Diplomarbeit

zum Thema

Analyzing Diabetes Data

ausgeführt am
Institut für Statistik und Wahrscheinlichkeitstheorie
der Technischen Universität Wien

unter der Anleitung von
O.Univ.-Prof. Dipl.-Ing. Dr.techn. Rudolf Dutter

durch

Bernhard Spangl
Matr.-Nr.: 9426095
1160 Wien, Baumeistergasse 26/4/4

Wien, am 8. Dezember 2002   Bernhard Spangl

Preface and Acknowledgements

It was very interesting to work with medical data. I have learned much about statistical analysis of practical data and it was fascinating to connect theory and praxis, i.e., to combine statistical models and medicine.

I hope I was able to write down the results of this statistical research in a way that others can also understand them.

I like to thank Rudi Dutter, my supervisor, for all the support he gave to me and for the long discussions about the meaning of statistics and the correct usage of well-known statistical phrases. Moreover, I am grateful to him for introducing to me the field of statistics from his point of view. I really appreciate working with him because he had so many suggestions for solving statistical questions which occurred during the research of the data that only a few of those ideas were used in this survey.

Thanks are also due to Univ.-Prof. Dr. Kinga Howorka for providing the diabetes data and for the support concerning medical questions.

Further I like to thank Peter Filzmoser for the stimulating discussions and his various suggestions and Herbert Guttmann for fixing all hard- and software problems concerning the computer terminals and accounts at the department.

Finally, I like to thank Barbara, her and my parents for their patience and for all the support they gave to me. This work would not have been possible without them.

Bernhard Spangl

e-mail: spangl@pc2.statistik.tuwien.ac.at

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The data for the present research were provided by Univ.-Prof. Dr. Kinga Howorka of the Department of Biomedical Engineering and Physics, General Hospital of Vienna. This medical data base contains data of diabetic patients like personal properties, data of medical check-ups and laboratory data. These data were collected over a period of more than 12 years.

The aim of our survey is to get a better understanding of how the values of different medical variables, especially those of *HbA1c*, influence the severity of late complications, such as kidney failure, retinopathy or high blood pressure (cf. Howorka, 1996).

However, the patients' data have been collected during medical check-ups and are mainly used for the patients' therapies. Therefore they are very inhomogeneous. Considering the development and usage of the data base it is obvious that the length of the time period between two check-ups of a single patient differs which causes heavy problems when analyzing the data. Moreover there are lots of missing values because different medical parameters were measured at different times.

In Chapter 2 we propose ways to solve these problems in order to statistically analyze the diabetes data. In Chapter 3 and 4 we introduce the methods which are used for multivariate and time series analysis. Further, in Chapter 5, we present in detail the results that are delivered by these methods and summarize our conclusions in Chapter 6. The data sets and variables used for statistical analysis are described in detail in Appendix A.

Throughout this work we will use the statistical software package ®. ® itself can be downloaded from the webpage `http://cran.r-project.org`. The listings of the algorithms can be found in Appendix B.

Chapter 2

Preparations

In this chapter we briefly introduce the diabetes data set and give a summary of the problems which occurred while preparing the data for further statistical analysis and of how we cope with these problems.

2.1 Data Facts

The medical data base provided by Univ.-Prof. Dr. Kinga Howorka of the Department of Biomedical Engineering and Physics, General Hospital of Vienna, contains data of diabetic patients. The data base has been generated with the data base program *Paradox 7.0*. It contains more than 130 tables where the patients' data like personal properties (e.g., gender, age), data of medical check-ups (e.g., blood pressure, weight) and laboratory data (e.g., blood and urine values) are stored. The data were collected from 1988 until 2000.

Our present research is based on a group of 852 patients whose data are mainly stored in two tables of the data base, namely *"laborbef.db"* and *"routine.db"*. In order to be able to refer to a single patient we will use the same number which has already been assigned to the patient in the medical data base. Because we do not take all patients of the data base into account the patients' numbers which are displayed in various plots range from 1 to 1198. Moreover in both tables there should also be a primary key for each observation as a one-to-one representation, i.e., for each observation the pair of date and the number of the patient should be unique. However, there are two cases where the date is missing and which we omit for further analysis.

Another important thing worth mentioning is that the data base contains personal information. Therefore it is only possible to publish data that is anonymous.

2.2 Problems of Real Life Data

First of all we may question if the data gathered at the beginning, e.g., in 1988, are comparable with the data gathered later. We may think about the changing standards of blood analysis in laboratories or of the different ways how the medical check-up is done. Therefore we *restandardize* the value of *HbA1c* according to the reference ranges for different time periods (for details see Section 2.3).

As we have mentioned above, the whole medical information is stored in many different tables, e.g., there is one for the medical check-ups (*routine.db*), one consists of the laboratory data (*laborbef.db*) and another one contains the retinopathy data (*retino.db*).

Considering the development and usage of the data base it is obvious that each patient has a different number of medical check-ups and also the length of the time period between two check-ups differs which causes heavy problems when analyzing the data.

Moreover there are lots of missing values. Neither the laboratory data, e.g., the blood or urine values, nor the data of the medical check-ups, e.g., blood pressure or weight, were gathered in the same way for all patients. That means, for example, that on one hand blood pressure and weight were never measured from some patients. On the other hand blood values, like *HbA1c*, *Cholesterol* or *Triglyceride*, and urine values may be measured for one patient at different times.

In order to do further statistical analysis we merge the two tables, *"laborbef.db"* and *"routine.db"*, in order to obtain one large data set. Unfortunately, this leads to further problems: the values of *HbA1c* and *blood glucose* are sometimes measured twice because they are recorded in the table *"laborbef.db"* as well as in the table *"routine.db"*. When looking at one patient we sometimes observe the value of *HbA1c* twice a day. In some cases we observe it only once because the value is missing in one of the former tables. In other cases an equal value is recorded twice but the date differs a few days. This happens, for example, if the patient first goes to the laboratory and a few days later to the medical check-up. Then the same value of *HbA1c* is

3

recorded on different days. The same is true for the values of *blood glucose*.

In order to illustrate the problems mentioned above the original values of the medical parameter *HbA1c* of some patients are plotted in Figure 2.1. The horizontal time axis represents 12 years.

Considering for example the time series of patient "171" we note that at least one value is missing at the beginning, indicated by the missing line between the second and third value. Moreover, we recognize that at the beginning of the last third the parameter of *HbA1c* is measured twice but the values are different.

Looking at patient "154" in Figure 2.1 we see at the beginning of the last third that the time period between two measurements is very long with respect to those of other patients which means that patient "154" had no medical check-ups in-between.

Furthermore the patients' series start at different times (for example, see the time series plot of patient "87" in Figure 2.1 compared to, e.g., patient "77" or "131").

2.3 Restandardization of *HbA1c*

When speaking about the value of *HbA1c*, denoted by z_{ij}, $i = 1, \ldots, I_j$ and $j = 1, \ldots, 4$, where I_j is the total number of observations in the period j given in Table 2.1, we refer to the patients' blood value which was measured by the laboratories and stored in the medical date base. Furthermore we had four reference ranges for the different periods of time

$$[\ell_j, u_j] \text{ together with } \mu_j := \frac{\ell_j + u_j}{2}, \quad j = 1, \ldots, 4, \tag{2.1}$$

where ℓ_j and u_j denote the lower and upper limit of the reference range in the period j (cf. Table 2.1).

In order to compare the values of *HbA1c*, z_{ij}, between two different periods, we transform them to get a common reference range $[\ell, u]$ and a common mean $\mu = (\ell + u)/2$, where ℓ and u denote the common lower and upper limit of the reference range. There are many ways to do this, e.g., taking the minimum $\ell_{\min} = \min\{\ell_1, \ell_2, \ell_3, \ell_4\}$ or the mean $\bar{\ell} = (1/4) \sum_{j=1}^{4} \ell_j$ of all four lower limits ℓ_j as common lower limit ℓ. We choose the fourth reference range $[\ell_4, u_4]$ of Table 2.1 as common reference interval because people, like diabetic patients or therapists, who daily have to deal with *HbA1c* values make use of the values of the latest period.

4
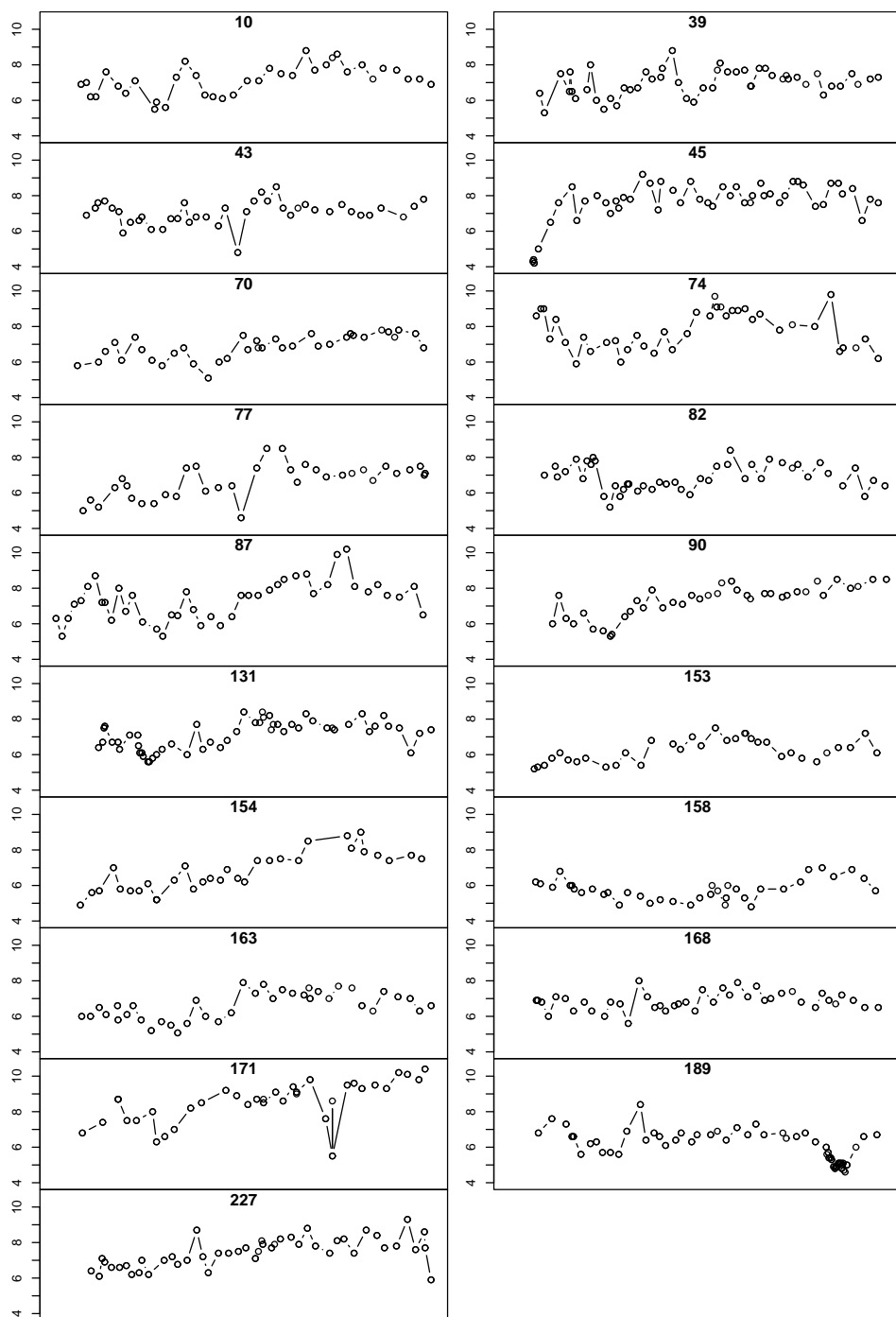
Figure 2.1: Original Data of the Variable *HbA1c*

Table 2.1: Reference Ranges of *HbA1c*

j	period	ℓ_j	u_j	μ_j
1	$\leq$ 1988-12-31	3.7	5.8	4.75
2	1989-01-01 to 1994-12-31	3.4	6.1	4.75
3	1995-01-01 to 2000-05-31	4.2	6.6	5.40
4	$\geq$ 2000-06-01	4.1	6.0	5.05

Therefore we get the transformed values of *HbA1c*, denoted by z_{ij}^*, using the following linear transformation T_j for each of the first three periods:

$$z_{ij}^* := T_j(z_{ij}), \quad j = 1, 2, 3, \tag{2.2}$$

where

$$T_j(z) = (z - \mu_j)\frac{\mu_4 - \ell_4}{\mu_j - \ell_j} + \mu_4, \quad j = 1, 2, 3. \tag{2.3}$$

We easily see that the above transformations T_j satisfy the following:

$$\begin{aligned}
T_j(\ell_j) &= \ell_4 \\
T_j(\mu_j) &= \mu_4 \\
T_j(u_j) &= u_4, \quad j = 1, 2, 3.
\end{aligned} \tag{2.4}$$

However, another way would be to use centered values of *HbA1c*, denoted by z_{ij}', instead of the absolute values z_{ij}, i.e.,

$$z_{ij}' := z_{ij} - \mu_j, \tag{2.5}$$

and afterwards apply a transformation T_j' similar to T_j to obtain common lower and upper limits for all four periods. This approach is not considered in the following survey.

2.4 The *Area Under the Curve (AUC)* Algorithm

Nevertheless the data show the following three dimensional structure: For each patient and each medical variable there is a time series.

For the subsequent multivariate analysis we shrink the time axis to get a data matrix where we only have one value for each patient and each medical variable. One possible way of shrinking is the slightly modified *Area Under the Curve*-algorithm (AUC):

$$AUC := \frac{1}{t_n - t_1}\left((t_2 - t_1)\frac{x_1 + x_2}{2} + \ldots + (t_n - t_{n-1})\frac{x_{n-1} + x_n}{2}\right)$$
$$= \frac{1}{2(t_n - t_1)}\left((t_2 - t_1)x_1 + \sum_{i=2}^{n-1}(t_{i+1} - t_{i-1})x_i + (t_n - t_{n-1})x_n\right) \quad,$$

where the pair (t_i, x_i), $i = 1, \ldots, n$, is the i-th measurement of a specific medical variable, denoted by x_i, at date t_i and n the number of available medical check ups of each patient. This statistic is often used in medical surveys (cf. Howorka et al., 1997, 1998b) and is a weighted mean where the observations are weighted by the length of the time interval between them.

There are some remarkable properties of AUC: when using AUC to estimate the location parameter we do not loose as much information as we would when using the mean or the median because it is weighted by the length of the time interval. Furthermore it does not matter if a variable were measured twice a day. Moreover, assuming equidistant time points, we obtain approximately the mean.

Applying AUC to our data set, we obtain a data matrix but, unfortunately, there are still missing values because for a few patients some medical variables were never measured by the laboratory or during a medical check-up.

Chapter 3

Nonlinear Estimation by Iterative Partial Least Squares

The *NIPALS* (Nonlinear Estimation by Iterative Partial Least Squares) algorithm was introduced by H. Wold in 1966 as an alternative method for principal component analysis (PCA). This algorithm and related methods, all based on the idea of the so-called *"criss-cross"* or *"alternating"* regression, were summarized by the name *NILES* (Nonlinear Estimation by Iterative Least Squares) procedures (cf. Wold, 1966).

In the book of Tenenhaus (1998) a version of the *NIPALS* algorithm is presented which is also suitable for data matrices with missing values. This slightly modified algorithm yields a decomposition into principal components, but it neither omits all the observations with missing values nor has to estimate the missing values for PCA. We will go into more details in Section 3.1.1.

An overview of the *NIPALS* algorithm and related topics can also be found in Geladi and Kowalski (1986) or in Kavšek (2002).

3.1 The *NIPALS* Algorithm

As mentioned before, the *NIPALS* algorithm was developed as an alternative algorithm for the calculation of principal components. Therefore we will first outline the main ideas of PCA.

The aim of PCA as well as of *NIPALS* is to derive a decomposition of a given data matrix $\boldsymbol{X}$. Let us suppose that the $n \times m$ data matrix $\boldsymbol{X}$ is

centered, i.e., $\sum_{i=1}^{n} \boldsymbol{x}_{ij} = 0$, $j = 1, \ldots, m$, and has rank r. Then we consider the following model

$$
\begin{aligned}
\boldsymbol{X} &= \boldsymbol{T}\boldsymbol{P}^{\top} + \boldsymbol{\varepsilon} \\
&= \sum_{\ell=1}^{k} \boldsymbol{t}_{.\ell}\boldsymbol{p}_{.\ell}^{\top} + \boldsymbol{\varepsilon},
\end{aligned}
\tag{3.1}
$$

where the matrix $\boldsymbol{T}$ has dimension $n \times k$, the matrix $\boldsymbol{P}$ $m \times k$ and $k \leq r$ is the number of components. The elements t_{ij} of the matrix $\boldsymbol{T}$ are called *scores* and the elements p_{ij} of $\boldsymbol{P}$ are called *loadings*. During this section the vector $\boldsymbol{t}_{.\ell}$ denotes the ℓ-th column of the matrix $\boldsymbol{T}$ and $\boldsymbol{p}_{.\ell}$ the ℓ-th column of the matrix $\boldsymbol{P}$ with $\ell = 1, \ldots, k$. We note that in Equation (3.1) each of the outer products $\boldsymbol{t}_{.\ell}\boldsymbol{p}_{.\ell}^{\top}$ of the two vectors $\boldsymbol{t}_{.\ell}$ and $\boldsymbol{p}_{.\ell}$ are approximations of the data matrix $\boldsymbol{X}$ of rank 1. The $n \times m$ matrix $\boldsymbol{\varepsilon}$ represents the error which will be made if we take less than r components to approximate the data matrix $\boldsymbol{X}$. This error term will be equal to $\boldsymbol{0}$ if and only if we use as many components k as the rank r of $\boldsymbol{X}$, i.e., if and only if $k = r$.

Hence, using PCA or *NIPALS* to decompose the matrix $\boldsymbol{X}$, we want to deduce k components (preferably $k < r$) which contain as much information of the data stored in the matrix $\boldsymbol{X}$ as possible. The way this decomposition is done in PCA is completely different to that applied by the *NIPALS* algorithm.

The main idea of classical PCA is to transform each observation of the centered data matrix $\boldsymbol{X}$, i.e., each row of $\boldsymbol{X}$. We obtain this linear orthogonal transformation by using the side condition that the variation of the transformed data is maximized in each component. This leads to a spectral decomposition which is implemented in the statistical software package Ⓡ either as eigenvalue decomposition or as singular value decomposition.

Contrary to *NIPALS*, where each component is derived after the other, classical PCA yields all principal components at once. It is important to know how many components we have to choose in order to get a good representation of the original data matrix $\boldsymbol{X}$. This is done by statistical tests, rules of thumb or graphically by the so-called *screeplot*. All methods depend on the proportion of the variation of each component on the total variation. For further detailed information on classical principal component analysis see Mardia et al. (1979) or Everitt and Dunn (2001).

Now Wold (1966) proposed the following iterative algorithm which only uses simple regressions to derive k principal components of a centered data matrix $\boldsymbol{X}$:

Step 0: First of all, we put

$$X^{(0)} := X. \tag{3.2}$$

For each component ℓ, $\ell = 1, \ldots, k$, we compute the following steps, where Steps 2 to 5 are iterated for each ℓ:

Step 1: Select a column j of $X^{(\ell-1)}$, e.g., the first one, as starting value for $t_{.\ell}$:

$$t_{.\ell} := x_{.j}^{(\ell-1)}, \quad \text{with } j = 1. \tag{3.3}$$

Step 2: Perform the following regression of $X^{(\ell-1)}$ on $t_{.\ell}$ with the model

$$X^{(\ell-1)} = t_{.\ell} p_{.\ell}^\top + \varepsilon_1$$

yielding the least squares (LS) solution

$$p_{.\ell}^\top := \frac{t_{.\ell}^\top X^{(\ell-1)}}{t_{.\ell}^\top t_{.\ell}} \ . \tag{3.4}$$

Step 3: Normalize $p_{.\ell}$:

$$p_{.\ell} := \frac{p_{.\ell}}{\|p_{.\ell}\|} \ . \tag{3.5}$$

Step 4: Perform the following regression of the transposed model, namely of $(X^{(\ell-1)})^\top$ on $p_{.\ell}$,

$$(X^{(\ell-1)})^\top = p_{.\ell} t_{.\ell}^\top + \varepsilon_2$$

yielding the LS solution

$$t_{.\ell} := \frac{X^{(\ell-1)} p_{.\ell}}{p_{.\ell}^\top p_{.\ell}} = X^{(\ell-1)} p_{.\ell} \ . \tag{3.6}$$

The equality in (3.6) holds because we have normalized $p_{.\ell}$ in (3.5).

Step 5: Stopping rule

One possible stopping rule is to calculate the norm of the vector of differences between the current $t_{.\ell}$ and the $t_{.\ell}$ of the previous iteration and look whether it is smaller than a given tolerance level or not. If it exceeds our tolerance limit then Steps 2 to 5 will be repeated, otherwise we will keep $t_{.\ell}$ and $p_{.\ell}$ as the final values or as best approximation of $X^{(\ell-1)}$ and go to Step 6.

Step 6: Calculate the residuals and return to Step 1 to compute the next component:

$$\boldsymbol{X}^{(\ell)} := \boldsymbol{X}^{(\ell-1)} - \boldsymbol{t}_{.\ell}\boldsymbol{p}_{.\ell}^{\top} . \tag{3.7}$$

The *NIPALS* algorithm delivers a decomposition of the $n \times m$ data matrix $\boldsymbol{X}$ into an $n \times k$ matrix $\boldsymbol{T}$ of scores and an $m \times k$ matrix $\boldsymbol{P}$ of loadings according to the model (3.1) which are at least approximately equal to those obtained by principal component analysis.

We note that all $\boldsymbol{t}_{.\ell}$ as well as all $\boldsymbol{p}_{.\ell}$, $\ell = 1, \ldots, k$, are orthogonal to each other like in classical principal component analysis. This is achieved by the LS regressions in (3.4) and (3.6) and the calculation of the residuals in (3.7). Moreover the $\boldsymbol{p}_{.\ell}$ are normalized to length one because of (3.5) as this is required in PCA.

Further we remark that both LS solutions of the multivariate regressions in (3.4) and (3.6) can also be obtained by simple regressions without intercept. We do not need to estimate an intercept because we assume that the data has been centered. The j-th coordinate $p_{j\ell}$, $j = 1, \ldots, m$, of the ℓ-th loadings vector $\boldsymbol{p}_{.\ell}$ is the regression coefficient of the regression of the j-th column $\boldsymbol{x}_{.j}^{(\ell-1)}$ of the matrix $\boldsymbol{X}^{(\ell-1)}$ on the scores vector $\boldsymbol{t}_{.\ell}$. Respectively, the same is true for $\boldsymbol{t}_{.\ell}$ in the second case. Furthermore, as $\boldsymbol{p}_{.\ell}$ is normalized in (3.5), we can interpret the coordinates $t_{i\ell}$, $i = 1, \ldots, n$, of $\boldsymbol{t}_{.\ell}$ as length of the projection of the i-th row $\boldsymbol{x}_{i.}^{(\ell-1)}$ of $\boldsymbol{X}^{(\ell-1)}$ onto the one-dimensional linear subspace engendered by the vector $\boldsymbol{p}_{.\ell}$.

However, because of these regressions, first the regression of the columns $\boldsymbol{x}_{.j}^{(\ell-1)}$, $j = 1, \ldots, m$, of the matrix $\boldsymbol{X}^{(\ell-1)}$ in (3.4) and then the regression of the rows $\boldsymbol{x}_{i.}^{(\ell-1)}$, $i = 1, \ldots, n$, of $\boldsymbol{X}^{(\ell-1)}$ in (3.6), Wold (1966) called this principle *"criss-cross"* or *"alternating"* regression. Hence, the *NIPALS* algorithm estimates the parameters of a non linear model–in fact bilinear–and, according to Tenenhaus (1998), the word *"partial"* comes from the fact that the algorithm uses the whole data and only one part of the parameters, namely those which are obtained by the regression before, to calculate the other ones.

Anyway, the iteration of the *NIPALS* algorithm usually stops quite soon in practical situations, which means that the convergence of $\boldsymbol{t}_{.\ell}$, $\ell = 1, \ldots, k$, is achieved.

Furthermore let us consider that we compute only the first component and assume that the *NIPALS* algorithm converges to the fixed point $\boldsymbol{t}_{.1}$ to see how the algorithm works. We also note that $\boldsymbol{t}_{.1}^{\top}\boldsymbol{t}_{.1}$ in (3.4) and $\|\boldsymbol{p}_{.1}\|$ in

(3.5) are scalars. Then we can rewrite Equation (3.4) and get

$$c_1 \boldsymbol{p}_{.1}^{\top} = \boldsymbol{t}_{.1}^{\top} \boldsymbol{X} \; , \tag{3.8}$$

where $c_1 := \boldsymbol{t}_{.1}^{\top} \boldsymbol{t}_{.1}$. Further we also rewrite Equation (3.6) which yields

$$c_2 \boldsymbol{t}_{.1} = \boldsymbol{X} \boldsymbol{p}_{.1} \; , \tag{3.9}$$

where $c_2 := \|\boldsymbol{p}_{.1}\|$. We remark that the vector $\boldsymbol{p}_{.1}$ in Equation (3.9) is the same as in Equation (3.8). Therefore we have to multiply the left side of Equation (3.9) by the factor c_2. Now we substitute Equation (3.9) into (3.8) and get

$$c_1 c_2 \boldsymbol{p}_{.1}^{\top} = \boldsymbol{p}_{.1}^{\top} \boldsymbol{X}^{\top} \boldsymbol{X} \; .$$

The above equation can be transformed into

$$(\boldsymbol{X}^{\top} \boldsymbol{X} - C \boldsymbol{I}_m) \boldsymbol{p}_{.1} = 0 \; , \tag{3.10}$$

where $C := c_1 c_2$ and $\boldsymbol{I}_m$ is the $m \times m$ identity matrix. We can also substitute Equation (3.8) into (3.9), respectively, to obtain

$$(\boldsymbol{X} \boldsymbol{X}^{\top} - C \boldsymbol{I}_n) \boldsymbol{t}_{.1} = 0 \; .$$

Moreover we note that $\boldsymbol{X}^{\top} \boldsymbol{X}$ is proportional to the estimated covariance matrix of the data and that Equation (3.10) is the eigenvalue/eigenvector equation used in the classical calculation of principal components.

3.1.1 The *NIPALS* Algorithm for Missing Values

The *NIPALS* algorithm for missing values, presented in the book of Tenenhaus (1998), is a modified version of the original *NIPALS* algorithm and delivers an approximation of the scores vectors $\boldsymbol{t}_{.\ell}$ and the loadings vectors $\boldsymbol{p}_{.\ell}$, $\ell = 1, \ldots, k$. As we have mentioned above, we neither need to estimate the missing values before applying the algorithm nor we need to omit the observations with missing values. Moreover the *NIPALS* algorithm for missing values yields estimations of the missing values of the data matrix $\boldsymbol{X}$.

The main idea of the *NIPALS* algorithm for missing values is to use only the existing values of the data matrix $\boldsymbol{X}$. Therefore, rewriting the model (3.1), we obtain the following for one element x_{ij} of the data matrix $\boldsymbol{X}$:

$$\begin{aligned} x_{ij} &= \boldsymbol{t}_{i.}^{\top} \boldsymbol{p}_{j.} + \varepsilon_{ij} \\ &= \sum_{\ell=1}^{k} t_{i\ell} p_{j\ell} + \varepsilon_{ij} \; , \end{aligned} \tag{3.11}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, m$.

Compared to the above definition of the *NIPALS* algorithm the following changes are made.

We replace (3.4) by the following:

Step 2*: For $j = 1, \ldots, m$ calculate

$$p_{j\ell} := \frac{\sum_{\{1 \le i \le n \,:\, x_{ij} \text{ and } t_{i\ell} \text{ are available}\}} x_{ij}^{(\ell-1)} t_{i\ell}}{\sum_{\{1 \le i \le n \,:\, x_{ij} \text{ and } t_{i\ell} \text{ are available}\}} t_{i\ell}^2} . \tag{3.12}$$

We note that it is also possible that the obtained vector $\boldsymbol{p}_{.\ell}$ in (3.12) contains missing values, i.e., one or more $p_{j\ell}$ cannot be calculated because neither values x_{ij} nor $t_{i\ell}$ are available for the same i.

Hence, the Euclidean norm in (3.5) has to be newly defined by

$$\|\boldsymbol{p}_{.\ell}\|_{new}^2 := \sum_{\{1 \le j \le m \,:\, p_{j\ell} \text{ are available}\}} p_{j\ell}^2 \tag{3.13}$$

and (3.5) is replaced by

Step 3*:

$$p_{j\ell} := \frac{p_{j\ell}}{\|\boldsymbol{p}_{.\ell}\|_{new}} , \tag{3.14}$$

for all available $p_{j\ell}$.

Consequently, we replace (3.6) by:

Step 4*: For $i = 1, \ldots, n$ calculate

$$t_{i\ell} := \frac{\sum_{\{1 \le j \le m \,:\, x_{ij} \text{ and } p_{j\ell} \text{ are available}\}} x_{ij}^{(\ell-1)} p_{j\ell}}{\sum_{\{1 \le j \le m \,:\, x_{ij} \text{ and } p_{j\ell} \text{ are available}\}} p_{j\ell}^2} . \tag{3.15}$$

Moreover we note that (3.15) cannot be simplified in the same way as (3.6) because some additional values x_{ij} may be missing.

As mentioned in Section 3.1, the LS solution of the multivariate regression of Step 2 in (3.4) or of Step 4 in (3.6) is the same as performing simple regressions without intercept yielding a least squares solution of each coordinate of $\boldsymbol{p}_{.\ell}$ or $\boldsymbol{t}_{.\ell}$, $\ell = 1, \ldots, k$. This is also done in (3.12) and (3.15) whereas only the existing values are taken into account. This means, we only use those pairs $\{(t_{i\ell}, x_{ij}^{(\ell-1)}) : 1 \le i \le n, x_{ij} \text{ and } t_{i\ell} \text{ are available}\}$ and $\{(p_{j\ell}, x_{ij}^{(\ell-1)}) : 1 \le j \le m, x_{ij} \text{ and } p_{j\ell} \text{ are available}\}$ to estimate the regression parameter $p_{j\ell}$ or $t_{i\ell}$, respectively.

It is obvious that if we use a full data matrix $\boldsymbol{X}$ with no missing values, both, the *NIPALS* algorithm and the above one, which is modified in order to be able to cope with missing values, will deliver the same results.

Furthermore, considering the model (3.11), we can easily estimate the missing values of the data matrix $\boldsymbol{X}$:

$$\widehat{x}_{ij} := \boldsymbol{t}_{i.}^{\top}\boldsymbol{p}_{j.} = \sum_{\ell=1}^{k} t_{i\ell}p_{j\ell} \ . \tag{3.16}$$

This means, we assume that the missing values are exactly in accordance with the model (3.1) or (3.11), respectively.

Nevertheless we have to make sure that neither a complete row nor an entire column of the data matrix $\boldsymbol{X}$ is missing because we do not get any estimations of these values. If this is the case we will have to omit these rows or columns before applying the *NIPALS* algorithm for missing values.

However, the *NIPALS* algorithm for missing values usually delivers good approximations of the scores vectors $\boldsymbol{t}_{.\ell}$ and the loadings vectors $\boldsymbol{p}_{.\ell}$, $\ell = 1, \ldots, k$, if the data matrix $\boldsymbol{X}$ does not consist of too many missing values.

3.2 Simulation

In this section we present the results of our simulations. First we check whether the principal components delivered by the *NIPALS* algorithm approximate the original data are as good as those delivered by the function *princomp*. Then we do the same comparing the original *NIPALS* algorithm and the *NIPALS* algorithm for missing values. The listing of the used algorithms is attached in Appendix B.

In both cases the data are multivariate normally distributed with mean vector $\boldsymbol{0}$ and a given covariance matrix $\boldsymbol{S}$. Instead of choosing an arbitrary covariance matrix first and scaling the simulated data matrix afterwards, we use a correlation matrix from the beginning. Assuming that the used data matrix is $n \times m$ we have an $m \times m$ correlation matrix $\boldsymbol{S}$. The elements of $\boldsymbol{S}$ are computed according to the following formula:

$$s_{ij} := 1 - \gamma(|i - j|) \quad i, j = 1, \ldots, m, \tag{3.17}$$

where $\gamma(.)$ is defined by

$$\gamma(h) := \begin{cases} C(\frac{3}{2}\frac{h}{a} - \frac{1}{2}(\frac{h}{a})^3) & : & h \leq a \\ C & : & h > a \end{cases} \ . \tag{3.18}$$

The function $\gamma(.)$ in (3.17) and (3.18) is called *spherical variogram* and is used in *geostatistics* (cf. Journel and Huijbregts, 1978). The parameter a is called the *range* and C is the *sill*. In our simulations we choose $a := m - 1$ and $C := 1$. Therefore all s_{ij}, $i, j = 1, \ldots, m$ are between zero and one with $s_{ii} = 1$, for all $i = 1, \ldots, m$, and $s_{1m} = s_{m1} = 0$.

Anyway we simulate a 500×6 data matrix $\boldsymbol{X}$ that is multivariate normally distributed with mean vector $\boldsymbol{0}$ and correlation matrix $\boldsymbol{S}$ as described above.

In order to compare the resulting approximation of the simulated data matrix $\boldsymbol{X}$ we compute k principal components, especially a $500 \times k$ scores matrix $\boldsymbol{T}$ and a $6 \times k$ loadings matrix $\boldsymbol{P}$, applying the function *princomp* and the *NIPALS* algorithm. (However, the function *princomp* delivers all principal components at once, in this case, we take only the first k ones.) With each algorithm an approximation $\widehat{\boldsymbol{X}}^{(\ell)}$, $\ell \in \{PCA,NIPALS\}$, of the originally simulated data matrix $\boldsymbol{X}$ is calculated according to the model (3.1). Then we subtract the approximated data matrix $\widehat{\boldsymbol{X}}^{(\ell)}$, $\ell \in \{PCA,NIPALS\}$, from the original $\boldsymbol{X}$. This yields two error matrices, denoted by $\boldsymbol{E}^{(PCA)}$ and $\boldsymbol{E}^{(NIPALS)}$. In order to compare the results we compute four different statistics of each error matrix, namely for $\ell \in \{PCA,NIPALS\}$,

$$\text{(a)} \quad e_{\text{Frobenius}}^{(\ell)} := \text{tr}(\boldsymbol{E}^{(\ell)\top} \boldsymbol{E}^{(\ell)}) = \text{tr}(\boldsymbol{E}^{(\ell)} \boldsymbol{E}^{(\ell)\top}) \,, \qquad (3.19)$$

$$\text{(b)} \quad e_{\max}^{(\ell)} := \max_{i=1,\ldots,500, \; j=1,\ldots,6} \left| e_{ij}^{(\ell)} \right| \,, \qquad (3.20)$$

$$\text{(c)} \quad e_{\text{mean}}^{(\ell)} := \text{ave}_{i=1,\ldots,500, \; j=1,\ldots,6} \left| e_{ij}^{(\ell)} \right| \,, \qquad (3.21)$$

$$\text{(d)} \quad e_{\text{median}}^{(\ell)} := \text{med}_{i=1,\ldots,500, \; j=1,\ldots,6} \left| e_{ij}^{(\ell)} \right| \,, \qquad (3.22)$$

where $\boldsymbol{E}^{(\ell)} = (e_{ij}^{(\ell)})$ with $\ell \in \{PCA, NIPALS\}$ and $tr(\boldsymbol{A})$ denotes the *trace* of an arbitrary square matrix $\boldsymbol{A}$. The functions $max(.)$, $ave(.)$ and $median(.)$ denote the *maximum*, the *mean* and the *median*, respectively.

In (3.19) we compute the *Frobenius* norm of both error matrices, which is the sum of squares of all elements $e_{ij}^{(\ell)}$ of $\boldsymbol{E}^{(\ell)}$, $\ell \in \{PCA,NIPALS\}$. Next, in (3.20), the maximum absolute value gives the largest deviation between the approximated data and the original one. Finally, in (3.21) and (3.22), we calculate the mean and the median of $|e_{ij}^{(\ell)}|$, $i = 1, \ldots, 500$, $j = 1, \ldots, 6$, respectively.

Moreover we determine the relative errors

$$e_{\text{rel}, a} \quad := \quad \frac{e_a^{(PCA)} - e_a^{(NIPALS)}}{e_a^{(PCA)}} \,,$$

$$a \in \{\text{Frobenius, max, mean, median}\} \ . \qquad (3.23)$$

This is done a hundred times. Afterwards we take the mean of the absolute errors, $e_a^{(PCA)}$ and $e_a^{(NIPALS)}$, and of the relative error, $e_{\text{rel, a}}$, $a \in \{\text{Frobenius, max, mean, median}\}$, and also compute the standard error (s.e.) of the mean according to the formula:

$$\text{s.e.} := \frac{s}{\sqrt{n}} \quad , \qquad (3.24)$$

where s denotes the standard deviation and n the number of simulations (100).

The mean errors for different numbers k of components along with their standard errors, given in parentheses, are printed in Table 3.1 and 3.2.

Table 3.1: Mean and Standard Error of $e_a^{(PCA)}$ and $e_a^{(NIPALS)}$

a	Components	*princomp*	*NIPALS*
Frobenius	2	25.1929(0.0484)	25.1929(0.0484)
	3	17.8970(0.0345)	17.8970(0.0345)
	4	13.3932(0.0287)	13.3951(0.0289)
	5	8.8661(0.0248)	8.8746(0.0259)
Maximum	2	1.8464(0.0155)	1.8463(0.0155)
	3	1.2606(0.0125)	1.2606(0.0125)
	4	1.0527(0.0111)	1.0514(0.0108)
	5	0.7540(0.0087)	0.7587(0.0091)
Mean	2	0.3627(0.0007)	0.3627(0.0007)
	3	0.2595(0.0006)	0.2595(0.0006)
	4	0.1876(0.0005)	0.1876(0.0005)
	5	0.1166(0.0006)	0.1167(0.0005)
Median	2	0.3006(0.0008)	0.3006(0.0008)
	3	0.2176(0.0007)	0.2176(0.0007)
	4	0.1476(0.0006)	0.1476(0.0006)
	5	0.0810(0.0010)	0.0809(0.0010)

We note that the mean errors in Table 3.1 comparing *princomp* and *NIPALS* are of the same order, which is also true for their standard errors, and

Table 3.2: Mean and Standard Error of $e_{\text{rel},\,a}$

a	Components	Mean Relative Error
Frobenius	2	$-2.65 \times 10^{-08}(1.72 \times 10^{-09})$
	3	$-9.26 \times 10^{-08}(1.22 \times 10^{-08})$
	4	$-1.41 \times 10^{-04}(2.15 \times 10^{-05})$
	5	$-9.41 \times 10^{-04}(2.57 \times 10^{-04})$
Maximum	2	$3.39 \times 10^{-05}(1.86 \times 10^{-05})$
	3	$2.17 \times 10^{-05}(2.55 \times 10^{-05})$
	4	$7.26 \times 10^{-04}(1.96 \times 10^{-03})$
	5	$-7.04 \times 10^{-03}(4.84 \times 10^{-03})$
Mean	2	$-4.60 \times 10^{-07}(5.51 \times 10^{-07})$
	3	$5.98 \times 10^{-07}(1.32 \times 10^{-06})$
	4	$-1.57 \times 10^{-04}(2.20 \times 10^{-04})$
	5	$-6.04 \times 10^{-04}(1.86 \times 10^{-03})$
Median	2	$2.73 \times 10^{-05}(2.85 \times 10^{-05})$
	3	$-5.55 \times 10^{-05}(5.68 \times 10^{-05})$
	4	$-2.96 \times 10^{-04}(8.66 \times 10^{-04})$
	5	$-7.61 \times 10^{-04}(5.61 \times 10^{-03})$

decrease with increasing number k of components. This is, because the original data will be approximated better and better if we take more and more components. Moreover the mean relative errors in Table 3.2 nearly vanish and are not significantly different from zero. This confirms the results of our example in Section 3.3.

Missing Values

In the same way we compare the results obtained by the *NIPALS* algorithm and the *NIPALS* algorithm for missing values.

Again we simulate a 500×6 data matrix $\boldsymbol{X}$ that is multivariate normally distributed with mean vector $\boldsymbol{0}$ and the same correlation matrix $\boldsymbol{S}$ as before. Additionally 15% of the data are randomly chosen and set to *NA* (*Not Available*).

Then k principal components of the full data matrix are calculated by the *NIPALS* algorithm and also k principal components of the data matrix

with missing values are computed by the *NIPALS* algorithm for missing values. Afterwards we obtain two error matrices, denoted by $\boldsymbol{E}^{(NIPALS)}$ and $\boldsymbol{E}^{(NIPNA)}$, in the same way as before by subtracting the approximated data matrix $\widehat{\boldsymbol{X}}^{(\ell)}$, $\ell \in \{NIPALS, NIPNA\}$, from the original one $\boldsymbol{X}$. Again we get absolute errors, $e_a^{(NIPALS)}$ and $e_a^{(NIPNA)}$, and are also able to calculate a relative error, $e_{\mathrm{rel},\,a}^{(miss)}$, according to the formula

$$e_{\mathrm{rel},\,a}^{(miss)} := \frac{e_a^{(NIPALS)} - e_a^{(NIPNA)}}{e_a^{(NIPALS)}} ,$$
$$a \in \{\text{Frobenius, max, mean, median}\} . \qquad (3.25)$$

In the same way as before, this is done a hundred times. The mean errors for different numbers k of components along with their standard errors, given in parentheses, are displayed in Table 3.3 and 3.4.

Table 3.3: Mean and Standard Error of $e_a^{(NIPALS)}$ and $e_a^{(NIPNA)}$

a	Components	*NIPALS*	*NIPALS* for M.V.
Frobenius	2	25.3380(0.0433)	27.4483(0.0517)
	3	18.1030(0.0352)	21.9209(0.0482)
	4	13.6193(0.0363)	19.2817(0.0539)
	5	9.1788(0.0326)	17.3349(0.0582)
Maximum	2	1.8691(0.0166)	2.6524(0.0437)
	3	1.2699(0.0120)	2.5284(0.0425)
	4	1.0573(0.0102)	2.5629(0.0438)
	5	0.7559(0.0084)	2.5953(0.0431)
Mean	2	0.3648(0.0007)	0.3874(0.0007)
	3	0.2626(0.0005)	0.2974(0.0006)
	4	0.1918(0.0006)	0.2424(0.0006)
	5	0.1244(0.0007)	0.1924(0.0007)
Median	2	0.3023(0.0008)	0.3140(0.0009)
	3	0.2202(0.0006)	0.2300(0.0007)
	4	0.1524(0.0007)	0.1693(0.0007)
	5	0.0913(0.0012)	0.1126(0.0010)

We note that, as before, the mean errors of the *NIPALS* algorithm in Table 3.3 decrease with increasing number k of components, but compared

Table 3.4: Mean and Standard Error of $e_{\text{rel, } a}^{(miss)}$

a	Components	Mean Relative Error
Frobenius	2	$-0.0833(0.0009)$
	3	$-0.2110(0.0020)$
	4	$-0.4162(0.0035)$
	5	$-0.8900(0.0070)$
Maximum	2	$-0.4261(0.0241)$
	3	$-1.0092(0.0389)$
	4	$-1.4471(0.0479)$
	5	$-2.4727(0.0681)$
Mean	2	$-0.0620(0.0007)$
	3	$-0.1326(0.0012)$
	4	$-0.2637(0.0021)$
	5	$-0.5503(0.0063)$
Median	2	$-0.0387(0.0016)$
	3	$-0.0444(0.0021)$
	4	$-0.1116(0.0036)$
	5	$-0.2476(0.0151)$

to them the mean errors of the *NIPALS* algorithm for missing values are significantly larger in each case. This causes the negative relative errors in Table 3.4. However, considering the *NIPALS* algorithm for missing values, the mean of $e_{\max}^{(NIPNA)}$ stays approximately constant for all numbers of components whereas the mean of $e_{\text{mean}}^{(NIPNA)}$ decreases in the same way as that of the *NIPALS* algorithm. The latter effect can also be found when looking at the mean of $e_{\text{Frobenius}}^{(NIPNA)}$ but it becomes most obvious when analyzing the mean of $e_{\text{median}}^{(NIPNA)}$ as the median is robust against outliers. (Note that the maximum absolute value in (3.20) only considers the extreme values of the error matrix $\boldsymbol{E}^{(\ell)}$, $\ell \in \{NIPALS, NIPNA\}$.)

3.3 An Example

In this section we give an example to get a better understanding of how the *NIPALS* algorithm works and missing values are estimated. As *NIPALS* is an alternative algorithm for principal component analysis (PCA) we will compare its results to those of other algorithms for PCA. As example we use the *euro86* data set which is described in Appendix A in detail.

In Appendix B the implemented versions of both *NIPALS* algorithms are printed.

First we calculate the principal components of the *euro86* data set using the *NIPALS* algorithm introduced in Section 3.1 and then we compare its results to those delivered by the function *princomp* of ℝ. For the classical PCA we use the function *princomp*, which performs an eigenvalue/eigenvector decomposition of the estimated covariance matrix, instead of the function *prcomp* that obtains the principal components by a singular value decomposition of the data matrix $\boldsymbol{X}$. At last we remove some elements of the *euro86* data and replace them by *"NA"* (*Not Available*). Then we calculate the principal components with the *NIPALS* algorithm for missing values introduced in Section 3.1.1 and compare the results. This will give us an idea of how the missing values are estimated.

As mentioned above, in contrary to the *NIPALS* algorithm all principal components are deduced at once by the function *princomp*. The number of components to be used for explaining the data set and further analysis is decided afterwards. The importance of each component is decreasing with increasing index. The decision of how many components we should choose can be done graphically by the so-called *screeplot*. The *screeplot* shows the proportion of each component on the total variation, i.e., for each scores vector $\boldsymbol{t}_{.\ell}$ we calculate

$$\frac{\operatorname{Var}(\boldsymbol{t}_{.\ell})}{\sum_{\ell=1}^{k} \operatorname{Var}(\boldsymbol{t}_{.\ell})} = \frac{\sum_{i=1}^{n}(t_{i\ell} - \bar{t}_{.\ell})^2}{\sum_{\ell=1}^{k}\sum_{i=1}^{n}(t_{i\ell} - \bar{t}_{.\ell})^2} \quad \ell = 1, \ldots, k \ , \qquad (3.26)$$

where $\operatorname{Var}(\boldsymbol{t}_{.\ell})$ denotes the empirical variance and $\bar{t}_{.\ell}$ the mean of each scores vector $\boldsymbol{t}_{.\ell}$. We note that if the scores are derived by classical principal component analysis the following equation will hold:

$$\operatorname{Var}(\boldsymbol{t}_{.\ell}) = \lambda_{\ell} \quad \ell = 1, \ldots, k \ ,$$

where λ_{ℓ} is the ℓ-th eigenvalue of the estimated covariance matrix $\widehat{\operatorname{Cov}}(\boldsymbol{X})$ of the data. Therefore for reasons of comparability also as many components as

are returned by the function *princomp* have to be calculated by the *NIPALS* algorithm.

To present the results gained with either of the algorithms we choose the well-known *biplot* (cf. Gower and Hand, 1996). This plot is a two dimensional approximation of the original data. In this graphic both, the observations and the variables of the two dimensional approximation of the data matrix $\boldsymbol{X}$, are printed. After an additional scaling each observation is represented by the first and second scores vector $\boldsymbol{t}_{.1}$ and $\boldsymbol{t}_{.2}$ and each variable is approximated by the first and second loadings vector $\boldsymbol{p}_{.1}$ and $\boldsymbol{p}_{.2}$. This representation of the data has some remarkable properties. The inner product of the two vectors $(t_{i1}, t_{i2})^\top$ and $(p_{j1}, p_{j2})^\top$ approximates x_{ij}, $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Furthermore the cosine of the angle between the two vectors $(p_{i1}, p_{i2})^\top$ and $(p_{j1}, p_{j2})^\top$ approximates the correlation between the i-th and j-th variable of $\boldsymbol{X}$, $i, j = 1, \ldots, m$.

When computing principal components with different algorithms the components sometimes differ by the multiplicative factor -1. Therefore, in order to compare both *biplots*, we change the direction of the first scores and loadings vector obtained by the function *princomp*, i.e., we multiply them with the factor -1.

As input of the algorithms we use mean-centered and scaled data matrices, the maximal number of iterations to be performed is set to 10 and as tolerance level of the precision of the scores vectors we choose 10^{-4}. Practice has shown that in general 10 iterations are sufficient to get quite a good result. Still, with a higher number of iterations and a lower tolerance level more exact components would be gained but the difference will normally only show up in the third or fourth decimal places which has no impact on practical surveys.

Now we calculate the principal components of the *euro86* data set using the *NIPALS* algorithm. The biplot of the first and second component is shown in Figure 3.1. They explain 68% of the total variation of the *euro86* data set. The screeplot of the results of the *NIPALS* algorithm is displayed in Figure 3.2. It shows a high importance of the first component followed by the components 2 to 7 with successive decreasing importance. The increasing value of component 5 is due to the algorithm.

Compared to the above biplot the function *princomp* yields principal components that are only slightly different (cf. Figure 3.3). The first and second component explain again 68% of the total variation of the data set. The corresponding screeplot of the function *princomp* is shown in Figure 3.4.
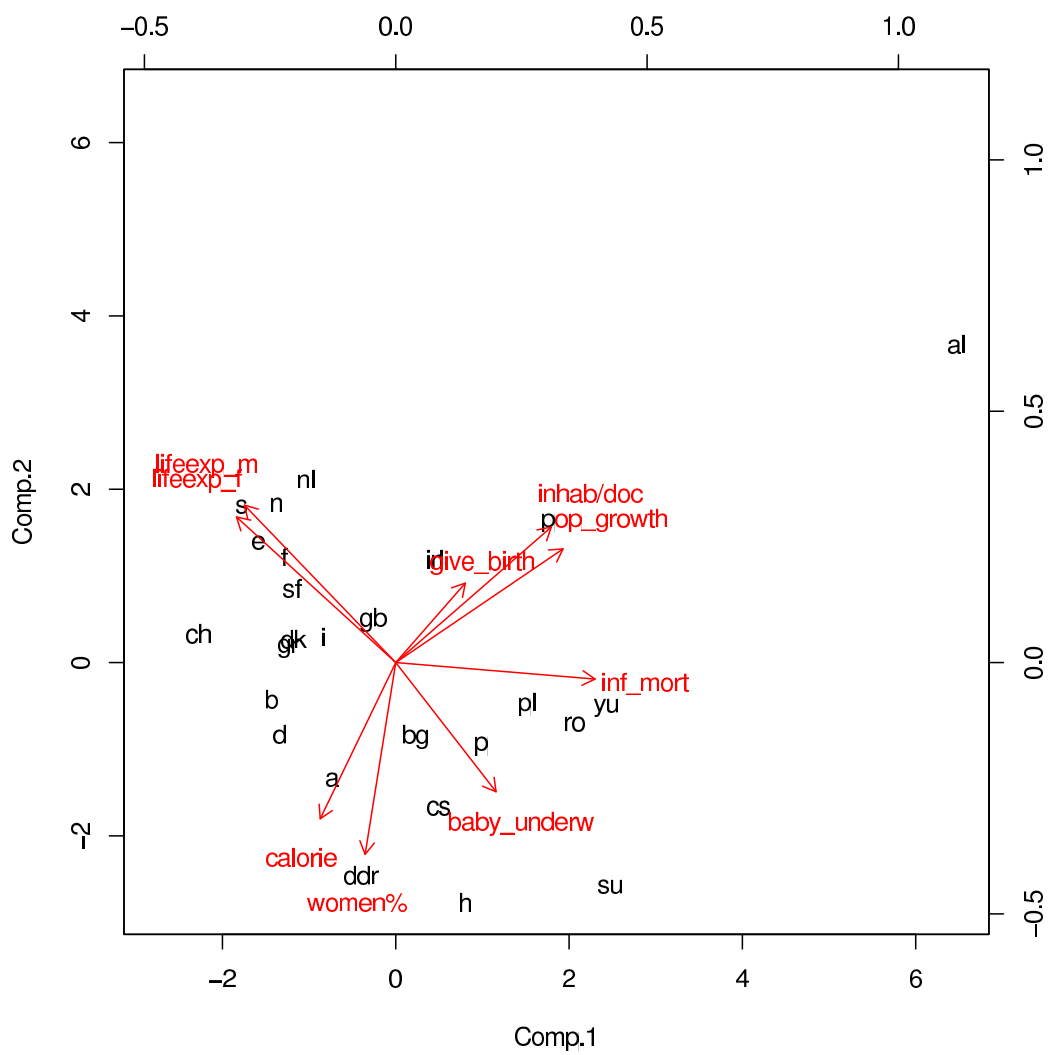
Figure 3.1: Biplot of *NIPALS* for *euro86*
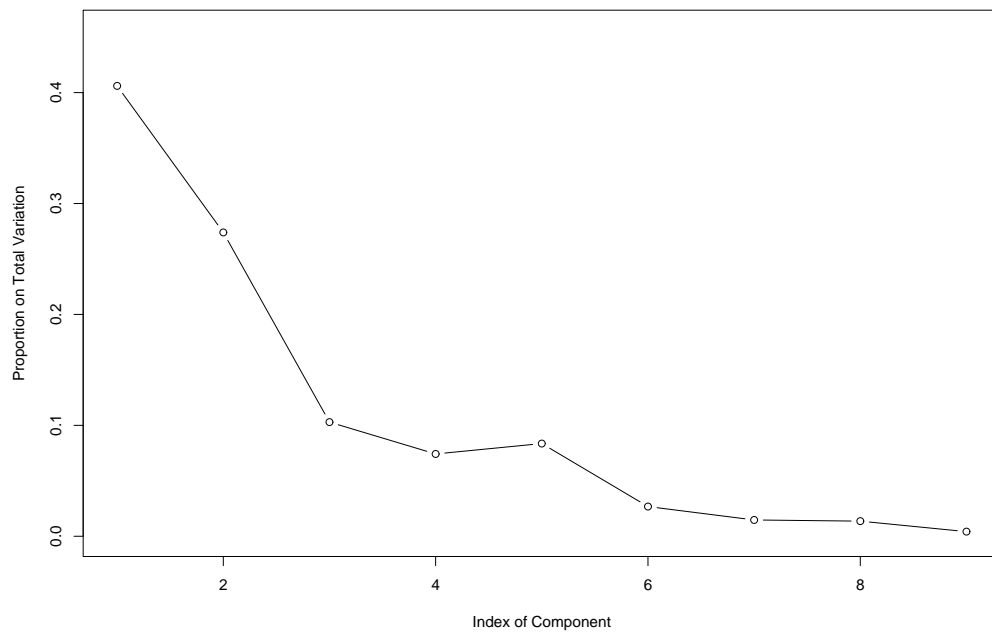
Figure 3.2: Screeplot of *NIPALS* for *euro86*
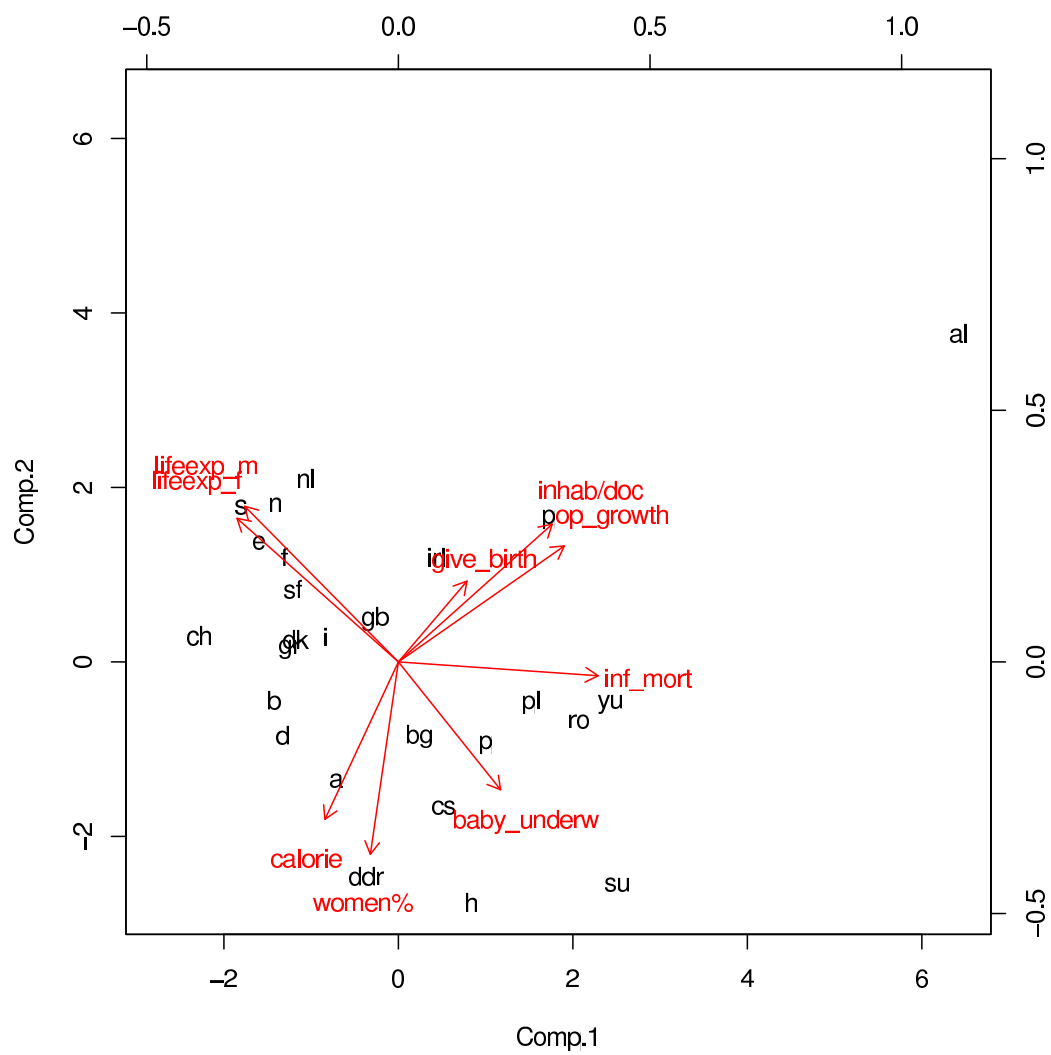
Figure 3.3: Biplot of *princomp* for *euro86*
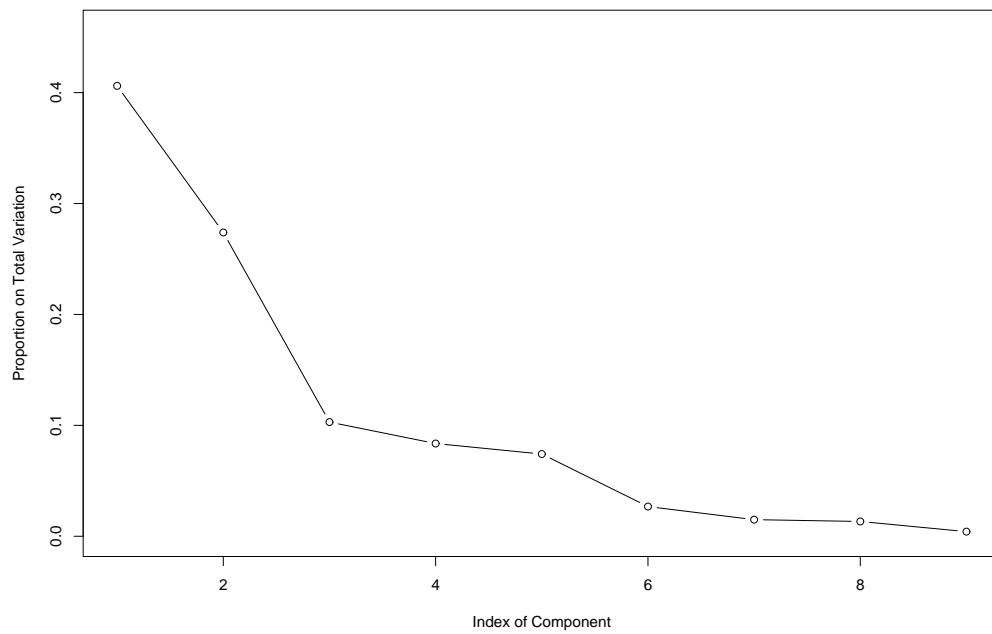
Figure 3.4: Screeplot of *princomp* for *euro86*

A first glance at the biplots shows an outlier: Albania (al). This suspect is proved by further investigation. Its impact on the size and direction becomes clear when we apply the *NIPALS* algorithm once more and look at the biplot of the *euro86* data set reduced by the observation Albania (cf. Figure 3.5). In this case the first and second component explain 58% of the total variation of the remaining data set (cf. Figure 3.6).
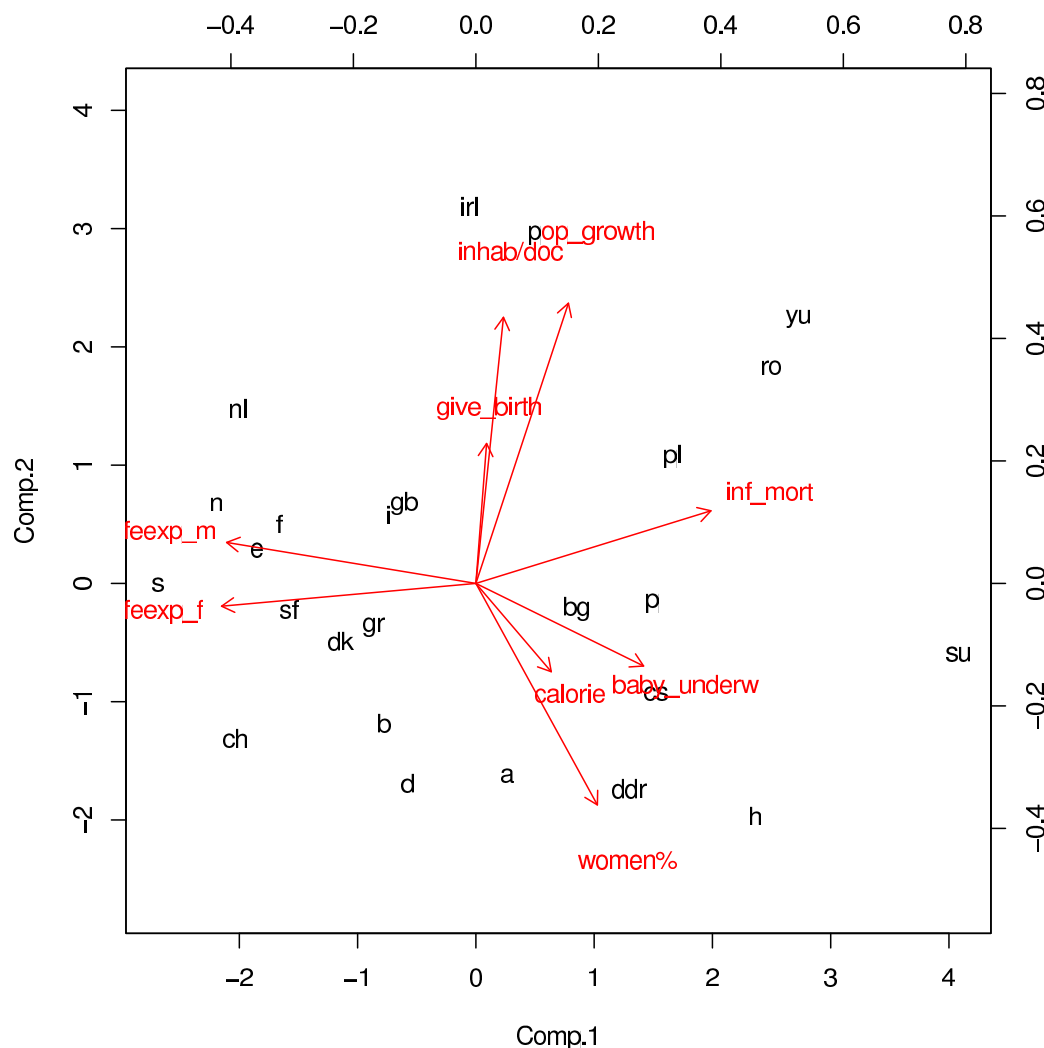
Figure 3.5: Biplot of *NIPALS* for *euro86* without Albania (al)

The same behavior can be found when applying *princomp* to the *euro86*
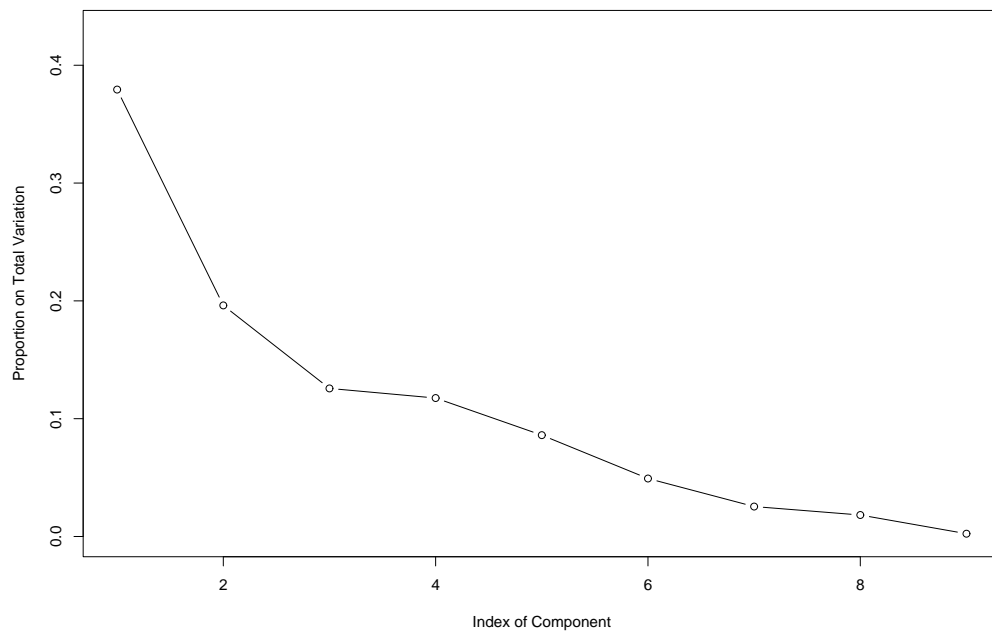
Figure 3.6: Screeplot of *NIPALS* for *euro86* without Albania (al)

data set without Albania (cf. Figure 3.7). Again the first and second component explain 58% of the total variation (cf. Figure 3.8).
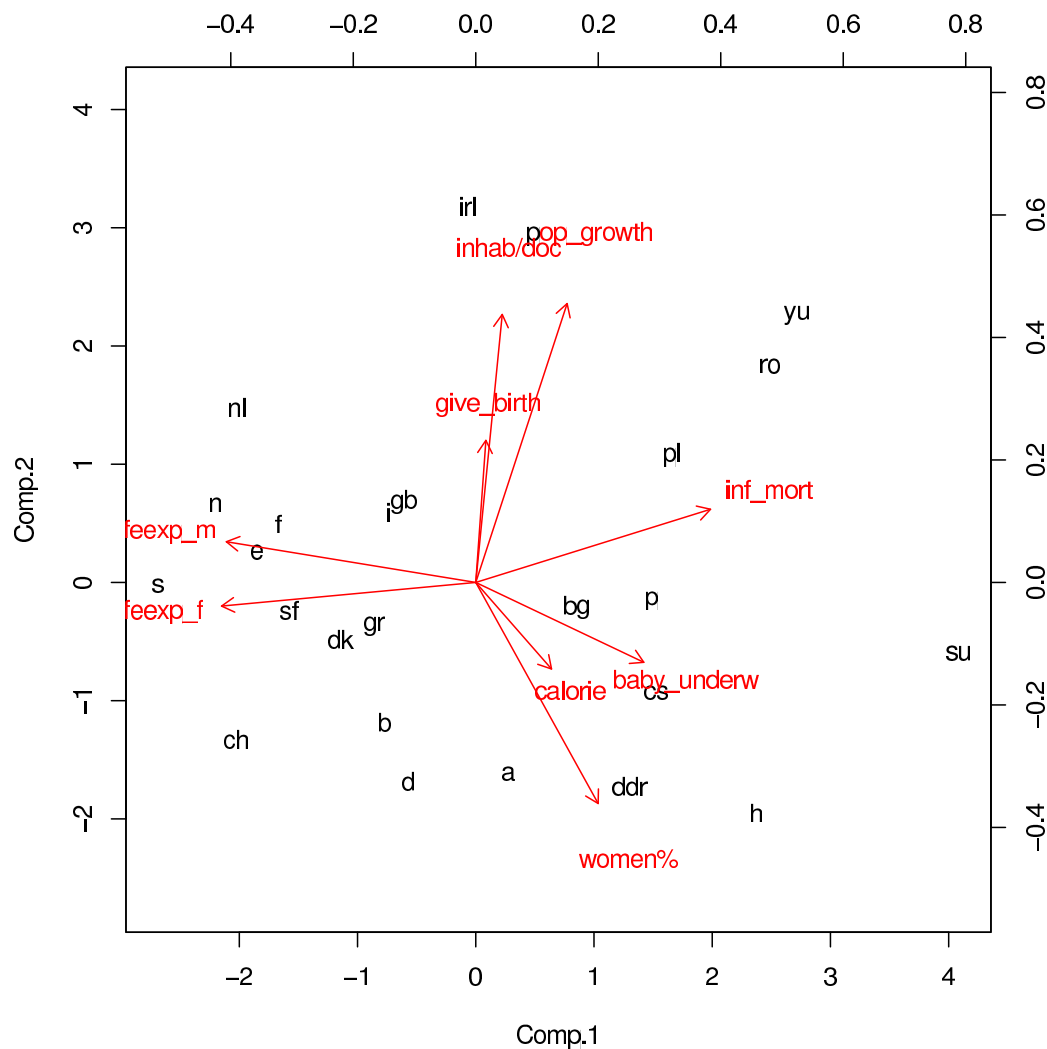
Figure 3.7: Biplot of *princomp* for *euro86* without Albania (al)

The change in the importance of the components calculated with either of the algorithms is striking (cf. Figures 3.2 and 3.6 for the *NIPALS* algorithm and Figures 3.4 and 3.8 for the function *princomp*).

We note that in the case of the reduced data set the *NIPALS* algorithm really gives components of decreasing importance. While the first result
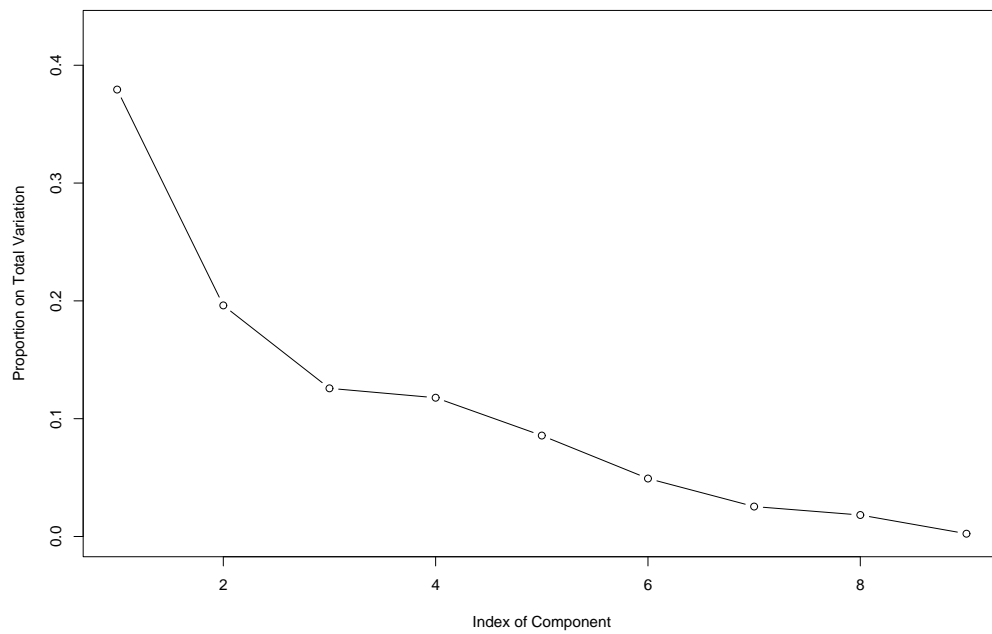
Figure 3.8: Screeplot of *princomp* for *euro86* without Albania (al)

with Albania shows great importance in the first and second component but only minor importance in components 3 to 5, there is a shift of importance mainly from the second towards components 3 to 6. So, in the case of further analysis, if we first decided to use five components (i.e., 94% of the total variation) when working with the contaminated data set, we would now, after the removal of Albania, revise our decision and also take the sixth component into account (i.e., 95% of the total variation, instead of only 90% for components 1 to 5).

As the function *princomp* and the *NIPALS* algorithm give similar results the interpretations of the first and second principal component are valid for both.

The first principal component of the *euro86* data set without Albania (al) has in its negative part female and male life expectation as the only influence. In this region we find the rich countries: the northern European countries (nl, n, s, sf, dk, b) but also Switzerland (ch), Spain (e), Greece (gr), Germany (d), Great Britain (gb), France (f) and Italy (i). Opposed to it, in the positive part, the infant mortality has strong influence along with baby underweight. This makes sense as better life circumstances reduce the cases of infant mortality. There we find the poor countries: the eastern European countries like the Soviet Union (su), Yugoslavia (yu), Romania (ro), Poland (pl), Hungary (h), Czechoslovakia (cs), Bulgaria (bg) and Eastern Germany (ddr), but also Portugal (p).

The second principal component is expressed by high values of calories per day and a higher percentage of women in the negative axis and high population growth combined with women in the right age for giving birth and many inhabitants per doctor in the positive part. Again many inhabitants per doctor reduce the life quality whereas higher portions of calories are a sign of wealth. It is also well-known that in poor countries the population grows faster than in developed countries where it even decreases.

Moreover the variables life expectation of men and women are positively correlated, so are population growth and inhabitants per doctor. On the other hand either of both life expectations and the variable infant mortality are negatively correlated.

The differences between the first two components of the *euro86* data set without Albania calculated with the function *princomp* and the *NIPALS* algorithm are shown in Figure 3.9.

Now, assuming that the values of the variables population growth and inhabitants per doctor of the observation Albania (al) are missing, we replace

them by *NAs* (*Not Available*). After centering and scaling of the data set, in order to obtain principal components we apply the *NIPALS* algorithm for missing values. The biplot of the first and second components is shown in Figure 3.10. They explain 63% of the total variation of the *euro86* data set with missing values. The corresponding screeplot of the *NIPALS* algorithm for missing values is displayed in Figure 3.11. We note that in this case the importance of each component decreases with increasing index.

Furthermore we are able to estimate the missing values according to (3.16). The estimated and original values are compared in Table 3.5.

Table 3.5: Estimated and Original Values of Albania

	estimated	original
pop_growth	1.18	1.80
inhab/doc	856.50	2100.00

We obtain the estimated values in Table 3.5 by applying the inverse calculation of the standardization. The big differences between the estimated and original values result from the fact that Albania appears to be an outlier. However, although the estimates of the variables population growth and inhabitants per doctor are smaller than the original values, they are still greater than the values of all other observations.
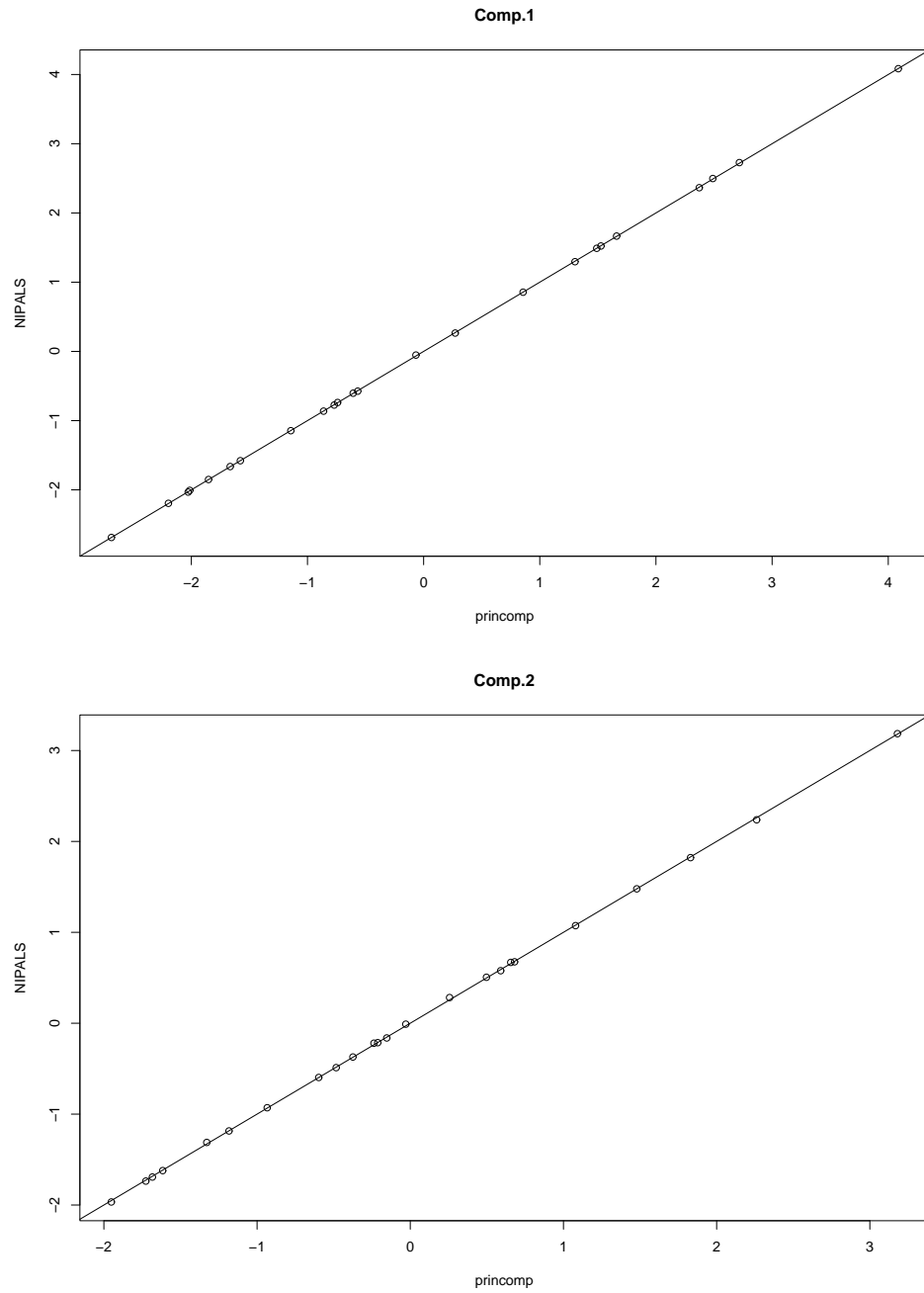
Figure 3.9: Scatterplot of *princomp* versus *NIPALS* in the First and Second Component for *euro86* without Albania (al)
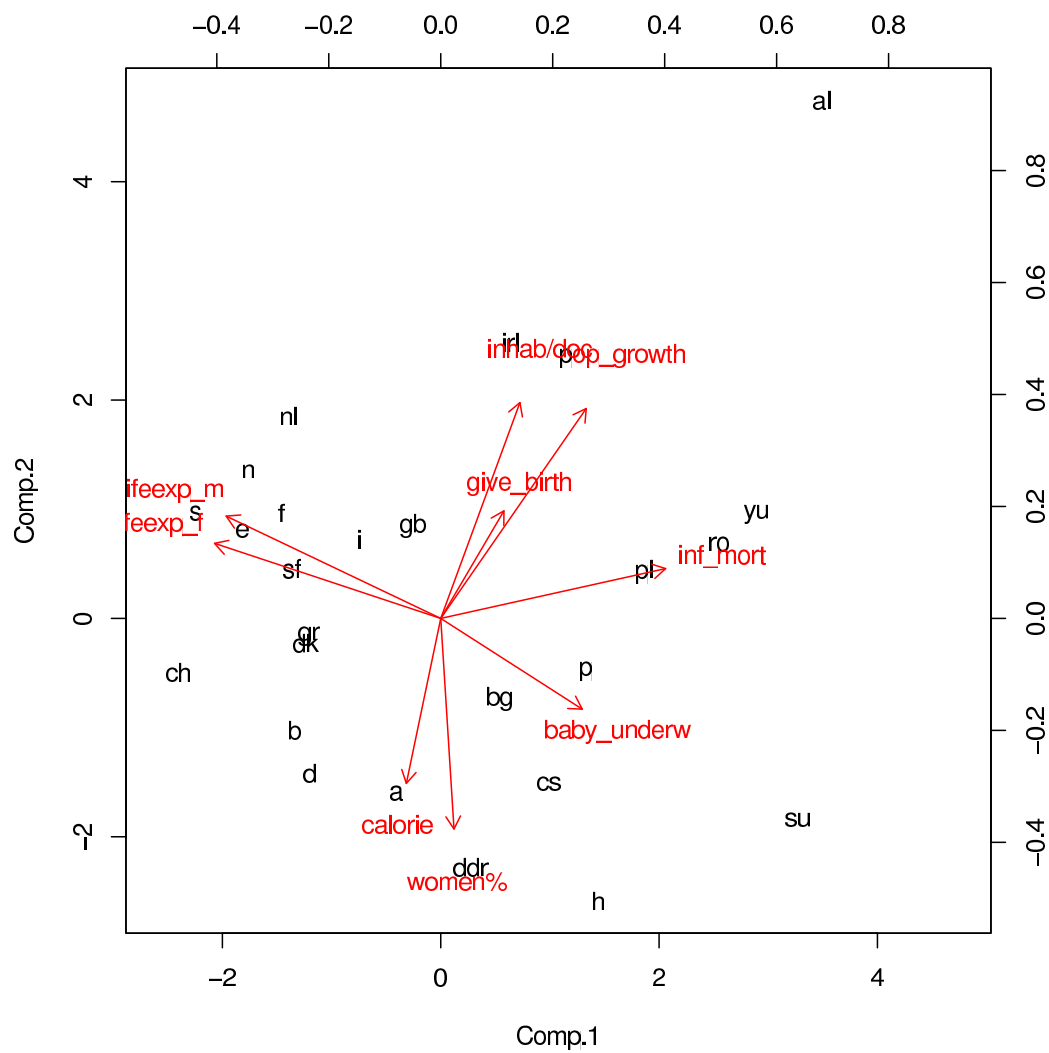
Figure 3.10: Biplot of *NIPALS* for *euro86* with Missing Values of Albania (al)
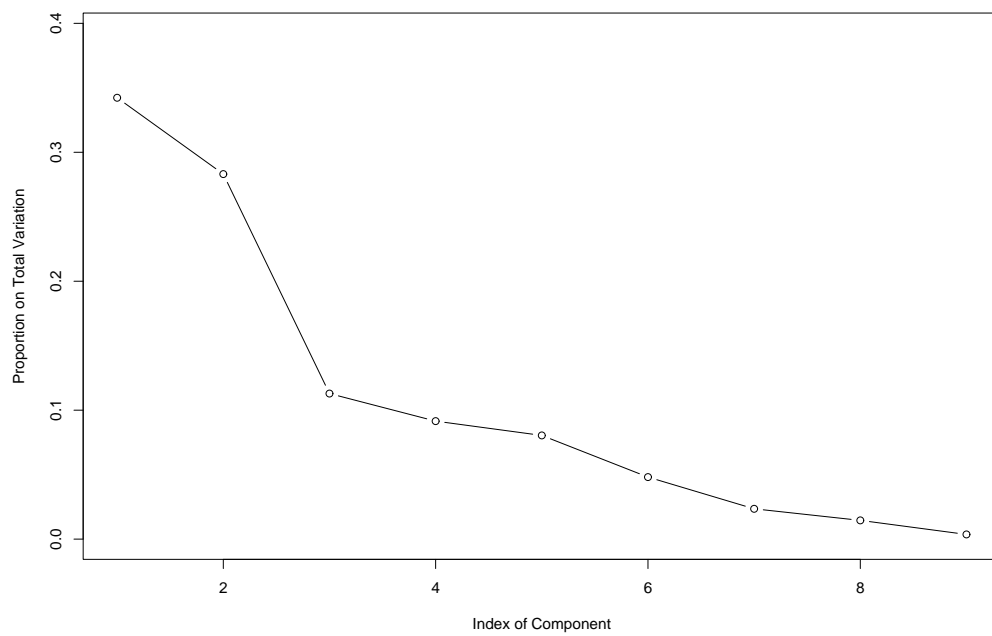
Figure 3.11: Screeplot of *NIPALS* for *euro86* with Missing Values of Albania (al)

Chapter 4

Time Series Analysis

In this section we present the methods which are used for the statistical time series analysis of the diabetes data in Chapter 5.

First, in Section 4.1, we summarize the methods that we use to extract proper time series from the diabetes data base. Then, in Section 4.2, we describe the algorithm that delivers appropriate estimates of the missing values. Last, in Section 4.3, some basic concepts of time series analysis are presented and we describe the autoregressive model with exogenous variables that will be fitted to the diabetes data.

4.1   Extraction of Time Series

In order to apply methods of time series analysis we first extract proper time series from the diabetes data base.

As mentioned before, for each patient the medical parameters have been gathered during medical check-ups and the time between two medical check-ups varies. Therefore, considering a patient's time series of a specific medical parameter, e.g., the time series of *HbA1c*, the time between two measurements is not constant.

Hence, in order to obtain equally spaced measurements in time, we divide the year into trimester because the average of the patients had three check-ups a year. For each medical variable of each patient we aim for getting a time series with one measurement each trimester. For each single patient, starting with that trimester that contains the first medical check-up, we continue until the last check-up as described in the following. Considering

only one medical variable first, if there are more measurements than one during a trimester we will randomly take one value. This is done because of statistical inference. If we took the mean instead this would lead to different variances of the measurements of a parameter's time series. If there is only one measurement during a trimester we will take this one. Finally, if there is no measurement during the trimester we put the value to *Not Available* (*"NA"*). This is done for each medical variable.

Now the resulting time series which we obtain for each medical variable of each patient after applying the above procedure are equally spaced in time with exactly one measurement each trimester which either exists or is *NA*. The way we cope with the *NA*s is described in detail in a later section (cf. Section 4.2).

Anyway, we note that, considering one patient, all time series of different medical variables start in the same trimester and have an equal number of measurements. On the contrary, considering different patients, the time series of the same medical variable may start in different trimester and may also have a different number of measurements. Therefore, in order to compare time series of different patients, we choose the trimester of the first medical check-up of each patient as common starting value.

For further time series analysis we concentrate on those patients with relatively "long" time series. Additionally we demand that the time series of those medical variables that we have selected for further investigations only have few missing values (see also Section A.3 for details).

4.2   Estimation of Missing Values

In this section let us consider only one single patient first. As mentioned before, the time series of some medical variables may still contain missing values. We now estimate these missing values taking the information of the remaining medical variables into account. This method that makes use of the correlative structure of the data is called *First-Order Regression* (*FOR*) and is described in Rao and Toutenburg (1995).

Let $\boldsymbol{X}$ be the $n \times m$ data matrix of a single patient with m medical variables and n trimester. We further pretend, only for the estimation of the missing values, that the n measurements of each variable are independent. Let

$$I_j := \{i : x_{ij} \text{ is missing}\} \ , \quad j = 1, \ldots, m \ , \tag{4.1}$$

36

denote the index set of the missing values of $\boldsymbol{x}_{.j}$, where $\boldsymbol{x}_{.j}$ is the j-th column of the data matrix $\boldsymbol{X}$. Further let $I = \bigcup_{j=1}^{m} I_j$. Now the dependence of any column $\boldsymbol{x}_{.j}$, $j \in \{1, \ldots, m\}$, with the remaining columns is modeled by additional regressions, i.e.,

$$x_{ij} = \theta_{0j} + \sum_{\substack{k=1 \\ k \neq j}}^{m} x_{ik}\theta_{kj} + \varepsilon_{ij} , \quad i \notin I . \tag{4.2}$$

The missing values x_{ij} of $\boldsymbol{X}$ are estimated and replaced by

$$\widehat{x}_{ij} = \widehat{\theta}_{0j} + \sum_{\substack{k=1 \\ k \neq j}}^{m} x_{ik}\widehat{\theta}_{kj} , \quad i \in I_j . \tag{4.3}$$

However, we note that when calculating $\widehat{x}_{ij}$, with $i \in I_j$ and j fixed, all values x_{ik}, $k \in \{1, \ldots, m\}$ and $k \neq j$, have to exist.

Therefore in case of non disjoint sets of indices I_j we have to choose some initial values. Hence, in order not to destroy a possible trend of the time series of each medical variable, we propose the following: If the first or the last value of the time series is missing we will replace it by the neighboring value, i.e., by the value of the second trimester or the trimester before the last, respectively. On the contrary, if k values $x_{i+\ell,j}$, $\ell = 1, \ldots, k$, of the j-th medical variable are missing in between, we will linearly interpolate them, i.e.,

$$x_{i+\ell,j} := \frac{x_{i+k+1,j} - x_{ij}}{k+1} \ell , \quad \ell = 1, \ldots, k . \tag{4.4}$$

The procedure described above is applied to all remaining patients. For each patient we now obtain m complete time series.

4.3 Autoregressive Moving Average Models

Generally a time series can be considered as a collection of random variables indexed according to the order they are obtained in time. This collection is usually referred to as a *stochastic process* and the observed values are referred to as a *realization* of the underlying stochastic process.

The simplest kind of time series is the *univariate time series*

$$x_t , \quad t = 1, \ldots, n , \tag{4.5}$$

which we use to introduce the basic concepts of time series analysis. In this text the index t will vary over the integers or some subset of the integers.

For example let us consider a collection of uncorrelated random variables w_t with mean 0 and finite variance σ_w^2. Such a generated time series is called *white noise*. An important white noise series is *Gaussian white noise*, wherein the w_t are *independent and identically distributed (iid) normal random variables*, with mean 0 and variance σ_w^2.

When there are more than one jointly measured time series it will be useful to consider the notion of a *vector-valued time series* $\boldsymbol{x}_t = (x_{1t}, \ldots, x_{mt})^\top$ which contains as its components m univariate time series.

In the case of the diabetes data we observe *cross-sectional data* $\boldsymbol{y}_{t\ell}$, where $\boldsymbol{y}_{t\ell}$ denotes the m-dimensional time series of the patient ℓ, $\ell = 1, \ldots, N$. Each time series $\boldsymbol{y}_{t\ell}$ contains as its components the univariate time series $y_{i,t,\ell}$ of m medical variables with $i = 1, \ldots, m$, and we assume that time series of different patients are independent.

4.3.1 Autocorrelation and Cross-correlation Function

The *mean value function* of a univariate time series x_t, $t = 1, \ldots, n$, is defined as

$$\mu_t = E(x_t) . \tag{4.6}$$

The lack of independence between two adjacent values x_s and x_t of the same time series can be assessed numerically, as in classic statistics, using the notion of covariance. Hence, the *autocovariance function* is defined as the second moment product

$$\gamma_x(s, t) = E((x_s - \mu_s)(x_t - \mu_t)) , \text{ for all } s \text{ and } t. \tag{4.7}$$

The autocovariance function measures the linear dependence between two points on the same series observed at different times.

The preceding definitions of the mean and autocovariance functions are completely general. However, in order to do some statistical inference, we have to assume that a sort of regularity may exist over time in the behavior of a time series. Therefore we introduce the concept of stationarity. A *weakly stationary time series* has to fulfill the following properties: (i) the mean value function μ_t in (4.6) is constant, i.e., it does not depend on time t, and (ii) the autocovariance function $\gamma_x(s, t)$ in (4.7) depends on s and t only by their difference $h = s - t$, where h is called the *lag*. Now we are able to

define the mean value and autocovariance function of a weakly stationary time series as

$$E(x_t) = \mu_x \tag{4.8}$$

and

$$\gamma_x(h) = E((x_{t+h} - \mu_x)(x_t - \mu_x)) , \tag{4.9}$$

where, for convenience, we write $\gamma_x(h)$ instead of $\gamma_x(t+h, t)$. We note that the autocovariance function satisfies $\gamma_x(h) = \gamma_x(-h)$. The *autocorrelation function (ACF)* of a stationary time series can be written as

$$\rho_x(h) = \frac{\gamma_x(t+h, t)}{\sqrt{\gamma_x(t+h, t+h)\gamma_x(t, t)}} = \frac{\gamma_x(h)}{\gamma_x(0)} . \tag{4.10}$$

When several stationary time series are available, say, x_t and y_t, we often would like to measure the predictability of the series y_t from the series x_t, leading to the notion of the *cross-covariance function* of stationary time series,

$$\gamma_{xy}(h) = E((x_{t+h} - \mu_x)(y_t - \mu_y)) . \tag{4.11}$$

We note that the cross-covariance function satisfies $\gamma_{xy}(h) = \gamma_{yx}(-h)$. The scaled version of the above, called *cross-correlation function (CCF)* of stationary time series, is defined as

$$\rho_{xy}(h) = \frac{\gamma_{xy}(h)}{\sqrt{\gamma_x(0)\gamma_y(0)}} . \tag{4.12}$$

Estimation

Assuming stationarity we are able to estimate the mean value function (4.8) if it is constant by replacing the average over the population, denoted by E, with an average over the sample, say,

$$\bar{x} = \frac{1}{n}\sum_{t=1}^{n} x_t \tag{4.13}$$

and the theoretical autocovariance (4.9) by the *sample autocovariance function*

$$\widehat{\gamma}_x(h) = \frac{1}{n}\sum_{t=1}^{n-h}(x_{t+h} - \bar{x})(x_t - \bar{x}) , \quad h = 0, \ldots, n-1 , \tag{4.14}$$

with $\widehat{\gamma}_x(-h) = \widehat{\gamma}_x(h)$. The estimator in (4.14) is generally preferred to the one that would be obtained by dividing by $n - h$ because (4.14) is a non-negative definite function. We note that neither dividing by n nor $n - h$ in (4.14) yields an unbiased estimator of $\gamma_x(h)$ (cf. Shumway and Stoffer, 2000; Deistler and Scherrer, 1994).

The *sample autocorrelation function* of a stationary time series is defined, analogously to (4.10), as

$$\widehat{\rho}_x(h) = \frac{\widehat{\gamma}_x(h)}{\widehat{\gamma}_x(0)} . \tag{4.15}$$

Further we can prove that if the series x_t is a white noise process, then $\widehat{\rho}_x(h)$ will be approximately normal with mean $\rho_x(h) = 0$ for all $h \neq 0$, and the standard error reduces to

$$\sigma_{\widehat{\rho}_x(h)} = \frac{1}{\sqrt{n}} . \tag{4.16}$$

Based on the above result, we obtain a rough method whether correlations are statistically significant at some lags by determining whether the observed peak in $\widehat{\rho}_x(h)$ is outside the interval $\pm z_{1-\alpha/2}/\sqrt{n}$, where $z_{1-\alpha/2}$ denotes the value of the $(1 - \alpha/2)$ quantile.

The estimators for the cross-covariance function $\gamma_{xy}(h)$ as given in (4.11) and the cross-correlation function $\rho_{xy}(h)$ in (4.12) are given by the *sample cross-covariance function* of stationary time series

$$\widehat{\gamma}_{xy}(h) = \frac{1}{n} \sum_{t=1}^{n-h} (x_{t+h} - \bar{x})(y_t - \bar{y}) , \quad h = 0, \ldots, n-1 , \tag{4.17}$$

where $\widehat{\gamma}_{xy}(-h) = \widehat{\gamma}_{yx}(h)$ determines the function for negative lags, and the *sample cross-correlation function*

$$\widehat{\rho}_{xy}(h) = \frac{\widehat{\gamma}_{xy}(h)}{\sqrt{\widehat{\gamma}_x(0)\widehat{\gamma}_y(0)}} . \tag{4.18}$$

Again we have a similar result as for the sample autocorrelation function. The large sample distribution of $\widehat{\rho}_{xy}(h)$ is normal with mean zero and

$$\sigma_{\widehat{\rho}_{xy}(h)} = \frac{1}{\sqrt{n}} , \tag{4.19}$$

if at least one of the processes is white noise.

In case of cross-sectional data we propose the following modification of the sample autocovariance function $\widehat{\gamma}_x(h)$ as given in (4.14) and define the *sample autocovariance function for cross-sectional data* as

$$\widehat{\gamma}_i(h) = \frac{1}{Nn} \sum_{\ell=1}^{N} \sum_{t=1}^{n-h} (y_{i,t+h,\ell} - \bar{y}_i)(y_{i,t,\ell} - \bar{y}_i) , \quad h = 0, \dots, n-1 , \quad (4.20)$$

where $\bar{y}_i = \frac{1}{Nn} \sum_{\ell=1}^{N} \sum_{t=1}^{n} y_{i,t,\ell}$ and $\widehat{\gamma}_i(-h) = \widehat{\gamma}_i(h)$, with $i = 1, \dots, m$.

As above the *sample autocorrelation function for cross-sectional data* is defined as

$$\widehat{\rho}_i(h) = \frac{\widehat{\gamma}_i(h)}{\widehat{\gamma}_i(0)} . \quad (4.21)$$

Similarly we modify the sample cross-covariance function $\widehat{\gamma}_{xy}(h)$ as given in (4.17) and the sample cross-correlation function $\widehat{\rho}_{xy}(h)$ in (4.18) and define the *sample cross-covariance function for cross-sectional data* as

$$\widehat{\gamma}_{ij}(h) = \frac{1}{Nn} \sum_{\ell=1}^{N} \sum_{t=1}^{n-h} (y_{i,t+h,\ell} - \bar{y}_i)(y_{j,t,\ell} - \bar{y}_j) , \quad h = 0, \dots, n-1 , \quad (4.22)$$

where $\widehat{\gamma}_{ij}(-h) = \widehat{\gamma}_{ji}(h)$ determines the function for negative lags, and the *sample cross-correlation function* as

$$\widehat{\rho}_{ij}(h) = \frac{\widehat{\gamma}_{ij}(h)}{\sqrt{\widehat{\gamma}_i(0)\widehat{\gamma}_j(0)}} . \quad (4.23)$$

4.3.2 Partial Autocorrelation Function

Formally, for a stationary time series x_t, $t = 0, 1, 2, \dots$, we define the *partial autocorrelation function (PACF)* ϕ_{hh}, $h = 1, 2, \dots$, by

$$\begin{aligned}
\phi_{00} &= 1 = \rho_x(0) \\
\phi_{11} &= \text{corr}(x_1, x_0) = \rho_x(1) \\
\phi_{hh} &= \text{corr}(x_h - x_h^{h-1}, x_0 - x_0^{h-1}) , \quad h \geq 2, \quad (4.24)
\end{aligned}$$

where x_t^{h-1} and x_{t-h}^{h-1} denote the best linear predictors of x_t and x_{t-h}, respectively, based on $\{x_{t-(h-1)}, \dots, x_{t-1}\}$ with $t = h$. We will go into detail later, for now we only note that both $(x_h - x_h^{h-1})$ and $(x_0 - x_0^{h-1})$ are uncorrelated with $\{x_1, \dots, x_{h-1}\}$. Because of stationarity the PACF ϕ_{hh} is the correlation

41

between x_t and x_{t-h} with the linear effect of $\{x_{t-(h-1)}, \ldots, x_{t-1}\}$ on each removed.

Moreover we can interpret the PACF in the following way. To ease notation we assume that x_t is stationary with mean zero. Let

$$\varepsilon_t = x_t - \sum_{j=1}^{h-1} a_j x_{t-j}$$

and

$$\delta_{t-h} = x_{t-h} - \sum_{k=1}^{h-1} b_k x_{t-k}$$

be the two residuals where $\{a_1, \ldots, a_{h-1}\}$ and $\{b_1, \ldots, b_{h-1}\}$ are chosen so that they minimize the mean square errors

$$E(\varepsilon_t^2) \text{ and } E(\delta_{t-h}^2) \ .$$

Then the PACF at lag h can be defined as the cross-correlation between ε_t and δ_{t-h}, i.e.,

$$\phi_{hh} = \frac{E(\varepsilon_t \delta_{t-h})}{\sqrt{E(\varepsilon_t^2) E(\delta_{t-h}^2)}} \ . \tag{4.25}$$

Relation Partial Autocorrelation Function and Best Linear Predictor

Let us suppose that x_t is a stationary time series and we want to predict future values x_{n+m}, $m = 1, 2, \ldots$, based on given data $\{x_{n-(h-1)}, \ldots, x_n\}$. Generally the *minimum mean square error predictor* of x_{n+m} is $x_{n+m}^h = E(x_{n+m}|x_n, \ldots, x_{n-(h-1)})$ because the conditional expectation minimizes the mean square error

$$E(x_{n+m} - g(x_{n-(h-1)}, \ldots, x_n))^2 \ , \tag{4.26}$$

where $g(x_{n-(h-1)}, \ldots, x_n)$ denotes a measurable function of the observations $x_{n-(h-1)}, \ldots, x_n$. Linear predictors of the form

$$x_{n+m}^h = a_0 + \sum_{k=1}^{h} a_k x_{n-(k-1)} \ , \tag{4.27}$$

that minimize the mean square error (4.26) are called *best linear predictors* *(BLP)*.

42

For a stationary process we have the following property. Given data $x_{n-(h-1)}, \ldots, x_n$, the best linear predictor, $x_{n+m}^h = a_0 + \sum_{k=1}^h a_k x_{n-(k-1)}$, of x_{n+m}, for $m \geq 1$, is found by solving

$$
\begin{aligned}
E(x_{n+m} - x_{n+m}^h) &= 0 \\
E((x_{n+m} - x_{n+m}^h)x_{n-(k-1)}) &= 0 , \quad k = 1, \ldots, h .
\end{aligned}
\tag{4.28}
$$

The equations specified in (4.28) are called the *prediction equations*.

Now we consider *one-step-ahead prediction* and assume, to ease notation, that $E(x_t) = 0$, which means $a_0 = 0$. The BLP of x_{n+1} is, given $x_{n-(h-1)}, \ldots, x_n$,

$$
x_{n+1}^h = \phi_{h1} x_n + \phi_{h2} x_{n-1} + \ldots + \phi_{hh} x_{n-(h-1)} ,
\tag{4.29}
$$

where we have written a_k in (4.27) as ϕ_{hk} in (4.29), for $k = 1, \ldots, h$. Using the prediction equations, the coefficients $\phi_{h1}, \ldots, \phi_{hh}$ satisfy

$$
E\left((x_{n+1} - \sum_{j=1}^h \phi_{hj} x_{n+1-j}) x_{n+1-k}\right) = 0 , \quad k = 1, \ldots, h ,
$$

or

$$
\sum_{j=1}^h \phi_{hj} \gamma_x(k - j) = \gamma_x(k) , \quad k = 1, \ldots, h .
\tag{4.30}
$$

The prediction equation (4.30) can be written in matrix notation as

$$
\mathbf{\Gamma}_h \boldsymbol{\phi}_h = \boldsymbol{\gamma}_h ,
\tag{4.31}
$$

where $\mathbf{\Gamma}_h = (\gamma_x(k - j))_{j,k=1}^h$ is an $h \times h$ matrix, $\boldsymbol{\phi}_h = (\phi_{h1}, \ldots, \phi_{hh})^\top$ is an $h \times 1$ vector, and $\boldsymbol{\gamma}_h = (\gamma_x(1), \ldots, \gamma_x(h))^\top$ is an $h \times 1$ vector.

Furthermore we can prove that the following equation holds:

$$
\phi_{hh} = \frac{\rho_x(h) - \widetilde{\boldsymbol{\rho}}_{h-1}^\top \boldsymbol{R}_{h-1}^{-1} \boldsymbol{\rho}_{h-1}}{1 - \widetilde{\boldsymbol{\rho}}_{h-1}^\top \boldsymbol{R}_{h-1}^{-1} \widetilde{\boldsymbol{\rho}}_{h-1}} = a_h ,
\tag{4.32}
$$

where $\boldsymbol{\rho}_{h-1} = (\rho_x(1), \ldots, \rho_x(h-1))^\top$, $\widetilde{\boldsymbol{\rho}}_{h-1} = (\rho_x(h-1), \ldots, \rho_x(1))^\top$ and $\boldsymbol{R}_h$ is the $h \times h$ matrix with elements $\rho_x(i - j)$, $i, j = 1, \ldots, h$ (cf. also Shumway and Stoffer, 2000; Hartung et al., 1998).

In the case of cross-sectional data we propose to use the results of the sample autocovariance function (4.20) or the sample autocorrelation function (4.21) in order to obtain estimators for the *partial autocorrelation function for cross-sectional data* according to (4.31) or (4.32), respectively.

43

4.3.3 Models

Now we proceed with the general definition of *autoregressive moving average (ARMA) models* for stationary time series. As before, to ease notation, we assume that the time series x_t has mean zero. A univariate time series x_t, $t = 0, \pm 1, \pm 2, \ldots$, is said to be ARMA$(p, q)$ if x_t is stationary and

$$x_t = \phi_1 x_{t-1} + \ldots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \ldots + \theta_q w_{t-q} , \qquad (4.33)$$

with $\phi_p \neq 0$ and $\theta_q \neq 0$. In (4.33) w_t denotes a white noise process with $\sigma_w^2 > 0$. The parameters p and q are called the autoregressive and moving average orders, respectively. If $q = 0$ the model is called an autoregressive model of order p, AR(p), and if $p = 0$ the model is called a moving average model of order q, MA(q).

Additionally we also require in (4.33) that $\phi(z)$ and $\theta(z)$ have no common factors, where the *AR* and *MA polynomials*, $\phi(z)$ and $\theta(z)$, are defined as

$$\phi(z) = 1 - \phi_1 z - \ldots - \phi_p z^p , \quad \phi_p \neq 0 , \qquad (4.34)$$

and

$$\theta(z) = 1 + \theta_1 z + \ldots + \theta_q z^q , \quad \theta_q \neq 0 , \qquad (4.35)$$

respectively, where z is a complex number.

Using the *backshift operator B*, defined as

$$Bx_t = x_{t-1} , \qquad (4.36)$$

we can write the ARMA(p, q) model in (4.33) as

$$\phi(B)x_t = \theta(B)w_t . \qquad (4.37)$$

Further in order to obtain models that do not depend on the future and are unique, we will require some additional restrictions on the model parameters in (4.33). An ARMA(p, q) model, $\phi(B)x_t = \theta(B)w_t$, is said to be *causal*, if the time series x_t, $t = 0, \pm 1, \pm 2, \ldots$, can be written as a one-sided linear process:

$$x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j} = \psi(B)w_t , \qquad (4.38)$$

where $\psi(B) = \sum_{j=0}^{\infty} \psi_j B^j$ and $\sum_{j=0}^{\infty} |\psi_j| < \infty$; we set $\psi_0 = 1$.

Moreover an ARMA(p, q) model, $\phi(B)x_t = \theta(B)w_t$, is said to be *invertible*, if the time series x_t, $t = 0, \pm 1, \pm 2, \ldots$, can be written as

$$\pi(B)x_t = \sum_{j=0}^{\infty} \pi_j x_{t-j} = w_t \; , \tag{4.39}$$

where $\pi(B) = \sum_{j=0}^{\infty} \pi_j B^j$ and $\sum_{j=0}^{\infty} |\pi_j| < \infty$; π_0 is set equal to one.

In general we have the following property. An ARMA(p, q) model is causal only if the roots of $\phi(z)$ lie outside the unit circle, i.e., $\phi(z) = 0$ only if $|z| > 1$. Analogously an ARMA(p, q) model is invertible only if the roots of $\theta(z)$ lie outside the unit circle.

Anyway, considering a causal AR(p) process, one can prove that

$$\phi_{hh} = 0 \; , \quad \text{for all } h > p \; ,$$

whereas the ACF will never cut off. On the other hand, for an invertible MA(q) process, we can prove that the PACF will never cut off, as in the case of an AR(p) process, whereas the ACF cuts off after lag q. We summarize these results in Table 4.1.

Table 4.1: Behavior of the ACF and PACF for Causal and Invertible ARMA Models

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

Vector-valued Models

The multivariate autoregressive model is a straight-forward extension of the univariate AR model. Considering an m-dimensional vector-valued time series $\boldsymbol{x}_t$, $t = 1, \ldots, n$, we define the multivariate *autoregressive model of order p with exogenous variables, ARX(p)*, as

$$\boldsymbol{x}_t = \boldsymbol{\Gamma}\boldsymbol{u}_t + \sum_{j=1}^{p} \boldsymbol{\Phi}_j \boldsymbol{x}_{t-j} + \boldsymbol{w}_t \; , \quad t = p+1, \ldots, n \; , \tag{4.40}$$

where $\boldsymbol{w}_t$ is an m-dimensional *vector-valued white noise process* and defined by its covariance matrix

$$E(\boldsymbol{w}_t \boldsymbol{w}_t^\top) = \boldsymbol{\Sigma}_w \ . \tag{4.41}$$

That means that w_{it} and w_{jt}, $i \neq j$ and $i, j \in \{1, \ldots, m\}$ may be correlated but $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$, $s \neq t$, are uncorrelated. The matrices $\boldsymbol{\Phi}_j$, $j = 1, \ldots, p$, are the $m \times m$ *transition matrices* which express the dependence of $\boldsymbol{x}_t$ on $\boldsymbol{x}_{t-j}$. The matrix $\boldsymbol{\Gamma}$ is an $m \times r$ parameter matrix. The X in ARX refers to the exogenous vector process which we have denoted here by $\boldsymbol{u}_t$. In the simplest case of exogenous variables, for example, considering that $\boldsymbol{x}_t$ has mean $\boldsymbol{\mu}$, we can set $r = 1$, $\boldsymbol{u}_t = 1$ and $\boldsymbol{\Gamma} = \boldsymbol{\alpha}$ with $\boldsymbol{\alpha} = (\boldsymbol{I} - \boldsymbol{\Phi}_1 - \ldots - \boldsymbol{\Phi}_p)\boldsymbol{\mu}$.

Using matrix notation, (4.40) can be written as

$$\boldsymbol{x}_t = \boldsymbol{B}\boldsymbol{z}_t + \boldsymbol{w}_t \ , \quad t = p+1, \ldots, n \ , \tag{4.42}$$

with $\boldsymbol{B} = (\boldsymbol{\Gamma}, \boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_p)$ and $\boldsymbol{z}_t = (\boldsymbol{u}_t^\top, \boldsymbol{x}_{t-1}^\top, \ldots, \boldsymbol{x}_{t-p}^\top)^\top$.

In this case the *ordinary least squares (OLS) estimator* for the matrix $\boldsymbol{B}$ in the above multivariate regression model is

$$\widehat{\boldsymbol{B}} = (\boldsymbol{X}\boldsymbol{Z}^\top)(\boldsymbol{Z}\boldsymbol{Z}^\top)^{-1} \ , \tag{4.43}$$

with $\boldsymbol{X} = (\boldsymbol{x}_{p+1}, \ldots, \boldsymbol{x}_n)$ and $\boldsymbol{Z} = (\boldsymbol{z}_{p+1}, \ldots, \boldsymbol{z}_n)$.

Further an estimator for the *error covariance* $\boldsymbol{\Sigma}_w$ is

$$\widehat{\boldsymbol{\Sigma}}_w = \frac{1}{n-p} \sum_{t=p+1}^{n} (\boldsymbol{x}_t - \widehat{\boldsymbol{B}}\boldsymbol{z}_t)(\boldsymbol{x}_t - \widehat{\boldsymbol{B}}\boldsymbol{z}_t)^\top \ . \tag{4.44}$$

Anyway, we can easily extend the model in (4.40) to handle cross-sectional data (cf. Anderson, 1978). The *ARX(p) model for cross-sectional data* is defined as

$$\boldsymbol{y}_{t\ell} = \boldsymbol{\Gamma}\boldsymbol{u}_{t\ell} + \sum_{j=1}^{p} \boldsymbol{\Phi}_j \boldsymbol{y}_{t-j,\ell} + \boldsymbol{w}_{t\ell} \ , \quad t = p+1, \ldots, n \ , \tag{4.45}$$

where, for $\ell = 1, \ldots, N$, $\mathrm{cov}(\boldsymbol{w}_{t\ell}) = \boldsymbol{\Sigma}_w$ and $\boldsymbol{u}_{t\ell}$ represents the $r \times 1$ vector of exogenous variables.

As before we can write this replicated ARX(p) model using matrix notation as

$$\boldsymbol{y}_{t\ell} = \boldsymbol{B}\boldsymbol{z}_{t\ell} + \boldsymbol{w}_{t\ell} \ , \tag{4.46}$$

for $\ell = 1, \ldots, N$ and $t = p + 1, \ldots, n$, where

$$z_{t\ell} = (u_{t\ell}^\top, y_{t-1,\ell}^\top, \ldots, y_{t-p,\ell}^\top)^\top \tag{4.47}$$

and the matrix B is analogously defined as given in (4.42).

Similarly, as in (4.43), the OLS estimator of B in this case is

$$\widehat{B} = \left(\sum_{\ell=1}^{N} \sum_{t=p+1}^{n} y_{t\ell} z_{t\ell}^\top \right) \left(\sum_{\ell=1}^{N} \sum_{t=p+1}^{n} z_{t\ell} z_{t\ell}^\top \right)^{-1} , \tag{4.48}$$

and an estimator of Σ_w is given, like in (4.44), as

$$\widehat{\Sigma}_w = \frac{1}{N(n-p)} \sum_{\ell=1}^{N} \sum_{t=p+1}^{n} (y_{t\ell} - \widehat{B} z_{t\ell})(y_{t\ell} - \widehat{B} z_{t\ell})^\top . \tag{4.49}$$

If we additionally assume that the error process $w_{t\ell}$ is normally distributed, we can evaluate the uncertainty in the estimators. Then the large sample standard error of the ij-th element of $B = (b_{ij})$, $i = 1, \ldots, m$ and $j = 1, \ldots, r + pm$, is

$$\text{s.e.}(b_{ij}) = \sqrt{\widehat{\sigma}_{ii} c_{jj}} , \tag{4.50}$$

where $\widehat{\sigma}_{ii}$ is the i-th diagonal element of $\widehat{\Sigma}_w$ and c_{jj} is the j-th diagonal element of

$$\left(\sum_{\ell=1}^{N} \sum_{t=p+1}^{n} z_{t\ell} z_{t\ell}^\top \right)^{-1} . \tag{4.51}$$

Chapter 5

Results

In this section we present the results of the principal component analysis (PCA) and of the time series analysis of the diabetes data set.

First, in Section 5.1, we show the results of the *NIPALS* algorithm for missing values applied to our diabetes data. Then, in Section 5.2, we summarize the results of the time series analysis and fit an autoregressive model with exogenous variables to our cross-sectional diabetes data.

5.1 Multivariate Analysis

Now, before applying the *NIPALS* algorithm for missing values, which is described in detail in Section 3.1.1 to our diabetes data to obtain principal components we give a brief summary what has happened to the data until now.

First we restandardized the value of *HbA1c* (cf. Section 2.3 for details). Then, applying the *AUC* algorithm, which is described in Section 2.4 to the diabetes data set we got a data matrix with one average value for each medical variable and each patient. Afterwards we used the function *log(log(.))* to transform the values of both kidney parameters because of their skewness. All nine variables used are described in detail in Section A.2.

However, there are still some missing values which means that for a few patients some medical variables were never measured by the laboratory or during a medical check-up. Moreover, as mentioned in Section 3.1.1, we have to omit all rows, i.e., all patients that only contain missing values. This yields to a data matrix of 848 different patients (observations) and nine different

medical parameters (variables) with 18% of the values missing.

After centering and scaling of the data set, in order to obtain principal components, we apply the *NIPALS* algorithm for missing values. The biplot of the first and second components is shown in Figure 5.1. They explain 36% of the total variation of the diabetes data set. The corresponding screeplot of the *NIPALS* algorithm for missing values is displayed in Figure 5.2. As we can see in Figure 5.2 the variation of the components does not decrease with increasing index. The peak of the sixth component (40%) is caused by the patients with numbers *"233"*, *"624"*, *"639"*, *"849"* and *"851"*. Of these patients only the *blood pressure* was measured once.

When looking at the biplot (cf. Figure 5.1) we can easily see some groups of patients that seem to be outliers and need to be investigated further. For the group of persons on the left side with the numbers *"860"*, *"579"* and *"586"* only the medical variable *HbA1c* was measured once and all of them do not suffer from diabetes. Further there is another group of patients on the lower side with the numbers *"519"*, *"875"*, *"393"*, *"575"* and *"345"*. The first four persons are also no diabetic patients. The *body mass index* of patient *"345"* is extremely high (value: 131.5) which appears to be a measurement error (cf. Section A.2). Moreover the patient *"923"* on the right side with a high *HbA1c* and *triglyceride* level had only once a medical check-up. Furthermore the average level of *HbA1c* of the patient *"240"* on the upper side again is very high but his/her blood pressure is lower than that of the mass of the observations.

However, if we take away the outliers described above and omit all non diabetic patients as well as all patients that had less than three medical check-ups we will get a data matrix of 502 patients and nine medical parameters with 6% of the values missing.

After centering and scaling we again apply the *NIPALS* algorithm for missing values once more to obtain principal components of our new diabetes data set without outliers. The biplot of the first and second components is displayed in Figure 5.3. They explain 54% of the total variation of the new diabetes data set. The corresponding screeplot of the *NIPALS* algorithm for missing values is shown in Figure 5.4. In contrary to the first screeplot we note that in this case the importance of each component decreases with increasing index.

The first principal component of the new diabetes data set without outliers is expressed in the positive axis by high values of both *kidney* parameters followed by *triglyceride*, *blood pressure* and *body mass index*. The second
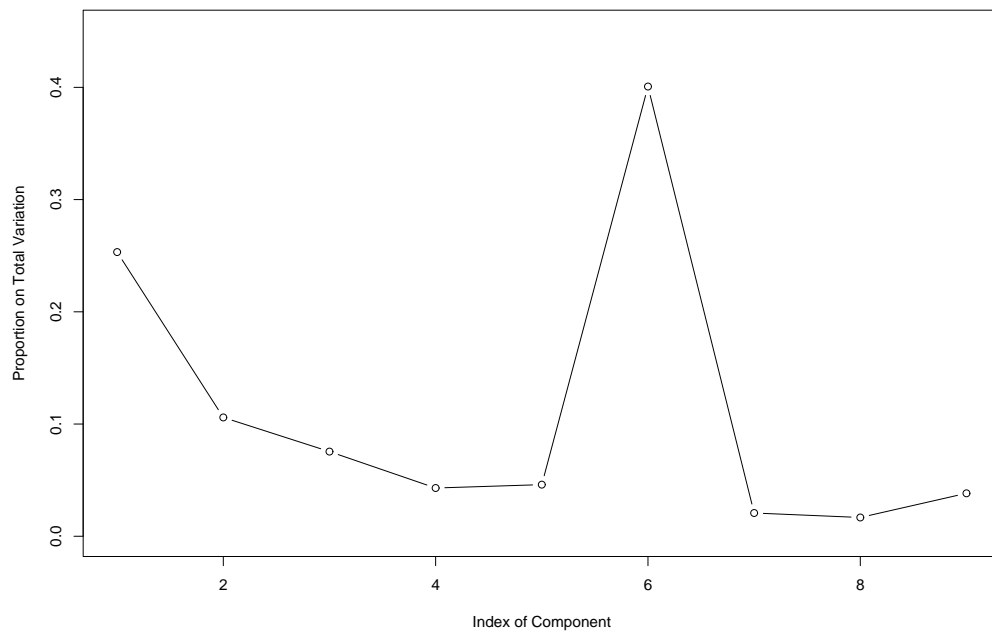
Figure 5.1: Biplot of the Diabetes Data Set

Figure 5.2: Screeplot of the Diabetes Data Set

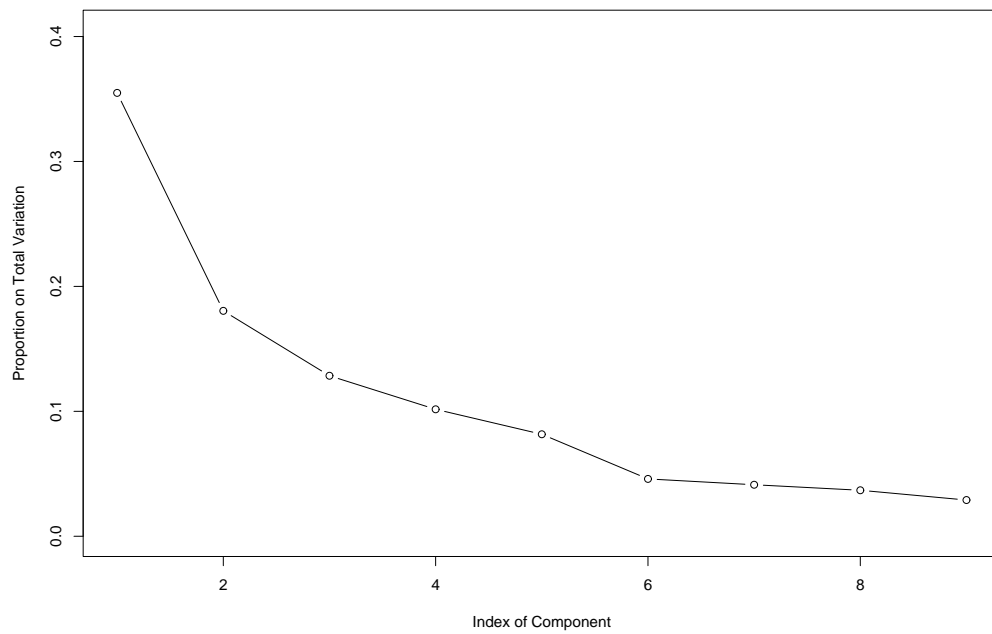Figure 5.3: Biplot of the Diabetes Data Set without Outliers

Figure 5.4: Screeplot of the Diabetes Data Set without Outliers

principal component has high values of *HbA1c* and *blood glucose* as main influence in its positive part. Opposed to it, in the negative part, again the *blood pressure* has strong influence along with *body mass index*. Details can be found in Table 5.1.

Table 5.1: The First Two Loadings $\boldsymbol{p}_{.1}$ and $\boldsymbol{p}_{.2}$ of the Diabetes Data Set

Variables	$\boldsymbol{p}_{.1}$	$\boldsymbol{p}_{.2}$
trans_HbA1c	0.22	0.56
Cholesterin	0.23	−0.12
Triglyceride	0.36	−0.08
BG	0.22	0.52
Mikroalbuminurie	0.39	0.19
Albuminexkr_rate	0.47	0.20
RR_syst	0.35	−0.37
RR_diast	0.32	−0.33
bmi	0.34	−0.28

Anyway, the variables *HbA1c* and *blood glucose* are highly positive correlated as well as the *systolic* and *diastolic pressure* and also both *kidney* parameters. Further either of the variables indicating the amount of glucose in the blood, i.e., *HbA1c* and *blood glucose*, are positively correlated with both *kidney* parameters. Moreover the variable *body mass index* and the *blood pressure* parameters are also positively correlated which indicates that a high *body mass index* value comes along with high *blood pressure*, i.e., overweight persons have high *blood pressure*. However, either *blood glucose* parameter (*HbA1c* and *blood glucose*) are nearly uncorrelated with the *blood pressure* parameters. This would mean that *blood pressure* does not influence the value of *HbA1c*. Furthermore the *blood pressure* parameters seem to be nearly uncorrelated with the *kidney* parameter.

Now we put together either of the three pairs of highly positively correlated variables to single parameters, i.e., we combine the variables *HbA1c* and *blood glucose* to a new *blood glucose* parameter by taking the average of the standardized variables and we do the same with both *kidney* and both *blood pressure* parameters (for details see Table A.5 in Section A.2). Hence, we get a data matrix of 502 patients and six medical parameters with 8% of the values missing.

Again we apply the *NIPALS* algorithm for missing values to our centered and scaled data. The biplot of the first and second components is shown in Figure 5.5. They explain 60% of the total variation of the data set. The corresponding screeplot is displayed in Figure 5.6.
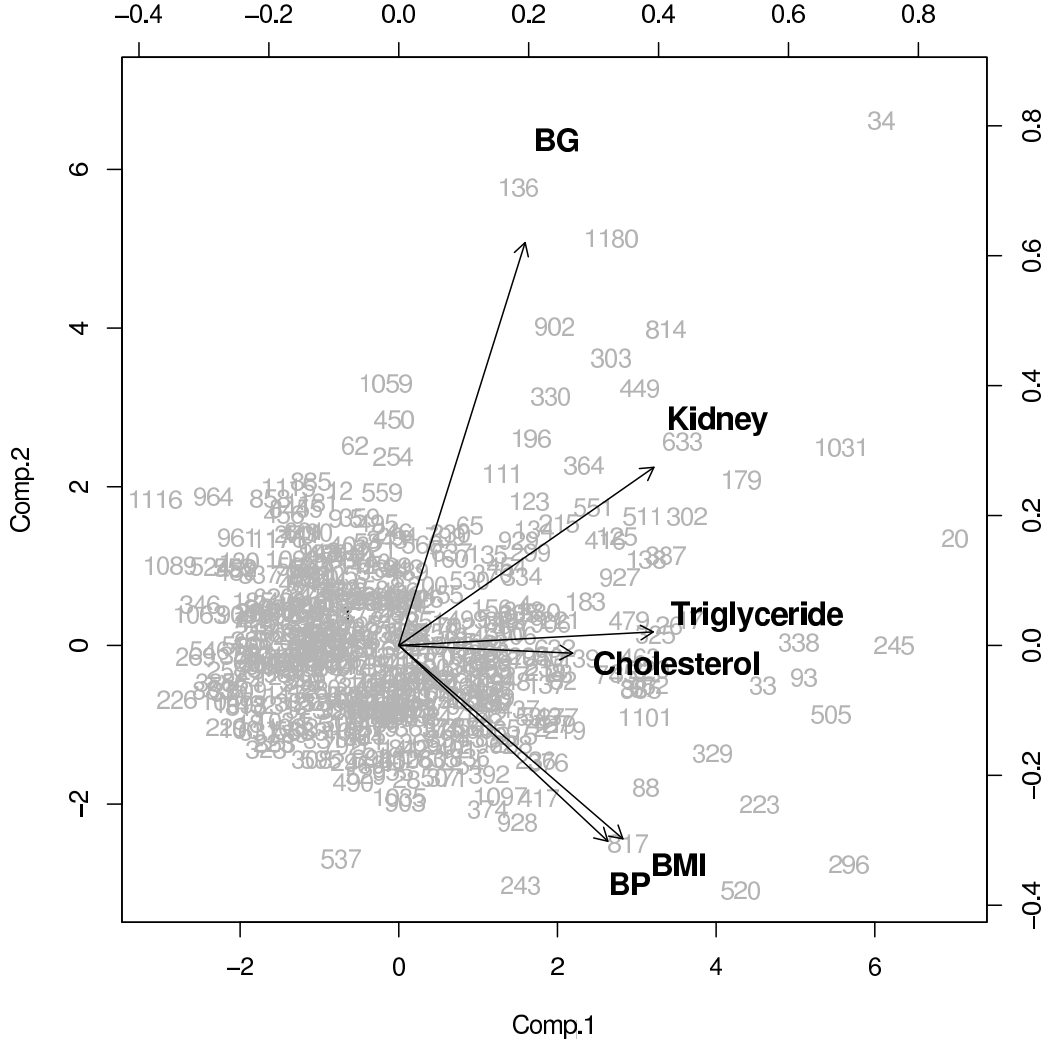
Figure 5.5: Biplot of the Reduced Diabetes Data Set without Outliers

The first principal component of the reduced diabetes data set is expressed in the positive axis by high values of the *triglyceride* and *kidney* parameters followed by a high *body mass index* and *blood pressure*. The second principal component has high values of *blood glucose* as main influence in its positive
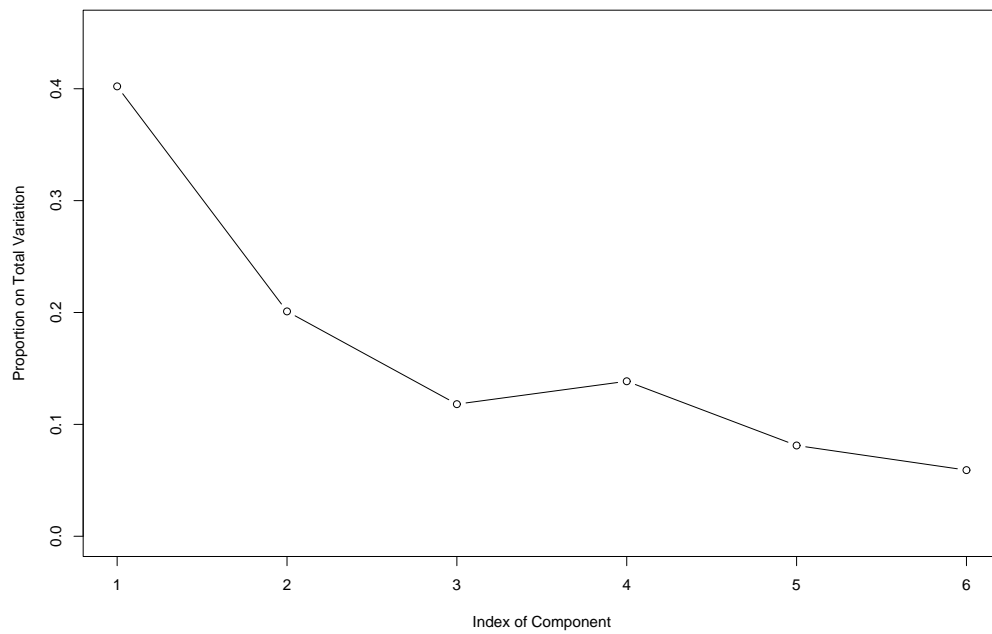
Figure 5.6: Screeplot of the Reduced Diabetes Data Set without Outliers

part. Opposed to it, in the negative part, again the variable *blood pressure* has strong influence along with the *body mass index*. For details confer Table 5.2.

Table 5.2: The First Two Loadings $\boldsymbol{p}_{.1}$ and $\boldsymbol{p}_{.2}$ of the Reduced Diabetes Data Set

Variables	$\boldsymbol{p}_{.1}$	$\boldsymbol{p}_{.2}$
BG	0.24	0.78
Cholesterol	0.33	−0.01
Triglyceride	0.49	0.03
Kidney	0.49	0.34
BP	0.40	−0.38
BMI	0.43	−0.37

As we see in Figure 5.5 the *body mass index* is again highly positively correlated with the *blood pressure*. As before the parameters of *blood glucose* and *kidney* are also positively correlated, and the *blood pressure* is again almost independent of the *blood glucose* level. This confirms our previous analysis.

5.2 Time Series Analysis

For time series analysis we use the 6-dimensional cross-sectional data $\boldsymbol{y}_{t\ell}$, $t = 1, \ldots, 30$ and $\ell = 1, \ldots, 19$, which are described in Section A.3 and the methods that we have developed in Section 4.3.

As mentioned before, the aim of this survey is to get a better understanding of how the values of different medical variables, like those of the parameter *HbA1c*, influence the severity of late complications, such as kidney failure. Therefore we are interested in using the remaining five variables, *HbA1c, Cholesterol, Triglyceride, Blood Pressure* and *BMI*, to explain some of the variation in the series of the parameter *Kidney*, denoted by $K_{t\ell}$.

In Figure 5.7 and Figure 5.8 the plots of the autocorrelation and partial autocorrelation functions for each variable of the cross-sectional diabetes data are displayed. Upper and lower dashed lines shown on the plots indicate

$\pm 1.96/\sqrt{570}$, so that upper values would be exceeded about 2.5% of the time if the series were white noise (cf. also Section 4.3.1).

Figure 5.9 shows the plots of the cross-correlation functions for the parameter *Kidney* versus all other variables.

Let us first consider the plot of the cross-correlation function (CCF) for the parameters *Kidney* and *HbA1c*. The CCF peaks at lag $h = 13$, showing that the *Kidney* series measured at time $t+13$ trimester is associated with the *HbA1c* series at time t. We could say the *HbA1c* series leads the *Kidney* series by 13 trimester. The sign of the CCF is positive, which means, increases in the *HbA1c* series at time t lead to increases in the *Kidney* series three to four years later.

The plots of the cross-correlation functions for the *Kidney* series versus the *Cholesterol* or *Triglyceride* series show no significant peaks, whereas the values of the cross-correlation functions for the parameter *Kidney* versus the variable *Blood Pressure* or *Body Mass Index* are positive with their maxima about lag $h = 0$.

In order to fit an appropriate ARX model to the *Kidney* series we again have a look at the plots of the autocorrelation and partial autocorrelation functions in Figure 5.7 and 5.8, respectively. We note that the autocorrelation function of the *Kidney* series tails off, whereas the partial autocorrelation function cuts off after lag 4. This leads to the conclusion that, according to Section 4.3.3, the *Kidney* series $K_{t\ell}$ follows an AR(4) process.

However, some preliminary fitting yields to the result that the inclusion of the *Kidney* parameter $K_{t-4,\ell}$ is not significant. Therefore we fit an AR(3) model to the centered data, that is,

$$K_{t\ell} = \alpha + \phi_1 K_{t-1,\ell} + \phi_2 K_{t-2,\ell} + \phi_3 K_{t-3,\ell} + w_{t\ell} \quad , \tag{5.1}$$

where $t = 4, \ldots, 30$ and $\ell = 1, \ldots, 19$.

The estimation was accomplished using the regression approach described in Section 4.3.3. In this case, the fitted model is

$$\begin{aligned}
\widehat{K}_{t\ell} &= 0.156 + 0.337_{(0.043)} K_{t-1,\ell} + 0.220_{(0.044)} K_{t-2,\ell} \\
&\quad + 0.174_{(0.043)} K_{t-3,\ell} + \widehat{w}_{t\ell} \quad ,
\end{aligned} \tag{5.2}$$

where $\widehat{\sigma}_w^2 = 0.094$. Each coefficient is highly significant, as seen from the estimated standard errors given in parentheses below each parameter estimate. A Q-Q plot, displayed in Figure 5.10, of the residuals of the fit $\widehat{w}_{t\ell}$ however,
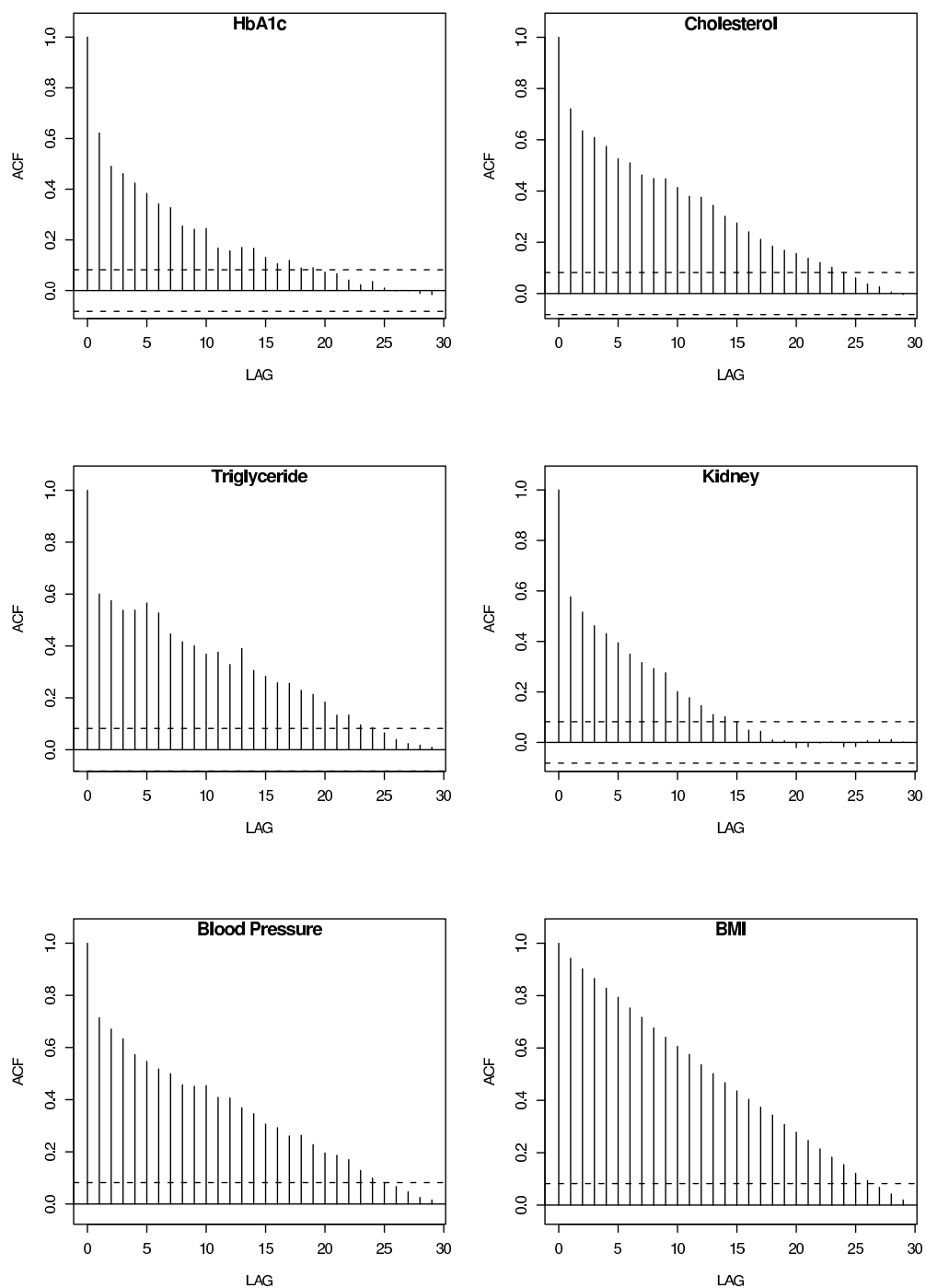
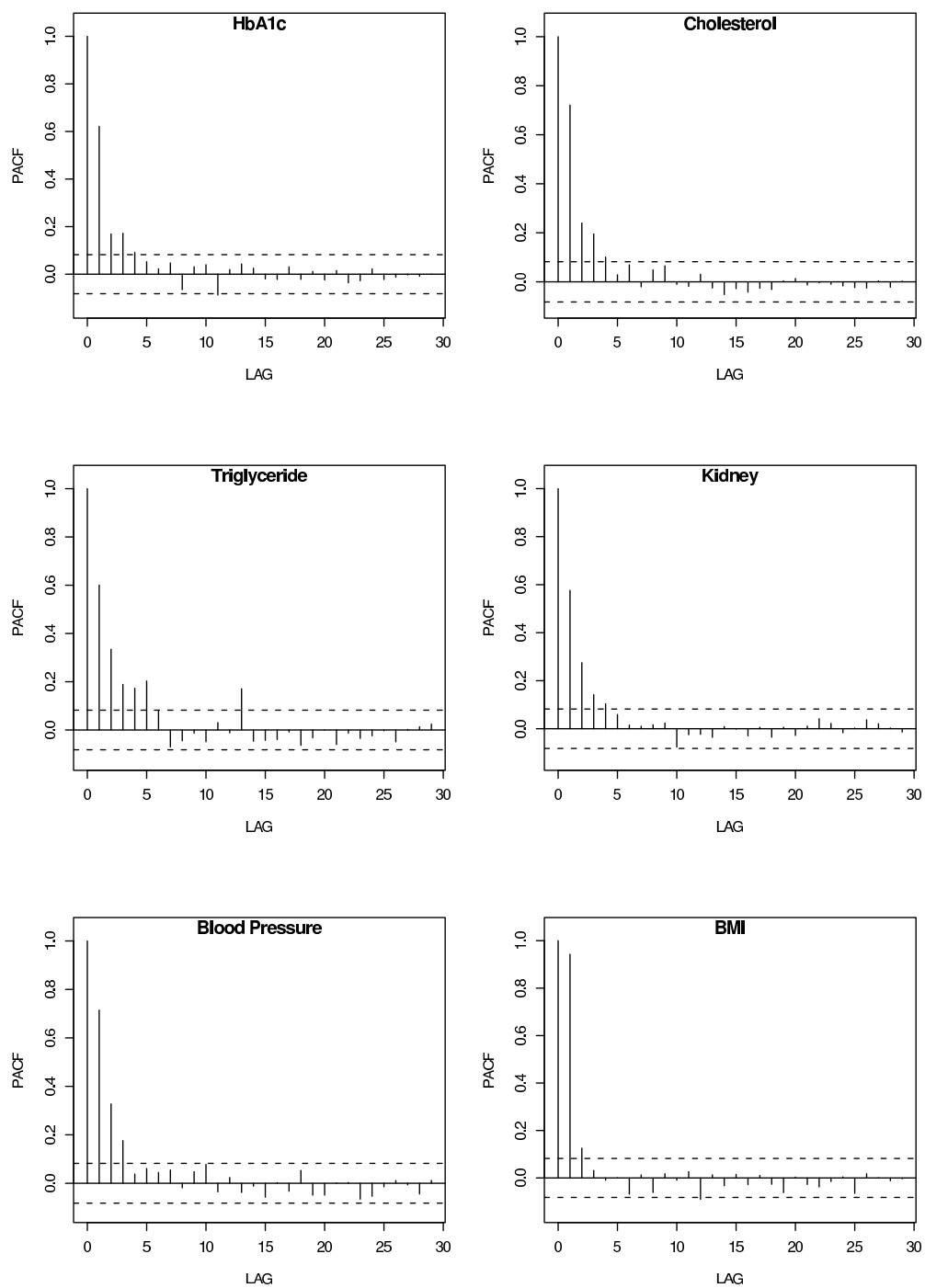Figure 5.7: Autocorrelation Function of the Diabetes Series

Figure 5.8: Partial Autocorrelation Function of the Diabetes Series
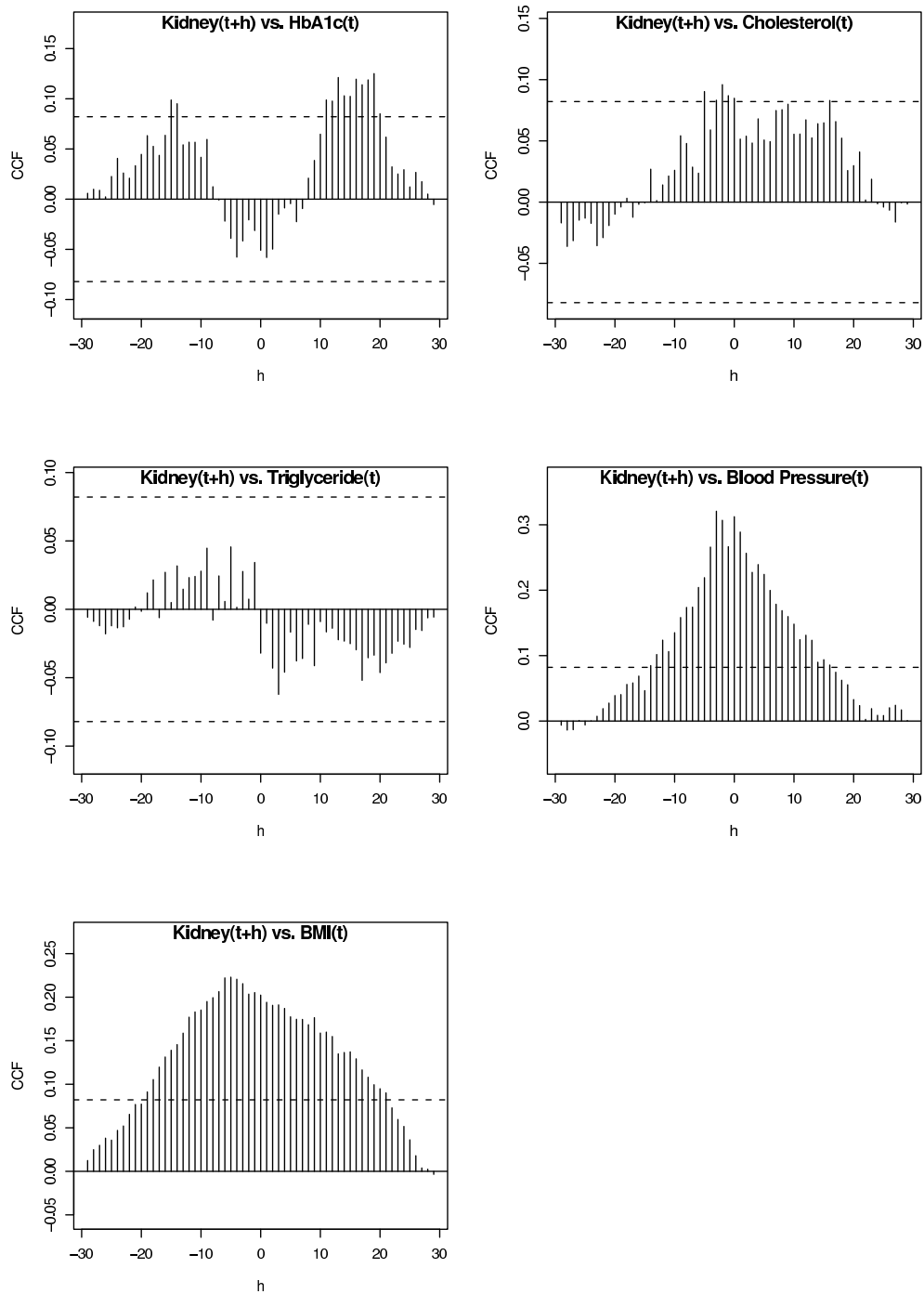
Figure 5.9: CCF between the Variable *Kidney* and all other Variables (Positive Lag Means the Respective Variable Leads *Kidney*)
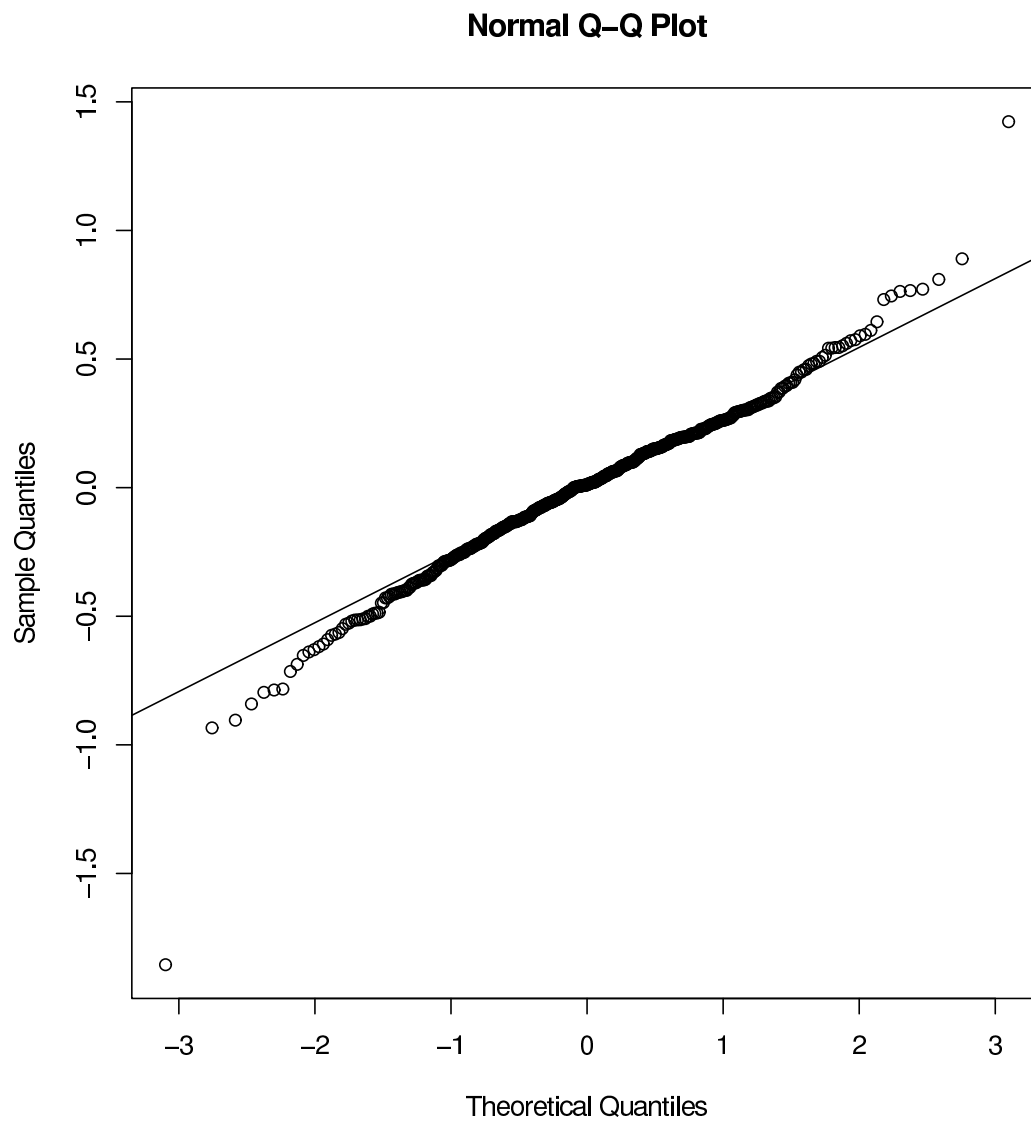
Normal Q–Q Plot

Figure 5.10: QQ-Plot of the Residuals $\widehat{w}_{t\ell}$ of the AR Model

62

shows a slight departure from the standard normal distribution, especially at the tails.

Hence, we are interested in using the remaining five variables, *HbA1c, Cholesterol, Triglyceride, Blood Pressure* and *BMI*, to explain some of the variation in the *Kidney* series $K_{t\ell}$. Therefore, using the residuals of the fit $\widehat{w}_{t\ell}$, we calculate the cross-correlation functions between the prewhitened *Kidney* series and the other variables shown in Figure 5.11.

We note that, unfortunately, there are no significant values any more in Figure 5.11. Nevertheless for our further analysis we take the largest values of the different cross-correlation functions into account.

Considering the cross-correlation function (CCF) of the prewhitened *Kidney* series and the *HbA1c* series (positive lag means *HbA1c* leads *Kidney*) positive correlations are seen at lags $h = 11$ and $h = 21$. Moreover for the CCF of prewhitened *Kidney* and *Cholesterol* we see correlations at lags $h = 0$ and $h = 2$, for the CCF of prewhitened *Kidney* and *Blood Pressure* at lag $h = 2$ and for the CCF of prewhitened *Kidney* and *Body Mass Index* at lag $h = 7$.

After some preliminary fitting, the final model uses the exogenous variables $\boldsymbol{u}_{t\ell} = (1, H_{t-11,\ell})^\top$, where $H_{t\ell}$ denotes the *HbA1c* series, along with an AR(3) model on the *Kidney* series $K_{t\ell}$. The inclusion of the particular *HbA1c* at lag 21 and of the other variables at different lags, mentioned above, yields to non significant parameter estimates. In this case the ARX model is

$$K_{t\ell} = \Gamma\boldsymbol{u}_{t\ell} + \phi_1 K_{t-1,\ell} + \phi_2 K_{t-2,\ell} + \phi_3 K_{t-3,\ell} + w_{t\ell} \quad, \qquad (5.3)$$

where $t = 12, \ldots, 30$, $\ell = 1, \ldots, 19$, $\Gamma = (\alpha, \beta)$ and $\boldsymbol{u}_{t\ell} = (1, H_{t-11,\ell})^\top$.

Again the estimation was accomplished using the regression approach described in Section 4.3.3. In this case the fitted model is

$$
\begin{aligned}
\widehat{K}_{t\ell} &= -0.200 + 0.057_{(0.025)} H_{t-11,\ell} + 0.348_{(0.051)} K_{t-1,\ell} \\
&\quad + 0.179_{(0.051)} K_{t-2,\ell} + 0.161_{(0.049)} K_{t-3,\ell} + \widehat{w}_{t\ell} \quad, \qquad (5.4)
\end{aligned}
$$

where $\widehat{\sigma}_w^2 = 0.098$. Each coefficient is significant, as seen from the estimated standard errors listed below each parameter estimate. Finally, an analysis of the residuals $\widehat{w}_{t\ell}$ shows, except for a few outliers, that the model fits well. In addition a Q-Q plot, displayed in Figure 5.12, shows no departure from the Gaussian assumption, except for the few outliers.

Our general conclusion is that the value of the *Kidney* parameter depends on those of the last year and an increase of the *Kidney* parameter is associated with higher *HbA1c* values about four years ago.
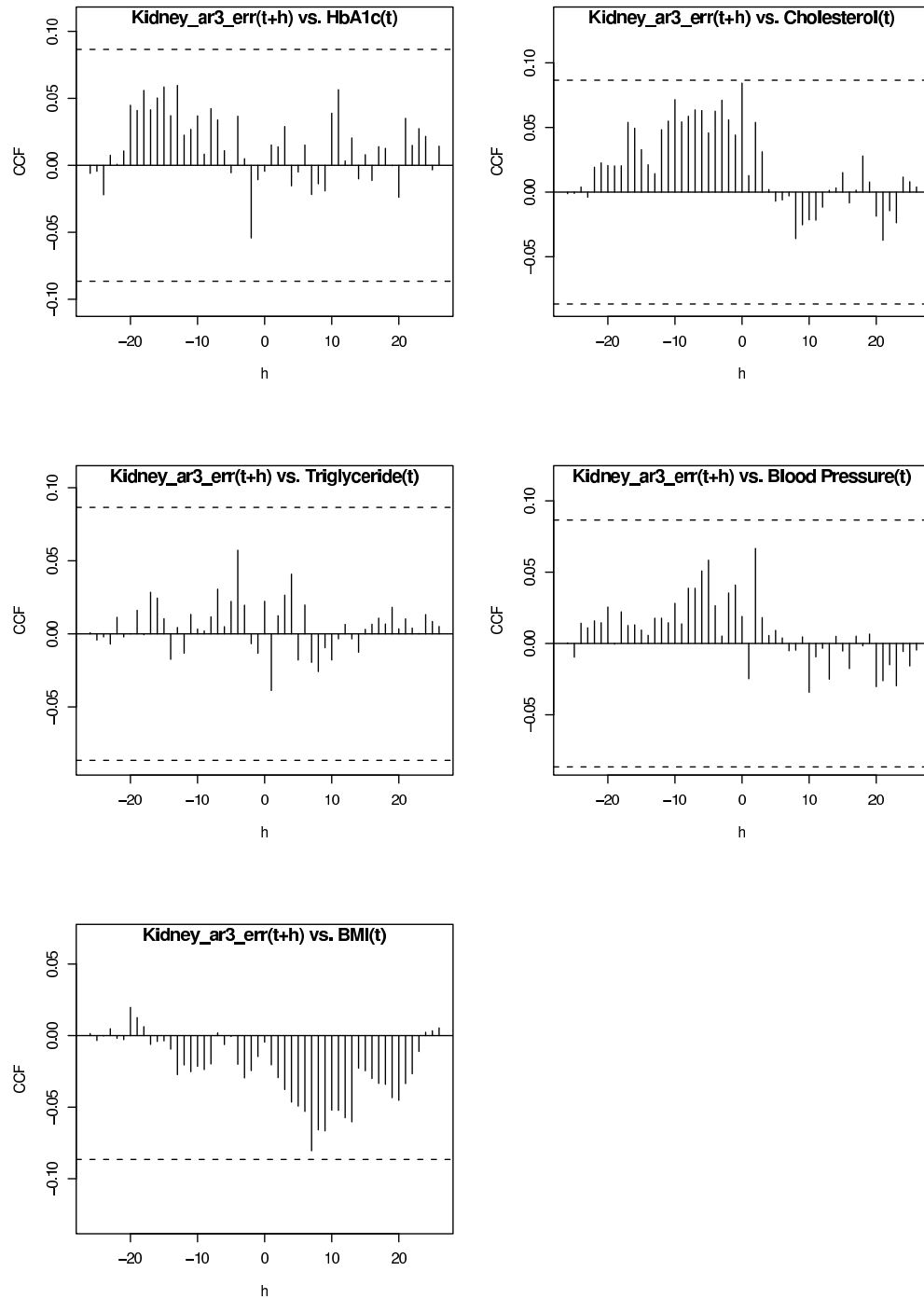
Figure 5.11: CCF between the Prewhitened Variable *Kidney* and all other Variables (Positive Lag Means the Respective Variable Leads *Kidney*)
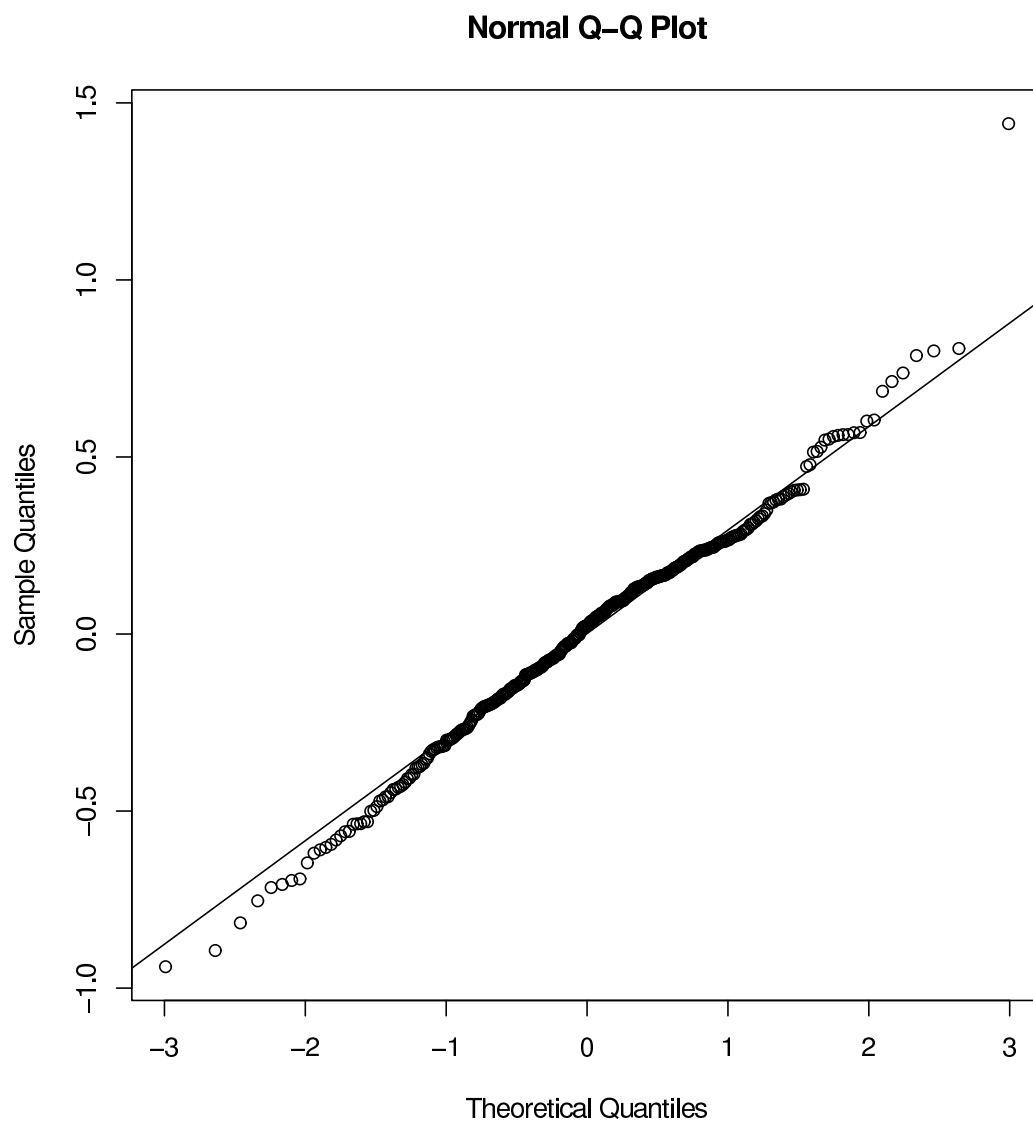
Normal Q–Q Plot

Figure 5.12: QQ-Plot of the Residuals $\widehat{w}_{t\ell}$ of the ARX Model

Chapter 6

Conclusions and Summary

In this work we presented two ways of analyzing our diabetes data which we found the most suitable: first, the principal component analysis (PCA) and, second, the time series analysis (TSA).

Our data consisted of many variables which could not be overlooked easily. Therefore the first idea was to extract some components which concentrate on the most important information of all these variables. This led to the usage of principal component analysis. On the other hand the patients' data had been collected during medical check-ups over a time period of more than 12 years so that time series analysis was the second idea. Additionally we had to deal with a lot of missing values which further complicated the analysis.

For principal component analysis we replaced the values of the medical variables which were obtained during that time for each patient by one weighted average value. In order to get principal components and not to loose too much information of the data we used the *NIPALS* algorithm for missing values. Although we neglected the effect of time in this case, we were able to detect groups of non diabetic patients and outliers.

Moreover, applying the *NIPALS* algorithm for missing values to our data, we recognized three groups with positively correlated variables: the first group contained the variables *HbA1c* and *Blood Glucose*, the second existed of both parameters that determine the severity of kidney disease, and in the third group we found the variables *Blood Pressure* and *Body Mass Index* as well as *Cholesterol* and *Triglyceride* (two different types of blood fat).

The variables of the first and second group were positively correlated whereas the variables of the third group seemed to be nearly independent of the variables of the first and second group.

Anyway, we may be surprised about the highly positive correlation of *HbA1c* and *Blood Glucose* because the value of *HbA1c* represents an average blood glucose level over the last few weeks whereas the value of *Blood Glucose* measures the actual amount of sugar in the blood. We note, however, that both values, *HbA1c* and *Blood Glucose*, which were used for the principal component analysis are weighted averages.

For time series analysis we tried to fit an autoregressive model with exogenous variables to the cross-sectional data which we obtained by extracting proper time series from our diabetes data. Hence, we modeled the time series of the *Kidney* parameter using other variables, e.g., *HbA1c*, to explain some of the variation in the *Kidney* series.

The final model used the *HbA1c* value which had been measured 11 trimester ago along with an autoregressive model of order three on the *Kidney* series. Our general conclusion is that the value of the *Kidney* parameter depends on those of the last year and an increase of the *Kidney* parameter is associated with higher *HbA1c* values about four years ago.

Moreover, in order to perform a better fit of the model, it would also be helpful to extend the model with further exogenous variables, like gender, but this would exceed the purpose of this work.

Anyway, we mainly focused on the modeling of the *Kidney* series but it would also be interesting to study other late complications like retinopathy.

Generally we think that handling the diabetes data as cross-sectional data was the most suitable way and led to proper results. However, we may think about another autocovariance function for cross-sectional data by replacement of the "global" mean, $\bar{y}_i$, in (4.20) by the mean of each patient, $\bar{y}_{i\ell}$, $\ell = 1, \ldots, N$. The procedure would be analogous for the cross-covariance function. This might lead to different models.

As presented in this survey much has been done but still much more analysis could be done on this diabetes data set.

Appendix A

Data

In this section we present the different data sets used in this work.

A.1 The euro86 Data Set

The *euro86* data set is collected from statistical year books and contains demographic data of the European countries around 1986.

For the 25 European countries (the observations) 9 variables were collected. In detail they are the average growth of population from 1986 to 2000, the percentage of women in the age to give birth (1985), number of women per 100 men (1985), life expectation of women (1986), life expectation of men (1986), infant mortality (1986), inhabitants per doctor (1981), daily provided calories per person (1985) and percentage of infants born underweighted (1984). The variable names used in the various plots are reprinted in Table A.1. A list of the abbreviations of the European countries (observations) is given in Table A.2.

One of the European countries, namely Albania (al), can easily be detected as an outlier. Compared with the median of the European values it has a 9 times higher population growth, a 4 times higher infant mortality, further, a 4.5 times higher number of inhabitants per doctor and 20 percent less calories per person provided daily.

Table A.1: Variables of *euro86* Data Set

1	pop_growth	4	lifeexp_f	7	inhab/doc
2	give_birth	5	lifeexp_m	8	calorie
3	women%	6	inf_mort	9	baby_underw

Table A.2: Observations of *euro86* Data Set

a	Austria	h	Hungary
al	Albania	i	Italy
b	Belgium	irl	Ireland
bg	Bulgaria	n	Norway
ch	Switzerland	nl	The Netherlands
cs	Czechoslovakia	p	Portugal
d	Western Germany	pl	Poland
ddr	Eastern Germany	ro	Romania
dk	Denmark	s	Sweden
e	Spain	sf	Finland
f	France	su	Soviet Union
gb	Great Britain	yu	Yugoslavia
gr	Greece		

A.2 The Diabetes Data Matrix for Multivariate Analysis

For multivariate analysis the diabetes data set is extracted from the medical data base provided by Univ.-Prof. Dr. Kinga Howorka and modified as described in Chapter 2.

For each patient we take nine variables for further analysis. *HbA1c* is the abbreviation for a specific particle of the blood, namely the hemoglobin *A1c* or glycosylated hemoglobin. The value of *HbA1c*–given as a percentage–is the amount of glucose that sticks to the red blood cell, which is proportional to the amount of glucose in the blood. It measures a person's average blood glucose level over the past few weeks. The value of *HbA1c* (*trans_HbA1c*) which we use during our survey is the *restandardized* one as described in

Section 2.3.

We also take the values of cholesterol (*Cholesterin*) and triglyceride (*Triglyceride*). Cholesterol is a type of fat produced by the liver and found in the blood whereas triglyceride is the storage form of fat in the body. High triglyceride levels may occur when diabetes is out of control.

The blood glucose level (BG) is the amount of glucose in a certain amount of blood. The latter three parameters are noted in milligrams per deciliter, or *mg/dl*.

Further we take two different kidney values into account both measuring the amount of the protein Albumin in the urine. The first value, *Mikroalbuminurie*, is a concentration, measured in *mg/l*, whereas the second parameter, *Albuminexkr_rate*, is a rate, given in *ng/min*. Generally small amounts of Albumin in the urine are called *microalbumin*. The condition in which the urine has more than normal amounts of Albumin is called *albuminuria*. Albuminuria may be a sign of kidney disease.

The blood pressure, both, the systolic pressure (RR_syst) when the heart pushes blood out into the arteries and the diastolic pressure (RR_diast) when the heart rests, are measured in *mmHg*.

The last variable is the *body mass index* (*bmi*) which is calculated by the following formula:

$$
\begin{aligned}
\text{bmi} \quad &:= \quad \frac{\text{weight in } pounds \times 703}{(\text{height in } in)^2} \\
&= \quad \frac{\text{weight in } kg \times 10003}{(\text{height in } cm)^2} \approx \frac{\text{weight in } kg}{(\text{height in } m)^2} \ ,
\end{aligned}
\qquad (A.1)
$$

where *weight* is the patient's average weight which is delivered by the AUC algorithm as described in Section 2.4 and *height* is the patient's height. The reference ranges of the body mass index are given in Table A.3.

Table A.3: Reference Ranges of the Body Mass Index

bmi $\leq$ 18.4	underweight
18.5 $\leq$ bmi $\leq$ 24.9	normal
25.0 $\leq$ bmi $\leq$ 29.9	overweight
30.0 $\leq$ bmi	obese

If one or both parameters are missing the value of the variable *bmi* will be put to *NA* (*Not Available*).

The variable names used in various plots are reprinted in Table A.4.

Table A.4: Variables of the Diabetes Data Set

1	trans_HbA1c	4	BG	7	RR_syst
2	Cholesterin	5	Mikroalbuminurie	8	RR_diast
3	Triglyceride	6	Albuminexkr_rate	9	bmi

For further analysis we reduce the number of variables. Hence, we combine the three pairs of highly correlated variables, namely *HbA1c* and *blood glucose*, both *kidney* variables as well as both *blood pressure* parameters, to three new variables by taking the average of the standardized variables. The new variable names of the reduced diabetes data set used in the biplot are listed in Table A.5.

Table A.5: Variables of the Reduced Diabetes Data Set

	new variable	old variables
1	BG	trans_HbA1c *and* BG
2	Cholesterol	Cholesterin
3	Triglyceride	Triglyceride
4	Kidney	Mikroalbuminurie *and* Albuminexkr_rate
5	BP	RR_syst *and* RR_diast
6	BMI	bmi

A.3 Cross-sectional Diabetes Data for Time Series Analysis

For time series analysis we extract proper time series from the medical diabetes data base and modify them as described in Sections 4.1 and 4.2.

Hence, we get 6-dimensional cross-sectional data $\boldsymbol{y}_{t\ell}$, with $t = 1, \ldots, 30$ and $\ell = 1, \ldots, 19$, which are stored in a 570×6 matrix, one patient after the

other. The first 30 rows are the measurements of the first patient according to the order they are obtained in time, and so on.

These 19 patients have the following in common:

- All of them are diabetic patients; non diabetic patients have been excluded.

- Before applying the method described in Section 4.2 all time series of the selected medical variables have less than ten missing values.

- We only choose persons for which measurements are available at least during ten years. In order to obtain time series which have the same number of measurements we only take the first ten years into account (still, there are only few patients where we have observations for more than ten years). Moreover we note that we have divided the year into trimester; therefore the time parameter t ranges from 1 to 30.

The six medical variables we have selected are given in Table A.6.

Table A.6: Variables of the Cross-sectional Diabetes Data Set

1	HbA1c	4	Kidney
2	Cholesterol	5	Blood Pressure
3	Triglyceride	6	BMI

The value of the first variable, *HbA1c*, is the one that is delivered by the restandardization as described in Section 2.3.

The variables *Cholesterol* and *Triglyceride* are equivalent to those used before.

The values of variable 4, *Kidney*, corresponds with the values of variable *Mikroalbuminurie* in Table A.4. We have transformed the values at each time t, $t = 1, \ldots, 30$, using the function *log(log(.))* because of their skewness.

The value of the variable *Blood Pressure* is the average of the systolic and diastolic pressure at each time t.

The value of the last variable, *Body Mass Index (BMI)*, is calculated according to (A.1) taking the patient's weight at each time t. Unfortunately the height of the female patient "189" is missing. In order to calculate the

Body Mass Index we have estimated her height taking the average of all female patients listed in the medical data base.

The data of all 19 patients used for time series analysis are plotted in Figure A.1 to A.6. The horizontal time axis represents 10 years.
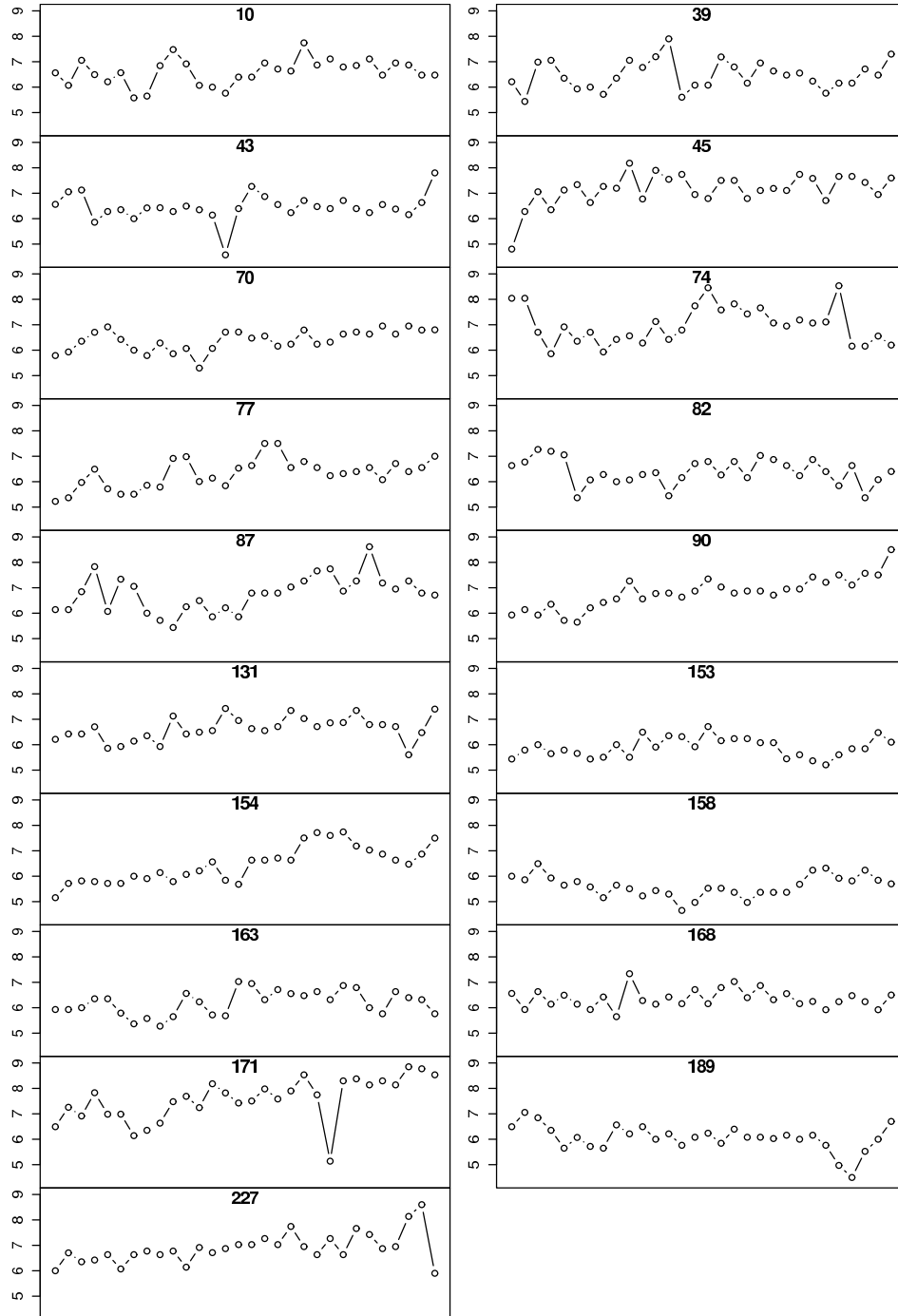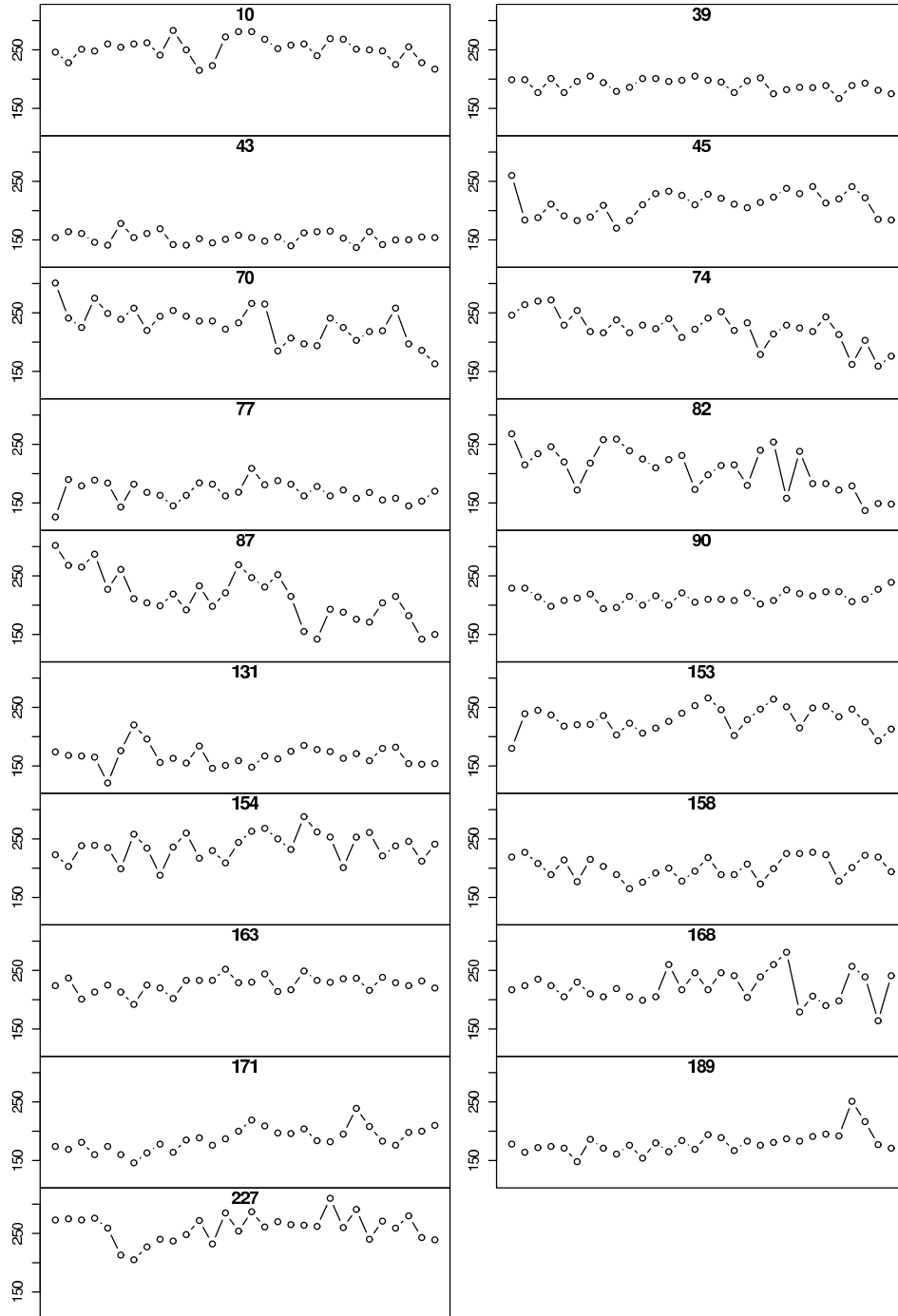
Figure A.1: Time Series of the Variable *HbA1c*

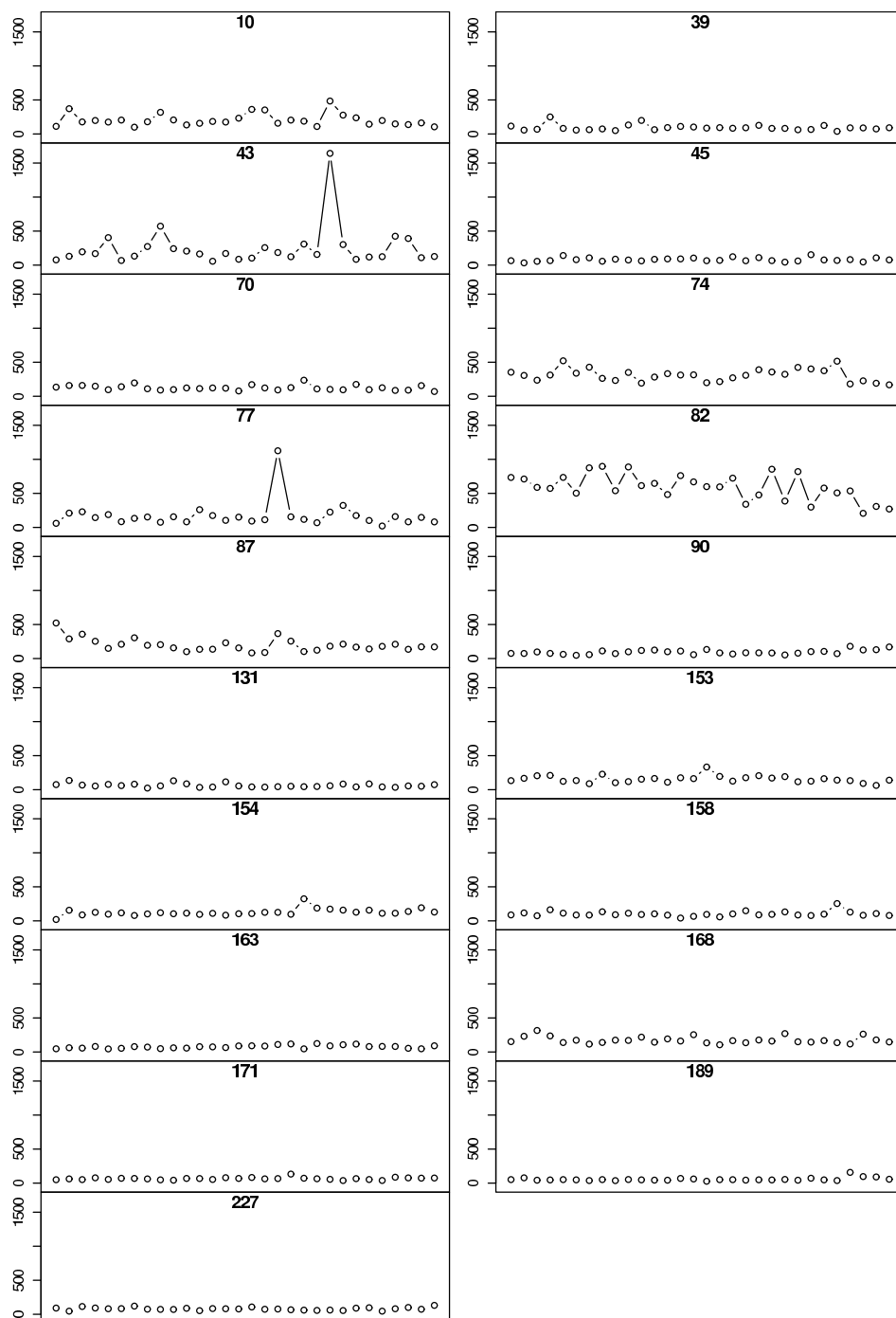Figure A.2: Time Series of the Variable *Cholesterol*

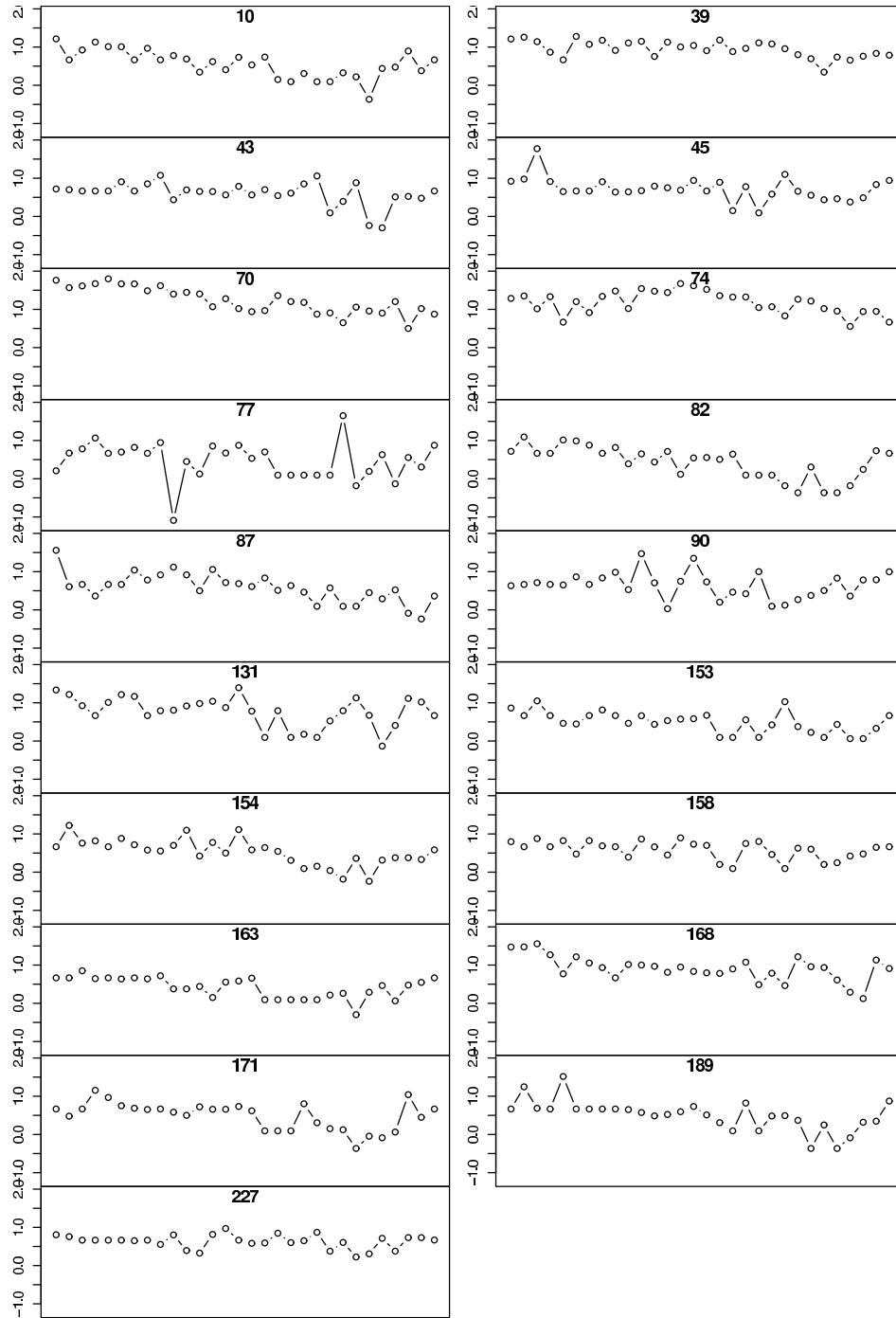Figure A.3: Time Series of the Variable *Triglyceride*

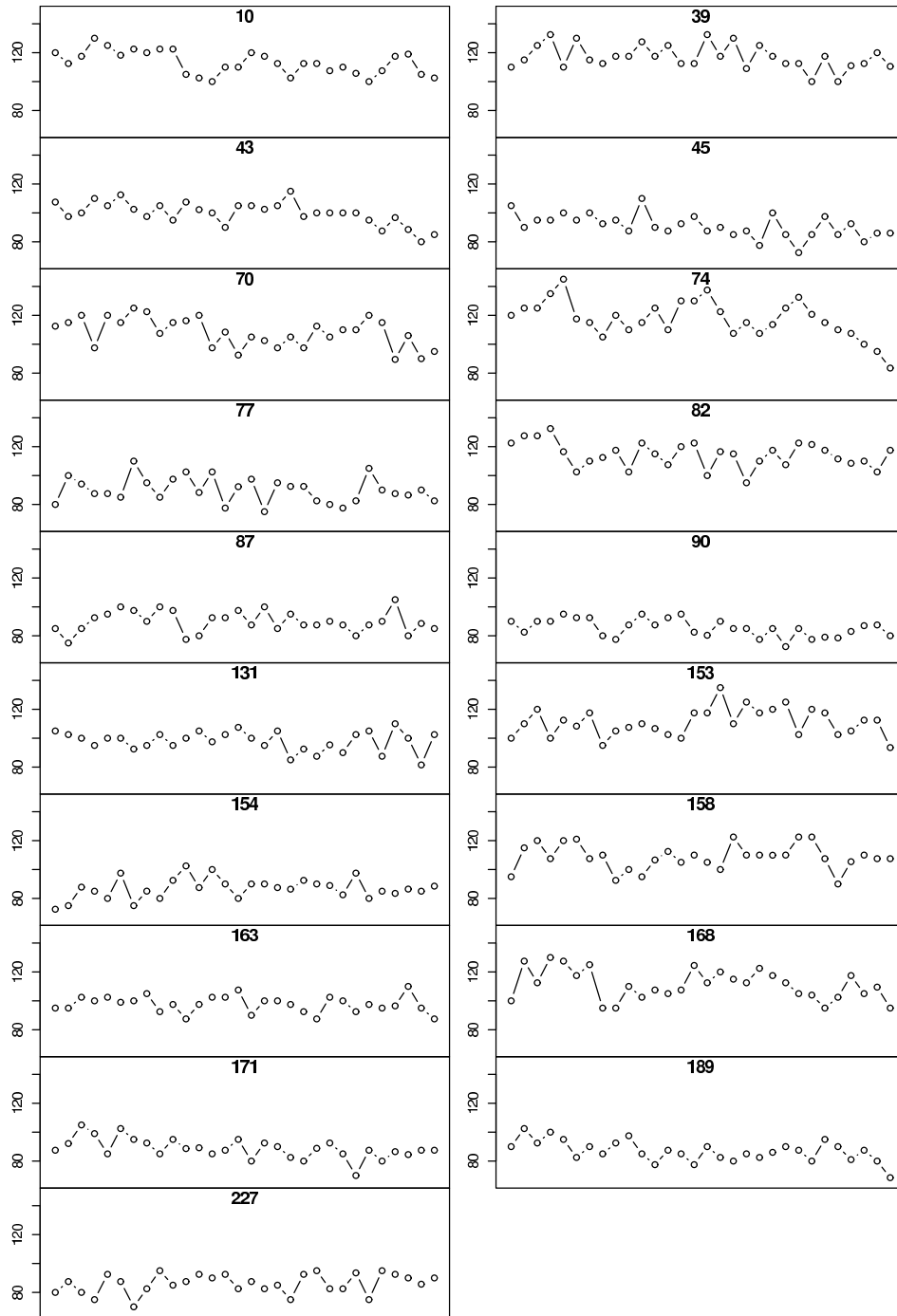Figure A.4: Time Series of the Variable *Kidney*

Figure A.5: Time Series of the Variable *Blood Pressure*

78
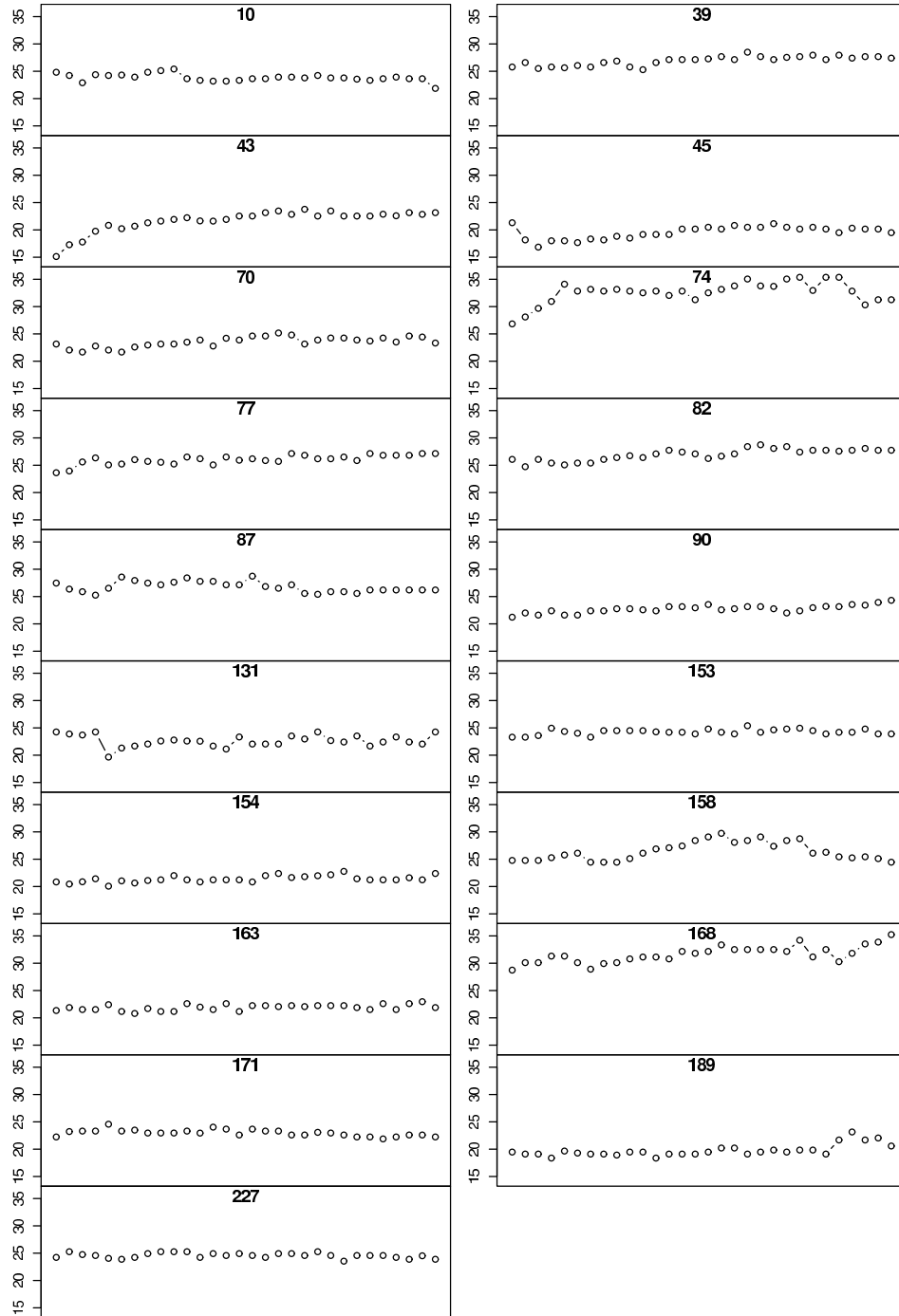
Figure A.6: Time Series of the Variable *Body Mass Index*

Appendix B

Listing of the Algorithms

In this section we give a listing of the algorithms which have been implemented in ℝ and used in this work.

B.1 The *NIPALS* Algorithm

```
f.nipals_function (X,k,it=10,tol=0.0001)
# function "f.nipals" calculates the principal components of a given
# data matrix X according to the NIPALS algorithm (cf. Wold, 1966).
# X...data matrix,
# k...number of components,
# it...maximal number of iterations per component,
# tol...(squared) precision tolerance for calculation of components
# date: 2002-05-11
{
cat("data matrix X: ",X,"\n")
# mean-centering of data matrix X
Xh <- scale(X,center=TRUE,scale=FALSE)
cat("Xh: ",Xh,"\n")
nr <- 0
T <- NULL
P <- NULL
for (h in 1:k){
  cat("h = ",h,"\n")
  th <- Xh[,1]  # starting value for th is 1st column of Xh
    ende <- FALSE
    # 3 inner steps of NIPALS algorithm
```

```
  while (!ende){
    nr <- nr+1
    # LS regression for ph
    ph <- t((t(th)%*%Xh) * as.vector(1/(t(th)%*%th)))
    # normalization of ph
    ph <- ph * as.vector(1/sqrt(t(ph)%*%ph))
    thnew <- t(t(ph)%*%t(Xh))  # LS regression for th
    prec <- t(th-thnew)%*%(th-thnew)  # calculate precision
    cat("actual precision: ",sqrt(prec),"\n")
    th <- thnew  # refresh th in any case
    cat("th: ",th,"\n")
    # check convergence of th
    if (prec <= (tol)) {
      ende <- TRUE
    }
    else if (it <= nr) {  # too many iterations
      ende <- TRUE
      cat("\nWARNING! Iteration stop in h=",h," without
          convergence!\n\n")
    }
  }
  Xh <- Xh-(th%*%t(ph))  # calculate new Xh
  T <- cbind(T,th)  # build matrix T
  cat("T: ",T,"\n")
  P <- cbind(P,ph)  # build matrix P
  cat("P: ",P,"\n")
  nr <- 0
}
return(T,P)
}
```

B.2   The *NIPALS* Algorithm for Missing Values

```
f.nipna_function (X,k,it=10,tol=0.0001,name="result")
# function "f.nipna" calculates the principal components of a
# given data matrix X according to the NIPALS algorithm for
# missing values (NA) (cf. Tenenhaus, 1998).
# X...data matrix,
```

```
# k...number of components,
# it...maximal number of iterations per component,
# tol...(squared) precision tolerance for calculation of components
# name...name of the text file which contains all comments
# date: 2002-07-14
{
sink(paste(name,".txt",sep=""))
cat("data matrix X: ",X,"\n\n")
cols <- ncol(X)
rows <- nrow(X)
# mean-centering of data matrix X
Xh <- scale(X,center=TRUE,scale=FALSE)
cat("Xh: ",Xh,"\n")
cat("number of columns: ",cols,"\n")
cat("number of rows: ",rows,"\n\n")
T <- NULL
P <- NULL
for (h in 1:k){
  nr <- 0
  cat("h = ",h,"\n")
  th <- Xh[,1]    # starting value for th is 1st column of Xh
  ende <- FALSE
  #3 inner steps of NIPALS algorithm
  while (!ende){
    nr <- nr+1
    ph <- rep(0,cols)
    hnorm <- 0
    prec <- 0
    #LS regression of ph
    for (l in 1:cols){
      denom1 <- 0
      flag <- FALSE
      for (m in 1:rows){
        if (!is.na(Xh[m,l]) & !is.na(th[m])){
          ph[l] <- ph[l]+th[m]*Xh[m,l]
          denom1 <- denom1+th[m]^2
          flag <- TRUE
        }
      }
      if (flag){
```

```
    cat("Entered loop to calculate LS regression of ph: \n")
    cat(l,"-th coordinate: denom1 = ",denom1,"\n\n")
    ph[l] <- ph[l]/denom1
  }
  else{
    ph[l] <- NA
  }
}
if (is.na(t(ph)%*%ph)){
  cat("WARNING! There exists at least one element of ph=",
     ph,"which is not available!\n\n")
}
#normalization of ph
for (q in 1:cols){
  if (!is.na(ph[q])){
    hnorm <- hnorm+ph[q]^2
  }
}
ph <- ph/sqrt(hnorm)
#LS regression of thnew
thnew <- rep(0,rows)
for (i in 1:rows){
  denom2 <- 0
  flag <- FALSE
  for (j in 1:cols){
    if (!is.na(Xh[i,j]) & !is.na(ph[j])){
      thnew[i] <- thnew[i]+ph[j]*Xh[i,j]
      denom2 <- denom2 + ph[j]^2
      flag <- TRUE
    }
  }

  if (flag){
    cat("Entered loop to calculate LS regression of thnew: \n")
    cat(i,"-th coordinate: denom2 = ",denom2,"\n\n")
    thnew[i] <- thnew[i]/denom2
  }
  else{
    thnew[i] <- NA
  }
```

```
    }
    if (is.na(t(thnew)%*%thnew)){
      cat("WARNING! There exists at least one element of thnew=",
          thnew,"which is not available!\n\n")
    }
    #calculate precision
    for (r in 1:rows){
      if (!is.na((th-thnew)[r])){
        prec <- prec+(th-thnew)[r]^2
      }
    }
    cat("actual precision: ",sqrt(prec),"\n")
    th <- thnew  #refresh th in any case
    cat("th: ",th,"\n")
    #check convergence of th
    if (prec <= (tol)) {
      ende <- TRUE
    }
    else if (it <= nr) {  #too many iterations
      ende <- TRUE
      cat("\nWARNING! Iteration stop in h=",h," without
          convergence!\n\n")
    }
  }
  Xh <- Xh-(th%*%t(ph))  #calculate new Xh
  T <- cbind(T,th)  #build matrix T
  cat("T: ",T,"\n")
  P <- cbind(P,ph)  #build matrix P
  cat("P: ",P,"\n")
  nr <- 0
}
dimnames(T)[[2]]_c(1:(dim(T)[2]))
dimnames(P)[[2]]_c(1:(dim(P)[2]))
scores <- T
loadings <- P
sink()
return(scores,loadings)
}
```

B.3 Simulations

In this section, as examples, we give the listings of the two algorithms which are used to calculate the errors $e_{\max}^{(PCA)}$, $e_{\max}^{(NIPALS)}$ and $e_{\max}^{(NIPNA)}$, along with their relative errors, $e_{\text{rel, max}}^{(1)}$ and $e_{\text{rel, max}}^{(2)}$. The results of the simulations are described in detail in Section 3.2.

```
f.sim1_function (anz,num,k,s,flag=FALSE)
# function "f.sim1" creates "anz" matrices X of a multivariate
# normal distribution specified by the correlation matrix "s"
# with dimension n by number of columns of "s".  For each X the
# principal components, especially a matrix of the scores and one
# of the loadings, are calculated by the functions "princomp(X)"
# and "f.nipals(X,k)" with "k" components. In order to compare
# the results a relative difference between the norms of the
# error matrices is calculated.
# anz...number of simulations
# num...number of observations of the data matrix X
# k...number of components that are calculated within the NIPALS
#     procedure
# s...correlation matrix of the multivariate normal distribution
# flag...logical variable which determine wether the parameter in
#         "mvrnorm(...)" is TRUE or FALSE
# date: 2002-08-17
{
begin <- date()
library(mass)
library(mva)
X <- NULL
X.pca <- NULL
X.nipals <- NULL
err.pca <- NULL
err.nipals <- NULL
a <- 0
b <- 0
res.pca <- NULL
res.nipals <- NULL
res.rel <- NULL
mittel <- 0
cols <- ncol(s)     # number of columns of s
```

```
if (k<cols){
  for (i in 1:anz){  #starting the simulations
    # creating a centered multivariate normal distributed
    # data matrix X with correlation matrix s
    X <- scale(mvrnorm(n=num, rep(0, cols), s, empirical=flag),
      center=TRUE, scale=FALSE)
    X.pca <- princomp(X)     # PCA of X
    X.nipals <- f.nipals(X,k)    # NIPALS of X
    # error matrix of PCA:
    err.pca <- X - X.pca$scores[,1:k]%*%t(X.pca$loadings[,1:k])
    # error matrix of NIPALS:
    err.nipals <- X - X.nipals$T%*%t(X.nipals$P)
    # maximum absolute value of error matrices:
    a <- max(abs(err.pca))
    b <- max(abs(err.nipals))
    res.pca <- c(res.pca,a)
    res.nipals <- c(res.nipals,b)
    res.rel <- c(res.rel,((a-b)/a))
  }
  mittel <- mean(res.rel)
  stderr <- (sqrt(var(res.rel)))/(sqrt(anz))
  end <- date()
  cat("starting time: ",begin," \n")
  cat("stopping time: ",end," \n\n")
  return(res.pca,res.nipals,res.rel,mittel,stderr)
}
else{
  cat("Warning! The number of used components ",k," should be
    smaller \n")
  cat("than the number of columns of the covariance matrix ",
    cols,"! \n\n")
}
}

f.sim2_function (anz,num,k,s,per)
# function "f.sim2" creates "anz" matrices X of a multivariate
# normal distribution specified by the covariance matrix "s"
# with dimension n by number of columns of "s". Some elements
# of the data matrix X are randomly set to "NA" according to the
# percentage given by the variable "per". For each X and each
# matrix with missing data the matrices of the scores and of
```

```
# the loadings are calculated by the functions "f.nipals(X,k)"
# and "f.nipna(X,k)" with "k" components. In order to compare the
# results a relative difference between the norms of the error
# matrices is calculated.
# anz...number of simulations
# num...number of observations of the data matrix X
# k...number of components that are calculated within the
#      NIPALS procedure
# s...covariance matrix of the multivariate normal distribution
# per...percentage of missing data
# date: 2002-08-17
{
begin <- date()
library(mass)
library(mva)
X <- NULL
Xna <- NULL
r.num <- 0
r.vec <- NULL
X.nipals <- NULL
X.nipna <- NULL
err.nipna <- NULL
err.nipals <- NULL
a <- 0
b <- 0
res.nipna <- NULL
res.nipals <- NULL
res.rel <- NULL
mittel <- 0
cols <- ncol(s)     #number of columns of s
if (k<cols){
  for (i in 1:anz){  #starting the simulations
    #creating a multivariate normal distributed data matrix X
    # with correlation matrix s
    X <- mvrnorm(n=num, rep(0, cols), s)
    Xna <- as.vector(X)
    r.num <- trunc(num*cols*per)
    r.vec <- sample(1:(num*cols),r.num)
    Xna[r.vec] <- NA
    Xna <- matrix(Xna,ncol=cols)
```

87

```
    X.nipals <- f.nipals(X,k)    #NIPALS of X
    X.nipna <- f.nipna(Xna,k)    #NIPALS of Xna
    #error matrix of NIPALS with NAs:
    err.nipna <- X - X.nipna$scores%*%t(X.nipna$loadings)
    #error matrix of NIPALS:
    err.nipals <- X - X.nipals$T%*%t(X.nipals$P)
    # maximum absolute value of error matrices:
    a <- max(abs(err.pca))
    b <- max(abs(err.nipals))
    res.nipna <- c(res.nipna,b)
    res.nipals <- c(res.nipals,a)
    res.rel <- c(res.rel,((a-b)/a))
  }
  mittel <- mean(res.rel)
  stderr <- (sqrt(var(res.rel)))/(sqrt(anz))
  end <- date()
  cat("starting time: ",begin," \n")
  cat("stopping time: ",end," \n\n")
  return(res.nipna,res.nipals,res.rel,mittel,stderr)
}
else{
  cat("Warning! The number of used components ",k," should be
      smaller \n")
  cat("than the number of columns of the covariance matrix ",
      cols,"! \n\n")
}
}
```

B.4   Time Series Analysis

```
f.acf_function (X,n,name="acf")
# The function "f.acf" calculates the autocorrelation
# of the time series that are stored in the the data
# matrix X.
# X...data matrix
# n...number of medical check-ups
# name...name of file
# date: 2002-09-12
{
```

```
rows <- nrow(X)
cols <- ncol(X)
store.name <- dimnames(X)[[2]]
G <- NULL
R <- NULL
for (j in 1:cols){
  mean.x <- mean(X[,j])
  g <- NULL
  for (h in 1:n){
    a <- 0
    for (l in 1:(rows/n)){
      for (k in 1:(n-(h-1))){
        a <- a + (X[(((l-1)*n)+k+(h-1)),j]-mean.x)*
          (X[(((l-1)*n)+k),j]-mean.x)
      }
    }
    g <- c(g,(1/rows)*a)
  }
  G <- cbind(G,g)
  R <- cbind(R,g/g[1])
}
dimnames(G)_list(0:(n-1),store.name)
dimnames(R)_list(0:(n-1),store.name)
sink(paste(name,".txt",sep=""))
cat("Matrix of Autocovariance: \n\n",G,"\n\n\n")
cat("Matrix of Autocorrelation: \n\n",R)
sink()
postscript(paste(name,".ps",sep=""),horizontal=F)
par(mfrow=c((ceiling(cols/2)),2))
for (j in 1:cols){
  plot(cbind(0:(n-1),R[,j]),typ="h",xlab="LAG",ylab="ACF",
    ylim=c(min(R[,j])-0.05,max(R[,j])+0.05))
  title(dimnames(X)[[2]][(j)],line=-1)
  abline(0,0)
  abline((1.96/sqrt(rows)),0,lty=2)
  abline((-1.96/sqrt(rows)),0,lty=2)
}
dev.off()
return(G,R)
}
```

```
f.pacf_function (G,R,N,name="pacf")
# The function "f.pacf" calculates the partial autocorrelations
# using the autocorrelations calculated by the function "f.acf".
# G...matrix of autocovariances
#     (result of function "f.acf")
# R...matrix of autocorrelations
#     (result of function "f.acf")
# N...number of patients
# name...name of file
# date: 2002-09-13
{
rows <- nrow(G)
cols <- ncol(G)
store.name <- dimnames(G)[[2]]
P <- NULL
for (k in 1:cols){
  p <- NULL
  for (h in 2:(rows-1)){
    M <- NULL
    for (i in 1:h){
      a <- NULL
      for (j in 1:h){
        a <- c(a,G[(abs(i-j)+1),k])
      }
      M <- rbind(M,a)
    }
    p <- c(p,lsfit(M,G[2:(h+1),k],intercept=F)$coef[h])
  }
  P <- cbind(P,p)
}
P <- rbind(R[1:2,],P)
dimnames(P)_list(0:(rows-1),store.name)
sink(paste(name,".txt",sep=""))
cat("Matrix of Partial Autocorrelations: \n\n",P)
sink()
postscript(paste(name,".ps",sep=""),horizontal=F)
par(mfrow=c((ceiling(cols/2)),2))
for (j in 1:cols){
  plot(cbind(0:(rows-1),P[,j]),typ="h",xlab="LAG",ylab="PACF",
      ylim=c(min(P[,j])-0.05,max(P[,j])+0.05))
```

```
  title(dimnames(P)[[2]][j],line=-1)
  abline(0,0)
  abline((1.96/sqrt(rows*N)),0,lty=2)
  abline((-1.96/sqrt(rows*N)),0,lty=2)
}
dev.off()
return(P)
}

f.ccf_function (X,i,n,g,name="ccf")
# The function "f.ccf" calculates the cross-correlations
# of the time series that are stored in the data
# matrix X.
# X...data matrix
# i...number of column
# n...number of medical check-ups
# g...vector of variances
# name...name of file
# date: 2002-09-12
{
rows <- nrow(X)
cols <- ncol(X)
x <- X[,i]
y <- X[,-i]
store.name <- dimnames(y)[[2]]
var.x <- g[i]
var.y <- g[-i]
G.ccf <- NULL
R.ccf <- NULL
for (j in 1:(cols-1)){
  mean.x <- mean(x)
  mean.y <- mean(y[,j])
  gx <- NULL
  gy <- NULL
  for (h in 1:n){
    a <- 0
    for (l in 1:(rows/n)){
      for (k in 1:(n-(h-1))){
        a <- a + ((x[(((l-1)*n)+k+(h-1))]-mean.x)*
           (y[(((l-1)*n)+k),j]-mean.y))
      }
```

```
    }
    gx <- c(gx,(1/rows)*a)
  }
  for (h in (n-1):1){
    b <- 0
    for (l in 1:(rows/n)){
      for (k in 1:(n-h)){
        b <- b + ((y[(((l-1)*n)+k+h),j]-mean.y)*
          (x[(((l-1)*n)+k)]-mean.x))
      }
    }
    gy <- c(gy,(1/rows)*b)
  }
  G.ccf <- cbind(G.ccf,c(gy,gx))
  R.ccf <- cbind(R.ccf,(c(gy,gx)/sqrt(var.x*(var.y[j]))))
}
dimnames(G.ccf)_list((-n+1):(n-1),store.name)
dimnames(R.ccf)_list((-n+1):(n-1),store.name)
sink(paste(name,"_ccf.txt",sep=""))
cat("Results of CCF for variable ",dimnames(X)[[2]][i],": \n\n")
cat("Matrix of Cross-Covariance: \n\n",G.ccf,"\n\n\n")
cat("Matrix of Cross-Correlation: \n\n",R.ccf)
sink()
postscript(paste(name,"_ccf.ps",sep=""),horizontal=F)
par(mfrow=c((ceiling((cols-1)/2)),2))
for (j in 1:(cols-1)){
  plot(cbind((-n+1):(n-1),R.ccf[,j]),typ="h",xlab="h",ylab="CCF",
    ylim=c(min(R.ccf[,j])-0.05,max(R.ccf[,j])+0.05))
  title(paste(dimnames(X)[[2]][(i)],"(t+h) vs. ",store.name[j],
    "(t)",sep=""),line=-1)
  abline(0,0)
  abline((1.96/sqrt(rows)),0,lty=2)
  abline((-1.96/sqrt(rows)),0,lty=2)
}
dev.off()
return(G.ccf,R.ccf)
}

f.tsa_function (X,n=30,j=4,trend=c(TRUE,TRUE),I=NULL,H=NULL,
   C=NULL,T=NULL,K=c(1:5),P=NULL,B=NULL)
# The function "f.tsa" estimates the parameters of
```

```
# the diabetes series using an univariate
# autoregressive model with exogeneous variables
# of arbitrary order specified by the parameters of
# the function.
# X...data matrix
# n...number of medical check-ups
# j...number of the variable which is analysed
# trend...logical vector, indicates whether a linear trend is
#          estimated or not
# exogenous variables:
# I...vector of input variables (e.g., gender,age,diabetes
#      type,...), constant in time
# H...vector of lags of the variable "HbA1c"
# C...vector of lags of the variable "Cholesterol"
# T...vector of lags of the variable "Triglyceride"
# K...vector of lags of the variable "Kidney"
# P...vector of lags of the variable "Blood Pressure"
# B...vector of lags of the variable "BMI"
# date: 2002-09-14
{
rows <- nrow(X)
name <- NULL
Y <- NULL
Z <- NULL
LSE <- NULL
B.est <- NULL
E <- NULL
S <- NULL
B.se <- NULL
k <- rows/n     # number of patients
X.new <- matrix(as.vector(X),nrow=n,byrow=FALSE)
# maximum lag:
r <- max(c(H,C,T,K,P,B))
# Y:
Y <- as.vector(t(X.new[((r+1):n),(((j-1)*k+1):(j*k))]))
# Z:
if (trend[1]){      # intercept
  Z <- rep(1,length(Y))
  name <- c(name,"trend")
}
```

```r
if (trend[2]){        # linear trend
  a <- NULL
  for (i in (r+1):n){
    a <- c(a,rep(i,k))
  }
  Z <- rbind(Z,a)
  name <- c(name,"trend")
}
if (length(I[1,])!=0){    # input variables, if availale
  Z <- rbind(Z,matrix(rep(as.vector(I),(n-r)),ncol=length(Y),
    byrow=FALSE))
  name <- c(name, rep("input",nrow(I)))
}
if (length(H)!=0){    # HbA1c, if available
  for (i in 1:(length(H))){
    Z <- rbind(Z,as.vector(t(X.new[((r+1-H[i]):(n-H[i])),1:k])))
    name <- c(name,"HbA1c")
  }
}
if (length(C)!=0){     # Cholesterol, if available
  for (i in 1:(length(C))){
    Z <- rbind(Z,as.vector(t(X.new[((r+1-C[i]):(n-C[i])),
      (k+1):(2*k)])))
    name <- c(name,"Chol")
  }
}
if (length(T)!=0){     # Triglyceride, if available
  for (i in 1:(length(T))){
    Z <- rbind(Z,as.vector(t(X.new[((r+1-T[i]):(n-T[i])),
      ((2*k)+1):(3*k)])))
    name <- c(name, "Trigl")
  }
}
if (length(K)!=0){     # Kidney, if available
  for (i in 1:(length(K))){
    Z <- rbind(Z,as.vector(t(X.new[((r+1-K[i]):(n-K[i])),
      ((3*k)+1):(4*k)])))
    name <- c(name, "Kidne")
  }
}
```

```
if (length(P)!=0){    # Blood Pressure, if available
  for (i in 1:(length(P))){
    Z <- rbind(Z,as.vector(t(X.new[((r+1-P[i]):(n-P[i])),
       ((4*k)+1):(5*k)]))))
    name <- c(name, "BP")
  }
}
if (length(B)!=0){    # Body Mass Index, if available
  for (i in 1:(length(B))){
    Z <- rbind(Z,as.vector(t(X.new[((r+1-B[i]):(n-B[i])),
       ((5*k)+1):(6*k)]))))
    name <- c(name, "BMI")
  }
}
dimnames(Z) <- list(name,NULL)
# Estimation of B.est and E:
LSE <- lsfit(t(Z),Y,intercept=FALSE)
B.est <- LSE$coef
E <- LSE$res
# Estimaton of S and Standard Error:
S <- (1/(k*(n-r)))*(t(E)%*%E)
B.est <- cbind(B.est,sqrt(S*diag(solve(Z%*%t(Z)))))
return(r,X.new,Y,Z,B.est,E,S)
}
```

Bibliography

T.W. Anderson. Estimation for autoregressive moving average models in the time and frequency domain. *Ann. Stat.*, 5:842–865, 1978.

M. Deistler and W. Scherrer. The Prague Lectures Econometrics II. Dept. of Econometrics, Operations Research and System Theory, Vienna University of Technology, Vienna, 1994.

J. Durbin and S.J. Koopman. *Time Series Analysis by State Space Methods.* Oxford University Press, New York, 2001.

B.S. Everitt and G. Dunn. *Applied Multivariate Data Analysis.* Arnold, London, 2001.

P. Geladi and B.R. Kowalski. Partial least–squares regression: A tutorial. *Analytica Chimica Acta*, 185:1–17, 1986.

J. Gower and D. Hand. *Biplots.* Chapman & Hall, New York, 1996.

J. Hartung, B. Elpelt, and H.-K. Klösener. *Statistik: Lehr- und Handbuch der angewandten Statistik.* Oldenbourg Verlag, München, 1998.

K. Howorka. *Functional Insulin Treatment: Principles, Teaching Approach and Practice.* Springer, Berlin, 2nd edition, 1996.

K. Howorka, G. Heger, A. Schabmann, P. Anderer, G. Tribl, and J. Zeitlhofer. Severe hypoglycaemia unawareness is associated with an early decrease in vigilance during hypoglycaemia. *Psychoneuroendocrinology*, 21:295–312, 1996.

K. Howorka, G. Heger, A. Schabmann, F. Skrabal, and J. Pumprla. Weak relationship between symptom perception and objective hypoglycaemia-induced changes of autonomic function in hypoglycaemia unawareness in diabetes. *Acta Diabetol*, 35:1–8, 1998a.

K. Howorka, J. Pumprla, P. Haber, J. Koller-Strametz, J. Mondrzyk, and A. Schabmann. Effects of physical training on heart rate variability in diabetic patients with various degrees of cardiovascular autonomic neuropathy. *Cardiovascular Research*, 34:206–214, 1997.

K. Howorka, J. Pumprla, B. Saletu, P. Anderer, M. Kieger, and A. Schabmann. Decrease of vigilance assessed by EEG-mapping in type I diabetic patients with history of recurrent severe hypoglycaemia. *Psychoneuroendocrinology*, 25:85–105, 2000.

K. Howorka, J. Pumprla, and A. Schabmann. Optimal parameters of short-term heart rate spectrogram for routine evaluation of diabetic cardiovascular autonomic neuropathy. *Journal of the Autonomic Nervous System*, 69:164–172, 1998b.

A.G. Journel and Ch.J. Huijbregts. *Mining Geostatistics*. Acad. Press, New York, 1978.

B. Kavšek. Partial Least Squares (PLS) Regression and its Robustification. Master's thesis, Dept. of Statistics and Probability Theory, Vienna University of Technology, Vienna, 2002.

K.V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate Analysis*. Acad. Press, London, 1979.

C.R. Rao and H. Toutenburg. *Linear Models: Least Squares and Alternatives*. Springer, New York, 1995.

R.H. Shumway and D.S. Stoffer. *Time Series Analysis and Its Applications*. Springer, New York, 2000.

D. Staas. *Paradox 7*. Hanser, München, 2000.

N. Stockinger and R. Dutter. Robust time series analysis: A survey. *Kybernetika*, 23, 1987.

M. Tenenhaus. *La Régression PLS*. Éditions Technip, Paris, 1998.

H. Wold. Nonlinear estimation by iterative least square procedures. In F.N. David, editor, *Research Papers in Statistics: Festschrift for Jerzy Neyman*, pages 411–444. Wiley, New York, 1966.