where one aims at predicting odds of belonging to a class, rather than predicting a classification directly.

Tree-based classifiers [5] are structured as a series of binary classifications; one can graphically illustrate every decision in a tree-like structure that is not unlike the dendrograms we have seen in the context of clustering. Especially in the medical sciences these tree-based classifiers are popular, perhaps because they resemble the structure of a medical diagnosis: first ask general questions and gradually go down in detail until one knows what is wrong. Moreover, they automatically provide a selection of relevant variables. Unfortunately, the predictions are usually of less quality than most other approaches[16].

Finally, one can apply powerful non-linear approaches such as Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) [5]. These offer superior modelling power, but this comes at a

[16] There is no such thing as a free lunch…

price: it is very easy to accidentally fit noise (the uninformative part of the data), which has a detrimental effect on prediction ability. In all these cases, validation is of utmost importance.

# 5. Multivariate regression

Massart et al. [1, 2]: Chapters 10.1-10.5, 35.6-35.7, 36

## 5.1 Introduction

In Chapters 3 and 4 we have concentrated on finding groups in data, or, alternatively, modelling a given grouping. The last situation is "supervised" in the sense that we use a set of examples, the training set, to build the model. In this chapter we will do something similar, but now we are not predicting a discrete class property, but rather a continuous variable. Put differently: given a set of independent variables (matrix $\boldsymbol{X}$), we want to build a model that allows prediction of $\boldsymbol{Y}$, consisting of one or possibly more dependent variables. As in almost all regression cases, we here assume that errors are only present in the dependent variables, or at least are so much larger in the dependent variables that errors in the independent variables can be ignored.

The usual multiple least-squares regression (MLR), taught in almost all statistics courses, has a number of very attractive features. It is simple, fast, and provides error estimates for regression coefficients and predictions. In matrix notation, the solution for

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{B} + \boldsymbol{\varepsilon} \tag{5.1}$$

where $\boldsymbol{B}$ is the matrix of regression coefficients and $\boldsymbol{\varepsilon}$ contains the residuals, is given by

$$\boldsymbol{B} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y} \tag{5.2}$$

and the variance-covariance matrix of the coefficients is given by

$$\mathrm{Var}(\boldsymbol{B}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\sigma^2 \tag{5.3}$$

where the residual variance $\sigma^2$ is typically estimated by

$$\hat{\sigma}^2 = \frac{1}{N-p-1}\sum_i (y_i - \hat{y}_i)^2 \tag{5.4}$$

The standard deviation of each coefficient is then obtained by taking the square root of the diagonal elements of the variance-covariance matrix $\mathrm{Var}(\boldsymbol{B})$.

This is also true when $\boldsymbol{Y}$ is multivariate, i.e., consists of more variables. Unfortunately, however, there are some drawbacks as well. In the context of the natural sciences, the most important perhaps is the sensitivity to correlation in the independent variables. This can be illustrated using the following example. Suppose we have a model that looks like this:

$$y = 2 + x_1 + 0.5x_2 - 2x_3$$

and further suppose that $x_2$ and $x_3$ are highly correlated ($r \approx 1.0$). This means that any of the following models will give more or less the same predictions:

$$\begin{aligned} y &= 2 + x_1 - 1.5x_2 \\ y &= 2 + x_1 - 1.5x_3 \\ y &= 2 + x_1 + 5.5x_2 - 7x_3 \\ y &= 2 + x_1 + 1000.5x_2 - 1002x_3 \end{aligned}$$

So what is so bad about that? If all predictions would exactly be the same, not even that much, but in practice there will be differences, especially when going away from the region of the training set. The differences will be larger when coefficients are larger, such as in the last example. Moreover, confidence intervals for the regression coefficients are based on the assumption of independence, which clearly is violated in this case: any coefficient value for $x_2$ can be compensated for by $x_3$. Obviously, variances for the coefficients for $x_2$ and $x_3$ will be infinite. Even in cases where there is less than perfect correlation, we will see more unstable models, in the sense that the variances of the coefficient estimates will get large and predictions less reliable.

To be fair, ordinary multiple regression will not allow you to calculate the model in pathological cases like the above: matrix $\boldsymbol{X}^T\boldsymbol{X}$ will be singular, indicating that infinitely many inverse matrices are possible. This is exactly what we saw in the example. Another case where the inverse cannot be calculated is the situation where there are more independent variables than samples (try drawing a straight line through one point). Although we have called the previous example "pathological", these cases often occur in practice. Think, for example, of a collection of spectra (IR, UV-Vis, ...) which we want to use to predict some property. In almost all cases the number of wavenumbers far exceeds the number of samples, and subsequent wavenumbers will be heavily correlated, so we experience both problems simultaneously.

The crucial point is the calculation of the inverse in Equation 5.2. We will treat a few popular ways to avoid this. Principal Components Regression (PCR) basically does a regression on the scores of $\boldsymbol{X}$, where a suitable number of principal components has to be chosen; the rather inaccurately named Partial Least Squares (PLS; many other acronyms have been suggested for these three letters) regression does something similar, but instead of PCA scores uses other latent variables, which are defined using additional information from the dependent variables. In both cases, however, the inversion is simple because of the orthogonality of the latent variables. The price we pay is the loss of inferential procedures: it is no longer possible to derive analytical expressions for the prediction error and the variance of individual regression coefficients (note that the latter is not too useful anyway because of the violated assumption of independence). To be able to say something about the optimal number of latent variables, and about the expected error of prediction, cross validation or similar techniques must be used. Finally, one can perform variable selection, either based on knowledge of the problem, or (more often) using brute-force optimization methods that select a small set of predictive variables. Often, these models with only a few variables perform pretty well and are easier to interpret.

## 5.2 Principal Components Regression (PCR)

The prime idea of PCR is to use scores rather than the original data for regression. This has two advantages: scores are orthogonal, so there are no problems with correlated variables, and secondly, the number of PCs taken into account usually is much lower than the number of original variables. This reduces the number of coefficients that must be estimated considerably, which in turn leads to more degrees of freedom for the estimation of errors. Of course, we have the added problem that we have to estimate how many PCs to retain.

### 5.2.1 The algorithm

For the moment, let us select $a$ PCs; matrices $\boldsymbol{T}$, $\boldsymbol{P}$, etcetera, will have $a$ columns. The regression model is then built using the reconstructed matrix $\widetilde{\boldsymbol{X}} = \boldsymbol{T}\boldsymbol{P}^T$ rather than the original matrix:

$$\boldsymbol{Y} = \widetilde{\boldsymbol{X}}\boldsymbol{B} + \boldsymbol{\varepsilon} = \boldsymbol{T}(\boldsymbol{P}^T\boldsymbol{B}) + \boldsymbol{\varepsilon} = \boldsymbol{T}\boldsymbol{A} + \boldsymbol{\varepsilon} \tag{5.5}$$

$$\boldsymbol{A} = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{Y} \tag{5.6}$$

where $\boldsymbol{A} = \boldsymbol{P}^T\boldsymbol{B}$ will contain the regression coefficients for the scores. The cross product matrix of $\boldsymbol{T}$ is diagonal (remember, $\boldsymbol{T}$ is orthogonal) so can be easily inverted. The regression coefficients for the scores can be back-transformed to coefficients for the original variables:

$$\boldsymbol{B} = (\boldsymbol{P}\boldsymbol{P}^T)^{-1}\boldsymbol{P}\boldsymbol{A} = \boldsymbol{P}\boldsymbol{A} = \boldsymbol{P}(\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{Y} \tag{5.7}$$

This can be simplified even further by resubstituting $\boldsymbol{T} = \boldsymbol{U}\boldsymbol{D}$ (from the SVD routine):

$$\boldsymbol{B} = \boldsymbol{P}(\boldsymbol{D}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{D})^{-1}\boldsymbol{D}\boldsymbol{U}^T\boldsymbol{Y} = \boldsymbol{P}\boldsymbol{D}^{-2}\boldsymbol{D}\boldsymbol{U}^T\boldsymbol{Y} = \boldsymbol{P}\boldsymbol{D}^{-1}\boldsymbol{U}^T\boldsymbol{Y} \tag{5.8}$$

where the equality follows from the orthonormality of $\boldsymbol{U}$.

In practice, one always performs PCR on a mean-centered data matrix. In Chapter 2 we have seen that without mean-centering the first PC often dominates and is very close to the mean of the data, taken over the columns, an undesirable situation. By mean-centering, we explicitly force a regression model without an intercept. The result is that the coefficient matrix $\boldsymbol{B}$ does not contain an abscissa vector $\boldsymbol{b}_0$; it should be calculated explicitly by taking the difference between the mean $y$-values and the mean predicted $y$-values.

$$\boldsymbol{b}_0 = \overline{\boldsymbol{y}} - \overline{\boldsymbol{X}\boldsymbol{B}}$$

For every variable in $\boldsymbol{Y}$, we will find one number in $\boldsymbol{b}_0$.

### 5.2.2 Selecting the optimal number of components

Up to now, we have not considered the value of $a$. Clearly, if $a$ is too small we will incorporate too little information, and our model will not be very useful. If the number of columns $p$ in the original $\boldsymbol{X}$ matrix is smaller than the number of rows, then the maximal value for the number of PCs to select is $a_{\max} = p$. Since in that case the reconstructed matrix $\widetilde{\boldsymbol{X}}$ equals $\boldsymbol{X}$, a PCR model with $p$ PCs will give exactly the same model as the MLR model! The only differences that can arise stem from the possibly numerically more stable inversion of the cross product matrix in the PCR case, but with modern software this is not very likely.

Since our aim is to define a model that gives good predictions with as few regressors as possible (why?), it makes sense to base the selection of the number of PCs to use on an estimate of prediction error. A conceptually simple way to obtain such an estimate is cross validation, already encountered in Chapter 4. We perform the regression for several numbers of PCs, calculate the prediction errors, put the results in a graph, *et voila*, we can pick the number we like best.

The prediction error is usually calculated by a measure we call the root-mean-square error (RMS or RMSE)—and with good reason:

$$RMS = \sqrt{\sum_i^n \frac{(\hat{y}_i - y_i)^2}{n}}$$

That is, we predict the left-out sample ($\hat{y}_i$), store the result, and afterwards sum the squared differences with the true values. We divide by the number of predictions to obtain the mean squared error, and finally take the root. One often sees the name RMSCV or RMSECV, to indicate that it is the RMS value derived from a cross validation procedure. Of course, this procedure can be used in other contexts as well; the RMSEP is associated with prediction, and RMSEC is the calibration error, indicating how well the model fits the training data. The RMSEC value is in almost all cases the lowest of the three; it measures how well the model represents the data on which it is based. The RMSECV is an estimate of the thing that really interests us, the error of prediction, and although it is an unbiased estimate in principle (so on average the RMSECV is not smaller or larger than the true, unknown, prediction error), it can be very variable. This means that individual RMSECV estimates may be far off. Moreover, there is a caveat when one uses the RMSECV for optimising parameters in the model; see below. The RMSEP value, obtained from applying the model to a test set that has not in any way been used in making the model, is the best possible estimate of prediction error, provided that the test set is not too small, and that the training data are representative for the test data.

Consider the stackloss data, also used in the examples in Chapter 10 of reference [1], (data are on page 268; the original stackloss data set, not used in Massart, contains four more objects but to be able to compare results with Massart et al. we have left them out here). Performing PCR leads to the LOO RMSECV estimates shown in Figure 5.1 (left plot). Note that there is also a value for zero PCs: that is the error obtained when predicting every object with the mean value of the training set. Obviously, the first PC already leads to a significant reduction in error; however, the third adds nothing. In this (not particularly challenging) case, we can safely pick two as the correct number of PCs for PCR. In Table 5.1 the regression coefficients are given for all four models. Clearly, models with two and three PCs are more or less equivalent. Compare these coefficients to the ones in Chapter 10 in Massart!

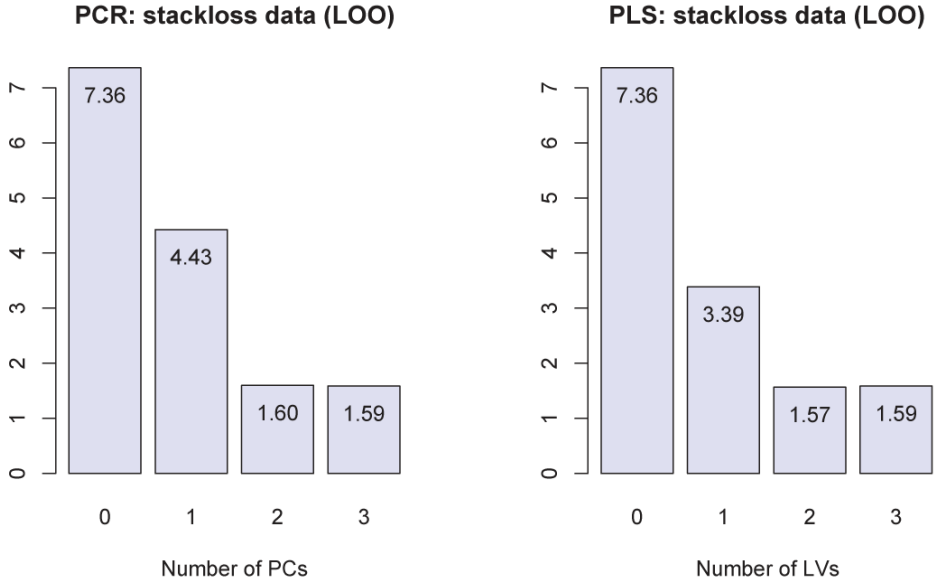


***Figure 5.1*** *- Root-mean-square error estimates from leave-one-out cross validation for the stackloss data, using PCR (left) and PLS (right).*

***Table 5.1*** *– Regression coefficients for PCR on the stackloss data set, depending on the number of PCs.*

| | *0 PC* | *1 PC* | *2 PC* | *3 PC* |
|---|---|---|---|---|
| $\boldsymbol{b_0}$ | 14.471 | -60.228 | -37.005 | -37.652 |
| $\boldsymbol{b_1}$ | - | 0.666 | 0.891 | 0.798 |
| $\boldsymbol{b_2}$ | - | 0.208 | 0.277 | 0.577 |
| $\boldsymbol{b_3}$ | - | 0.373 | -0.066 | -0.067 |

### 5.2.3 Discussion

Note that there is no *a priori* reason why the PCs with the largest singular values should be most useful for regression. Since we routinely pick PCs starting from number 1 and going up, there is a real chance that we include variables that actually do not contribute to the regression model. Put differently: we compress information in $\boldsymbol{X}$ without regard to what is to be predicted, so we can never be sure that the essential part is preserved. Although some people claim that selecting specific PCs (e.g. nrs 2, 5 and 6) on which to base the regression, rather than a sequence of PCs starting from one, leads to better models, this only increases the difficulties one faces: selection is a much more difficult process than determining a threshold.

A very important aspect is the estimation of the prediction error for future samples. Cross validation is often used for this, and since we already have performed cross validation to estimate the optimal number of PCs, it is tempting to use the corresponding RMSECV value as the estimate for prediction error. However, this will give a biased (i.e., too low) estimate: one has picked the number of PCs that minimises this error (well, in a way) for this particular training set. We need a separate way to estimate prediction error. Often, an independent test set is used. So the procedure is exactly the same as already seen in the previous chapter on classification:

1. Divide data in (random) training and test sets
2. Perform cross validation on the training set to obtain the optimal number of latent variables
3. Using the whole training set, build the model
4. Using the test set, estimate the prediction error: RMSEP

The same warnings as in the case of classification apply, too. First of all, the training set should be *representative*. This should be checked! Especially the presence of outliers can do serious harm... Note that too much tinkering with the composition of training and test sets will again lead to a biased estimate of prediction error. In general, a random division will work; when in doubt, one can perform multiple random divisions.

Secondly, one should make sure that any scaling is done in the same way for both training and test sets. So if mean-centering is applied, it should be the mean of the training set that is subtracted from the test set! Scaling the test set independently of the training set (in the mean-centering case this would mean subtracting the mean of the columns of the test set instead of using the mean of the columns of the training set) is an often-made error. The same goes for the prediction of new objects: they should be scaled in exactly the same way as the training set.

The previous point also has implications for the cross validation: since this in fact is a repeated division in training and test sets, one should also *within the cross validation* repeatedly scale the new training and test sets, of course only in the case where the scaling depends on the composition of the

training set. If one scales by only taking logarithms, there is no problem. The result of incorrect scaling (again) is bias in the prediction estimates; since data have been used which in principle should have been unknown. In general, it will make the prediction look better than it in fact is. In many (commercial) software packages, one scales the data before entering the cross validation and ignores the effects this has on the prediction errors. The rationale is that it is much faster to do the scaling only once, and since it is not the absolute values of the errors that is important in the cross validation here, but the relative magnitudes (one is going to pick the optimal number of latent variables, remember), a little bias is not too bad.

## 5.3 Partial Least Squares Regression (PLS)

Just like PCR, PLS defines orthogonal latent variables to compress the information and throw away irrelevant stuff. However, PLS explicitly aims to construct latent variables in such a way as to capture most information in $\boldsymbol{X}$ and $\boldsymbol{Y}$ and to maximize the correlation between these matrices. Put differently: it maximizes the covariance between $\boldsymbol{X}$ and $\boldsymbol{Y}$ [2] (page 336). So it seems we keep all the advantages, and get rid of the less desirable aspects of PCR. The algorithm is a bit more complicated than PCR; in fact, there exist tens of almost equivalent algorithms to perform PLS, all of which are more or less similar with respect to the resulting regression coefficients. The differences are caused by either variations in the criterion that is optimized, different implementations to obtain speed improvements in specific situations, or by different choices for scaling intermediate results.

### 5.3.1 The algorithm(s)

For PLS as well as PCR, it is customary to perform mean-centering of the data so that there is no need to estimate an intercept vector; this is obtained afterwards. The notation is a bit more complicated than with PCR, as already mentioned, since now both $\boldsymbol{X}$ and $\boldsymbol{Y}$ have scores and loadings. Moreover, in many algorithms one employs additional weight matrices. One other difference with PCR is that the components of PLS are extracted sequentially whereas the PCs in PCR can be obtained in one step. In each iteration in the PLS algorithm, the variation associated with the estimated component is removed from the data, and the remainder (indicated with $\boldsymbol{E}$ for the "deflated" $\boldsymbol{X}$ matrix, and $\boldsymbol{F}$ for $\boldsymbol{Y}$) is used to estimate the next component. This continues until the user decides it has been enough, or until all components have been estimated. The first component is obtained from an SVD of the cross product matrix $\boldsymbol{S} = \boldsymbol{X}^T\boldsymbol{Y}$, thereby including information on both variation in $\boldsymbol{X}$ and $\boldsymbol{Y}$, and on the correlation between both. The first left singular vector, $w$, can be seen as the direction of maximal variance in the cross product matrix, and is usually indicated with the somewhat vague description of "weights". The projections of matrix $\boldsymbol{X}$ on this vector are called "X scores":

$$t = \boldsymbol{X}w = \boldsymbol{E}w \tag{5.9}$$

Eventually, these scores $t$ will be gathered in a matrix $\boldsymbol{T}$ that fulfils the same role as the score matrix in PCR; it is a low-dimensional estimate of the information in $\boldsymbol{X}$. Therefore, regressing $\boldsymbol{Y}$ on $\boldsymbol{T}$ is easy (there are very few, orthogonal, columns), and the coefficient vector for $\boldsymbol{T}$ can be converted to a coefficient vector for the original variables.

The next step in the algorithm is to obtain loadings for $\boldsymbol{X}$ and $\boldsymbol{Y}$ by regressing against the same score vector $t$:

$$p = \boldsymbol{E}^T t/(t^T t) \tag{5.10}$$

$$q = \boldsymbol{F}^T t/(t^T t) \tag{5.11}$$

Notice that one divides by the sum of all squared elements in $t$: this leads to "normalized" loadings. It is not essential that the scaling is done in this way, in fact, there are numerous possibilities to scale either loadings, weights, or scores. Unfortunately, this can make it difficult to compare the scores and loadings of different PLS implementations.

Finally, the data matrices are "deflated": the information related to this latent variable, in the form of the outer products $tp^T$ and $tq^T$ , is subtracted from the (current) data matrices.

$$\boldsymbol{E}_{n+1} = \boldsymbol{E}_n - tp^T \tag{5.12}$$

$$\boldsymbol{F}_{n+1} = \boldsymbol{F}_n - tq^T \tag{5.13}$$

The estimation of the next component then can start from the SVD of the cross product matrix $\boldsymbol{E}_{n+1}^T \boldsymbol{F}_{n+1}$. After every iteration, vectors $w$, $t$, $p$ and $q$ are saved as columns in matrices $\boldsymbol{W}$, $\boldsymbol{T}$ , $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively. One complication is that columns of matrix $\boldsymbol{W}$ can not be compared directly: they are derived from successively deflated matrices $\boldsymbol{E}$ and $\boldsymbol{F}$. It has been shown that an alternative way to represent the weights, in such a way that all columns relate to the original $\boldsymbol{X}$ matrix, is given by

$$\boldsymbol{R} = \boldsymbol{W}(\boldsymbol{P}^T \boldsymbol{W})^{-1} \tag{5.14}$$

Matrix $\boldsymbol{R}$ has some interesting properties, one of which is that it is a generalized inverse for $\boldsymbol{PT}$ . It also holds that $\boldsymbol{T} = \boldsymbol{XR}$. For interpretation purposes, one sometimes also calculates so-called y-scores $\boldsymbol{U} = \boldsymbol{YQ}$.

Now, we are in the same position as in the PCR case: instead of regressing $\boldsymbol{Y}$ on $\boldsymbol{X}$, we use scores $\boldsymbol{T}$ to calculate the regression coefficients $\boldsymbol{A}$, and later convert these back to the realm of the original variables:

$$\boldsymbol{Y} = \boldsymbol{TA} + \boldsymbol{\varepsilon} = \boldsymbol{XB} + \boldsymbol{\varepsilon} \tag{5.15}$$

$$\boldsymbol{A} = (\boldsymbol{T}^T \boldsymbol{T})^{-1} \boldsymbol{T}^T \boldsymbol{Y} \tag{5.16}$$

$$\boldsymbol{B} = \boldsymbol{RA} = \boldsymbol{RQ}^T \tag{5.17}$$

Note the similarities with the PCR algorithm: the difference lies first and foremost in the calculation of $\boldsymbol{T}$, which now includes information on $\boldsymbol{Y}$, and in the case of PLS involves $\boldsymbol{R}$ rather than $\boldsymbol{P}$. Again, the difficulties in calculating the inverse are solved by using orthogonal (in this case even orthonormal) variables $\boldsymbol{T}$.

Finally, the coefficient for the abscissa is obtained in the same way as with PCR:

$$\boldsymbol{b}_0 = \overline{\boldsymbol{y}} - \overline{\boldsymbol{XB}}$$

where, of course, the number of coefficients in $\boldsymbol{b}_0$ equals the number of columns in $\boldsymbol{Y}$.

One popular alternative formulation, SIMPLS [2], deflates matrix $\boldsymbol{S}$ rather than matrices $\boldsymbol{E}$ and $\boldsymbol{F}$; especially for large data matrices this leads to a much faster algorithm. It can be shown that SIMPLS actually optimises the PLS criterion, whereas the above algorithm is a (good) approximation. Other

algorithms have been designed that for instance perform SVD on cross product matrices of cross product matrices ($\boldsymbol{X}^T\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{X}$) or do not deflate $\boldsymbol{Y}$. In general, these modifications are designed to obtain speed improvements in certain situations (e.g. many more variables than objects, or the other way around). Moreover, given that one has a certain freedom to decide where exactly in the algorithm to normalize, the outcome of different implementations may seem to vary significantly. However, the regression coefficients, and therefore the predictions, of all these are usually virtually identical. For all practical purposes there is no reason to prefer the outcome of one algorithm over another.

### 5.3.2 Interpretation

PLS models give separate scores and loadings for $\boldsymbol{X}$ and $\boldsymbol{Y}$, and additionally in most implementations some form of a weight matrix. Probably most people concentrate on the matrix of regression coefficients $\boldsymbol{B}$ which is independent of algorithmic details such as the location of the normalisation procedure, and is directly comparable to regression coefficients from other methods like PCR. However, there is something useful in the other model elements as well.

As can be seen in the algorithm, the vectors $w$ constitute the direction of most variation in the cross product matrix $\boldsymbol{X}^T\boldsymbol{Y}$. The scores $t$ are the coordinates of the objects on this axis. Loadings for $\boldsymbol{X}$ and $\boldsymbol{Y}$ are obtained by regressing both matrices against the scores, and the products of the scores and loadings for $\boldsymbol{X}$ and $\boldsymbol{Y}$ are removed from data matrices $\boldsymbol{E}$ and $\boldsymbol{F}$.

The interpretation of the scores and loadings is similar to PCA: a score indicates how much a particular object contributes to a latent variable, while a loading indicates the contribution of a particular variable. Biplots can be made, showing the relations between scores and loadings, just like in the PCA case. One can in this way inspect the influence of individual PLS components, where the regression coefficient matrix $\boldsymbol{B}$ gives a more global view, summarizing the influence of all PLS components (for latent variables 1 to $a$).

An additional source of information is the relation between scores $\boldsymbol{U}$ and $\boldsymbol{T}$. Usually one hopes to see a nice and linear relation; if this is not the case one can replace the linear regression in Eqs. 5.10 and 5.11 by a multiple regression (using columns of powers of t) or even non-linear regression. There are however not too many reports where this has led to significant improvements.

### 5.3.3 Discussion

By definition, with the same number of latent variables, PCR will explain more of $\boldsymbol{X}$ and PLS will explain more of $\boldsymbol{Y}$; this is also visible in Figure 5.1, where especially with two latent variables PLS seems to have a lower error than the corresponding PCR model. Although PLS will usually need less latent variables than PCR, the final, optimized, models are often very similar. To illustrate this, the regression coefficients for the modified stackloss data set, obtained with PLS, are given in Table 5.2; compare these with the MLR results in Massart and the PCR results from Table 5.1. In practice, there is no good reason to prefer the one over the other, although PLS has become the more popular of the two.

For multivariate $\boldsymbol{Y}$, there is one other difference between PLS and PCR. With PCR, separate models are fit for each $\boldsymbol{Y}$ variable: the algorithm does not try to make use of any correlation between the separate dependent variables. With PLS, this is different. Of course, one can fit separate PLS models for each $\boldsymbol{Y}$ variable (this is often indicated with the acronym PLS1), but one can also do it all in one go

(PLS2). In that case, the same $\boldsymbol{X}$-scores $\boldsymbol{T}$ are used for all dependent variables; although a separate set of regression coefficients will be generated for every $y$-variable, implicitly information from the other dependent variables is taken into account. This can be an advantage, especially when there is appreciable correlation between the $y$-variables, but in practice there is often little difference between the two approaches. Most people prefer multiple PLS1 models (analogous to PCR regression) since they seem to give better fits.

There is no difference in the practical use of PLS and PCR; in both cases one has to perform cross validation to pick the optimal number of latent variables, and in both cases one has to have an additional way to get a reliable estimate of prediction error for future samples.

***Table 5.2*** *– Regression coefficients for PLS on the stackloss data set.*

| | *0 PC* | *1 PC* | *2 PC* | *3 PC* |
|---|---|---|---|---|
| $\boldsymbol{b_0}$ | 14.471 | -54.805 | -36.889 | -37.652 |
| $\boldsymbol{b_1}$ | - | 0.751 | 0.875 | 0.798 |
| $\boldsymbol{b_2}$ | - | 0.250 | 0.335 | 0.577 |
| $\boldsymbol{b_3}$ | - | 0.243 | -0.071 | -0.067 |

## 5.4 Variable selection

A simpler way to be able to use MLR on large collinear data sets is to select variables, either by hand, or by automatic procedures. This approach, when successful, presents many advantages. First of all, models with few variables are usually easier to understand than large complex models with thousands of regression coefficients. The (few) regression coefficients may provide important chemical information, e.g. to design new drugs or choose better experimental conditions, that are difficult to extract from larger sets of coefficients. Furthermore, they are often more robust.

The downside, of course, is that such a small set of variables with good predictive abilities may not exist, and even if it does exist, it is almost impossible to find. So-called variable-selection methods are designed to search in spaces of huge dimensions: the number of possible combinations within a set of 1000 variables is astronomically large, and there is no guarantee whatsoever that any procedure will find the optimal subset. However, in some cases, especially when knowledge of the chemical or physical processes involved can be used, it is possible to rule out certain variables, which already can be a big improvement.

Note that variable selection is an important topic in more or less every statistical activity: the choice which variables to include in a model to a large degree determines the outcome of the model. This is true for every single technique discussed in this reader, be it PCA, clustering methods, classification methods, or regression. In the unsupervised approaches, uninformative variables can obscure the "real" picture, and distances between objects can become meaningless. In the supervised cases (both classification and regression), there is the danger of chance correlations, leading to models with low predictive power. Unfortunately, there is no way out of this conundrum, and we are forced to accept correlated and uninformative variables as facts of life.

## 5.5 Example

In this example we will study a well-known data set relating octane numbers to infrared spectra of sixty gasoline samples [6]. The data are available on the course web site, and it is instructive to try

and reproduce the results in this example; because of different divisions in training and test sets, you will see small differences in the results.

The first thing to do is to divide the data set into a training set and a test set. We pick (randomly) 15 samples for the test set, leaving a training set of 45 samples. A plot of the infrared data is shown in Figure 5.2; the wavelengths run from 900 nm to 1700 nm in 2 nm intervals, giving 401 variables in total. We do not perform any explicit scaling prior to the regression; inside the PLS and PCR functions the data are mean-centered. This plot (and others) shows no signs of outliers or other strange phenomena, and after checking that the training data cover both the $\boldsymbol{X}$ and $\boldsymbol{Y}$ spaces (not shown here), we can proceed with the modelling.

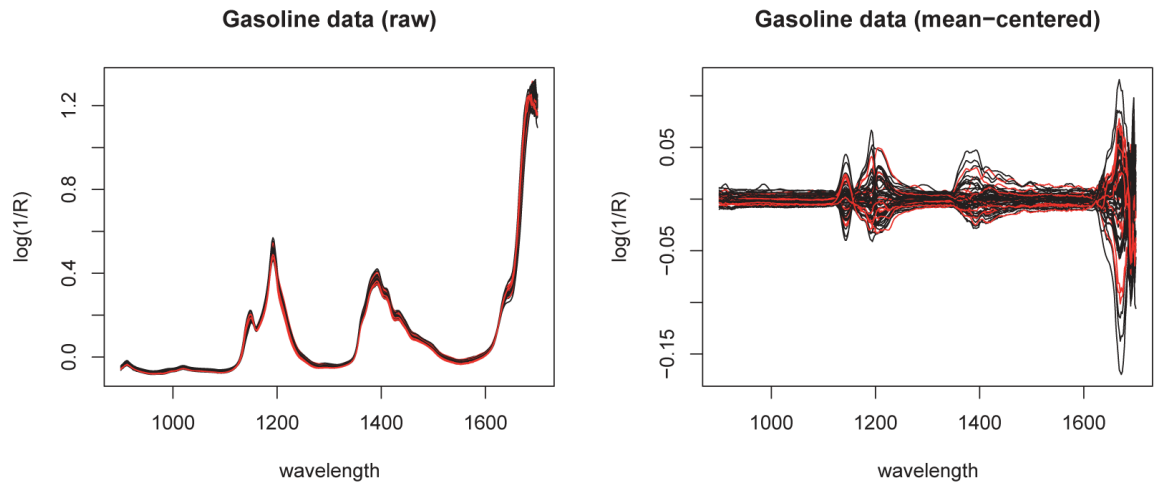


***Figure 5.2*** *- Infrared spectra of sixty gasoline samples. Raw data are shown at the left; mean-centered data right. Training set spectra are shown in black; test set spectra in red.*

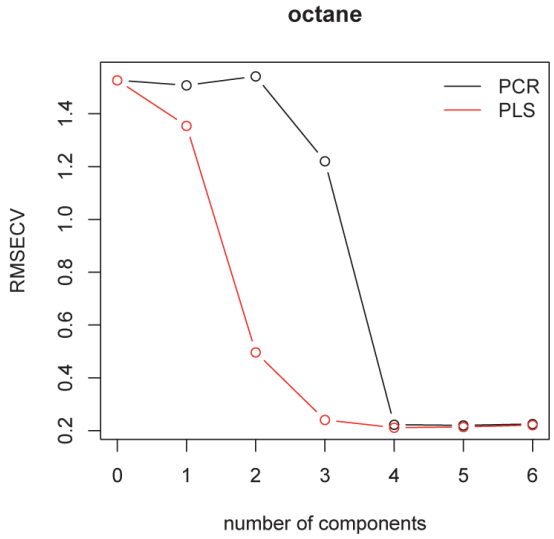


***Figure 5.3*** *- RMSECV values for different numbers of latent variables, for both PLS (red) and PCR (black).*

We will use both PCR and PLS. Performing LOO on the training set leads to the RMSECV values shown in Figure 5.3. Prediction with zero components is also included in the plot; this corresponds to an

equal prediction for all samples, *viz.* with the mean octane number. Interestingly, the PCR models using one and two PCs give more or less the same error as prediction with the mean. This indicates that the dimension reduction with PCA does not include useful information - for the prediction, at least - in the first two PCs. On the other hand, PLS immediately extracts the relevant information. Based on this graph, we would pick three components for the PLS model, and four for PCR. Adding more components does not lead to a significant reduction of RMSECV (although $y$-scales in these figures can be deceptive!). This pattern is quite common: both PCR and PLS eventually arrive at more or less the same prediction errors, but PLS needs fewer components.

Now, we build models using the complete training set, with the optimal number of latent variables. In Figure 5.4 the regression vectors for both the PLS and PCR models are shown. It should come as no surprise that the predictions for both models are so similar: apart from the rather uninteresting area below 1100 nm the two are virtually identical. The biggest coefficients (in an absolute sense) occur at locations in the spectra where there is information, i.e. at peak positions.

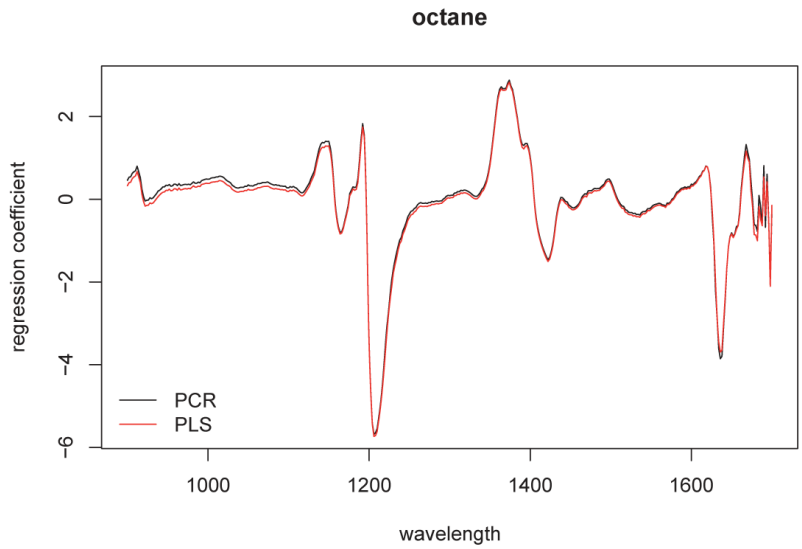


***Figure 5.4*** *- Regression coefficients for the PLS model with three LVs (red) and the PCR model with four PCs (black). The abscissa values are not shown: these are 100.25 and 103.43 for the PCR and PLS models, respectively.*

The proof of the pudding is in the eating; all the work we have done up until now is worth nothing if the predictions are bad. Therefore, we are going to apply the models to the test set, and derive an RMSEP value, which can be seen as an estimate of the expected prediction error for future data. On top of that, we can inspect the predictions graphically to see whether there are any peculiarities that should lead to modifications. This is depicted in Figure 5.5. The true values are always plotted on the $x$-axis; also note that the $x$ and $y$-scales are identical so that the line $y = x$ corresponds with perfect predictions. Both models seem to do pretty well, although the RMSEP values seem to be 50% higher than the RMSECVs. This illustrates the caution that is required in "trusting" error estimates...

## 5.6 Discussion

It is no coincidence that chemistry has been the first area in which PLS was really heavily used. Analytical chemists had been measuring spectra and trying to relate them to chemical properties for years, when this regression technique finally provided them with the tool to do so. Other disciplines such as statistics were slow to follow, but eventually PLS has found its place in a wide range of fields.

Also the theoretical background has now been clarified: PLS started out as an algorithm that was really poorly understood. Nowadays, there are very few people who dispute that PLS is an extremely useful method, but it has been overhyped somewhat. In practical applications, its performance is very, very similar to techniques like PCR. Careful thinking of experimental design, perhaps variable selection, and appropriate preprocessing the data is likely to be far more important than the exact choice of multivariate regression technique.

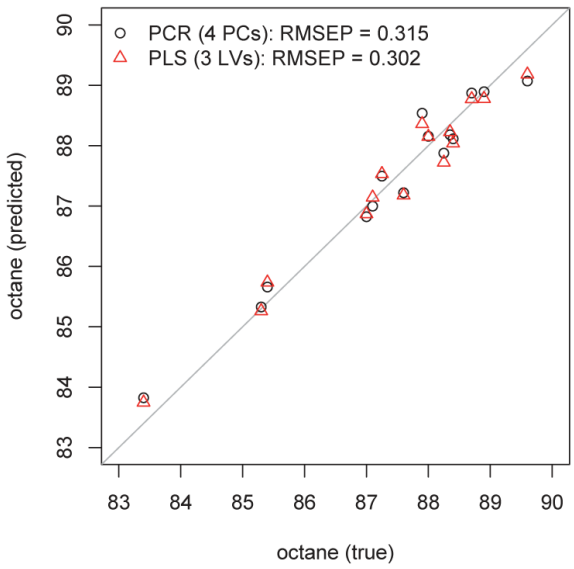


***Figure 5.5*** *- Predictions of octane numbers for the PLS and PCR models. The line $y = x$ is added for reference.*

# Bibliography

[1] D.L. Massart, B.G.M. Vandeginste, L.M.C. Buydens, S. de Jong, P.J. Lewi, and J. Smeyers-Verbeke, editors. Handbook of Chemometrics and Qualimetrics: Part A, volume 20A of Data Handling in Science and Technology. Elsevier Science Publishers, 1997.

[2] B.G.M. Vandeginste, D.L. Massart, L.M.C. Buydens, S. de Jong, P.J. Lewi, and J. Smeyers-Verbeke, editors. Handbook of Chemometrics and Qualimetrics: Part B, volume 20B of Data Handling in Science and Technology. Elsevier Science Publishers, 1998.

[3] W.J. Ewens and G.R. Grant. Statistical methods in bioinformatics. Springer, New York, 2001.

[4] K. Mardia, J. Kent, and J. Bibby. Multivariate analysis. Academic Press, 1979.

[5] T. Hastie, R. Tibshirani, and J.H. Friedman. The elements of statistical learning. Springer, New York, 2001.

[6] J. Kalivas. Two data sets of near infrared spectra. Chemom. Intell. Lab. Syst, 37:255- 259, 1997.

## Version history

1.03 (Sept. 2015): small textual modifications in the Introduction

1.04 (Jan. 4, 2016): small textual modifications in the Introduction. ‘Version history’ added.

## Acknowledgements

The course Chemometrics I, first given in this form in 2006/2007, profits from feedback from many people. In particular I would like to thank Willem Melssen, Tom Bloemberg, Patrick Krooshof and Lutgarde Buydens for their suggestions and meticulous proof-reading. The computer exercises were designed by Patrick Krooshof, Bulent Ustun, Willem Melssen and Tom Bloemberg; the latter also converted the Matlab-oriented exercises to R so that a student population with very diverse backgrounds can be addressed.

Of course, any errors and inconsistencies remain my responsibility, and I am grateful for any suggestions for further improvements.

*Ron Wehrens*
Nijmegen, September 2008