LMU LUDWIG MAXIMILIANS UNIVERSITY MUNICH

DEPARTMENT OF STATISTICS

Master Thesis

Multiple Imputation in Generalized Linear Mixed Models

Author:
Rudolf Jagdhuber

Supervisor:
Prof. Dr. Christian Heumann

Ludwig Maximilians University Munich

Winter Semester of 2015 / 2016

Statutory Declaration

I hereby declare, that I have authored the enclosed Master Thesis independently, that I have not used sources or means without declaration in the text, and that any thoughts from others or literal quotations are clearly marked.

Munich, 5. März 2016

_____

(Rudolf Jagdhuber)

Acknowledgement

This thesis was developed at the Department of Statistics at Ludwig Maximilians University Munich.

I would particularly like to thank Prof. Dr. Christian Heumann, who supported me with plenty of helpful discussions and enabled a successful outcome of this project.

Apart from the technical assistance, i also give special thanks to my parents and to my girlfriend, who always provided an accommodation and personal support for me during this very busy period of time.

Contents

Page

Statutory Declaration **I**

Acknowledgement **II**

1. Introduction **1**

2. Linear Mixed Models **3**

2.1. The Linear Mixed Model (LMM) 4

2.1.1. General Definition of the LMM 4

2.1.2. Common Special Cases of LMMs 6

2.1.3. Marginal and conditional perspective 10

2.1.4. Estimation of the LMM 11

2.2. The Generalized Linear Mixed Model (GLMM) 13

2.2.1. Introduction 13

2.2.2. Example of a GLMM with Binary Response 14

2.2.3. Estimation of the GLMM 18

2.2.4. GLMMs in `R` using the package `lme4` 24

2.3. Bootstrapping from Mixed Model Data 27

3. Imputation of Missing Data with Amelia II **29**

3.1. Missingness Assumptions 31

3.2. Multiple Imputation 32

3.3. The Imputation Model 34

3.4. The Estimation Algorithm . . . 36
3.4.1. The Sweep operator $\mathcal{S}wp(\mathbf{X}, \{s\})$ . . . 36
3.4.2. Sufficient Statistics . . . 37
3.4.3. The ML and Bayesian Estimation Approach . . . 38
3.4.4. The EM algorithm . . . 39
3.4.5. Imputing Multiple Missing Values . . . 42
3.5. TSCS-Data Imputation . . . 43
3.6. Practical Use in `R` . . . 44
3.6.1. Calling the Function `amelia()` . . . 44
3.6.2. Return Value of `amelia()` . . . 46

4. Simulations **47**
4.1. Design of the Simulations . . . 47
4.2. Generating Test Data . . . 48
4.2.1. Independent Variables . . . 48
4.2.2. Summary of Covariable Dependencies . . . 59
4.2.3. Response Variable . . . 60
4.3. Generating Missing Values . . . 65
4.4. Estimating the Theoretical Standard Error of $\hat{\boldsymbol{\beta}}$ . . . 68
4.5. Simulation I: No Missing Values . . . 69
4.5.1. Overview . . . 69
4.5.2. Implementation . . . 70
4.5.3. Results . . . 72
4.6. Simulation II: Complete Case Analysis . . . 76
4.6.1. Overview . . . 76
4.6.2. Implementation . . . 78
4.6.3. Results . . . 79
4.7. Simulation III: Multiple Imputation (Amelia II) . . . 83
4.7.1. Overview . . . 83
4.7.2. Implementation . . . 85

4.7.3. Results 88
4.8. Results of Simulation I, II and III in comparison 93

5. Summary **97**

A. R Code and Further Descriptions **99**
A.1. Theoretical variance of β: `theosd()` 99
A.2. Parallel computing using the package `snow` 102
A.3. Bootstrap Percentil Interval `bpi()` 103
A.4. Other discussed combination methods for $\hat{d}$ 104
A.4.1. Two-Stage Design 104
A.4.2. Probability Weighting Approach 105

B. Additional Plots **107**
B.1. Densities of the Fixed Effects Estimators 107
B.2. Bootstrap Percentile Intervals for 1000 Runs 110

C. Digital Supplement on CD **114**

List of Figures **116**

List of Tables **117**

References **118**

Mathematical Notation

Fonts and Operations

a	:	A scalar value
$\mathbf{a}$	:	A column vector of scalar values
$\mathbf{A}$	:	A matrix of scalar values
$\mathbf{A}^T$	:	The transposed matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	:	The inverse of matrix $\mathbf{A}$
$\hat{a}$	:	An estimator for the scalar a

Globally Defined Variables

$\mathbf{y}$	:	The response vector of a regression model
$\mathbf{X}$	:	The matrix of covariables for the fixed effects
$\mathbf{Z}$	:	The matrix of covariables for the random effects
β	:	A fixed effect
b	:	A random effect
ϵ	:	The residual
η	:	The linear predictor
$\mathbf{\Sigma}$	:	The variance covariance matrix of the residuals
$\mathbf{D}$	:	The variance covariance matrix of the random effects
$\mathfrak{D}$	:	A data set comprising covariables and response
$\mathcal{M}$	:	The missingness matrix

Globally Defined Indices

m	:	The number of individuals
n_i	:	The number of observations for individual i
N	:	The overall number of observations $= \sum_i n_i$

p	:	The number of fixed effects
q	:	The number of random effects
M	:	The number of imputations

Distributions

$\mathcal{N}(\mu, \sigma^2)$	:	Normal distribution
$\mathcal{N}_q(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	:	q-dimensional multivariate Normal distribution
$\mathcal{B}(\pi)$	:	Bernoulli distribution
$\mathcal{B}(n, \pi)$	:	Binomial distribution
$\mathcal{B}\mathfrak{e}(a, b)$	:	Beta distribution

Special Characters and Abbreviations

$\mathbb{1}_n$	:	A column vector of length n containing ones
$\mathbf{I}_n$	:	The identity matrix of dimension $n \times n$
$\sim$	:	"is distributed"
$\overset{\text{ind}}{\sim}$	:	"is independently distributed"
$\overset{\text{iid}}{\sim}$	:	"is identically and independently distributed"
∇	:	The gradient
$\mathbb{E}$	:	The expected value
$\mathbb{V}\text{ar}$	:	The variance
$\mathbb{C}\text{ov}$	:	The covariance

Functions

$p(\cdot)$	:	The density function
$\mathcal{N}(x\|\mu, \sigma^2)$	:	An alternative notation for the density $p(x)$, having $x \sim \mathcal{N}(\mu, \sigma^2)$
$\mathcal{L}(\cdot)$	:	The likelihood function
$\ell(\cdot)$	:	The log-likelihood function
$g(\cdot)$	:	The link function
$h(\cdot)$	:	The response function
$\mathcal{S}wp(\cdot)$	:	The sweep operator

1. Introduction

A multitude of studies in the fields of medicine or public health are driven by longitudinal data. If repeated measurements of one individual are obtained over time, the data situation is often referred to as "time-series cross-sectional data" (TSCS). An appropriate modeling approach for this structure of observations is the (generalized) linear mixed model (GLMM). It addresses dependencies between measurements of the same individual by extending the linear predictor.

The computation of these models - or regression models in general - requires a rectangular data set without missing values. In many instances of data collection though, incomplete observations are an inevitable issue.
A frequently used method to address the missingness problem is listwise deletion. It excludes incomplete data rows from the analysis and hence reduces the data set to fully observed ones. This approach however not only discards valuable information, it may also induce an estimation bias, if the missingness of a data point is dependent on observed variables.

The contrary method to deleting missing data rows is to impute incomplete fragments, i.e. to estimate the values utilizing available information. A corresponding approach, which gained a lot of recognition, is the multiple imputation of "Amelia II: A program for missing data" (Honaker et al. 2011). The general idea of multiple imputation is to impute missing values $M > 1$ times. Thereby, the additional uncertainty of the imputation process can be taken into account. Statistical procedures are afterwards applied

to all completed data sets. Combining their outcomes by a simple procedure yields the final result.

The aim of this thesis was to analyze the multiple imputation of Amelia II in generalized linear mixed models. Special notice was given to the combination method of the model parameters. Whereas theoretical approaches for combining fixed effects and their corresponding variance estimates already exist, methods for the combination of the random effects parameters were not thoroughly analyzed up to this point. Different strategies were applied in a simulation study to evaluate the performance.

The thesis begins with an introduction on linear mixed regression models in chapter 2. Particularly linear mixed models and generalized linear mixed models including the respective estimation process and practical applications in `R` are described.
Chapter 3 concerns the multiple imputation of missing data with Amelia II in the context of TSCS data. Again, applications of the theory in `R` are introduced at the end.
Chapter 4 presents the implementation and analysis of the simulation studies. After the generation and preparation of artificial test data, GLMM estimations were reviewed having

- a full data set (Simulation I).
- a complete case data set obtained by applying listwise deletion (Simulation II).
- an imputed data set using Amelia II (Simulation III).

In the final section of this chapter, the obtained results are compared and evaluated.
Concluding remarks on the researched framework are given in chapter 5.

2. Linear Mixed Models

Mixed models are convenient in situations, where the data is naturally divided into clusters. This can e.g. be the case, if multiple measurements of each individual subject are obtained over a period of time.
In linear and generalized linear regression models, the predictor η_i of an individual i is defined as $\mathbf{x}_i^T\boldsymbol{\beta}$ comprising the covariable vector $\mathbf{x}_i$ alongside its associated fixed effects $\boldsymbol{\beta}$. The idea of a mixed model approach is to extend the predictor by so called random effects. These effects model cluster-specific influences on the response. For clus- ters $i = 1, \ldots, m$ and measurements $j = 1, \ldots, n_i$ the linear predictor can be written as

$$\eta_{ij} = \mathbf{x}_{ij}^T\boldsymbol{\beta} + \mathbf{z}_{ij}^T\mathbf{b}_i$$

Here $\mathbf{z}_{ij}$ often represents a partial vector of $\mathbf{x}_{ij}$. $\mathbf{b}_i$ denotes the vector of random effects.

Compared to estimating models with fixed effects for each cluster, the mixed model approach has several advantages. Depending on the number of clusters, a substantially smaller amount of parameters needs to be estimated. These parameters are also less affected by small sample sizes within the clusters. Finally, mixed models are able to account correlations of the repeated measurements.

This chapter starts by presenting the linear mixed model (LMM). The second section firstly introduces the generalized linear model by broadening the spectrum of possible response distributions of the linear model. To make this model applicable on data situations with repeated measurements, the generalized linear model is extended by introducing random effects. This leads to the generalized linear mixed model (GLMM). The chapter ends with a description on how to bootstrap in these TSCS data situations.

2.1. The Linear Mixed Model (LMM)

The classical linear model (LM) for observations y_i $(i = 1, \ldots, m)$ and covariables $\mathbf{x}_i$ can be specified in a variety of ways. An often used definition is the representation

$$y_i = \mathbf{x}_i^T \boldsymbol{\beta} + \epsilon_i \tag{2.1}$$

with

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2) \tag{2.2}$$

Assumption 2.2 is not necessarily required[1] for the LM, but is often used to enable further methods like maximum likelihood estimation.

Suppose a given data situation, where n_i observations for each of the m individuals are measured. Its structure can be illustrated by two indices.

$$(y_{ij}, \mathbf{x}_{ij}) \quad , \quad i = 1, \ldots, m \quad , \quad j = 1, \ldots, n_i \tag{2.3}$$

This type of data can be modeled by extending the model equation of the LM.

2.1.1. General Definition of the LMM

A common notation for the model equation of the LMM is

$$y_{ij} = \mathbf{x}_{ij}^T \boldsymbol{\beta} + \mathbf{z}_{ij}^T \mathbf{b}_i + \epsilon_{ij} \tag{2.4}$$

The response of 2.4 depends partly on the fixed effects $\boldsymbol{\beta}$ with their associated covariables $\mathbf{x}_{ij}$, and partly on random effects $\mathbf{b}_i$ for each statistical unit i with covariables $\mathbf{z}_{ij}$. For a compact notation, one can use vectors and matrices to summarize response, dependent and error variables. This leads to the "individual notation"

$$\mathbf{y}_i = \mathbf{X}_i \boldsymbol{\beta} + \mathbf{Z}_i \mathbf{b}_i + \boldsymbol{\epsilon}_i \tag{2.5}$$

where

$$\mathbf{y}_i = \begin{pmatrix} y_1 \\ \vdots \\ y_{n_i} \end{pmatrix}_i , \quad \mathbf{X}_i = \begin{pmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_{n_i}^T \end{pmatrix}_i , \quad \mathbf{Z}_i = \begin{pmatrix} \mathbf{z}_1^T \\ \vdots \\ \mathbf{z}_{n_i}^T \end{pmatrix}_i , \quad \boldsymbol{\epsilon}_i = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_{n_i} \end{pmatrix}_i \tag{2.6}$$

[1] Alternatively: $\mathbb{E}(\epsilon_i) = 0 \;\; ; \;\; \mathbb{V}\mathrm{ar}(\epsilon_i) = \sigma$

The following distribution assumptions are made on this model:

$$\mathbf{b}_i \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{D}) \tag{2.7}$$

$$\boldsymbol{\epsilon}_i \overset{\text{ind}}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_i) \tag{2.8}$$

To assure 2.7, it is usually necessary, that all covariables of $\mathbf{Z}_i$ are also included in $\mathbf{X}_i$. The only theoretical exception would be, if the associated fixed effects are exactly 0.
The $(q \times q)$ covariance matrix of the normal distributed $\mathbf{b}_i$ is $\mathbf{D}$. Assumptions on the structure of this matrix are sometimes used to simplify the model complexity. A diagonal matrix $\mathbf{D}$ would e.g. assume all random effects to be independent of each other. This is called a variance component model.
For the error covariance $\boldsymbol{\Sigma}_i$, it is also conceivable to use a diagonal matrix $\sigma_i^2 \mathbf{I}_{n_i}$ (or $\sigma^2 \mathbf{I}_{n_i}$), if measurements of the same cluster are assumed to be uncorrelated. This can e.g. be the case in time-series data, if the time gap between two points is very large.
Another approach, which is often used for longitudinal data is to partition the error vector.

$$\boldsymbol{\epsilon}_i = \boldsymbol{\epsilon}_i^{(1)} + \boldsymbol{\epsilon}_i^{(2)} \tag{2.9}$$

$\boldsymbol{\epsilon}_i^{(2)}$ represents the serial correlation. It typically decreases with the distance of two measurements. $\boldsymbol{\epsilon}_i^{(1)}$ is an additional error term, which is independent of $\boldsymbol{\epsilon}_i^{(2)}$. The distributions of those errors are

$$\boldsymbol{\epsilon}_i^{(1)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}^2 \mathbf{I}_{n_i}) \tag{2.10}$$

$$\boldsymbol{\epsilon}_i^{(2)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\tau}^2 \mathbf{H}_i) \tag{2.11}$$

The (j,k)-element h_{ijk} of the $n_i \times n_i$ correlation matrix $\mathbf{H}_i$ is usually modeled as a function of the distance between two data points.

$$h_{ijk} = g(|t_{ij} - t_{ik}|) \tag{2.12}$$

where t_{ij} is the time of measurement j for person i and g is a decreasing function with $g(0) = 1$. Examples for g would be the exponential serial correlation $g(u) = \exp(-\Phi u)$, or the gaussian serial correlation $g(u) = \exp(-\Phi u^2)$ with parameter $\Phi > 0$.

Note that in any of these serial correlation cases $\mathbf{\Sigma}_i$ only depends on i over the number of observations n_i.

Individuals from the notation of 2.5 can be summarized to get a more compact notation for the LMM.

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_m \end{pmatrix}, \quad \boldsymbol{\epsilon} = \begin{pmatrix} \boldsymbol{\epsilon}_1 \\ \vdots \\ \boldsymbol{\epsilon}_m \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_m \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_m \end{pmatrix},$$
$$\mathbf{Z} = \begin{pmatrix} \mathbf{Z}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{Z}_m \end{pmatrix}, \quad \mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{\Sigma}_m \end{pmatrix}, \quad \mathcal{D} = \begin{pmatrix} \mathbf{D}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{D}_m \end{pmatrix} \tag{2.13}$$

where $\mathbf{D}_i \equiv \mathbf{D}$.

Altogether the LMM can be defined by

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon} \tag{2.14}$$

with

$$\begin{pmatrix} \mathbf{b} \\ \boldsymbol{\epsilon} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathcal{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma} \end{pmatrix} \right) \tag{2.15}$$

Like stated earlier for the LM, the normal distribution assumption is not necessarily needed for every inferential statement. Yet, it is standardly used for (restricted) maximum likelihood estimations of the unknown parameters in $\mathbf{\Sigma}$ and $\mathcal{D}$ (see section 2.1.4).

2.1.2. Common Special Cases of LMMs

This section concerns two special cases of LMMs in the context of longitudinal data. The first one is the random intercept model, which is the most simple type of linear mixed model. In the subsequent part, it is extended by a random time effect, which leads to the random slope model. Both models are very commonly used in practical applications and are good starting points to understand the concept of LMMs.

The Random Intercept Model

As the name indicates, the Random Intercept Model (RIM) is characterized by including an individual intercept as sole random effect.
Therefore, the matrix $\mathbf{Z}_i$ of equation 2.5 reduces to a vector of ones and the vector $\mathbf{b}_i$ reduces to a scalar.
The model equation of the RIM in individual notation reads as follows:

$$\mathbf{y}_i = \mathbf{X}_i\boldsymbol{\beta} + \mathbb{1}_{n_i} b_i + \boldsymbol{\epsilon}_i \tag{2.16}$$

$$b_i \overset{\text{iid}}{\sim} \mathcal{N}(0, d^2) \qquad ; \qquad \boldsymbol{\epsilon}_i \overset{\text{ind}}{\sim} \mathcal{N}(0, \boldsymbol{\Sigma}_i) \tag{2.17}$$

Written in general matrix notation, 2.16 equals

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n_i} \end{pmatrix}_i = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1(p-1)} \\ 1 & x_{21} & x_{22} & \dots & x_{2(p-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n_i1} & x_{n_i2} & \dots & x_{n_i(p-1)} \end{pmatrix}_i \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{p-1} \end{pmatrix} + \begin{pmatrix} b \\ b \\ \vdots \\ b \end{pmatrix}_i + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_{n_i} \end{pmatrix}_i \tag{2.18}$$

In this case, the distribution of the random effect is a one-dimensional normal distribution with variance d^2. To illustrate an application of the RIM, the fictional data of figure 2.1 is used.
This data set comprises of 4 individuals with 10 measurements each. It represents a special kind of TSCS data, which is called "balanced data". Measurements for all individuals are taken at fixed (here also equidistant) points in time without missing values. In practice, most data sets may not be as simple structured as in this example, but for ease of exposition a small amount of persons without intersecting sequences was chosen.

The model equation of a RIM for this example could read

$$\mathbf{y}_i = \boldsymbol{\beta}_0 + \mathbf{x}_{i,\text{time}}\beta_{\text{time}} + \mathbf{b}_i + \boldsymbol{\epsilon}_i \tag{2.19}$$

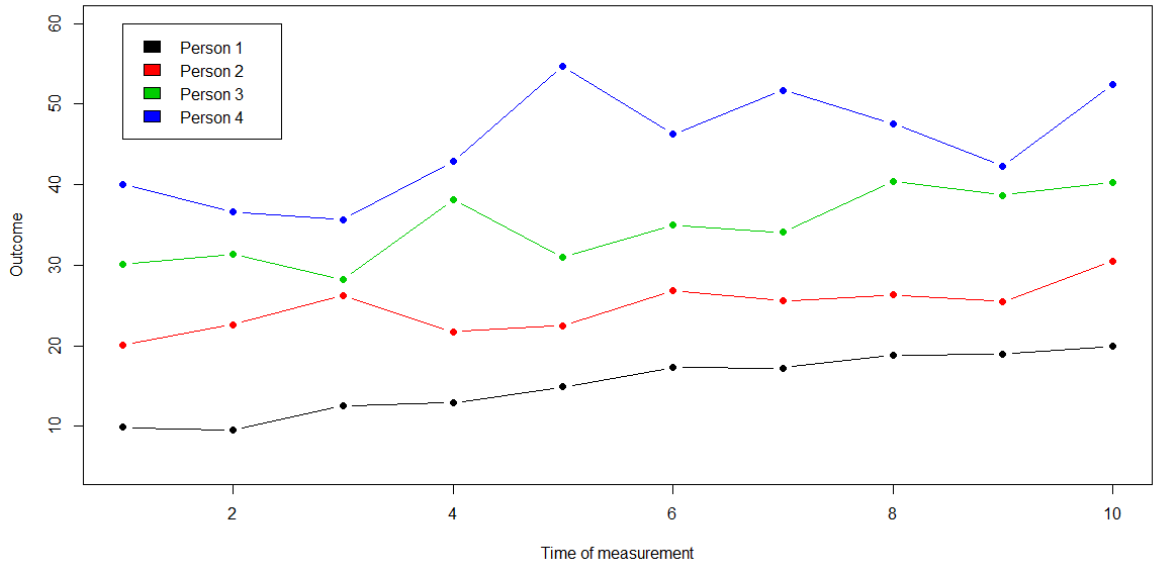


Figure 2.1.: Illustration of a simple time series cross section data set.

The model of 2.19 is visualized in figure 2.2.

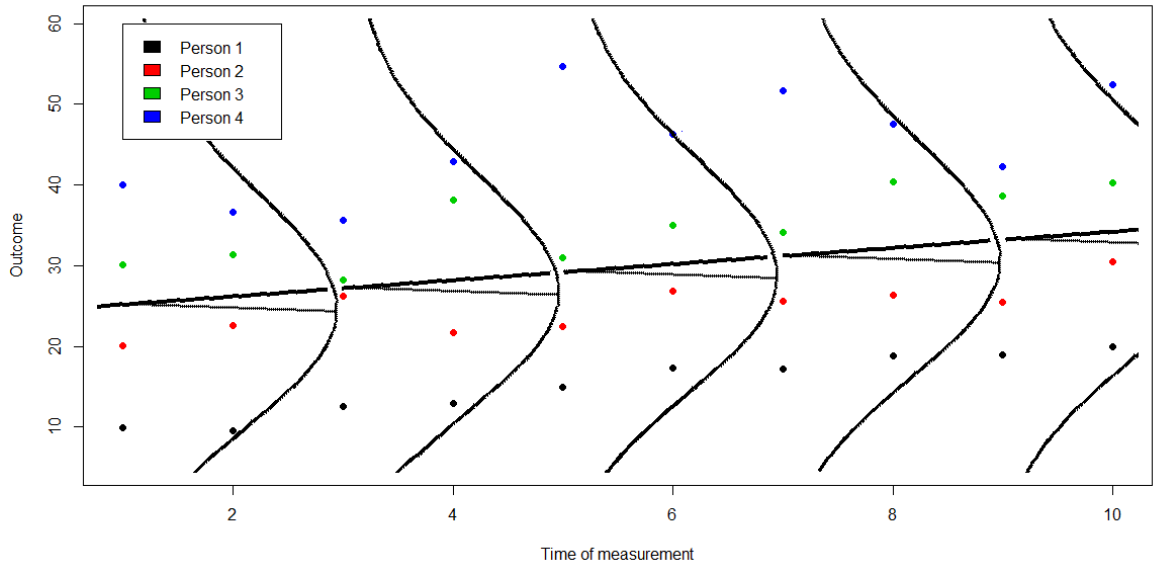


Figure 2.2.: Illustration of the RIM. The straight black line corresponds to the fixed effects (i.e. marginal estimation). The added normal distributed random effect is visualized by the superimposed density functions.

The Random Slope Model

Another commonly used LMM is the Random Slope Model (RSM). It extends the RIM by adding a random time effect.

A RSM can be appropriate, if the individual trends tend to differ in slope. In example 2.1, this seemed not to be the case and a RIM would be satisfactory. However, if a data situation as shown in figure 2.3 is given, adding a random time effect may be plausible.

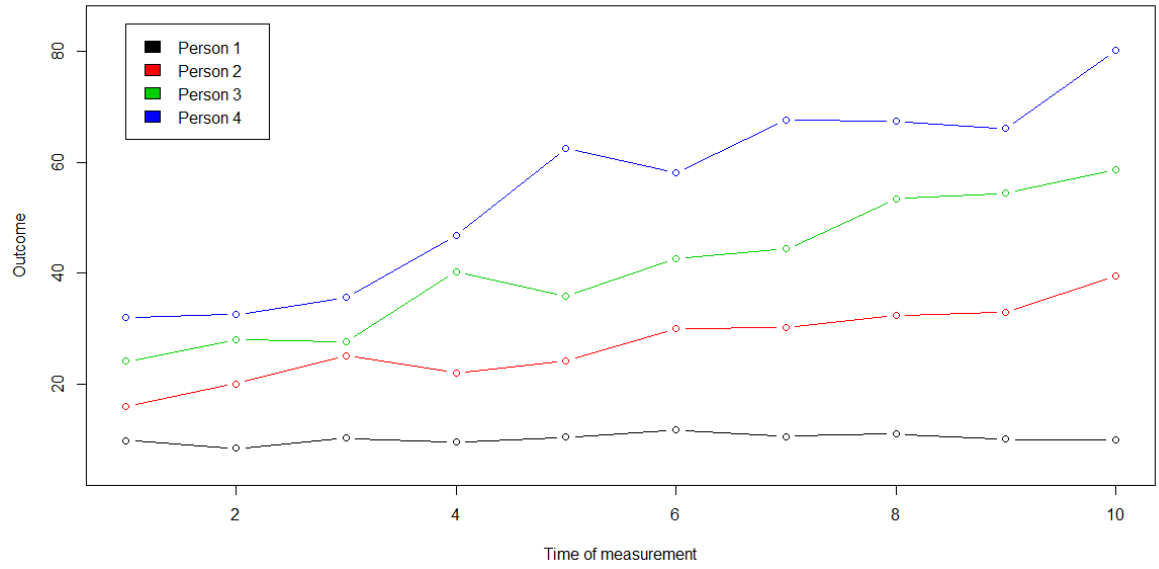


Figure 2.3.: Exemplary data situation, where the random slope model appears to be suited.

A model equation, which includes an individual slope could read

$$\mathbf{y}_i = \boldsymbol{\beta}_0 + \mathbf{x}_{\text{time},i}\beta_1 + \mathbf{b}_{0,i} + \mathbf{x}_{\text{time},i}\mathrm{b}_{1,i} + \boldsymbol{\epsilon}_i \tag{2.20}$$

In this situation, the random effects $\mathbf{b}_i$ are assumed to be two-dimensionally normal distributed.

$$\begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{pmatrix}\right) \tag{2.21}$$

2.1.3. Marginal and conditional perspective

The LMM can be perceived and interpreted from two perspectives.
If the main interest is to interpret the cluster influences on the response variable, the **conditional perspective** is used.

$$\mathbf{y}|\mathbf{b} \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b}, \boldsymbol{\Sigma}) \tag{2.22}$$

with

$$\mathbf{b} \sim \mathcal{N}(\mathbf{0}, \mathcal{D})$$

This hierarchical notation models the expected value of $\mathbf{y}$ as a function of population effects alongside random effects, which are interpreted as individual effects here.

The alternative point of view is the **marginal perspective**. Here, the random effects induce a correlation structure to enable a statistically valid analysis of the correlated data situation. The expected value of y_i is averaged over the population and modeled as function of fixed effects only.

$$\mathbf{y} \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \mathbf{V}) \tag{2.23}$$

with

$$\mathbf{V} = \mathbf{Z}\mathbf{D}\mathbf{Z}^T + \boldsymbol{\Sigma}$$

The marginal perspective follows directly from the conditional one by integrating over the random effects. However, the conditional perspective does not follow from the marginal one, because the latter does not assume random effects to represent heterogeneity.
The interpretation of the fixed effects is equivalent in both perspectives.

Consequently, the estimation process, which is presented in the next section differs between fixed and random effects. The fixed effects are estimated on the marginal distribution of $\mathbf{y}$, whereas the prediction of $\mathbf{b}$ needs to be conducted using the hierarchical notation $\mathbf{y}|\mathbf{b}$.

2.1.4. Estimation of the LMM

This section first elaborates the estimation of the LMM using maximum likelihood (ML) and moves then on to the more regularly used restricted maximum likelihood estimation (REML). In the following, the parameters of interest $(\boldsymbol{\beta}, \mathbf{D}, \boldsymbol{\Sigma})$ are summarized as $\boldsymbol{\theta}$.

Maximum Likelihood Estimation

ML estimation for $\boldsymbol{\theta}$ is performed on the marginal distribution $p(\mathbf{y}|\boldsymbol{\theta})$. Using the distribution assumptions of 2.15 and the model equation 2.14, one can obtain the distribution of $\mathbf{y}$ for known random effects as a shift of the normal distributed $\boldsymbol{\epsilon}$ by $\mathbf{X}\boldsymbol{\beta} + \mathbf{Zb}$.

$$p(\mathbf{y}|\mathbf{b}, \boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Zb}, \boldsymbol{\Sigma}) \tag{2.24}$$

By applying the sum and product rule of probability[2], the marginal distribution of $\mathbf{y}$ can be written as

$$p(\mathbf{y}|\boldsymbol{\theta}) = \int_{\mathbb{R}^q} p(\mathbf{y}|\mathbf{b}, \boldsymbol{\theta}) p(\mathbf{b}|\mathbf{D}) \, \mathrm{d}\mathbf{b} \tag{2.25}$$

Both elements in the integral are known normal distributions. If a conditional normal distribution is multiplied with its condition, which is also normal, the resulting joint distribution is again a normal distribution.

$$p(\mathbf{y}|\boldsymbol{\theta}) = \int_{\mathbb{R}^q} p\left(\begin{pmatrix}\mathbf{y}\\ \mathbf{b}\end{pmatrix}|\boldsymbol{\theta}\right) \mathrm{d}\mathbf{b} = \int_{\mathbb{R}^q} \mathcal{N}\left(\begin{pmatrix}\mathbf{X}\boldsymbol{\beta}\\ \mathbf{0}\end{pmatrix}, \begin{pmatrix}\mathbf{V} & \mathbf{ZD}\\ \mathbf{DZ}^T & \mathbf{D}\end{pmatrix}\right) \mathrm{d}\mathbf{b} \tag{2.26}$$

where $\mathbf{V} = \mathbf{ZDZ}^T + \boldsymbol{\Sigma}$

The marginal distribution of this partitioned Gaussian can be easily obtained.[3]

$$p(\mathbf{y}|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \mathbf{V}) \tag{2.27}$$

Based on the marginal model, a suitable estimator for the fixed effects is given by the ML- or generalized least squares estimator

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{V}^{-1}\mathbf{y} \tag{2.28}$$

[2] cf. Bishop 2006 page 14, eq. 1.10 and 1.11

[3] cf. Bishop 2006 page 90, eq. 2.98

To estimate the components of the covariance matrices $\mathbf{\Sigma}$ and $\mathcal{D}$ (in the following summarized as $\boldsymbol{\vartheta}$), a profile likelihood approach is used.
The log-likelihood of $\boldsymbol{\theta}$ excluding additive constants is

$$\ell(\boldsymbol{\theta}) = -\frac{1}{2}\{\log|\mathbf{V}| + (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\} \tag{2.29}$$

By inserting the estimator $\hat{\boldsymbol{\beta}}$ of 2.28 into this log-likelihood, the profile log-likelihood is obtained.

$$\ell_P(\boldsymbol{\vartheta}) = -\frac{1}{2}\{\log|\mathbf{V}| + (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^T\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})\} \tag{2.30}$$

Maximizing $\ell_P(\boldsymbol{\vartheta})$ with respect to $\boldsymbol{\vartheta}$ gives the ML estimator $\hat{\boldsymbol{\vartheta}}_{\text{ML}}$. Numerical approaches to realize this optimization are e.g. the Newton-Raphson or Fisher-Scoring algorithm.

Restricted Maximum Likelihood Estimation

A main disadvantage of the maximum likelihood estimation is the induced bias of variance parameters, which arises from inserting fixed effects estimations into the log-likelihood. One way of solving this problem, is to optimize the restricted log-likelihood instead of the profile log-likelihood. It is obtained by integrating the likelihood with respect to $\boldsymbol{\beta}$.

$$\ell_R(\boldsymbol{\vartheta}) = \log\left(\int \mathcal{L}(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\beta}\right) \tag{2.31}$$

The resulting $\ell_R(\boldsymbol{\vartheta})$ differs from the profile-log-likelihood only by an additive term.

$$\ell_R(\boldsymbol{\vartheta}) = \ell_P(\boldsymbol{\vartheta}) - \frac{1}{2}\log|\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X}| \tag{2.32}$$

Analogous to the ML estimator, $\hat{\boldsymbol{\vartheta}}_{\text{REML}}$ is computed iteratively.

Due to the unbiased estimation of the variance components, REML is the commonly used method to estimate $\boldsymbol{\vartheta}$. However, despite eliminating the bias of the estimator, it can not generally be ensured, that the mean squared error is reduced by performing REML estimation.

2.2. The Generalized Linear Mixed Model (GLMM)

2.2.1. Introduction

The Generalized Linear Model (GLM)

Section 2.1, which presented the LMM, opened with the classical linear model. Analogously, to introduce the GLMM a short revision of the generalized linear model will be made here.

The GLM is based on two main assumptions. The first one is the **structural assumption**:

$$\begin{aligned} \mathbb{E}(y_i|\mathbf{x}_i) = \mu_i = h(\eta_i) = h(\mathbf{x}_i^T\boldsymbol{\beta}) \\ g(\mu_i) = \eta_i \end{aligned} \tag{2.33}$$

The expected value of the response is connected to the linear predictor η_i over the response function $h(\eta_i) = \mu_i$ (respectively over the link function $g(\mu_i) = \eta_i$, having $g(\cdot) = h^{-1}(\cdot)$). This link function could e.g. be $g(\mu_i) = \log(\mu_i)$ for count data, or $g(\mu_i) = \log(\frac{\mu_i}{1-\mu_i})$ for binary data.

The second elementary assumption is the **distribution assumption**. It states, that the distribution of the response is a member of the exponential family.

$$\begin{aligned} p(y_i|\theta) &= \exp\left(\frac{y_i\theta(\mu_i) - b(\theta(\mu_i))}{\phi_i} - c(y_i,\phi_i)\right) \\ \text{giving} \qquad \mathbb{E}(y_i|\mathbf{x}_i) &= b'(\theta(\mu_i)) \qquad \text{and} \qquad \mathbb{V}\text{ar}(y_i|\mathbf{x}_i) = \phi_i b''(\theta(\mu_i)) \end{aligned} \tag{2.34}$$

Here, $\theta(\cdot)$ is called the canonical function and ϕ denotes the scale parameter of the distribution. $b(\cdot)$ and $c(\cdot)$ are functions of the respective parameters.

Some examples of exponential family distributions are

- the Binomial distribution ($\theta(\mu) = \log(\frac{\mu}{1-\mu})$, $\phi = 1$)
- the Poisson distribution ($\theta(\mu) = \log(\mu)$, $\phi = 1$)
- the Normal distribution ($\theta(\mu) = \mu$, $\phi = \sigma^2$)

The Generalized Linear Mixed Model

To enable an analogical modeling approach in the context of longitudinal or cluster data, it is necessary to extend the GLM. This particular data structure can be addressed, by adapting the linear predictor of the model using a regulating component - the random effects term.

$$\eta_{ij} = \mathbf{x}_{ij}^T\boldsymbol{\beta} + \mathbf{z}_{ij}^T\mathbf{b}_i \tag{2.35}$$

Altogether the GLMM can now be defined by introducing a third assumption.

$$\begin{gathered}\mathbb{E}(y_{ij}|\eta_{ij}) = h(\mathbf{x}_{ij}^T\boldsymbol{\beta} + \mathbf{z}_{ij}^T\mathbf{b}_i) \\ y_{ij}|\eta_{ij} \overset{\text{ind}}{\sim} [\text{exponential family}] \qquad ; \qquad \mathbf{b}_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \mathbf{D})\end{gathered} \tag{2.36}$$

This model can be understood as a combination of the GLM and the LMM. It is a flexible tool to model cross-sectional data and to additionally enable a broad spectrum of distributions for the dependent variable.

2.2.2. Example of a GLMM with Binary Response

In this thesis, artificial time-series cross-sectional data sets were analyzed. The response variable throughout this analyses was binary. Therefore, the concept of the GLMM is exemplified on an analogous data situation in this section.
To specify a GLMM, three assumptions have to be made: The distribution assumption for the response, the structural assumption and the distribution assumption for the random effects.

The **distribution assumption for $\mathbf{y}|\boldsymbol{\eta}$** having a binary response is

$$y_{ij}|\eta_{ij} \overset{\text{ind}}{\sim} \mathcal{B}(\pi_{ij}) \tag{2.37}$$

The Bernoulli distribution $\mathcal{B}(\pi)$ is a commonly known member of the exponential family. It's density can be rearranged to

$$p(y|\pi) = \pi^y(1-\pi)^{1-y} = \exp\left(\frac{y \cdot \log\left(\frac{\pi}{1-\pi}\right) - \log\left(1 + \exp\left(\log\left(\frac{\pi}{1-\pi}\right)\right)\right)}{1} + 0\right) \tag{2.38}$$

This way of formulating the density illustrates the components of formula 2.34.

$$\begin{array}{rcc} \theta(\mu) & = & \log\left(\frac{\pi}{1-\pi}\right) \\ b(\theta(\mu)) & = & \log(1+\exp(\theta(\mu))) \\ \phi & = & 1 \\ c(y,\phi) & = & 0 \\ \mathbb{E}(y) = b'(\theta(\mu)) & = & \pi \\ \mathbb{V}\mathrm{ar}(y) = \phi b''(\theta(\mu)) & = & \pi(1-\pi) \end{array} \tag{2.39}$$

The choice of the **structural assumption** depends on the type of response variable. A variety of different options exist for every member of the exponential family. A popular and universal choice is the canonical/natural link function. It is given by the exponential family as

$$g(\mu_{ij}) = \theta(\mu_{ij}) \tag{2.40}$$

The canonical link function of the Bernoulli distribution is the logistic function.

$$g(\mu_{ij}) = \log\left(\frac{\pi_{ij}}{1-\pi_{ij}}\right) \tag{2.41}$$

Other common options for a binary response variable would be:

- Probit link

  $$g(\mu_{ij}) = \Phi^{-1}(\mu_{ij})$$

  with Φ denoting the distribution function of the normal distribution

- Complementary log-log link

  $$g(\mu_{ij}) = \log(-\log(1-\mu_{ij}))$$

Nevertheless, a practical advantage of using the natural link function here is, that the regression coefficients are interpretable as an effect on the log-odds for $y_{ij} = 1$.

The **distribution assumption for the random effects** is the final assumption to be made. A typical choice here is the (multivariate) normal distribution. Yet, other options could be considered as well, like e.g. a (multivariate) t-distribution to create a more robust estimation. For this thesis however, the random effects $\mathbf{b}_i$ were simply assumed to be normal distributed.

A summary of the logistic GLMM defined in this section is given in 2.42.

$$y_{ij}|\eta_{ij} \overset{\text{ind}}{\sim} \mathcal{B}(\pi_{ij}) \quad ; \quad g(\mu_{ij}) = \log\left(\frac{\pi_{ij}}{1-\pi_{ij}}\right) \quad ; \quad \mathbf{b}_i \overset{\text{iid}}{\sim} \mathcal{N}_q(\mathbf{0}, \mathbf{D}) \tag{2.42}$$

As mentioned earlier, the fixed effects of this GLMM can be interpreted, if the natural link function of 2.41 is used. A rearranged model equation, which illustrates the relation of the odds for $y_{ij} = 1$ and the covariables is shown in 2.43.

$$\frac{\pi_{ij}}{1-\pi_{ij}} = \exp(\mathbf{x}_{ij}^T\boldsymbol{\beta}) \cdot \exp(\mathbf{z}_{ij}^T\mathbf{b}_i) \tag{2.43}$$

This rearrangement can be used to interpret the fixed effects of the model as an exponential factor influencing the odds $\frac{\pi_{ij}}{1-\pi_{ij}}$. It is important to note, that all interpretations of $\boldsymbol{\beta}$ are only valid given the random effects (e.g. "... for an average person ($\mathbf{b}_i = \mathbf{0}$) ..."). Contrary to the LMM, $\mathbb{E}(y_{ij}) = \mathbb{E}(\mathbb{E}(y_{ij}|b_i))$ does not hold for GLMMs in general. Marginal estimations of $\boldsymbol{\beta}$ are less or equal to the conditional ones, which are computed in the model. Hence, estimations of the fixed effects obtained from the GLMM can not be interpreted as parameters of the whole population, like in LMs, LMMs or GLMs, but should be treated as cluster-individual parameters.
If one is interested in a population based interpretation, the marginal parameters need to be computed separately.
The interpretation of both, the model parameters and the marginal parameters is valid, but addresses different issues. A parameter of the GLMM quantifies the expected change of the odds for a given individual, if a covariable is altered for this person, whereas the marginal parameter quantifies the expected change in the population, if the covariable

is altered for everyone.

The first case would e.g. be of interest for a doctor treating a given person, while the second case would rather apply to an epidemiologist studying the benefit of a treatment in the general population.

Probability curves of an exemplary GLMM with one covariable $\mathbf{x}$ are shown in figure 2.4. Comparing the curve of an average individual (red) with the curve of the population average (blue) illustrates the so called "shrinkage effect" on the marginal fixed effects.

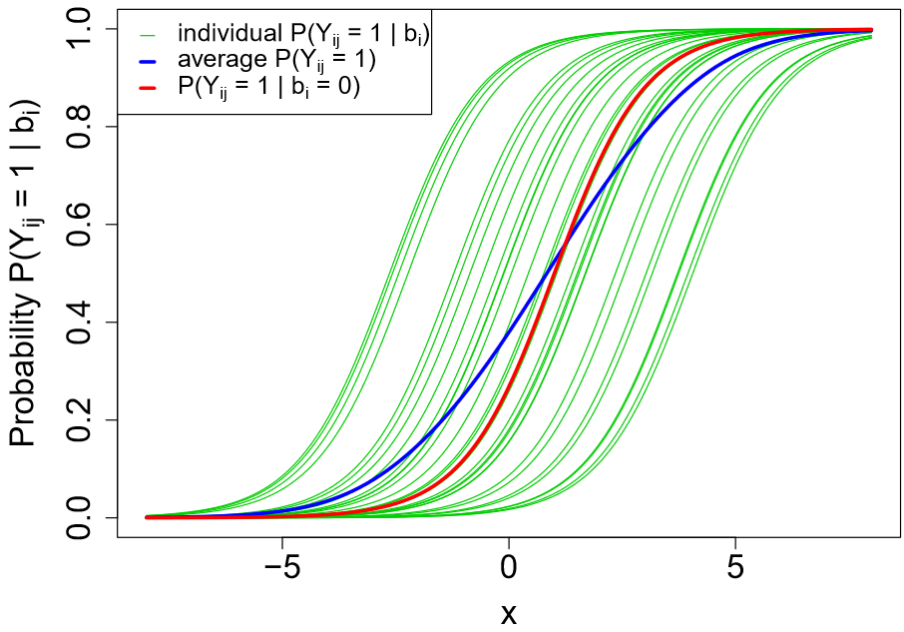


Figure 2.4.: Comparison of the individual probability curves, the overall population average probability curve and the probability curve for the average individual ($b_i = 0$) over a single continuous covariable $\mathbf{x}$.

Note: No general formula exists to convert marginal to conditional parameters for most GLMMs. The special case of a probit link random intercept GLMM is one exception to this. Here, it is possible to explicitly formulate the shrinkage effect.

$$\beta_{\text{marg.}} = \frac{\beta_{\text{cond.}}}{\sqrt{\tau^2 + 1}} \tag{2.44}$$

with τ^2 denoting the variance of the random intercept's normal distribution.

2.2.3. Estimation of the GLMM

Broadening the spectrum of distributions for the response variable results in additional difficulties for the estimation process. The main problem is, that contrary to the LMM, a closed form solution of the marginal likelihood cannot be obtained analytically in general.

Likelihood

Recapitulating formula 2.25, the marginal distribution of $\mathbf{y}_i$ is given by

$$p(\mathbf{y}_i|\boldsymbol{\theta}) = \int_{\mathbb{R}^q} \prod_{j=1}^{n_i} p(y_{ij}|\mathbf{b}_i, \boldsymbol{\theta}) p(\mathbf{b}_i|\mathbf{D}) \, \mathrm{d}\mathbf{b}_i \tag{2.45}$$

The marginal likelihood for all individuals therefore results as

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{m} \int_{\mathbb{R}^q} \prod_{j=1}^{n_i} p(y_{ij}|\mathbf{b}_i, \boldsymbol{\theta}) p(\mathbf{b}_i|\mathbf{D}) \, \mathrm{d}\mathbf{b}_i \tag{2.46}$$

For the LMM estimation of section 2.1.4, both densities in the integral arose from normal distributions, resulting in an analytical solution for the likelihood.
Here however, the distribution assumptions are

$$y_{ij}|\mathbf{b}_i, \boldsymbol{\theta} \overset{\text{ind}}{\sim} [\text{exponential family}] \qquad ; \qquad \mathbf{b}_i|\mathbf{D} \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{D}) \tag{2.47}$$

which gives no closed form solution for the likelihood in general.

Maximizing the product of m q-dimensional integrals over the random effects is difficult to perform and approximate solutions are still an area of active research.
Three main ideas can be considered to solve this problem:

- approximate the data (Penalized Quasi Likelihood approach)
- approximate the integrand (Laplace approximation)
- approximate the integral (Gaussian quadrature)

All three approaches are highly time-consuming. Nevertheless, the Laplace approximation is the computationally fastest of these algorithms. For the simulation experiments of chapter 4, a high number of models needed to be computed in a reasonable amount of time. Because of that, the Laplace approximation was chosen to perform the multitude of GLMM estimations in this thesis. It will be introduced in the following paragraph. Theory on the other approaches can be found e.g. in (Greven 2015) or (Greven and Cederbaum 2015).

The Laplace Approximation

Laplace's method is able to approximate the likelihood of 2.25 using a Taylor expansion. The general concept of this approach is described in the following.

Suppose one wants to approximate an expression of the form

$$\int f(x)\,\mathrm{d}x \tag{2.48}$$

First, a maximum value x_0 of $f(x)$ needs to be computed. I.e. a value x_0, for which $f'(x_0) = 0$. This value is employed as the centering point for a quadratic Taylor expansion of the logarithm of $f(x)$.

$$\log(f(x)) \overset{\text{Taylor}}{\approx} \log(f(x_0)) - \frac{s}{2}(x - x_0)^2 \tag{2.49}$$

with

$$s = -\left.\frac{\mathrm{d}^2}{\mathrm{d}x^2}\log f(x)\right|_{x=x_0}$$

Note that the first-order term of the Taylor expansion vanishes, because $f'(x_0) = 0$ and therefore $\left.\frac{\mathrm{d}}{\mathrm{d}x}\log f(x)\right|_{x=x_0} = 0$.

Taking the exponential of both sides of 2.49 results in

$$f(x) \approx f(x_0)\exp\left\{-\frac{(x - x_0)^2}{2s^{-1}}\right\} \tag{2.50}$$

The rear part of this equation represents the core of a normal distribution with mean x_0 and variance $\frac{1}{s}$. By adding the inverse normalization factor one obtains

$$f(x) \approx f(x_0) \cdot \sqrt{2\pi s^{-1}} \cdot \mathcal{N}(x|x_0, s^{-1}) \tag{2.51}$$

$\mathcal{N}(x|x_0, s^{-1})$ is called the Laplace approximation of the density given by a normalized $f(x)$. An example using a gamma density function is illustrated in figure 2.5.

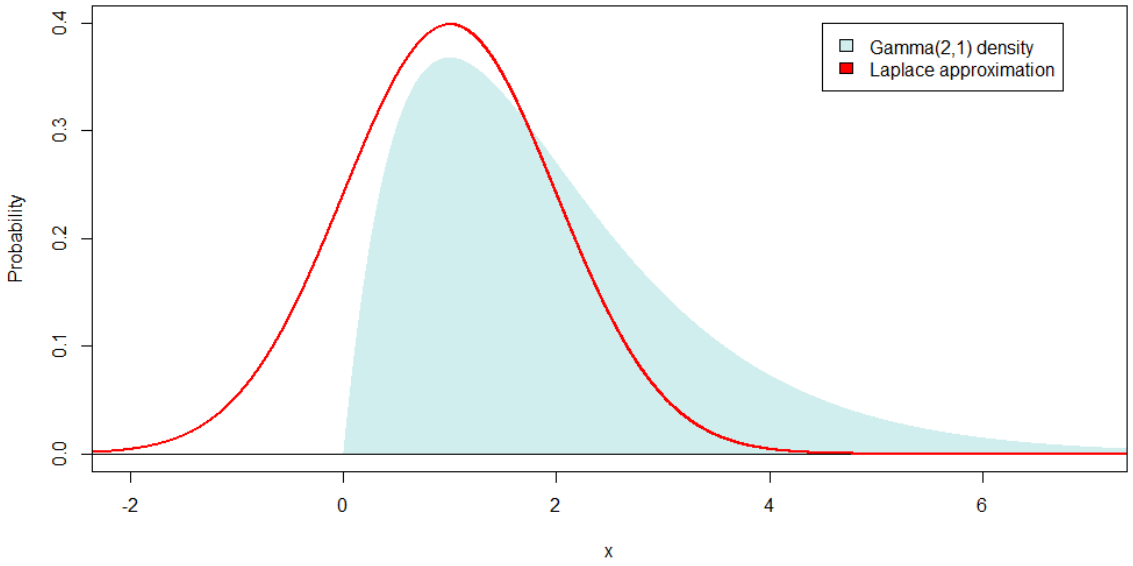


Figure 2.5.: Laplace approximation (red) of the gamma density with shape parameter 2 and scale parameter 1 (cyan). The illustrated gamma density is very skewed, which makes the Laplace approximation rather crude here.

The big advantage of approximating the function by means of Laplace's method is, that integration of $f(x)$ over x simplifies greatly. If 2.51 is plugged in 2.48 one obtains

$$\int f(x)\,\mathrm{d}x \overset{\text{Laplace}}{\approx} f(x_0)\sqrt{2\pi s^{-1}} \int \mathcal{N}(x|x_0, s^{-1})\,\mathrm{d}x = f(x_0)\sqrt{2\pi s^{-1}} \tag{2.52}$$

The only part of the approximated $f(x)$ depending on x is the normal density, which simply integrates to 1.

The concept of the Laplace approximation can be extended to a function over a q-dimensional space. $\mathbf{x}_0$ now represents a stationary point of the multi-dimensional function $f(\mathbf{x})$, for which the gradient will consequently vanish.

$$\nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0} \tag{2.53}$$

Performing a quadratic Taylor expansion of the logarithm of $f(\mathbf{x})$ around this point gives

$$\log(f(\mathbf{x})) \approx \log(f(\mathbf{x}_0)) - \frac{1}{2}(\mathbf{x}-\mathbf{x}_0)^T\mathbf{S}(\mathbf{x}-\mathbf{x}_0) \tag{2.54}$$

with a $q \times q$ Hessian matrix $\mathbf{S}$ defined by

$$\mathbf{S} = -\nabla\nabla \log f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0}$$

Taking the exponential of 2.54 results in

$$f(\mathbf{x}) \approx f(\mathbf{x}_0)\exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_0)^T\mathbf{S}(\mathbf{x}-\mathbf{x}_0)\right\} \tag{2.55}$$

which represents the core of a q-dimensional normal distribution. It can thus be obtained by multiplying the inverse normalization constant.

$$f(\mathbf{x}) \approx f(\mathbf{x}_0)\sqrt{\frac{(2\pi)^q}{|\mathbf{S}|}}\mathcal{N}_q(\mathbf{x}|\mathbf{x}_0, \mathbf{S}^{-1}) \tag{2.56}$$

where $|\mathbf{S}|$ denotes the determinant of $\mathbf{S}$.

Analogous to the univariate case, the approximation makes the integration of the function over $\mathbf{x}$ simple.

$$\int_{\mathbb{R}^q} f(\mathbf{x})\,\mathrm{d}\mathbf{x} \approx f(\mathbf{x}_0)\sqrt{\frac{(2\pi)^q}{|\mathbf{S}|}} \tag{2.57}$$

Returning to the estimation of the GLMM, the components for using the Laplace method can be identified easily.

$$\mathcal{L}_i(\boldsymbol{\theta}) = \int_{\mathbb{R}^q} p(\mathbf{y}_i|\mathbf{b}_i, \boldsymbol{\theta})p(\mathbf{b}_i|\mathbf{D})\,\mathrm{d}\mathbf{b}_i := \int_{\mathbb{R}^q} f(\mathbf{b}_i)\,\mathrm{d}\mathbf{b}_i \tag{2.58}$$

Applying the steps described in this section, the likelihood can be approximated to

$$\mathcal{L}_i(\boldsymbol{\theta}) = \mathcal{L}_i(\boldsymbol{\theta}, \tilde{\mathbf{b}}_i)\sqrt{\frac{(2\pi)^q}{|\mathbf{S}|}} \tag{2.59}$$

where $\tilde{\mathbf{b}}_i$ denotes the maximum of the likelihood for $\mathbf{b}_i$.

$$\mathbf{S} = -\frac{\partial^2}{\partial\mathbf{b}_i\partial\mathbf{b}_i^T}\log(\mathcal{L}_i(\boldsymbol{\theta}))\bigg|_{\mathbf{b}_i=\tilde{\mathbf{b}}_i} = -\frac{\partial}{\partial\mathbf{b}_i^T}\mathrm{sc}(\mathbf{b}_i)\bigg|_{\mathbf{b}_i=\tilde{\mathbf{b}}_i} = -\mathcal{F}(\tilde{\mathbf{b}}_i) \tag{2.60}$$

with $\mathrm{sc}(\cdot)$ denoting the score function and $\mathcal{F}(\cdot)$ denoting the observed Fisher information. However, to approximate the likelihood with Laplace's method here, one first has to estimate the maximum value $\tilde{\mathbf{b}}_i$. This can e.g. be done using the P-IRLS Algorithm.

The P-IRLS Algorithm

The optimization of the likelihood with respect to $\mathbf{b}$ can be performed using a Penalized Iteratively Re-weighted Least Squares algorithm (P-IRLS). The P-IRLS approach is described in this section.

Analogous to the numerical optimization algorithm of the GLM, the system of score equations for $\mathbf{b}$ is approximated by a linear Taylor expansion around $\mathbf{b}_0$.

$$\text{sc}(\mathbf{b}) \overset{\text{Taylor}}{\approx} \text{sc}(\mathbf{b}_0) - \mathcal{F}(\mathbf{b}_0)(\mathbf{b} - \mathbf{b}_0) \overset{!}{=} 0 \tag{2.61}$$

Using the expected Fisher information matrix $\mathcal{I}(\mathbf{b}_0)$ ("Fisher-Scoring") instead of the observed Information matrix $\mathcal{F}(\mathbf{b}_0)$ and rearranging the rear equation of 2.61 gives

$$\mathcal{I}(\mathbf{b}_0)\mathbf{b} = \mathcal{I}(\mathbf{b}_0)\mathbf{b}_0 + \text{sc}(\mathbf{b}_0) \tag{2.62}$$

This is the main equation for the P-IRLS algorithm. Exemplified on a GLMM with canonical link function, the score function and expected Fisher information can be written as

$$\begin{aligned} \text{sc}(\mathbf{b}) &= \frac{\partial}{\partial \mathbf{b}}\ell(\boldsymbol{\theta}) = \frac{1}{\phi}\mathbf{Z}^T(\mathbf{y} - \boldsymbol{\mu}) - \mathcal{D}^{-1}\mathbf{b} \\ \mathcal{I}(\mathbf{b}) &= -\mathbb{E}\left(\frac{\partial^2}{\partial \mathbf{b}\partial \mathbf{b}^T}\ell(\boldsymbol{\theta})\right) = \mathbf{Z}^T\mathbf{W}\mathbf{Z} + \mathcal{D}^{-1} \end{aligned} \tag{2.63}$$

with $\mathbf{W} = \frac{1}{\phi}\text{diag}\left(\frac{\partial^2}{\partial \mathbf{b}\partial \mathbf{b}^T}b(\boldsymbol{\theta})\right)$, where $b(\cdot)$ referring to the function given in the definition of the exponential family (see equation 2.34)

Plugging these solutions in 2.62 gives

$$\begin{aligned} &\mathcal{I}(\mathbf{b}_0)\mathbf{b} = (\mathbf{Z}^T\mathbf{W}\mathbf{Z} + \mathcal{D}^{-1})\mathbf{b}_0 + \frac{1}{\phi}\mathbf{Z}^T(\mathbf{y} - \boldsymbol{\mu}) - \mathcal{D}^{-1}\mathbf{b}_0 \quad \Leftrightarrow \\ &\mathcal{I}(\mathbf{b}_0)\mathbf{b} = \mathbf{Z}^T\mathbf{W}\mathbf{Z}\mathbf{b}_0 + \frac{1}{\phi}\mathbf{Z}^T(\mathbf{y} - \boldsymbol{\mu}) \quad \Leftrightarrow \\ &\mathcal{I}(\mathbf{b}_0)\mathbf{b} = \mathbf{Z}^T\mathbf{W}(\mathbf{Z}\mathbf{b}_0 + \frac{1}{\phi}\mathbf{W}^{-1}(\mathbf{y} - \boldsymbol{\mu})) \quad \Leftrightarrow \\ &\mathcal{I}(\mathbf{b}_0)\mathbf{b} = \mathbf{Z}^T\mathbf{W}\tilde{\mathbf{y}} \end{aligned} \tag{2.64}$$

with a "working response" $\tilde{\mathbf{y}}$ defined as $\tilde{\mathbf{y}} = \mathbf{Z}\mathbf{b}_0 + \frac{1}{\phi}\mathbf{W}^{-1}(\mathbf{y} - \boldsymbol{\mu})$.

An estimation algorithm can now be formulated as follows.

Algorithm 1: P-IRLS for GLMMs

Set starting values $\tilde{\mathbf{b}}^{[0]}$.

Iterate until convergence:

- Compute working responses

$$\tilde{\mathbf{y}}^{[u]} = \mathbf{Z}\tilde{\mathbf{b}}^{[u]} + \frac{1}{\phi}\mathbf{W}^{-1}(\mathbf{y} - \boldsymbol{\mu})$$

- Update $\tilde{\mathbf{b}}$ with

$$\tilde{\mathbf{b}}^{[u+1]} = \mathcal{I}(\tilde{\mathbf{b}}^{[u]})^{-1}\mathbf{Z}^T\mathbf{W}\tilde{\mathbf{y}}^{[u]}$$

Note: The dependencies of $\mathbf{W}$ and $\boldsymbol{\mu}$ on $\tilde{\mathbf{b}}$ were not highlighted in this algorithm for ease of exposition.

The Estimation Algorithm for $\boldsymbol{\theta}$

The modes $\tilde{\mathbf{b}}_i$ used in the Laplace approximation depend on the unknown parameters in $\boldsymbol{\theta}$. Therefore, the numerical maximization of the likelihood has to iterate between estimating $\tilde{\mathbf{b}}$ using $\hat{\boldsymbol{\theta}}$ and maximizing the likelihood after inserting $\tilde{\mathbf{b}}$ to obtain updated estimations for $\boldsymbol{\theta}$.

Algorithm 2: Estimation of θ using the Laplace Approximation

Set starting values $\hat{\boldsymbol{\theta}}^{[0]}$.

Iterate until convergence:

- Compute the maximizer $\hat{\mathbf{b}}^{[u]}$ with a P-IRLS approach (described in Algorithm 1) using $\hat{\boldsymbol{\theta}}^{[u]}$
- Use $\hat{\mathbf{b}}^{[u]}$ to optimize the likelihood approximated with the Laplace method to compute $\hat{\boldsymbol{\theta}}^{[u+1]}$

On the one hand, the estimation of GLMM parameters via the Laplace approximation has the advantage of being computationally fast compared to the other common approaches. On the other hand, it is also more imprecise than the more CPU-intensive methods. This especially applies for small cluster sizes and strong discreteness among the clusters.
However, to practically enable the computation of the vast amount of models necessary for the simulations of chapter 4, the lower precision associated with the Laplace approximation needed to be accepted here.

2.2.4. GLMMs in `R` using the package `lme4`

This section gives a short introduction, on how to estimate GLMMs in `R`. The package `lme4` (Bates 2010) was used for this purpose throughout the thesis. It is one of the standard packages for mixed model estimation in `R`.

The Function `glmer()`

Suppose, one is interested in fitting a GLMM for the data set shown in tabular 2.1, having clusters i given by the variable `subject`.

subject	y	time	age	x
1	1	1	37	1
1	1	2	37	1
1	0	3	37	0
1	1	4	37	0
1	1	5	37	1
2	1	1	32	1
2	1	2	32	0

Table 2.1.: Excerpt from an exemplary TSCS data set.

An applicable model equation to this data set would e.g. be

$$\mathbb{E}(y_{ij}|\eta_{ij}) = h(\beta_0 + \texttt{time}_{ij}\beta_1 + \texttt{age}_i\beta_2 + \texttt{x}_{ij}\beta_3 + b_i)$$

The corresponding function call in R reads

```
1 glmer(formula = y ~ time + age + x + (1|subject),
2       family = binomial(link="logit"),
3       data = D)
```

The formula parameter denotes a formula statement, comparable to the one used in the `lm()` or `glm()` functions. First, the response variable is indicated, followed by a tilde operator and the fixed effects separated with "+" operators. After another "+" and within round brackets, the random effects are specified analogously to the fixed effects followed by a "|" operator and the cluster variable. In this formula, the random effects only consist of an intercept to create a RIM. If one wants to extend this model to a RSM, the corresponding formula argument would need to be

$$\texttt{y} \sim \texttt{time + age + x + (time|subject)}$$

Note, that the random intercept is included by default and does not need to be declared explicitly as soon as one or more random effect variable is declared. If the intercept should be removed, the expression "`-1`" has to be added to the statement.

The second argument is named family. It specifies the distribution and structural assumption of the GLMM. The first part declares the distribution of the response. Here, y is binary, which leads to the Bernoulli, respectively the Binomial distribution. The link function is specified as argument of the distribution expression. It can be defined either by keywords like "logit" or "probit", or explicitly as an R function.

The third argument determines the data set, which incorporates the used variables.
A variety of further optional parameters are available in the glmer function. They can be found in Bates 2015. The described arguments however provide the basis for the computation of the GLMMs in this thesis.

Output of the Estimation with `glmer()`

Calling the `glmer()` function with the arguments presented in the last section leads to the following (shortened) output:

```
1  Generalized linear mixed model fit by maximum likelihood
2     (Laplace Approximation) ['glmerMod']
3
4   Family: binomial  ( logit )
5  Formula: y ~ time + age + x + (1 | subject)
6     Data: D
7
8  Random effects:
9   Groups  Name        Std.Dev.
10  subject (Intercept) 0.9005
11
12 Fixed Effects:
13  (Intercept)          time           age             x
14     -1.02054       0.09962       0.04344      -1.14988
```

The output starts with a caption naming the applied estimation algorithm, which is by default the Laplace approximation. After recapitulating the specified arguments of the function call, the estimation of the variance covariance matrix for the random effects is displayed. In this case, it is a single value denoting the standard deviation of the random intercept ($d \approx 0.9$). The fixed effects estimations can be found below.
For a more detailed output (p-values, correlation matrix and variances of the fixed effects, ...), one can use the `summary()` function having the model object as its argument.

The CPU-time required to estimate this random intercept GLMM with the Laplace approximation having a data set of 100 clusters with 5 measurements each lies in the region of 0.6 seconds using an "AMD-FX8320" CPU.

2.3. Bootstrapping from Mixed Model Data

The simulation analyses of chapter 4 used a bootstrap approach to obtain empirical densities for GLMM parameters. Bootstrapping is a computational resampling method. For a data set comprising N observations, the method independently draws N times a random data row, which in combination gives a sample data set of the same size as the original one. Observations of the data can occur in the sample either one time, multiple times, or not at all. In this way, B sample data sets are created. Next, a GLMM is fitted on each of the bootstrap samples. Every model parameter is thus obtained B times, resulting in an empirical density for it.

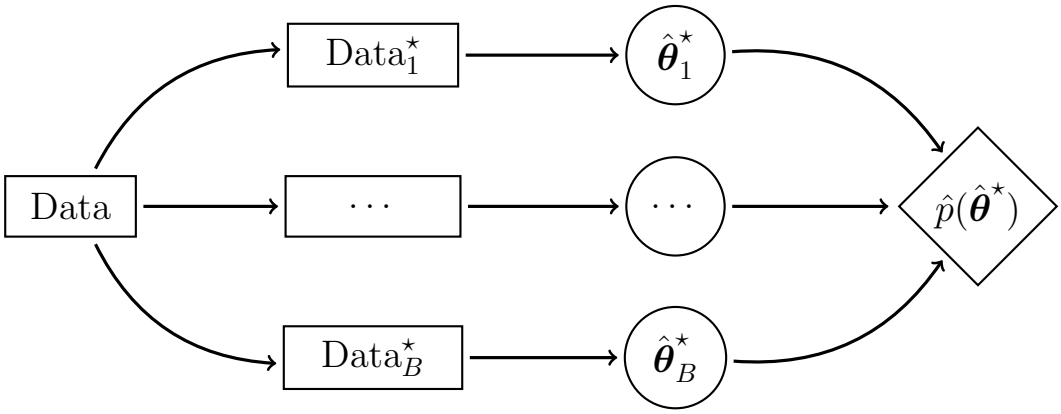


Figure 2.6.: Schematic illustration of realizing an empirical bootstrap density of GLMM estimators. The asterisk represents a sampled data set, $\boldsymbol{\theta}$ denotes the parameters of a GLMM and $\hat{p}(\cdot)$ refers to the empirical density of it's argument.

Having obtained multiple estimates of $\boldsymbol{\theta}$ and therefore the empirical bootstrap density, one can also compute 95% bootstrap percentile intervals.

$$BPI = \left[\; \hat{\boldsymbol{\theta}}^\star_{(0.025)} \; ; \; \hat{\boldsymbol{\theta}}^\star_{(0.975)} \; \right] \tag{2.65}$$

with $\hat{\boldsymbol{\theta}}^\star_{(\frac{\alpha}{2})}$ denoting the $(B \cdot \frac{\alpha}{2})$-st value[4] of the ordered sequence of $\hat{\boldsymbol{\theta}}^\star_1, \ldots, \hat{\boldsymbol{\theta}}^\star_B$.

[4]resp. if $(B \cdot \frac{\alpha}{2})$ is a non-integer value, choose the $(\lfloor (B+1) \cdot \frac{\alpha}{2} \rfloor)$-st and the $(B+1-\lfloor (B+1) \cdot \frac{\alpha}{2} \rfloor)$-st values as borders of the BPI

One essential prerequisite of the bootstrap is, that the units, which are drawn from the data set to obtain a bootstrap sample are independent. However, having a cross sectional data set, this does not account for the observations of one individual. The bootstrap therefore has to be adapted to handle the data structure.
Measurements of one person may not be independent, yet, the persons themselves actually are. Hence, one solution to apply the bootstrap method on cross-sectional data is to draw cluster-wise from the data set. One bootstrap sample can then contain all measurements from one individual either once, multiple times, or not at all. This approach preserves the structure of the dependent measurements and creates a proper bootstrap sample of the data situation.

An implementation in `R` is given below. The parameters of the function are a balanced, cross-sectional data set and the number of bootstrap samples to compute.

```
1 bootstrap <- function(data, B=200){
2   pers = unique(data$subject)
3   lapply( 1:B, function(y){
4     smp = do.call(rbind, lapply(sample(pers, replace=T),
5                           function(x)data[data$subject==x,]))
6     smp$subject = rep(pers, each=nrow(data)/length(pers))
7     return(smp)} )
8 }
```

First, the names of all cross-sections are stored in the variable `pers`. The call `lapply` returns the resulting sampled data sets of it's inner function as a list. One bootstrap sample is created by repeatedly drawing all measurements of a random person. The code `do.call(rbind, lapply(...)` stores all drawn persons to a list, which gets then combined to a data set. Finally, before returning it, distinct names have to be given to each drawn person to preserve a balanced structure.

3. Imputation of Missing Data with Amelia II

Missing values are a common issue in a lot of practical data situations. Participants of studies drop out, devices fail to measure values (resp. measure highly unplausible values), or questions of a survey are not answered; there are many reasons that lead to incomplete data. Nevertheless, most common methods of statistical analysis require rectangular data sets without missing observations.
One approach, which is widely used among researchers to handle this problem is listwise deletion. It eliminates all data rows holding one or more missing values. However, beside discarding a considerable amount of information, this method may also induce biased parameter estimates, if the data contains information on the missingness.

An alternative strategy to omit the disadvantages of listwise deletion is to impute missing values. I.e.: Estimate a plausible value for the missing data point from the observed measurements. However, if one is not able to replace the missing values with the true ones, the analysis software falsely assumes, that the data has more observation, than were actually measured. This leads to an overestimated confidence in the results, which arises from biased standard errors, hence biased confidence intervals.
One way to address this problem is to impute multiple values for a single unobserved one. Consequently the uncertainty of the imputation can be taken into account, by including the discrepancy between the imputed values in the final estimation. The expected value, or "best guess" of a missing observation would then be the mean of the

imputations across the completed data sets.

In this thesis, multiple imputation via Amelia II was used to perform this task. The package is named after a famous missing person (see 3.1). Compared to other common algorithms for multiple imputation, like the imputation-posterior approach, or the expectation maximization importance sampling, Amelia II has the advantage of being far less computational intensive. Furthermore, it includes adaptions to deal with TSCS data, which altogether makes it well suited for the applications of this thesis.

The chapter starts with a description of the missingness assumptions typically made for incomplete data. Section 3.2 introduces the basic concepts of multiple imputation. Subsequently, the imputation model used by Amelia II is presented. Section 3.4 concerns the EMB-Algorithm and practical computation of imputations. After that, the adaptions to deal with TSCS data are presented in section 3.5. The last part of this chapter describes the practical application of Amelia II in `R`.

Figure 3.1.: The `R` package Amelia was named after Amelia Earhart, who was an American aviation pioneer and author. During a flight in 1937, she disappeared over the central Pacific Ocean.

3.1. Missingness Assumptions

The process, by which data becomes missing can be divided into three cases, which are summarized by the following assumptions. The applicability of statistical methods strongly depends upon them.
Let $\mathfrak{D}$ be a matrix comprising all dependent and independent variables.

$$\mathfrak{D} = \{\mathbf{Y}, \mathbf{X}\} \tag{3.1}$$

This data matrix can be divided into observed and missing portions.

$$\mathfrak{D} = \{\mathfrak{D}_{\text{obs}}, \mathfrak{D}_{\text{mis}}\} \tag{3.2}$$

Let $\mathcal{M}$ be a matrix of the same dimension indicating, if an element of $\mathfrak{D}$ is missing.

$$\mathcal{M}_{ik} = 1 \Leftrightarrow \mathfrak{D}_{ik} \in \mathfrak{D}_{\text{mis}} \tag{3.3}$$

The first possible assumption is, that missing values are **"Missing Completely At Random" (MCAR)**. This means the probability of a cell being not observed does not depend on the data. In other words, the data contains no information about the missingness. MCAR is an important requirement for the application of listwise deletion, to not induce biased estimators.
Formally, the assumption can be written as

$$p(\mathcal{M}|\mathfrak{D}) = p(\mathcal{M}) \tag{3.4}$$

The second assumption is the less restrictive **"Missing At Random" (MAR)**. It allows the missingness of a value to be dependent on observed, but not on unobserved values. This assumption is e.g. made for the multiple imputation of Amelia II.

$$p(\mathcal{M}|\mathfrak{D}) = p(\mathcal{M}|\mathfrak{D}_{\text{obs}}) \tag{3.5}$$

The last assumption allows no simplification of $p(\mathcal{M}|\mathfrak{D})$. The probability of a value being missing can depend on observed, as well as on unobserved data. This assumption is called **"Not Missing At Random" (NMAR)**. By definition, NMAR cannot be proven from the observed data, which makes it impossible in practical data situation to absolutely verify the validity of Amelia's multiple imputation model. Nevertheless, MCAR can often be empirically discarded in favor of MAR.
Altogether, these assumptions are crucial for the validity of any approach to treat incomplete data and should be applied with careful thought.

3.2. Multiple Imputation

Contrary to imputing a missing data point once, the idea of multiple imputation is to impute M values for each missing cell, therefore creating M completed data sets. The multiple values of each unobserved cell reflect the uncertainty of the imputation. Observed values stay the same for all data sets. The analyst can then apply any statistical method individually on each imputed data set and subsequently combine the results with a simple procedure. Figure 3.2 schematizes a multiple imputation analysis.

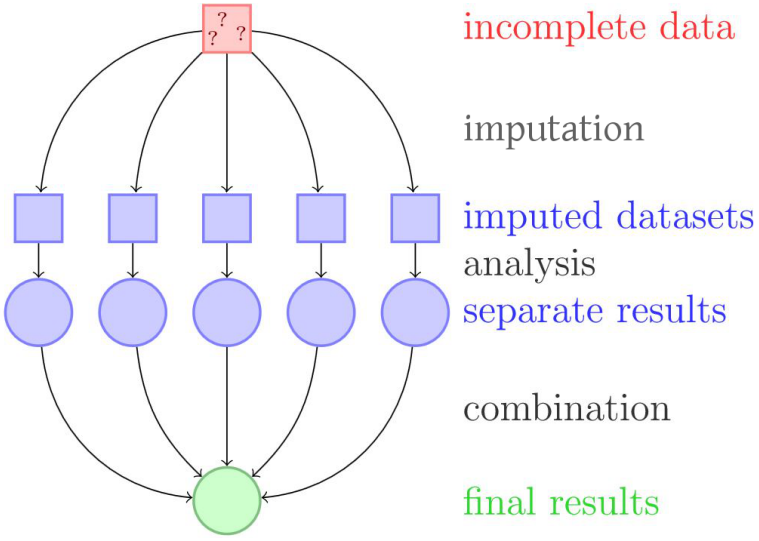


Figure 3.2.: Scheme of a multiple imputation analysis approach.

Combination Methods

The quantity of interest Q is obtained from each imputed data set: $Q_1, \ldots, Q_M$.
Q can e.g be a predicted probability, a univariate mean, or like in this thesis a regression coefficient of a generalized linear mixed model. The combined estimate for Q is most typically the mean of all separate estimates Q_j $(j = 1, \ldots, M)$.

$$\bar{Q} = \frac{1}{M} \sum_{j=1}^{M} Q_j \tag{3.6}$$

Multiple imputation enables to express the uncertainty of estimating missing values. Obtained variances of Q have to be altered, to include this additional uncertainty (cf. King et al. 2001). Let $\widehat{\mathbb{V}\mathrm{ar}}(Q_j)$ denote the estimated variance of Q from data set j. This can be seen as the variance component, that lies "within" each data set. In addition to this exists the uncertainty of the imputation, which is projected "between" the data sets. It can be calculated by the sample variance across the M data sets multiplied by $\frac{M+1}{M}$ to correct for bias.
The overall estimate for the variance of Q then follows as

$$\widehat{\overline{\mathbb{V}\mathrm{ar}}}(Q) = \underbrace{\frac{1}{M} \sum_{j=1}^{M} \widehat{\mathbb{V}\mathrm{ar}}(Q_j)}_{\text{"within"}} + \underbrace{\frac{M+1}{M(M-1)} \sum_{j=1}^{M} (Q_j - \bar{Q})^2}_{\text{"between"}} \tag{3.7}$$

For multivariate $\mathbf{Q}$, equation 3.7 reads

$$\widehat{\overline{\mathbb{C}\mathrm{ov}}}(\mathbf{Q}) = \frac{1}{M} \sum_{j=1}^{M} \widehat{\mathbb{C}\mathrm{ov}}(\mathbf{Q}_j) + \frac{M+1}{M(M-1)} \sum_{j=1}^{M} (\mathbf{Q}_j - \bar{\mathbf{Q}})(\mathbf{Q}_j - \bar{\mathbf{Q}})^T \tag{3.8}$$

Finally, if one is only interested in estimating the best guess for the missing values, the imputed data sets can also be combined by calculating the mean.

$$\bar{\mathfrak{D}} = \frac{1}{M} \sum_{j=1}^{M} \mathfrak{D}_j \tag{3.9}$$

Of course, when averaging over the data sets, it's only necessary to average the imputed values and simply carry over observed data points.

A popular mistake done by researchers is to use the averaged data set $\bar{\mathfrak{D}}$ to calculate the quantity of interest. The problem with this way of proceeding is, that the variance of Q will be underestimated, due to ignoring the variance between the imputed data sets, which represents the uncertainty of the imputation process. Imputed cells are seen as observed values by analysis software and all information gained on the imputation variance is discarded, which makes multiple imputation pointless. Moreover, in Amelia II, unnecessary additional estimation uncertainty is added, because the original data is bootstrapped internally.

Altogether, understanding the idea of multiple imputation and the correct way of analyzing the obtained data situation is essential to exploit the advantages of imputing more than one value for a missing data point.

3.3. The Imputation Model

To implement multiple imputation, a model, which estimates missing values is required. The modeling approach of Amelia II will be introduced in this section.

Observed values oftentimes carry information about the unobserved ones. This makes the distribution of the missing values predictable by existing data. A main assumption, to enable an unbiased estimation in this context is MAR (see 3.5). The second assumption made in Amelia II is, that the data variables are jointly multivariate normal distributed. This obviously is an approximation, because only few data sets have e.g. only continuous, or unbounded variables. Yet research has discovered, that this approximation works as well as more complicated approaches, which were adapted specifically for those data types. (Schafer 1997)

Let $\mathfrak{D}_i$ be the vector of values for observation i ($i = 1, \ldots, N$). If $\mathfrak{D}_i$ was fully observed, this vector is assumed to be multivariate Normal distributed with mean vector $\boldsymbol{\mu}$ and a non-diagonal covariance matrix $\boldsymbol{\Sigma}$.

The likelihood function for the complete data then is

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathfrak{D}) \propto \prod_{i=1}^{N} \mathcal{N}(\mathfrak{D}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.10}$$

Now let $\mathfrak{D}_{i,\text{obs}}$ denote the observed elements for row i of the data $\mathfrak{D}$, with $\boldsymbol{\mu}_{i,\text{obs}}$ and $\boldsymbol{\Sigma}_{i,\text{obs}}$ being the corresponding subvector and submatrix of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. The likelihood of the observed data is obtained by integrating the full data likelihood over $\mathfrak{D}_{i,\text{mis}}$.

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathfrak{D}_{\text{obs}}) \propto \int \mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathfrak{D})\, \mathrm{d}\mathfrak{D}_{i,\text{mis}} \propto \prod_{i=1}^{N} \mathcal{N}(\mathfrak{D}_{i,\text{obs}}|\boldsymbol{\mu}_{i,\text{obs}}, \boldsymbol{\Sigma}_{i,\text{obs}}) \tag{3.11}$$

The difficulty of this likelihood is, that the observed values typically have a changing compositions over i, thus contributing information to different portions of the parameters. This makes optimization complex to implement. In other words, for p variables in the vector $\mathfrak{D}_i$, there are 2^p possible missingness patterns, which may need to be combined for the final parameter estimation.

Amelia II imputes missing values by means of a linear regression model.
Let $\hat{\mathfrak{D}}_{ik}$ be the imputation estimation for a missing value of observation i and variable k. Further, let $\mathfrak{D}_{i,\text{obs}}$ be the vector of all non missing values of observation i. $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ contain all available information on the data. They can be used to calculate the regression coefficient $\hat{\boldsymbol{\beta}}$.
The imputation is then performed similar to

$$\hat{\mathfrak{D}}_{ij} = \mathfrak{D}_{i,\text{obs}}\hat{\boldsymbol{\beta}} + \hat{\epsilon}_i \tag{3.12}$$

The expected imputation is therefore a linear function of the observed data, with added uncertainty ϵ_i calculated from the covariance matrix $\boldsymbol{\Sigma}$.
For the actual estimation algorithm in detail see section 3.4.

If a categorical variable is to be imputed, the continuous estimation $\hat{\mathfrak{D}}_{ij}$ can be rounded off to the nearest integer (as recommended by Schafer 1997). However, if the variable is ordinal, one could also consider analyzing the data with non-integer imputations, to get more precise conclusions. For a binary variable, a Bernoulli draw with probability π being equal to $\hat{\mathfrak{D}}_{ij}$ and truncated to $[0, 1]$ is a possible solution. But again, if the analysis allows it, using the continuous value $\hat{\mathfrak{D}}_{ij}$ should be taken into consideration.

3.4. The Estimation Algorithm

The multiple imputation of missing values is performed by a bootstrap in combination with an EM algorithm. A detailed description of the practical estimation process will be presented in this section. To fully understand the approach, the sweep operator and the concept of sufficient statistics are introduced first.

3.4.1. The Sweep operator $\mathcal{S}wp(\mathbf{X}, \{s\})$

Sweeping is an operation performed on a quadratic matrix. This thesis uses the definition of the sweep operator having two parameters. The first one is a matrix of dimension $r \times r$ to perform the operation on. The second is a binary vector of length r, indicating the row and column to be swept over.

Let the sweep operation for the $r \times r$ matrix $\mathbf{A}$ over the k-th row/column result in a matrix $\mathbf{B}$. I.e.

$$\mathcal{S}wp(\mathbf{A}, \{\mathbf{k}\}) = \mathbf{B} \quad ,$$

where $\mathbf{k}$ denotes a vector of length r, with the k-th element being 1 and all other elements being 0.

The resulting matrix $\mathbf{B}$ is then composed of

$$\begin{aligned} B_{ij} &= A_{ij} - \frac{A_{ik}A_{kj}}{A_{kk}} \qquad ; \qquad B_{kk} = \frac{1}{A_{kk}} \\ B_{ik} &= \frac{A_{ik}}{A_{kk}} \qquad\qquad\quad ; \qquad B_{kj} = \frac{A_{kj}}{A_{kk}} \end{aligned} \tag{3.13}$$

Though at first sight, the aim of the sweep operator may not be very obvious, it has multiple fields of application. One of them is the ordinary least squares method of (multivariate) regression problems, which will be introduced in the following.

Let $\mathfrak{D}$ be a design matrix comprising a column of ones, covariables $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_5$ and responses $\mathbf{x}_1$ and $\mathbf{x}_4$:

$$\mathfrak{D} = \{ \ \mathbf{1} \quad \mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \mathbf{x}_4 \quad \mathbf{x}_5 \ \} \tag{3.14}$$

One is now interested in computing the parameters of the linear regression models

$$x_1 = \beta_0^{(1)} + x_2\beta_2^{(1)} + x_3\beta_3^{(1)} + x_5\beta_5^{(1)} + \epsilon_{(1)} \qquad \text{(Model 1)}$$

$$x_4 = \beta_0^{(2)} + x_2\beta_2^{(2)} + x_3\beta_3^{(2)} + x_5\beta_5^{(2)} + \epsilon_{(2)} \qquad \text{(Model 2)}$$

A computational efficient way to do this, without having to invert any matrices is feasible with the sweep operator. The sufficient statistic $\mathbf{T} = \mathfrak{D}^T\mathfrak{D}$ (see section 3.4.2) is swept consecutively over all non-response columns $\mathbf{c} = (1, 0, 1, 1, 0, 1)$, giving

$$\mathcal{S}wp(\mathbf{T}, \{\mathbf{c}\}) = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{1} & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 & \mathbf{x}_5 \end{array} \\ \begin{pmatrix} \cdot & \beta_0^{(1)} & \cdot & \cdot & \beta_0^{(2)} & \cdot \\ \cdot & \sum_i \epsilon_{(1)}^2 & \cdot & \cdot & \mathbb{C}\mathrm{ov}(x_1, x_4) & \cdot \\ \cdot & \beta_2^{(1)} & \cdot & \cdot & \beta_2^{(2)} & \cdot \\ \cdot & \beta_3^{(1)} & \cdot & \cdot & \beta_3^{(2)} & \cdot \\ \cdot & \mathbb{C}\mathrm{ov}(x_1, x_4) & \cdot & \cdot & \sum_i \epsilon_{(2)}^2 & \cdot \\ \cdot & \beta_5^{(1)} & \cdot & \cdot & \beta_5^{(2)} & \cdot \end{pmatrix} \end{array} \tag{3.15}$$

Here, the dots represent portions of the matrix no longer of use for this example.
All parameters of interest for the regression problem can be extracted from this matrix.

3.4.2. Sufficient Statistics

A statistic $\mathbf{T}(\mathbf{X})$ is sufficient for the parameters $\boldsymbol{\theta}$ of the distribution of $\mathbf{X}$, if it contains all information on $\boldsymbol{\theta}$ available from $\mathbf{X}$.
More formally:

$$f_{X|T}(\mathbf{X}|\mathbf{T}(\mathbf{X}) = \mathbf{T}, \boldsymbol{\theta}) = f_{X|T}(\mathbf{X}|\mathbf{T}(\mathbf{X}) = \mathbf{T}) \tag{3.16}$$

Like mentioned in an earlier section, the data set, which is imputed by Amelia is assumed to be multivariate normal distributed. Hence, the focus here lies on applications for this distribution.

A sufficient statistic for $\mathfrak{D}$ can be summarized as

$$\mathbf{T} = \mathfrak{D}^T\mathfrak{D} = \begin{pmatrix} N & \sum_i x_{i1} & \dots & \sum_i x_{i5} \\ \sum_i x_{i1} & \sum_i x_{i1}^2 & \dots & \sum_i x_{i1}x_{i5} \\ \vdots & & \ddots & \\ \sum_i x_{i5} & \dots & & \sum_i x_{i5}^2 \end{pmatrix} \tag{3.17}$$

When transforming this matrix by means of the sweep operator, the parameter estimates of the underlying distribution can be obtained. $\mathbf{T}$ is swept over the first row/column resulting in

$$\mathcal{S}wp(\mathbf{T}, \{1\ 0\ 0\ 0\ 0\ 0\}) = \begin{pmatrix} N^{-1} & \hat{\boldsymbol{\mu}}^T \\ \hat{\boldsymbol{\mu}} & \hat{\boldsymbol{\Sigma}} \end{pmatrix} \tag{3.18}$$

This is the most common way to express the sufficient statistic of a multivariate normal distribution, since all parameters of $\mathfrak{D}_{i,-1} \sim \mathcal{N}_5(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are found in this version of $\mathbf{T}$. (Here $\mathfrak{D}_{i,-1}$ denotes a row of the data set without the first column)

3.4.3. The ML and Bayesian Estimation Approach

Two approaches exist within Amelia II to estimate imputations for incomplete data. If expert knowledge on missing values exists, a Bayesian framework can be applied. Otherwise, the likelihood is maximized solely based on the data set. Both approaches will be described simultaneously in this section, as they pursue a similar concept.

The assumed distribution for an individual observation $\mathfrak{D}_{ij}$ is

$$\hat{\mathfrak{D}}_{ij}|\mathbf{T} \sim \mathcal{N}\left(\ \mu_{ij} = \mathbb{E}(\mathfrak{D}_{ij})\ ,\ \sigma_{ij}^2 = \hat{\sigma}_{ij}^2\ \right) \tag{3.19}$$

Now let a potential prior for this value with $\mathcal{M}_{ij} = 1$ be defined as

$$p(\mathfrak{D}_{ij}) = \mathcal{N}(m_{ij}, \lambda_{ij}^2) \tag{3.20}$$

The corresponding posterior distribution then results as

$$\hat{\mathfrak{D}}_{ij}|m_{ij},\lambda_{ij},\mathbf{T} \sim \mathcal{N}\left(\ \mu^*_{ij} = \frac{m_{ij}\lambda_{ij}^{-2} + \mathbb{E}(\mathfrak{D}_{ij})\hat{\sigma}_{ij}^{-2}}{\lambda_{ij}^{-2} + \hat{\sigma}_{ij}^{-2}}\ ,\ \sigma^{2*}_{ij} = \frac{1}{\lambda_{ij}^{-2} + \hat{\sigma}_{ij}^{-2}}\ \right) \tag{3.21}$$

The Bayesian framework uses this posterior to draw imputations, whereas the ML approach maximizes the likelihood to obtain estimators for the parameters in 3.19.
Regardless of the estimation method however, the difficulty lies in obtaining the ML-estimators for $\mathbb{E}(\mathfrak{D}_{ij})$ and $\hat{\sigma}^2_{ij}$. Like stated earlier, the data likelihood cannot be optimized without further ado, because the missingness composition may vary among the data rows. Yet, ML-estimates of a completed data set would be easily obtained, especially under the assumption of multivariate normality. A powerful tool applicable in this kind of situations is the Expectation Maximization (EM) algorithm.

3.4.4. The EM algorithm

The main idea of the EM algorithm is to iterate between estimating the missing values of the data from the ML parameters and updating these parameters by maximization of the likelihood using the completed data set. The algorithm starts (step $t = 0$) by setting initial values for the parameters. After that, the E- and M-step are executed alternately.

The E-step

In the E-step, missing values of the data set are imputed either, using the optimized μ_{ij} of the likelihood, or, when incorporating a prior, using the respective μ^*_{ij} of the posterior distribution. To calculate them, the expectations of $\mathfrak{D}_{ij}$ for all missing values are computed. They are obtained from a linear regression model having the observed data points as covariables. (Here, the expression $\mathfrak{D}^0_i$ denotes, that all `NA` values are replaced with zeros)

$$\mathbb{E}(\mathfrak{D}_{ij}) = \mathfrak{D}^0_i\hat{\boldsymbol{\beta}} = \mathfrak{D}^0_i\mathcal{S}wp(\mathbf{T}^{(t)}, \{1 - \mathcal{M}_i\})_j \tag{3.22}$$

The regression coefficients $\hat{\boldsymbol{\beta}}$ for the response variable j are obtained from the j-th column of the matrix, which results from sweeping the current sufficient statistic $\mathbf{T}$ over all observed variables (cf. section 3.4.1). With $\mathfrak{D}_i^0$ being 0 for all unobserved elements, non-β values drop out of the regression term.
Next, the sum of squared errors $\hat{\sigma}_{ij}^2$ is estimated. This value can analogously be calculated by sweeping the sufficient statistic.

$$\hat{\sigma}_{ij}^2 = \mathcal{S}wp(\mathbf{T}^{(t)}, \{1 - \mathcal{M}_i\})_{jj} \tag{3.23}$$

A completed data set is obtained, using the values of 3.22 (and 3.23) to compute the (posterior) expectation.

$$\begin{aligned} \hat{\mathfrak{D}}_{ij}^{(t+1)} &= \mathbb{E}(\mathfrak{D}_{ij}) && \text{(ML)} \\ \hat{\mathfrak{D}}_{ij}^{(t+1)} &= \mu_{ij}^* = \frac{m_{ij}\lambda_{ij}^{-2} + \mathbb{E}(\mathfrak{D}_{ij})\hat{\sigma}_{ij}^{-2}}{\lambda_{ij}^{-2} + \hat{\sigma}_{ij}^{-2}} && \text{(Bayes)} \end{aligned} \tag{3.24}$$

The M-step

The M-step updates the sufficient statistic using the completed data set. Additionally to the single values $\mathfrak{D}_{ij}$, $\mathbf{T}$ also contains the products $\mathfrak{D}_{ij}\mathfrak{D}_{ik}$ $\forall i, j, k$. Their expectation is given by

$$\mathbb{E}(\mathfrak{D}_{ij}\mathfrak{D}_{ik}) = \begin{cases} \mathfrak{D}_{ij}\mathfrak{D}_{ik} & \text{, if } \mathcal{M}_{ij} = 0 \quad \text{, } \mathcal{M}_{ik} = 0 \\ \underbrace{\mathbb{E}(\mathfrak{D}_{ij})\mathfrak{D}_{ik}}_{\text{ML}} \quad \text{or} \quad \underbrace{\mu_{ij}^*\mathfrak{D}_{ik}}_{\text{Bayes}} & \text{, if } \mathcal{M}_{ij} = 1 \quad \text{, } \mathcal{M}_{ik} = 0 \\ \mathbb{E}(\mathfrak{D}_{ij}\mathfrak{D}_{ik}) & \text{, if } \mathcal{M}_{ij} = 1 \quad \text{, } \mathcal{M}_{ik} = 1 \end{cases} \tag{3.25}$$

The first and second case are simply computed from the data. The third one additionally involves the estimated (posterior) covariance of both variables conditional on the observed data for measurement i. This can be obtained as the jk-element of the swept matrix.

$$\begin{aligned} \mathbb{E}(\mathfrak{D}_{ij}\mathfrak{D}_{ik}) &= \mathbb{E}(\mathfrak{D}_{ij})\mathbb{E}(\mathfrak{D}_{ik}) &&+ \mathcal{S}wp(\mathbf{T}^{(t)}, \{1 - \mathcal{M}_i\})_{jk} && \text{(ML)} \\ \mathbb{E}(\mathfrak{D}_{ij}\mathfrak{D}_{ik}) &= \mu_{ij}^*\mu_{ik}^* &&+ \mathcal{S}wp(\mathbf{T}^{(t)}, \{1 - \mathcal{M}_i\})_{jk} && \text{(Bayes)} \end{aligned} \tag{3.26}$$

In the Bayesian case, a prior covariance of 0 is assumed, as it is computationally convenient. However, other options are possible as well and would involve an adaption of equation 3.26. Contrary to the covariance, the prior variance of $\mathfrak{D}_{ij}$ is not assumed to be 0, which results in a posterior expectation of the quadratic case (i.e. $j = k$) as

$$\mathbb{E}(\mathfrak{D}_{ij}^2) = \mu_{ij}^{*2} + \left(\frac{1}{\lambda_{ij}^2} + \frac{1}{\hat{\sigma}_{ij}^2}\right)^{-1} \tag{3.27}$$

For the ML method, equation 3.26 can be applied to this case without any adaption.

The final requirement to compute an updated sufficient statistic is the variance-covariance matrix for the missing values within one observation. Let $\hat{\mathbf{\Sigma}}_i$ denote the portion of this $p \times p$ matrix contributed by observation i. The variance of non-missing values in $\hat{\mathbf{\Sigma}}_i$ is therefore 0. It can be obtained with

$$\hat{\mathbf{\Sigma}}_i = \mathcal{M}_i \mathcal{M}_i^T * \mathcal{S}wp(\mathbf{T}, \{1 - \mathcal{M}_i\}) \tag{3.28}$$

where the asterisk denotes an element-wise product.

In a Bayesian framework, if multiple priors within one observation i shall be incorporated, a matrix notation is helpful to formally define the corresponding posterior-equivalent to $\hat{\mathbf{\Sigma}}_i$.
Assume a prior for the whole observation i of $p(\mathfrak{D}_i) = \mathcal{N}(\mathbf{m}_i, \mathbf{\Lambda}_i)$. $\mathbf{m}_i$ denotes a vector of prior means if $\mathcal{M}_{ij} = 1$ and 0s otherwise. $\mathbf{\Lambda}_i$ represents a diagonal matrix comprising the individual prior variances λ_{ij}^2, resp. 0s for non-missing elements.

The posterior variance-covariance matrix is then given by

$$\hat{\mathbf{\Sigma}}_i^* = (\hat{\mathbf{\Sigma}}_i^{-1} + \mathbf{\Lambda}_i^{-1})^{-1} \tag{3.29}$$

Having computed all necessary elements, the M-step of the algorithm can be declared

as

$$\begin{aligned} \mathbf{T}^{(t+1)} &= (\hat{\mathfrak{D}}^{(t+1)})^T \hat{\mathfrak{D}}^{(t+1)} + \sum_{i=1}^{N} \left(\hat{\mathbf{\Sigma}}_i^{(t+1)} \right) \qquad \text{(ML)} \\ \mathbf{T}^{(t+1)} &= (\hat{\mathfrak{D}}^{(t+1)})^T \hat{\mathfrak{D}}^{(t+1)} + \sum_{i=1}^{N} \left(\hat{\mathbf{\Sigma}}_i^{*(t+1)} \right) \qquad \text{(Bayes)} \end{aligned} \tag{3.30}$$

The presented E- and M-steps are iterated, until there is convergence in $\mathbf{T}$.

3.4.5. Imputing Multiple Missing Values

To generate multiple imputation values, the incomplete data set is bootstrapped in the beginning. The (Bayesian) EM approach is then performed on each bootstrap sample, resulting in multiple estimated sets of parameters (resp. sufficient statistics $\mathbf{T}$).
In the ML framework, the final imputations of observation i are drawn from the assumed data distribution using the converged parameters in $\mathbf{T}$.

$$\begin{gathered} \hat{\mathfrak{D}}_i | \mathbf{T}^{(t)}, \mathcal{M} \sim \mathcal{N} \left(\tilde{\boldsymbol{\mu}}_i \ , \ \tilde{\mathbf{\Sigma}}_i \right) \\ \tilde{\boldsymbol{\mu}}_i = \mathbb{E}(\mathfrak{D}_i) = \mathfrak{D}_i^0 \mathcal{S}wp(\mathbf{T}^{(t)}, \{1 - \mathcal{M}_i\}) * \mathcal{M}_i + \mathfrak{D}_i^0 \qquad ; \qquad \tilde{\mathbf{\Sigma}}_i = \hat{\mathbf{\Sigma}}_i \end{gathered} \tag{3.31}$$

with $\mathfrak{D}_i^0$ denoting row i of the initial incomplete data set, having missing values replaced by 0s, and $\hat{\mathbf{\Sigma}}_i$ being given at 3.28. The element wise product with $\mathcal{M}_i$, combined with adding $\mathfrak{D}_i^0$ sets $\tilde{\boldsymbol{\mu}}_{ij} = \mathfrak{D}_{ij}$ for non missing values.

In a Bayesian approach, the imputations are drawn from the posterior distributions of observation i.

$$\begin{gathered} \hat{\mathfrak{D}}_i | \mathbf{m}_i, \mathbf{\Lambda}_i, \mathbf{T}^{(t)}, \mathcal{M} \sim \mathcal{N} \left(\tilde{\boldsymbol{\mu}}_i \ , \ \tilde{\mathbf{\Sigma}}_i \right) \\ \tilde{\boldsymbol{\mu}}_i = \boldsymbol{\mu}_i^* = \hat{\mathbf{\Sigma}}_i^* (\hat{\mathbf{\Sigma}}_i^{-1} \mathbb{E}(\mathfrak{D}_i) + \mathbf{\Lambda}_i^{-1} \mathbf{m}_i) + \mathfrak{D}_i^0 \qquad ; \qquad \tilde{\mathbf{\Sigma}}_i = \hat{\mathbf{\Sigma}}_i^* \end{gathered} \tag{3.32}$$

with $\hat{\mathbf{\Sigma}}_i^*$ given at 3.29.

For this thesis, no prior information on the missing values was incorporated. However, expert knowledge can often drastically improve the quality and numerical stability of the imputation and is, if available, highly recommended to include in the algorithm.

3.5. TSCS-Data Imputation

The standard imputation approach models missing values as a linear function of observed ones. Yet, when dealing with time-series cross-sectional data, this model has to be adapted to apply to the nature of the data.

Two modifications are made for that purpose. Both follow the concept of supplementing the data set with additional variables prior to running the imputation algorithm. To include possible shifts between cross-sections, an additional categorical variable indicating the cross-section is introduced (de facto: $m-1$ dummy coded variables representing m cross-section). With that, the imputation model now fits an individual effect for each cross-section, resulting in a more flexible estimation.

To adapt the imputation model for trends in time, the data is extended by smooth basis functions of the time variable. These can be created via LOESS, splines, wavelets or simply using q-order polynomials. For this thesis, the latter approach was chosen. It is economical on CPU resources, yet still satisfactory for many applications. The main disadvantage of polynomials lies in the extrapolation of values. More complex methods could be favorable in these situations.
Since time effects may vary between cross-sections, it is sometimes also convenient to include interactions between the basis functions and the cluster units. However, for a high number of cross-sections, this would lead to a multitude of variables in the imputation model. E.g.: Having m cross-sections and using a polynomial of order q would require adding $qk+k-1$ variables to the data frame. This approach is only feasible for large cluster sizes and was therefore not included for the simulations of chapter 4.

Further extensions of the data matrix, including lags and leads, or incorporating expert knowledge (in addition to the priors of 3.4.3) are analogously possible. These ideas are not discussed here, but can be found in King and Honaker 2008.

3.6. Practical Use in R

The described multiple imputation of this chapter is implemented in the R-package `Amelia`. It's main function will be introduced in the following.

3.6.1. Calling the Function `amelia()`

The function `amelia()` is used to perform multiple imputation on an incomplete data set. It's main parameters are listed below along with a short description.

`x =`
The incomplete data set, on which the multiple imputation shall be performed.

`m =`
The number of imputed data sets to create. A typical value here is 5.

`ts =` , `cs =`
These parameters are used to identify the time-series and cross-section variable of the data set. They can either be specified by the column number or the variable name.

`polytime =`
An integer value between 0 and 3, defining the power of a polynomial to model the time-series.

`intercs =`
A logical value indicating, if the time effects should vary across the cross-sections.

`prior =`
A matrix, to declare the prior parameters for missing values. It's rows have the composition: `c(NA.row, NA.column, prior.mean, prior.sd)`

`p2s =`
Set the screen output during execution: 0 $\hat{=}$ disabled; 1 $\hat{=}$ normal; 2 $\hat{=}$ detailed.

Other parameters, that are not mentioned here, can be found in the R Documentation for `amelia()` (see Honaker et al. 2015).

The multiple imputations of this thesis were performed using the following function call:

```
amelia( x = Data,
        m = 5,
        ts = "time",
        cs = "subject",
        polytime = 1,
        p2s = 0 )
```

5 imputed data sets were created. Unless the rate of missingness is very high, this is a conventional size.

To model the time-series variable, a linear time trend was chosen (`polytime = 1`). The autoregressive model (see section 4.2.1) used to create the time effect for one of the incomplete variables in the simulation may have an overall non-linear trend, however, the strong dependency of consecutive measurements paired with the small count of observations made the approximation of a linear trend satisfactory.

One problem shortly remarked in the previous section is, that interactions between the time series and cross section variables were not properly implementable here, due to the small number of measurements per cluster. Fitting individual polynomials (or even splines) on every cluster would have produced a tremendous increase of imputation model parameters, which would therefore have resulted in either a poor fit, or no possible fit at all.

The last parameter `p2s` controls the screen output. Having a high number of `amelia()` function calls in the simulations, a cluttering up of the console can be prevented by disabling this feature (`p2s=0`).

3.6.2. Return Value of `amelia()`

After executing the function described in 3.6.1, a list object is returned. It contains the imputed data set along with various imputation parameters (e.g. posterior means of each imputation). In the simulations of chapter 4, only the completed data sets were stored for later analysis. They can be obtained using

```
1 amelia( ... )$imputations
```

A summary of the other parameters given in the returned list, together with a short description is presented below.

`$ m` , `$ arguments`
`$arguments` returns all arguments, that were set in the function call, except for the number of imputations to be performed, which is returned by `$m`.

`$ orig.vars` , `$ missMatrix`
`$orig.vars` returns a vector containing the variable names of the data set. `$missMatrix` returns the missingness matrix $\mathcal{M}$ of the data.

`$ theta` , `$ mu` , `$ covMatrices`
`$theta` returns the converged sufficient statistic of 3.18 for every EM chain. It's elements $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ can also be returned individually with `$mu` and `$covMatrices`

`$ code` , `$ message`
These elements represent an internal exit-code and -message from the function.

`$ iterHist` , `$ overvalues`
Both are diagnostic elements of the imputation fit. `$iterHist` gives a history of the EM chain convergence (see Honaker et al. 2012 section 4.10). `$overvalues` can be used to create a cross validation study and consequently judge the fit of the imputation model (see Honaker et al. 2012 section 4.7.2).

4. Simulations

4.1. Design of the Simulations

The main goal of the simulation was to examine the quality of the GLMM estimation in the presence of missing values. For this purpose, an artificial data set was created, which incorporated associations between its variables via predefined "true" parameters. A model fitted on the data should detect these underlying parameters. To review a correct detection, the empirical bootstrap distribution of the model parameters was analyzed. 95% bootstrap percentile intervals were computed and tested for holding the true values. When reiterating this procedure, the underlying parameters should be held by the intervals in approximately 95% of the runs.

Three simulations were made in this thesis. The first one estimated parameters on a full data set. With no missing values, the results could be seen as a reference for the following analyses. The second simulation incorporated missing values into the data. The problem of incomplete data was here addressed by applying listwise deletion. Because of it's simplicity, this is a very common approach amongst many researchers. The final simulation also incorporated missingness into the data, but addressed the problem using multiple imputation. An important part in this context was to find an appropriate combination method for all estimators, which was researched here as well.

This chapter starts with presenting the construction of test data and the implementation of a function to generate missing values. After that, the function to obtain estimates

for the true $sd(\hat{\boldsymbol{\beta}})$ is shown. The analysis of the three simulations is presented in the following sections. Conclusively, all individual results are compared.

4.2. Generating Test Data

This section concerns the construction of test data. By properly creating a custom data situation, it was possible to define the underlying "true" parameters of a regression model. This enabled the evaluation of parameter estimations from the associated model. The aspired data had a balanced time-series cross-section design and comprised six covariables plus a binary response.

4.2.1. Independent Variables

The aim of the independent variables was to cover a variety of variable types found in practical data situations. Furthermore, dependencies among those variables were implemented to achieve a more realistic design. This section continues, by describing each created covariable in an individual paragraph.
The practical implementation of the data creation algorithm has soft coded variables for the number of clusters and their sizes. Nevertheless, their explicit values were used in the following code samples for ease of exposition.

Cross Section Variable (`subject`)

Like stated before, the aim was to construct a cross-sectional data design. Therefore, a variable indicating the cluster was needed. Here, 100 individuals comprising 5 measurements each were created. The cross section variable therefore simply followed as

$$\mathbf{x}_{\text{subject}} = (1, 1, 1, 1, 1, 2, 2, \ldots, 99, 100, 100, 100, 100, 100)^T \tag{4.1}$$

and was created in `R` with the code line

```
1 subject = rep(1:100,each=5)
```

Time Series Variable (`time`)

The second design-composing component was the time-series variable. A balanced design with equidistant observations was chosen. It can be written as

$$\mathbf{x}_{\text{time}} = (1, 2, 3, 4, 5, 1, 2, 3, 4, 5, \ldots, 1, 2, 3, 4, 5)^T \tag{4.2}$$

Analogous to the cross-section variable, the time-series component could be created in `R` using the `rep()` function.

```
1 time = rep(1:5,100)
```

Note:

Because of the simplicity of the time-series and cross-section variables, both can also be created simultaneously by the `R` function: `expand.grid(1:5,1:100)`

Discrete Baseline Variable (`age`)

A common type of covariable is given by a discrete variable, which is constant over the measurements of an individual. Common examples (if data points are measured in a rather small time-span) are the age of a person, their height, or their weight. In this thesis, a variable representing the age of a person was destined.

The first considered solution was to draw random observations from a binomial distribution. To achieve realistic values for the age of a person, the draws also had to satisfy upper and lower bounds. Here, the lowest age observable was defined to be 18. An age of 60 was chosen to be the maximum value.

Potential realizations of a Binomial distribution $\mathcal{B}(n, p)$ are $(0, 1, \ldots, n)$. To transform this range, one can add an integer value to the binomial draw. A possible solution for the aspired boundaries is

$$f(x) = x + 18 \quad ; \quad x \sim \mathcal{B}\left(42, \frac{1}{2}\right) \quad ; \quad \Rightarrow \quad f(x) \in (18, 19, \ldots, 60) \tag{4.3}$$

The distribution is illustrated in figure 4.1 (left).

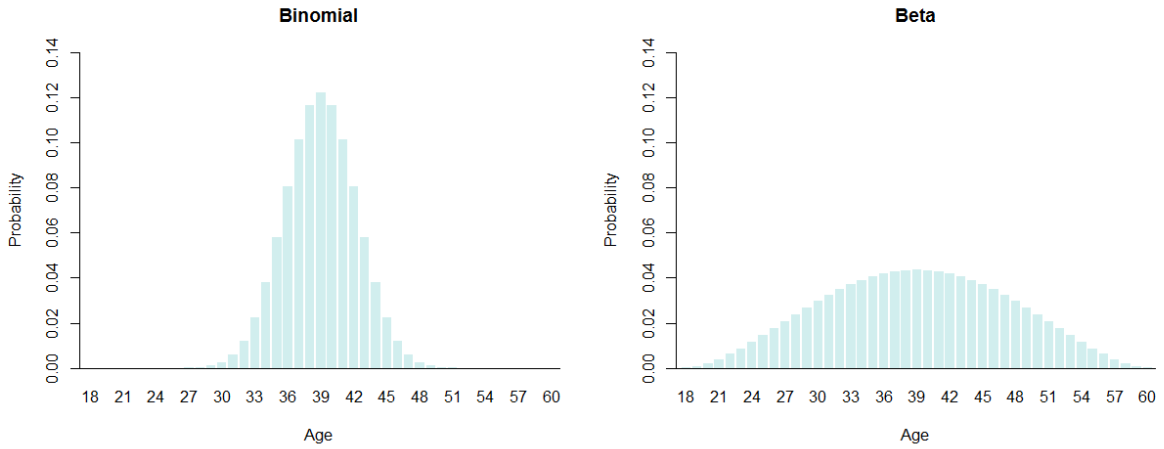


Figure 4.1.: Comparison of the Binomial distribution of 4.3 and the discretized Beta distribution of 4.4 for simulating the variable `age`.

The main problem of using the binomial distribution was the small variance of the random draws. It can be calculated for $x_{\text{age}} \sim \mathcal{B}(n, \pi)$ by $\mathbb{V}\text{ar}(x_{\text{age}}) = n\pi(1-\pi)$. The parameter n is fixed to ensure the bounds of x_{age} and the function $\pi(1-\pi)$ for $\pi \in [0, 1]$ has it's maximum value at $\pi = \frac{1}{2}$. Therefore the solution of 4.3 has the maximum variance possible here. Yet, observing values outside of the interval $[31, 47]$ with less than 1% probability seemed very limiting.

An alternative solution to draw x_{age} was given by a discretized version of the very flexible Beta distribution. Realizations of $\mathcal{Be}(a, b)$ are continuous and lie in the interval $[0, 1]$. It's shape is governed by the two parameters a and b. A bell curve can be created by setting both parameters to the same value greater than 1. This value is inversely proportional to the variance of the realization. Here, the parameters were set to 3 to obtain a rather flat density function. To discretize and adjust the range of the Beta distribution to the interval $[18, 60]$, a transformation was used.

The final function to obtain adequate random draws for x_{age} was created as

$$f(x) = \lceil 43x \rceil + 17 \quad ; \quad x \sim \mathcal{Be}(3, 3) \quad ; \quad \Rightarrow \quad f(x) \in (18, 19, \ldots, 60) \tag{4.4}$$

with $\lceil \cdot \rceil$ denoting the ceiling function.

The density of 4.4 is visualized in 4.1 (right).

In R, the variable was implemented using

```
1 age = rep(ceiling(43*rbeta(100,3,3)) + 17, each=5)
```

100 draws for age were obtained and repeated for all 5 observations of a person. It is reasonable, that the age of a person does not depend on one of the other covariables here and hence was drawn independently.
An exemplary vector of the age variable is given below.

$$\mathbf{x}_{\text{age}} = (39, 39, 39, 39, 39, 42, 42, \ldots, 25, 25, 33, 33, 33, 33, 33)^T \tag{4.5}$$

Continuous Baseline Variable (height)

The next covariable to be created was named height. It represented a baseline variable similar to age, but had a continuous range of values.
Of course, real life data never is truly continuous, yet approximate examples for this type of variable would be the duration of a process, the weight of a sampling unit, or the height of a person. The latter one was simulated for this data set.

A linear model was used to compose the covariable and to induce a dependency onto the age variable.

$$x_{\text{height},i} = \beta_0 + x_{\text{age},i}\beta_1 + \epsilon_i \tag{4.6}$$

with $\epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$

The average height of German men obtained from physical examinations for the military service[1] is 1.82 meters. This was chosen to be the expected value for individuals of age 18. Typically, older persons tend to be smaller than younger ones (over 18). The value of β_1, which determined how the age influences the height therefore needed to have a negative sign. By assuming, that a 60 year old person is on average 1.70 meters, one

[1] see online reference: Verteilung des Merkmals Körpergröße

could formulate two equations

$$\beta_0 + 18\beta_1 = 1.82 \quad ; \quad \beta_0 + 60\beta_1 = 1.7 \tag{4.7}$$

which lead to the following approximate solutions for the β coefficients.

$$\beta_0 \approx 1.871 \quad ; \quad \beta_1 \approx -0.003 \tag{4.8}$$

The final component to be defined was the variance of ϵ_i. A reasonable value would be $\sigma^2 = 0.005$, which gave a standard deviation of approximately 7 centimeters.
Altogether, the full model to create the covariable `height` resulted as

$$x_{\text{height},i} = 1.871 - 0.003x_{\text{age},i} + \epsilon_i \tag{4.9}$$

with $\epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, 0.005)$

This model could be realized in `R` using

```
1 height = 1.871 - 0.003*age + rep(rnorm(100,0,sqrt(0.005)),
2                                  each=5)
```

The vector `age` already contained every measurement repeated 5 times. Therefore, only the individual ϵ_i needed to be drawn and repeated, to create a cross-sectional baseline design.
By simulating random draws of height, one could create an empirical density function for this covariable. It is illustrated in figure 4.2.

The range of the variable had no theoretical upper and lower bounds. Yet, unrealistic values were highly improbable. The shortest and tallest full-aged men on earth[2] span a range of [0.546m , 2.72m]. The probability of simulating a person to be less than 0.546m is approximately $2 \cdot 10^{-55}$ [3]. Respectively for a person to be taller than 2.72m

[2] Guinness World Records 2015: Chandra Bahadur Dangi (0.546m) and Robert Wadlow (2.72m)
[3] $\sum_{i=18}^{60} \Phi_i(0.546) \approx 2 \cdot 10^{-55}$, where $\Phi_i(\cdot)$ denotes the CDF of $\mathcal{N}(1.894 - 0.004i, 0.005)$

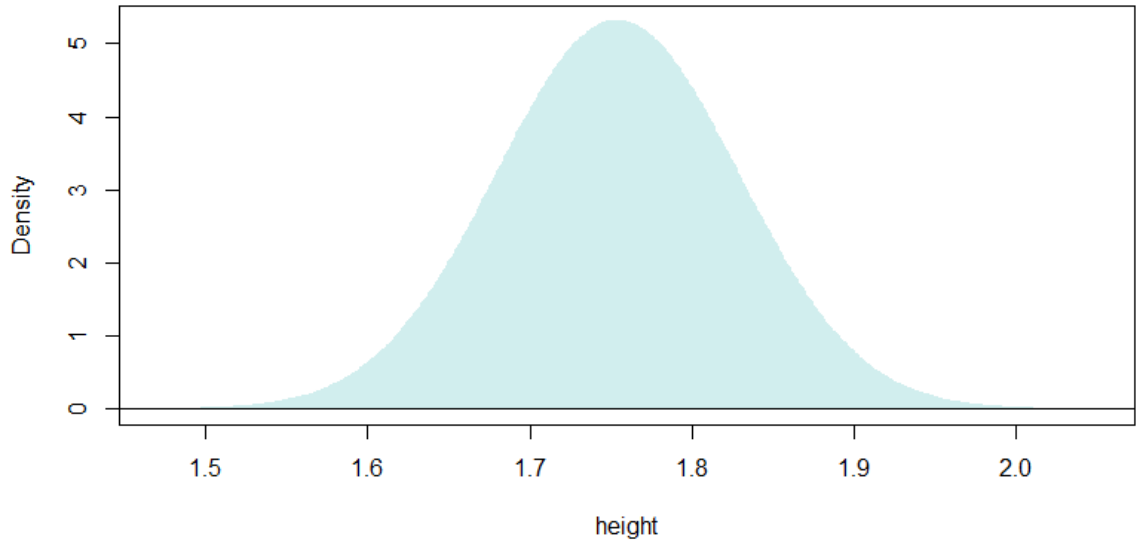


Figure 4.2.: Empirical distribution of `height` created by drawing 10^7 observations of `age` and subsequently computing `height` using 4.9.

it is $6 \cdot 10^{-37}$ [4]. Because of that, no restriction to the range was made. Furthermore, unrealistic values do not impair the imputation of missing values. Hence, the actual aim of the project would either way not have been compromised.

Self Dependent Variable (`blood`)

Many medical variables (e.g. blood parameters) have the property, that consecutive measurements tend to be correlated (often depending on the period of time, that lies in between them). If e.g. one is to measure the blood pressure of a patient two times in 3 minutes, the results will most likely be very similar. The simulation of a variable with self dependent measurements was realized using an autoregressive model of order 1 (AR(1)).

An AR(1) model can be defined recursively with fixed starting value x_0 as

$$x_t = \rho x_{t-1} + \epsilon_t \quad ; \quad t = 1, \ldots, n \tag{4.10}$$

[4] $\sum_{i=18}^{60}(1 - \Phi_i(2.72)) \approx 6 \cdot 10^{-37}$, where $\Phi_i(\cdot)$ denotes the CDF of $\mathcal{N}(1.894 - 0.004i, 0.005)$

having

$$\begin{aligned} \mathbb{E}(\epsilon_t) &= 0 \\ \mathbb{V}\text{ar}(\epsilon_t) &= \tau^2 \\ \mathbb{C}\text{ov}(\epsilon_t, \epsilon_s) &= 0 \\ -1 < \rho &< 1 \end{aligned} \tag{4.11}$$

The t-th observation of a person is given by the (t-1)-th observation multiplied with a known factor ρ and summed up with a random error ϵ_t.
By defining a vector $\boldsymbol{\varepsilon}_t$ and $\boldsymbol{\rho}_t$ of length $(t+1)$ as

$$\begin{aligned} \boldsymbol{\varepsilon}_t &= (\ x_0\ \ \epsilon_1\ \ \ldots\ \ \epsilon_t\ )^T \\ \boldsymbol{\rho}_t &= (\ \rho^t\ \rho^{t-1} \ldots\ \rho^0\ )^T \end{aligned} \tag{4.12}$$

the prediction for the t-th value of this model can also be written without recursion as

$$x_t = \sum_{k=0}^{t} \rho^{t-k} \varepsilon_k = \boldsymbol{\varepsilon}_t^T \boldsymbol{\rho}_t \tag{4.13}$$

One characteristic property of this model is, that the correlation between x_t and a delayed value x_{t-s} decreases with the magnitude of s. This process can be illustrated with the Autocorrelation function (ACF), which is given for the AR(1) as

$$ACF(s) = \frac{\mathbb{C}\text{ov}(x_t, x_{t-s})}{\mathbb{V}\text{ar}(x_s)} = \rho^s \tag{4.14}$$

Figure 4.3 shows the ACF for different values ρ. If ρ is positive, the correlation decreases geometrically. For negative ρ , it additionally decreases with alternating sign. The larger the absolute value of ρ is, the faster this decrease of the ACF happens.

The main intention of using the AR(1) model here was to create (strongly) dependent observations. Therefore a large absolute value for ρ was chosen in this thesis ($\rho = 0.9$).

To practically implement a vector of n consecutive AR(1) draws in `R`, the recursive definition of the model was used, since it is less computationally intensive than simulating each value individually using $\boldsymbol{\varepsilon}_t^T \boldsymbol{\rho}_t$. The input parameters of the generated `ar1()`

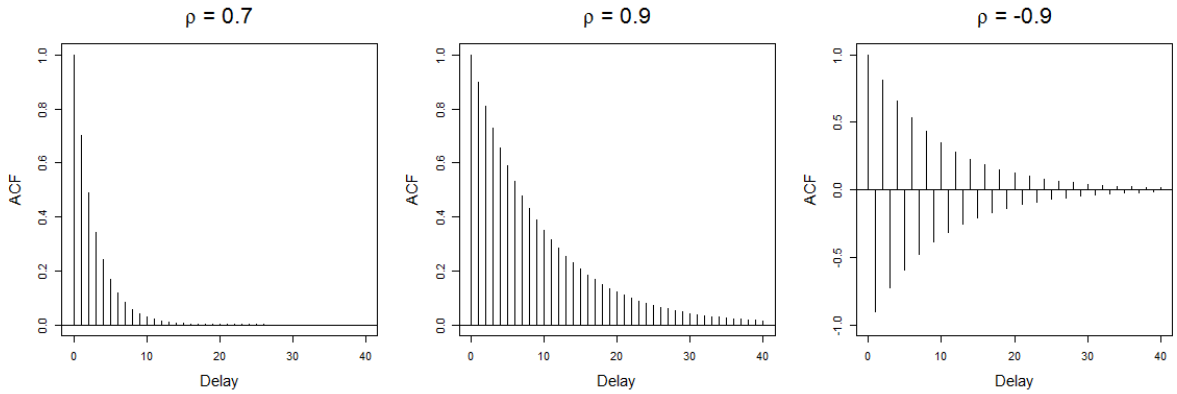


Figure 4.3.: The Autocorrelation function of the AR(1) for different values of ρ.

function were the number of values to be created n, the error standard deviation τ, the starting value x_0 and the correlation parameter ρ.

```
1 ar1 = function(n, x0, rho, tau){
2   eps = rnorm(n,0,tau)
3   x = vector(length = n)
4   x[1] = x0
5   for(t in 2:n) x[t] = rho*x[t-1] + eps[t-1]
6   return(x)
7 }
```

With the help of this `ar1()` function, the self depending variable `blood` was created by the following code.

```
1 blood = as.vector( sapply(
2           unique(height),
3           function(x) ar1(5, x+rnorm(1,0,0.2), 0.9, 0.7)
4         ) )
```

An additional dependency with the baseline variable `height` was induced, by utilizing these observations (with added noise) as starting points of the `ar1()` series.

Figure 4.4 illustrates five exemplary `ar1()` sequences with different starting points.

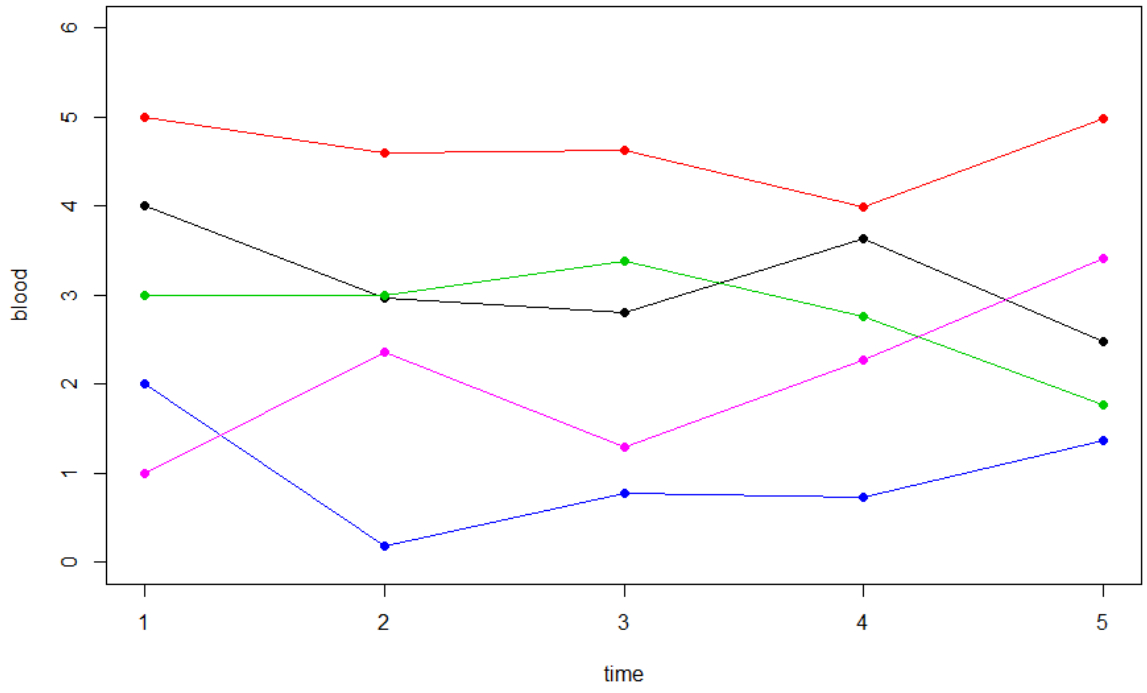


Figure 4.4.: Five realizations of length 5 from the `ar1()` function with starting points $x_0 = 1, \ldots, 5$. The strong dependency of values from the same individual is conspicuous.

Binary Variable (`pain`)

The last covariable to be created was named `pain`. It is a binary variable with dependencies on all other covariables created so far and could represent an indicator, if a person is in pain or not. To model these dependencies, a GLMM was used. This section presents the main components used for this model. A full explanation on how to

technically compose a variable from a GLMM will be given in section 4.2.3, where the response variable of the data set was created analogously.
The model used the following linear predictor for `subject` i and measurement j

$$\eta_{ij} = \beta_0 + x_{\text{time},ij}\beta_1 + x_{\text{age},i}\beta_2 + x_{\text{height},i}\beta_3 + x_{\text{blood},ij}\beta_4 + b_i \tag{4.15}$$

To implement a GLMM, the three main assumptions described in section 2.2.1 needed to be specified.

- The distribution of the model response given the linear predictor:

$$x_{\text{pain},ij}|\eta_{ij} \overset{\text{ind}}{\sim} \mathcal{B}(\pi_{ij})$$

- The link-/ response function:

$$\eta_{ij} = g(\mu_{ij}) = \log\left(\frac{\pi_{ij}}{1-\pi_{ij}}\right) \qquad ; \qquad \pi_{ij} = h(\eta_{ij}) = \frac{1}{1+\exp(-\eta_{ij})}$$

- The distribution the random effects:

$$b_i \overset{\text{iid}}{\sim} \mathcal{N}(0, d^2)$$

The linear predictor η_{ij} can be calculated by specifying the true underlying values of $\boldsymbol{\beta}$ and d. They were chosen to be

$$\boldsymbol{\beta} = (0.5, -0.2, -0.02, -0.25, 0.5)^T \qquad ; \qquad d = 0.5 \tag{4.16}$$

The probability π_{ij} for the response to be 1 was computed by inserting η_{ij} into the response function. With this, $x_{\text{pain},ij}$ could be realized as a random draw from a Bernoulli distribution with parameter π_{ij}.
The `R`-Code drawing the variable `pain` as GLMM realizations of the described model is shown below. It is a shortened representation of the analogously structured code to create the response variable of the data set.

```
1 pain = rbinom(n=500, size=1,prob=1/(1+exp(-(
2                 cbind(1,time,age,height,blood) %*%
3                 c(0.5,-0.2,-0.02,-0.25,0.5) +
4                 rep(rnorm(n=100, sd=0.5), each=5)))))
```

Using the parameters of 4.16, an average of approximately 37% of all measurements were simulated to have pain. The data excerpt shown in figure 4.5 illustrates the variable `pain` for different subjects at the five points of measurement. It was used to examine, if the arrangement of the pain measurements was well balanced over the subject and time variable. This seemed to be the case here.

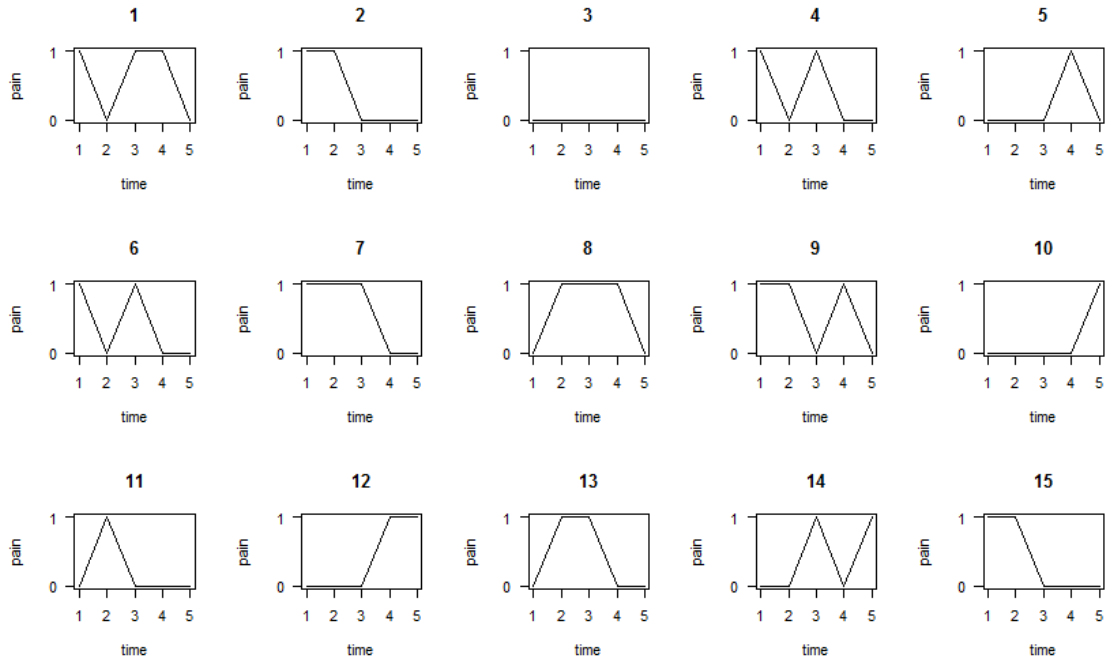


Figure 4.5.: Sample of the variable `pain` from 15 subjects over the five points of measurement. Interpolations between the discrete points were added to illustrate the progress over time.

4.2.2. Summary of Covariable Dependencies

Dependencies between covariables are essential to enable multiple imputation with Amelia II. Missing values are modeled by the observed ones. Therefore, the more information these variables contain on each other, the more precise can the imputation algorithm predict values from one another.
Figure 4.6 gives an overview over all induced dependencies between the covariables.

The variables are influenced in the following ways:

(1) as starting value of the AR(1) model creating the variable

(2) as covariable of the LM creating `height`

(3) as covariable of the GLMM creating `pain`

(4) as grouping variable of the GLMM creating `pain`

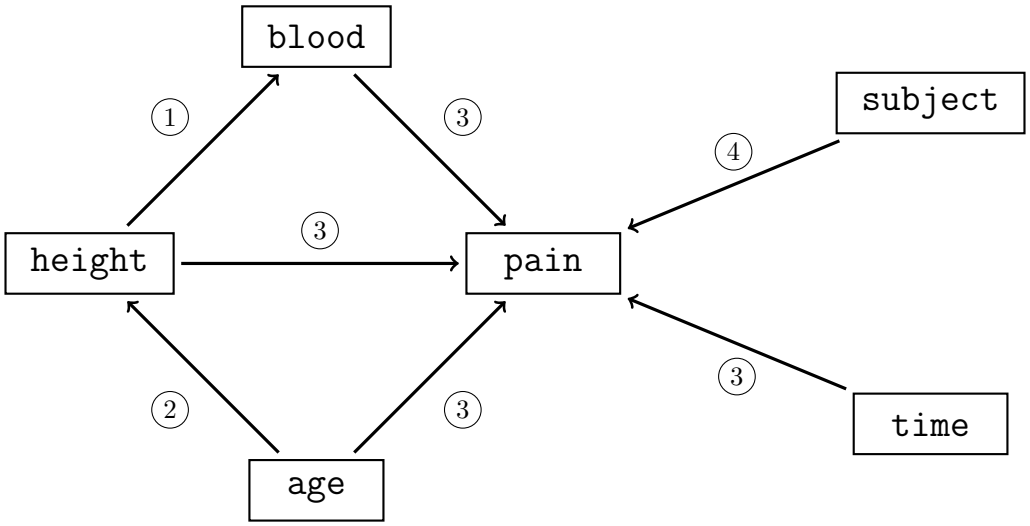


Figure 4.6.: Directed acyclic graph representing the dependencies of the covariables. The numbers describe the respective way, that the covariables are influenced (cf. notes above).

4.2.3. Response Variable

The response variable of the data set was chosen to be binary. It could e.g. represent an indicator, if a person is ill. The state of health should depend on all created covariables of section 4.2.1. For this purpose, a GLMM with the following linear predictor for measurement j and individual i was used.

$$\eta_{ij} = \beta_0 + x_{\text{time},ij}\beta_1 + x_{\text{age},i}\beta_2 + x_{\text{height},i}\beta_3 + x_{\text{blood},ij}\beta_4 + b_i \tag{4.17}$$

The model was defined by the three assumptions of 4.18.

Distribution of $y_{ij}\|\eta_{ij}$	**Link function**	**Distribution of** b_i
$y_{ij}\|\eta_{ij} \overset{\text{ind}}{\sim} \mathcal{B}(\pi_{ij})$	$g(\mu_{ij}) = \text{logit}(\pi_{ij})$	$b_i \overset{\text{iid}}{\sim} \mathcal{N}(0, d^2)$

(4.18)

The Regression Coefficients $\boldsymbol{\beta}$

An important part of creating a response variable is the determination of the regression coefficients. Careful thought should be given to this, since the values of $\boldsymbol{\beta}$ highly affect the predictability of a model. If one coefficient is chosen too small, the effect of this variable may be unidentifiable. On the other hand if chosen too large, the probability of $y_{ij} = 1$ reaches extreme values, which could result in masking other effects.
To visualize these borders of $\boldsymbol{\beta}$, the following procedure was used:

1. Set every $\beta = 0$, except the one to analyze.
2. Map a plausible interval for the analyzed β by a sequence of 200 values, and perform the following steps for each of the 200 β candidates:
   - Create a data set using the current regression coefficients
   - Estimate the probability of $y = 1$ from this data set by

$$\hat{\pi} = \frac{1}{2500}\sum_{i=1}^{500}\sum_{j=1}^{5} y_{ij}$$

3. Plot the obtained probabilities against the corresponding β values

The plots of all fixed effects are illustrated in figure 4.7

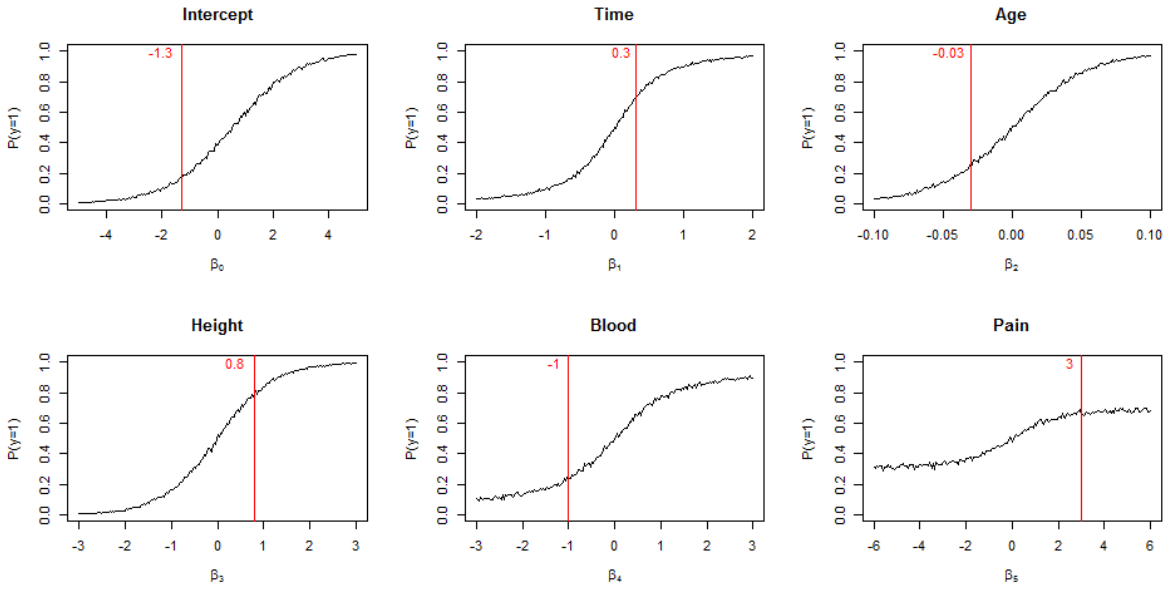


Figure 4.7.: Estimations for $\mathbb{P}(y_{ij} = 1)$ over a sequence of 200 values for each fixed effect c.p. . The red vertical line illustrates the chosen value for the variable.

Note, that the limits of the β_5-curve were not found to be $[0, 1]$, but approximately $[0.35, 0.65]$. This resulted from the dichotomy of the variable `pain`, having $\approx 37\%$ ones. High parameter values only influenced the observations, where $x_{\text{pain}} = 1$, hence making these values (nearly) guaranteed ones or zeros. Yet, the other 63% of the values, where $x_{\text{pain}} = 0$ had a linear predictor of 0 regardless of the chosen effect, which resulted in a probability of 0.5 for the response to be 1. That is why $\mathbb{P}(y_{ij} = 1)$ was not bounded at $[0, 1]$ for an effect of a binary variable , but depended on the probability of this variable to be 1.

To avoid the problems described earlier, the respective β-curves of plot 4.7 were analyzed. The following guidelines were considered to chose the $\boldsymbol{\beta}$ values from the plot.

- The chosen β should not induce extreme values for $\mathbb{P}(y_{ij} = 1)$
- Extremely small β values (i.e. $\mathbb{P}(y_{ij} = 1) \approx 0.5$) were avoided to create a notable

effect of the covariable.

- The effects should have opposing signs to avoid extreme overall values for $\mathbb{P}(y_{ij} = 1)$

Altogether, the chosen regression coefficients to create the response variable were

$$\boldsymbol{\beta} = (-1.3, 0.3, -0.03, 0.8, -1, 3)^T \tag{4.19}$$

The variance of the random intercept was defined to be $d^2 = 0.5^2$.

These parameters resulted in an average probability for $y_{ij} = 1$ of approx. 42%.

Implementation in R

The first step to draw a response variable of a GLMM was to compute the linear predictor of all individuals $\boldsymbol{\eta} = (\boldsymbol{\eta}_1^T, \ldots, \boldsymbol{\eta}_{100}^T)^T$.
Having a random intercept model, it was given by

$$\boldsymbol{\eta} = \begin{pmatrix} \mathbf{1} & \mathbf{x}_{1,\text{time}} & \mathbf{x}_{1,\text{age}} & \mathbf{x}_{1,\text{height}} & \mathbf{x}_{1,\text{blood}} & \mathbf{x}_{1,\text{pain}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{1} & \mathbf{x}_{100,\text{time}} & \mathbf{x}_{100,\text{age}} & \mathbf{x}_{100,\text{height}} & \mathbf{x}_{100,\text{blood}} & \mathbf{x}_{100,\text{pain}} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_5 \end{pmatrix} + \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{100} \end{pmatrix} \tag{4.20}$$

The design matrix `X` was obtained by binding all created covariables and a vector of ones column-wise.

```
1 X = cbind(1,time,age,height,blood,pain)
```

Repeating 100 draws from the $N(0, 0.5^2)$-distribution 5 times constructed the vector of random intercepts.

```
1 b = rep( rnorm(100,0,0.5) , each=5 )
```

With the fixed effects vector `beta` being set to the value of 4.19, the linear predictor could be computed by

```
1 eta = X %*% beta + b
```

The next step was to obtain the expected value `pi` of the response by applying the response function to the linear predictor. The natural link function of the Bernoulli distribution is the logit-link. The corresponding response function is

$$h(\eta_{ij}) = \frac{1}{1 + \exp(-\eta_{ij})} \tag{4.21}$$

```
1 pi = 1/(1+exp(-eta))
```

The vector `pi` contained the probability of the response being 1 for each observation. To realize actual binary values, the last step was to draw the response y_{ij} from the Bernoulli distribution with parameter π_{ij}.
This could be performed in `R` using

```
1 y = rbinom(500, 1, pi)
```

An excerpt of the response variable created with the presented code is shown in figure 4.8.

The final data set was constructed, by binding the covariables and the response into a data-frame object.

The first 7 rows of an exemplary data set are illustrated in table 4.1

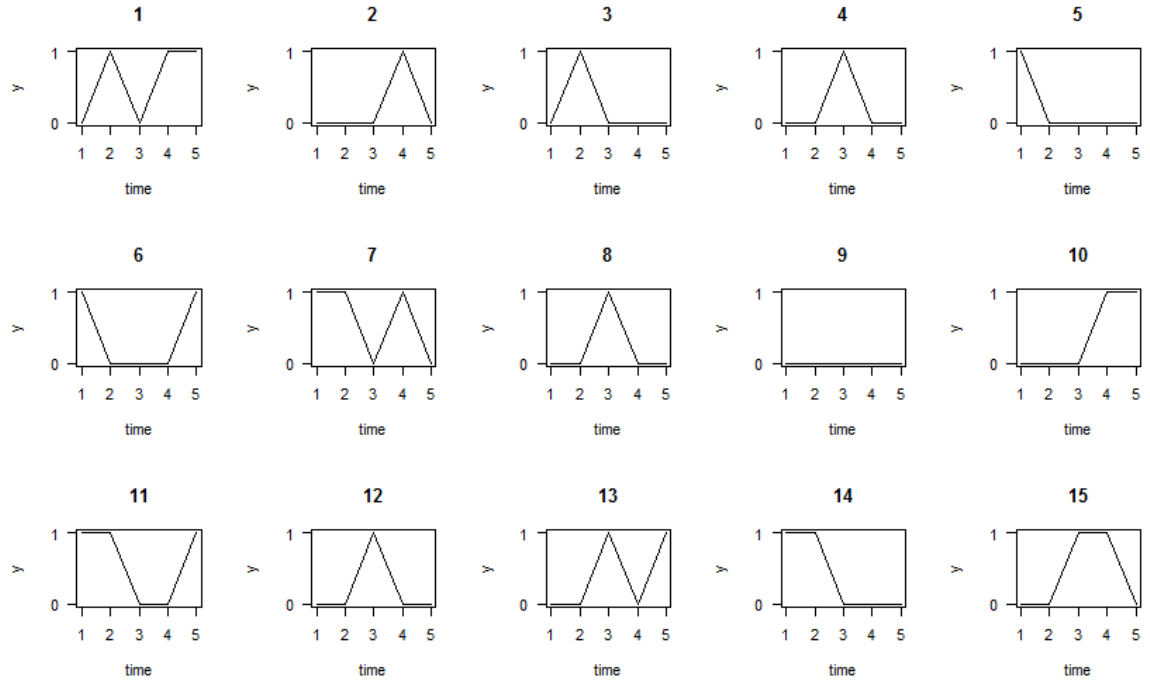


Figure 4.8.: Sample of the response variable `y` for 15 subjects over the five points of measurement. Interpolations between the discrete points were added to illustrate the progress over time.

	subject	time	y	age	height	blood	pain
1	1	1	0	43	1.7413	1.7464	1
2	1	2	1	43	1.7413	2.2221	0
3	1	3	0	43	1.7413	2.3840	1
4	1	4	1	43	1.7413	1.6517	1
5	1	5	1	43	1.7413	1.3506	0
6	2	1	0	53	1.6049	1.5802	1
7	2	2	0	53	1.6049	0.7719	1

Table 4.1.: Excerpt of a data set, created by the methods of section 4.2.1 and 4.2.3.

4.3. Generating Missing Values

Simulation II and III required the presence of missing observations in the analyzed data. This section concerns the construction of an artificial incomplete data situation, by deleting observed values from a full data set.

In real-life data, the missingness of a value often depends on influencing factors. If those influences are available in the form of observed variables, the missing values are assumed to be "Missing At Random" (see section 3.1).

To simulate such a situation, a GLM was used. Let $\mathcal{M}_k$ be a missingness vector indicating, if a value of covariable k is missing. This vector shall be the response of the GLM. The model assumptions were

$$\begin{array}{cc} \textbf{Distribution of } \mathcal{M}_{ijk}|\eta_{ij} & \textbf{Link function} \\ \mathcal{M}_{ijk}|\eta_{ijk} \overset{\text{ind}}{\sim} \mathcal{B}(\pi_{ijk}) & g(\mathbb{E}(\mathcal{M}_{ijk})) = \text{logit}(\pi_{ijk}) \end{array} \tag{4.22}$$

For the analyses of this thesis, the linear predictor η_{ijk} was specified to consist of one other observed covariable with a regression coefficient of 1 and an intercept.

$$\eta_{ijk} = \gamma_k + x_{ijl} \tag{4.23}$$

Choosing this kind of predictor enables an explicit solution for the intercept γ_k, if a certain percentage of missing observations is aspired. After applying the response function, the mean of all expected values for the observations to be missing, i.e. the average percentage of missing values to be created is given by

$$\bar{\pi}_k = \frac{1}{500} \sum_{i,j} \pi_{ijk} = \frac{1}{500} \sum_{i,j} \left(\frac{1}{1 + \exp(-\gamma_k - x_{ijl})} \right) \tag{4.24}$$

For a given $\bar{\pi}_k$, one obtains an equation with one unknown variable. Yet, an analytical solution for gamma is not feasible, which is why numerical optimization methods are required. In `R`, the function `uniroot()` can be used to solve univariate equations of the form $f(x) = 0$. A suitable value for γ_k to create an average percentage of $\tilde{\pi}_k$ missing

values could therefore be obtained by solving

$$\frac{1}{500}\sum_{i,j}\left(\frac{1}{1+\exp(-\gamma_k - x_{ijl})}\right) - \tilde{\pi}_k = 0 \tag{4.25}$$

for γ_k.

The code to compute the solution of this equation is given below.

```
1 uniroot(function(g)sum(1/(1+exp(-(g+data[,dep]))))/500-pi,
2         interval=c(-100,10) )
```

`data[,dep]` selected the column of the dependent variable (x_{ijl}) from the data set. The searching interval was chosen to be comparatively large[5], to ensure finding a solution. This could easily be done, because of the small amount of CPU-time used for the function.

In this thesis, missing values were incorporated for two covariables. An overview over the explicit parameters chosen is given in tabular 4.2.

Target variable	Dependent variable	Missing values
`pain`	`blood`	30%
`blood`	`age`	40%

Table 4.2.: Overview of the missingness structure.

How to draw realizations from a custom model was already described in detail in section 4.2.3. A short summary is given in the following.

After computing an approximate value for γ, the linear predictor can be obtained. Applying the response function to η gives the expected probability of a value being

[5] the average values of γ_k for the dependent variables and missingness percentages used in the simulation studies are approximately -2.4 (sd=0.06) and -41.3 (sd=1.09)

unobserved. This probability is used to draw the values of the missingness vector from a Bernoulli distribution. Finally, values indicated by $\mathcal{M}$ are replaced with `NA`.

The `R` code implementing these steps is shown below.

```
1 eta_miss = gamma + data[,dep]
2 is.miss = rbinom(n = length(data[,1]), size = 1,
3                  prob = 1/(1+exp(-eta_miss)))
4 data[,tar] = ifelse(is.miss, NA, data[,tar])
```

The overall function to incorporate missing values to a data set was named

`create.miss( data, pi, tar, dep )`

It used four parameters.

- `data`: The data frame to be used.
- `pi`: The percentage of missing values to be created. ($\in [0, 1]$)
- `tar`: The column of `data` holding the variable to induce missing values to.
- `dep`: The column of `data` holding the dependent variable.

To create missing values in more than one variable, `create.miss()` needs to be called multiple times passing the same data set. When repeating the function on incomplete data, these variables cannot be chosen as a dependent covariable any more.

4.4. Estimating the Theoretical Standard Error of $\hat{\boldsymbol{\beta}}$

The true values of the fixed effects and the standard deviation of the random intercept were known, as they have been defined in the construction of the data set. However, the theoretical value of the standard errors for the fixed effects depended on the circumstances of the estimation (e.g. the number of observations). To obtain a reference value for the analyses of the simulation, a numerical procedure was applied.

The regression coefficients $\hat{\boldsymbol{\beta}}$ were estimated on repeatedly new generated data. Hence, the theoretical standard error could be obtained, by computing the sample variance of these fixed effects and taking the square root.

Implementation

The function implementing the presented approach was called `theosd`. It's parameters were the number of iterations to perform, and an integer indicating the data situation to simulate. The return value was a matrix containing all estimations of the fixed effects. A main advantage of returning all estimations was the extensibility. The function could e.g. be executed parallel, with individual results being combined subsequently. Altogether, 1000 runs were performed.
The full `R`-code and description of the function `theosd` can be found in appendix A.1.

Results

The obtained estimations of $sd(\hat{\boldsymbol{\beta}})$ for each simulation are given in table 4.3.

	$sd(\hat{\beta}_0)$	$sd(\hat{\beta}_1)$	$sd(\hat{\beta}_2)$	$sd(\hat{\beta}_3)$	$sd(\hat{\beta}_4)$	$sd(\hat{\beta}_5)$
Full Data	3.589	0.088	0.018	1.858	0.149	0.315
Complete Case	5.675	0.149	0.041	2.879	0.267	0.485
Multiple Imputation	4.056	0.102	0.022	2.105	0.249	0.462

Table 4.3.: Theoretical standard errors of the fixed effects for the three simulations.

4.5. Simulation I: No Missing Values

The first simulation was performed on a complete data set. It serves the purpose of providing a reference to more complex situations, where strategies on similar, yet incomplete data sets are analyzed.

4.5.1. Overview

The main goal of this simulation was to examine the quality of the GLMM estimation. Therefore, the empirical bootstrap distribution of the model parameters was reviewed, considering the theoretical values used for creating the data.

One simulation run comprised the following steps:

(1) Construct a data set using specified estimators $\boldsymbol{\theta}$.

(2) Create bootstrap samples from this data according to the method described in section 2.3.

(3) Compute GLMMs on each bootstrap sample and extract the estimators of interest.

(4) Generate 95% bootstrap percentile intervals of the estimations from the models.

(5) Examine, if the true value of every estimator is held by the respective interval.

After repeating this simulation run 1000 times, the count of intervals holding their respective true value was obtained.

The setup of one simulation run is illustrated in figure 4.9.

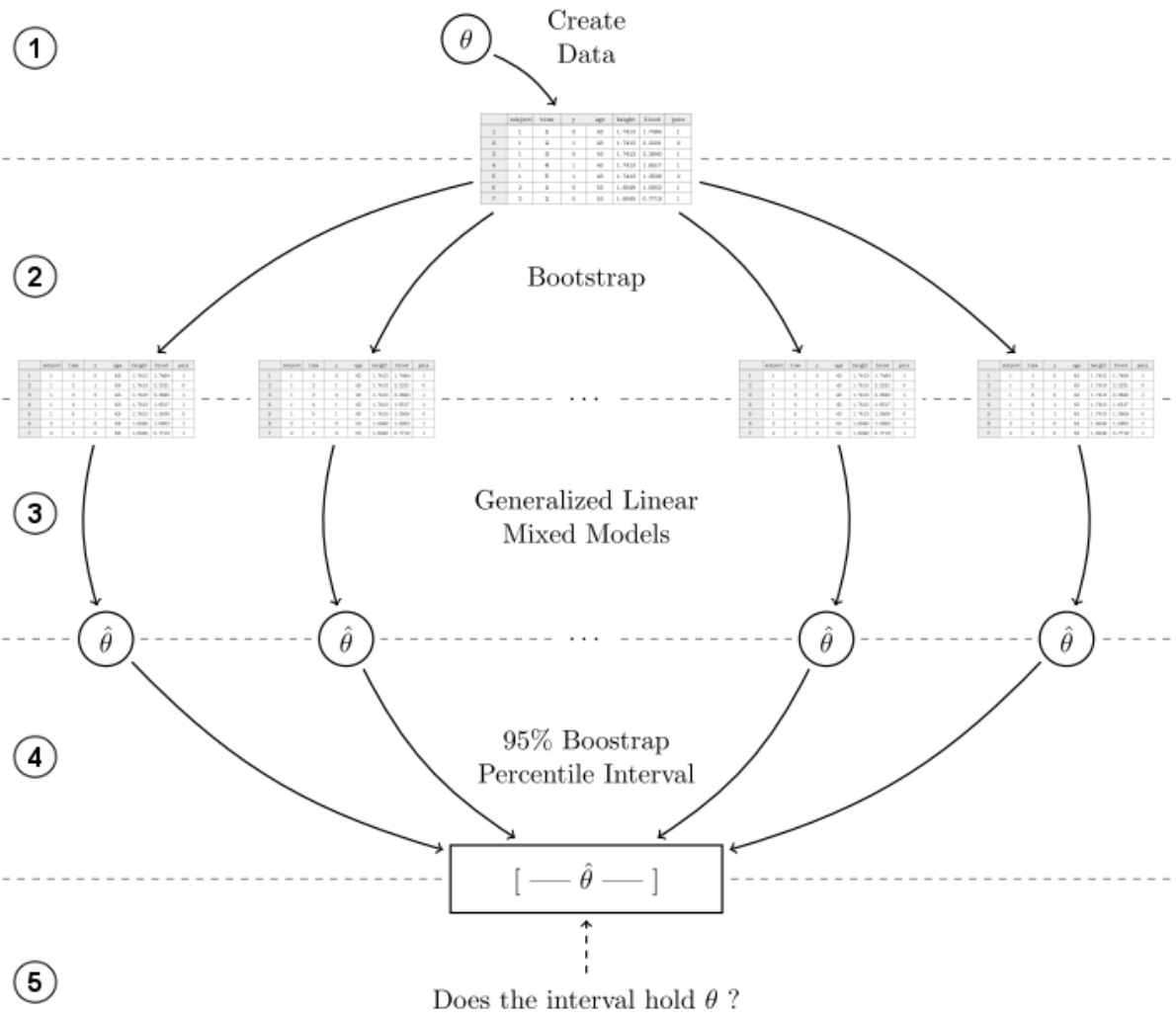


Figure 4.9.: Schematic illustration of one run of Simulation I. Descriptions on the circled numbers can be found in the text.

4.5.2. Implementation

To realize 1000 simulation runs, parallel computing by the `R` package `snow` was used. A short introduction to this package is given in appendix A.2. The main parts of the simulation code, apart from the parallelized framework are presented in the following.

One simulation run was implemented with the function `step()`. It starts with setting up a matrix, where all computed parameters will be written to in the end. After that, a data set is created and B Bootstrap samples are drawn from it.

```
step = function(B){
    Pm = matrix( nrow = B, ncol = 13 )
    DB = bootstrap( create.D(), B )
    ...
```

A `for()` loop is used to loop through all bootstrap samples. For every entry of the list, a GLMM is fitted and all estimators of interest are stored in a row of the parameter matrix. Those estimators are the fixed effects of the covariables, the standard deviation of the random intercept and the estimated standard errors of the fixed effects.

```
    ...
    for(i in 1:B){
      glmmi = glmer( y~time+age+height+blood+pain+(1|subject),
                     data = DB[[i]], family = binomial() )
      Pm[i, 1:6]  = fixef(glmmi)
      Pm[i, 7]    = sqrt(as.numeric(VarCorr(glmmi)$subject))
      Pm[i, 8:13] = summary(glmmi)$coefficients[,2]
    }
    ...
```

The final step is to return the completed parameter matrix `Pm`.

```
    ...
    return(Pm)
}
```

Note, that no simulation run computes bootstrap percentile intervals, or checks if the true parameters were reproduced. Instead, the full matrix of parameters is returned. The idea behind this was to store every information obtained, which enabled further analysis of the estimated parameters. The theoretical simulation steps (4) and (5) were

performed afterwards, using the function `bpi()` (see appendix A.3).

To implement the full simulation, the `step()` function was repeated by a `for()` loop. The results were stored in a list object.

```
1 FD = list()
2 for(i in 1:1000) FD[[i]] <- step(200)
```

Using parallel computing, the CPU-time of this function was approximately 5 hours. (20 processor cores, 1000 iterations, 200 bootstrap samples and 500 data observations)

4.5.3. Results

Additionally to the bootstrap percentile intervals, the overall bootstrap distribution of the obtained parameters was analyzed. Figure 4.10 illustrates density estimations for one fixed effect ($\hat{\beta}_1$), its estimated standard error ($\widehat{\text{sd}}(\hat{\beta}_1)$), and the standard deviation of the random intercept .

The distribution of the random intercept standard deviation and the standard error of $\hat{\beta}_1$ were both skewed. The latter was a mixture distribution, having a point mass at 0. The fixed effects followed a normal distribution with mean β_1.
All specified true values seemed to be covered very well by the empirical densities. Yet, to clarify the performance of the estimation, it was necessary, to evaluate the number of 95% bootstrap percentile intervals, that held the respective true value.

The function implementing this was called `bpi()`. It is described in appendix A.3.

Tabular 4.4 shows the percentage of the respective intervals holding the true value for each parameter.

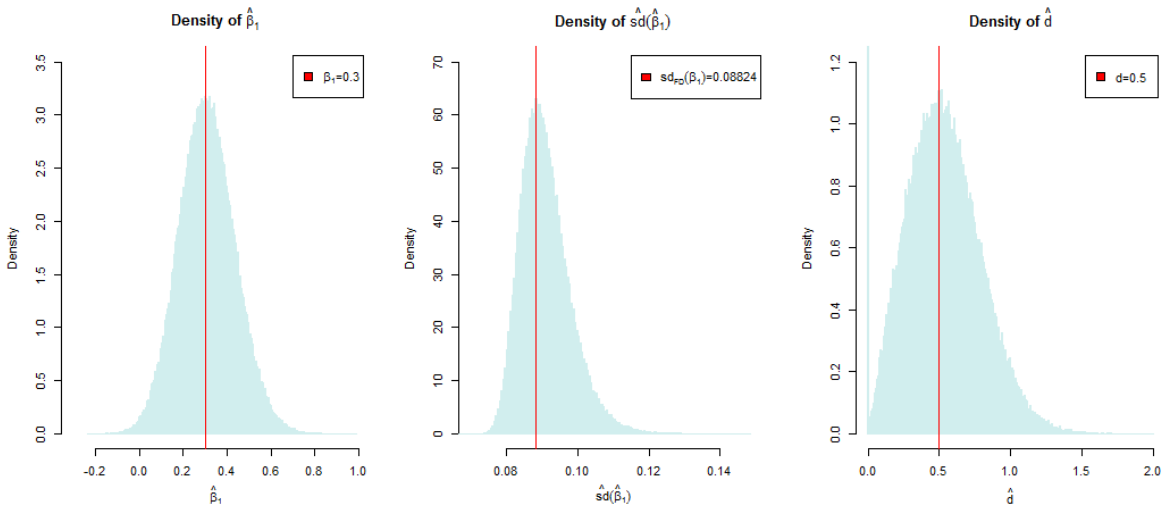


Figure 4.10.: Full Data Simulation: Empirical distribution of $\hat{\beta}_1$, $\widehat{\text{sd}}(\hat{\beta}_1)$ and $\hat{d}$ using the parameter estimations of all 1000 simulation runs.

	(Intercept) $\hat{\beta}_0$	time $\hat{\beta}_1$	age $\hat{\beta}_2$	height $\hat{\beta}_3$	blood $\hat{\beta}_4$	pain $\hat{\beta}_5$	(subject) $\hat{d}$
$\theta \in [\text{-}\hat{\theta}\text{-}]$	93.0	93.5	93.2	93.4	93.7	94.0	89.7

	(Intercept) $\widehat{sd}(\hat{\beta}_0)$	time $\widehat{sd}(\hat{\beta}_1)$	age $\widehat{sd}(\hat{\beta}_2)$	height $\widehat{sd}(\hat{\beta}_3)$	blood $\widehat{sd}(\hat{\beta}_4)$	pain $\widehat{sd}(\hat{\beta}_5)$
$\theta \in [\text{-}\hat{\theta}\text{-}]$	95.0	96.6	94.7	95.3	96.7	96.6

Table 4.4.: Simulation I: Percentage of 95% bootstrap percentile intervals holding the respective true value.

A Wald confidence interval for the expected number of holds after 1000 runs can be

computed by

$$1000 \cdot 0.95 \pm \Phi(0.975)\sqrt{1000 \cdot 0.95 \cdot 0.05} \quad \approx \quad [\ 936.5\ ,\ 963.5\ ] \tag{4.26}$$

with $\Phi(\cdot)$ denoting the distribution function of $\mathcal{N}(0, 1)$.

It could be observed, that the percentages of holds were lower than the expected 95% for all fixed effects. The smallest value was found at the standard deviation of the random intercept with 89.7%. These small percentages are a typical side-effect of the bootstrap percentile method. Therefore, they could also be expected to occur in the more complex simulations later. The intervals for the standard errors of the fixed effects on the other hand tended to hold the true value more often than expected here. Nevertheless, most of these percentages did not differ widely from the calculated Wald interval.

To further analyze the bootstrap percentile intervals, they were plotted with the respective true values. An illustration for the first 100 simulation runs is shown in figure 4.11. The full plot of all 1000 runs is given in appendix B.3. It can be used to find structural errors, or to examine the confidence interval sizes.

After reviewing this plot, there seemed to be no abnormalities in the structure. Intervals, that did not hold the true value evenly distributed below and above of it. For the fixed effects, intervals were mostly of similar size throughout the simulation, whereas intervals for the standard errors tended to differ more widely in size over the runs.

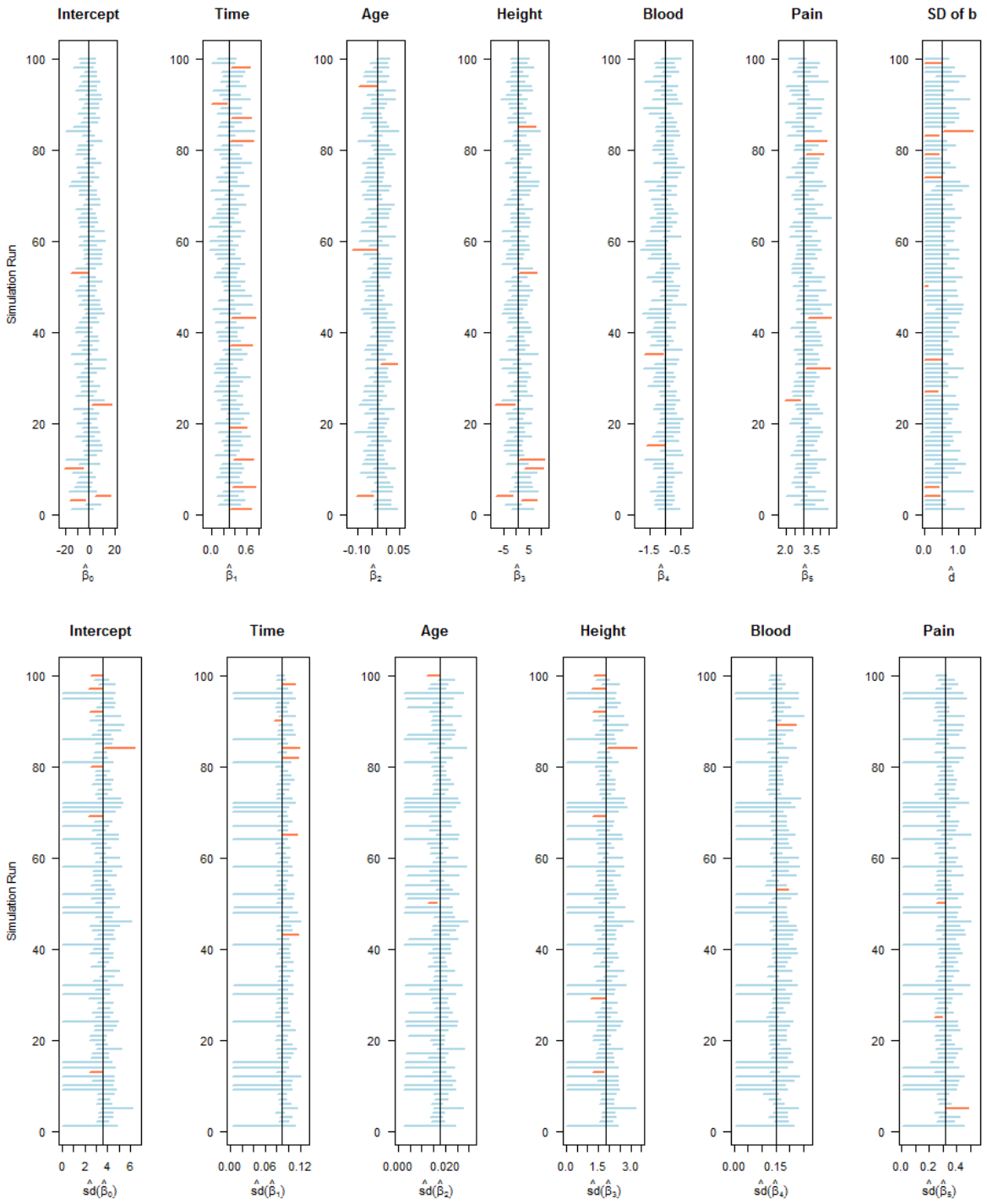


Figure 4.11.: Full Data Simulation: 95% bootstrap percentile intervals of all parameters of interest for the first 100 simulation runs. Intervals are colored blue if they hold the true value (black line), and red if they don't.

4.6. Simulation II: Complete Case Analysis

The second simulation exacerbated the estimation process, by incorporating missing values into the data. One strategy often applied in such data situations is the complete case analysis. Listwise deletion is performed, removing every row, that contains a missing value. Analyses are then performed on the resulting subset of fully observed individuals.

4.6.1. Overview

The idea of this simulation, similarly to the full data case, was to analyze the empirical bootstrap distribution of the model parameters to assess the quality of the estimation after listwise deletion.

One simulation run can be summarized as follows.

(1) Construct the data set using the true parameters $\boldsymbol{\theta}$.

(2) Induce Missing values to the data according to section 4.2.

(3) Perform a bootstrap on the incomplete data set .

(4) Apply listwise deletion on every bootstrap sample to address the problem of incomplete data.

(5) Compute GLMMs on each data subset and extract the estimators of interest.

(6) Create 95% bootstrap percentile intervals of the extracted estimators.

(7) Examine, if the true value for every estimator is held by the respective interval.

After 1000 runs, the number of intervals holding the true parameters could be analyzed.

The setup of a simulation run is illustrated in figure 4.12.

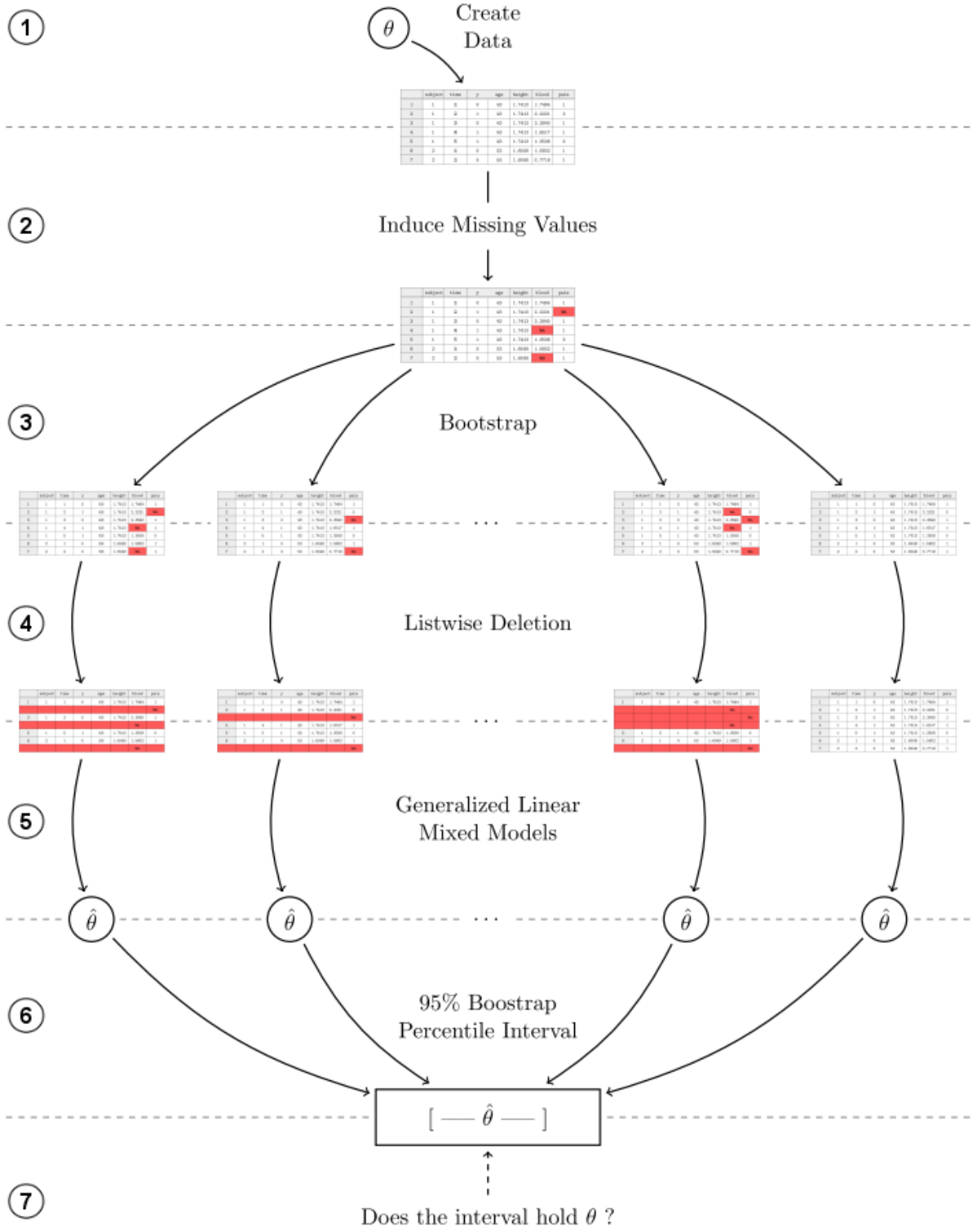


Figure 4.12.: Schematic illustration of one run of Simulation II. Descriptions on the circled numbers can be found in the text.

4.6.2. Implementation

This section presents the main elements of the R-code implementing the complete case simulation.

One run was performed by a function named `step()`. It starts with initializing a matrix, where all parameters to be extracted from the models will be stored in. After that, a data set is created and missing values are incorporated to variable 7 (`pain`) and variable 6 (`blood`). The constructed incomplete data set is then bootstrapped, resulting in a list of bootstrap samples.

```
1 step = function(B){
2     Pm = matrix(nrow = B, ncol = 13 )
3     Dat = create.miss(create.D(),0.3,7,6)
4     Dat = create.miss(Dat,0.4,6,4)
5     DB = bootstrap(Dat,B)
6     ...
```

A `for()` loop is used to iterate through all elements of `DB`.

Each row of the current bootstrap sample is checked for containment of `NA`s and as the case may be deleted (listwise deletion). Practically, when applying the function `is.na()` to a data set, a boolean matrix was returned indicating, if a value was not observed. Adding up the rows of this matrix resulted in a vector containing 0s (for fully observed rows) and values > 0 (for rows, that contained `NA`s). Finally, appending the logical comparison "`==0`" to this expression returned a boolean vector, indicating rows without missing values. The data was then reduced to these completely observed rows.

```
1     for(i in 1:B){
2       CC = DB[[i]][apply(is.na(DB[[i]]),1,sum)==0,]
3       ...
```

The following steps are similar to the previous simulation.
GLMMs are fitted on the data and parameters are extracted and stored in the parameter matrix. A simulation run ends by returning the completed matrix `Pm`.

```
1       ...
2       glmmi = glmer(y~time+age+height+blood+pain+(1|subject),
3                     data = CC, family = binomial())
4       Pm[i, 1:6] = fixef(glmmi)
5       Pm[i, 7] = sqrt(as.numeric(VarCorr(glmmi)$subject))
6       Pm[i, 8:13] = summary(glmmi)$coefficients[,2]
7     }
8     return(Pm)
9 }
```

[6]

The overall simulation was implemented, by repeating the `step()` function 1000 times and storing the results in a list object.

```
1 CC = list()
2 for(i in 1:1000) CC[[i]] = step(200)
```

To ensure convergence, an extension of the `glmerControl` option was necessary [6]. This lead to a notable increase of the CPU-time compared to the first simulation.
The duration of executing the complete case simulation was approximately 13 hours.

4.6.3. Results

The first part of the analysis concerns the overall bootstrap distribution of the parameters. It is illustrated on the same parameters as in the previous simulation ($\hat{\beta}_1$, $\widehat{\text{sd}}(\hat{\beta}_1)$

[6] Excluded in this code sample for ease of exposition: The severe reduction of observations required an additional adjustment of the glmer parameters to ensure convergence:
`control=glmerControl(optimizer="bobyqa", optCtrl = list(maxfun = 100000))`

and $\hat{d}$) in figure 4.13. The respective true values for the full data situation are given as vertical red lines. The plot for the standard error of $\hat{\beta}_1$ also includes the true value for the complete case situation as a vertical blue line.

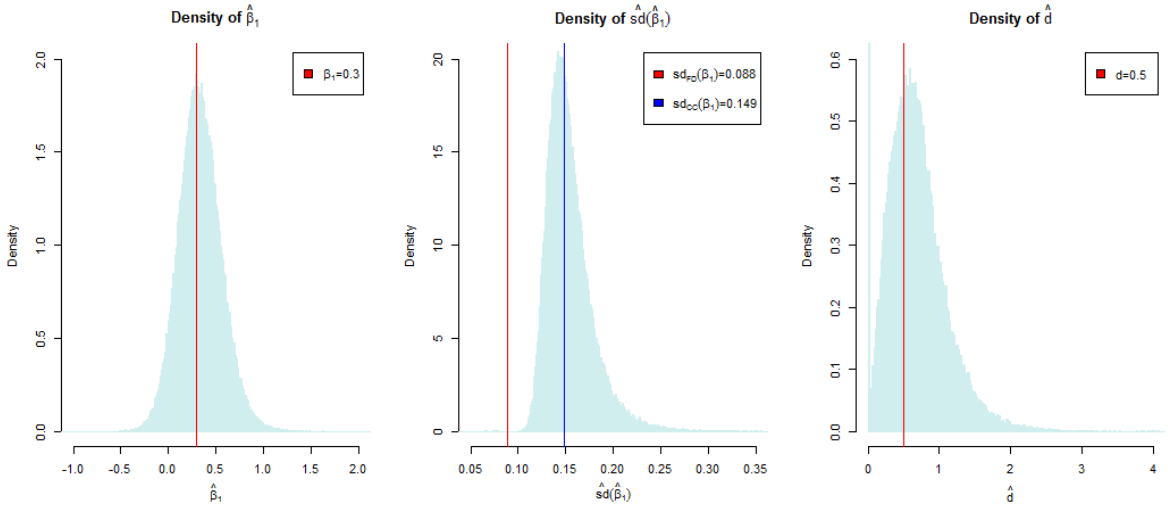


Figure 4.13.: Complete Case Simulation: Empirical distribution of $\hat{\beta}_1$, $\widehat{\text{sd}}(\hat{\beta}_1)$ and $\hat{d}$ using the parameter estimations of all 1000 simulation runs.

Similarly to the first simulation, the density of $\hat{\beta}_1$ still seemed to be centered around the true value. Yet, an increase of the standard error became apparent, when inspecting the second plot. The distribution of $\widehat{\text{sd}}(\hat{\beta}_1)$ was located mostly above the full data value. This resulted from reducing the size of the data set by performing listwise deletion. Models were fitted on fewer observations, which increased the uncertainty of the parameter estimations. The same applied to the estimators for d.

For the second part of the analysis, the number of 95% bootstrap percentile intervals, that hold the respective true value were examined. (Standard error intervals were tested for holding the true value for the complete case analysis.)

The results are given in table 4.5.

	(Intercept) $\hat{\beta}_0$	time $\hat{\beta}_1$	age $\hat{\beta}_2$	height $\hat{\beta}_3$	blood $\hat{\beta}_4$	pain $\hat{\beta}_5$	(subject) $\hat{d}$
$\theta \in [\text{-}\hat{\theta}\text{-}]$	92.1	92.9	91.9	92.2	92.2	89.8	87.6

	(Intercept) $\widehat{sd}(\hat{\beta}_0)$	time $\widehat{sd}(\hat{\beta}_1)$	age $\widehat{sd}(\hat{\beta}_2)$	height $\widehat{sd}(\hat{\beta}_3)$	blood $\widehat{sd}(\hat{\beta}_4)$	pain $\widehat{sd}(\hat{\beta}_5)$
$\theta \in [\text{-}\hat{\theta}\text{-}]$	98.6	98.6	95.7	98.7	98.3	98.6

Table 4.5.: Simulation II: Percentage of 95% bootstrap percentile intervals holding the respective true value.

Similar to the results for the full data situation, the observed percentages of holds were lower than 95% for all fixed effects. Here, this effect seemed to be even stronger, which may result from the severe data reduction by performing listwise deletion. The smallest number of fixed effects holds was observed at the binary variable `pain`. Overall, the lowest percentage was spotted on the standard deviation estimate for the random intercept. Contrary, all intervals for the standard error estimates of the fixed effects were holding the computed true value more often than 95%. These tendencies were also observed in the previous simulation.
Altogether, no severe irregularities of the complete case results could be detected here.

An illustration of the individual bootstrap percentile intervals for 100 simulation runs is given in figure 4.14. It could be used to discover potential abnormalities or structural errors.
Substantially, most plots looked similar to the corresponding ones from simulation I. Yet, intervals were generally larger than before. Inspecting the plot of all 1000 runs (Figure B.4) revealed, that the incomplete variable `blood` underestimated it's true effect more often, while the other incomplete variable `pain` compromisingly overestimated it. Therefore, the incorporated missingness, which followed the MAR assumption, seemed to have induced structural errors for the estimation process.

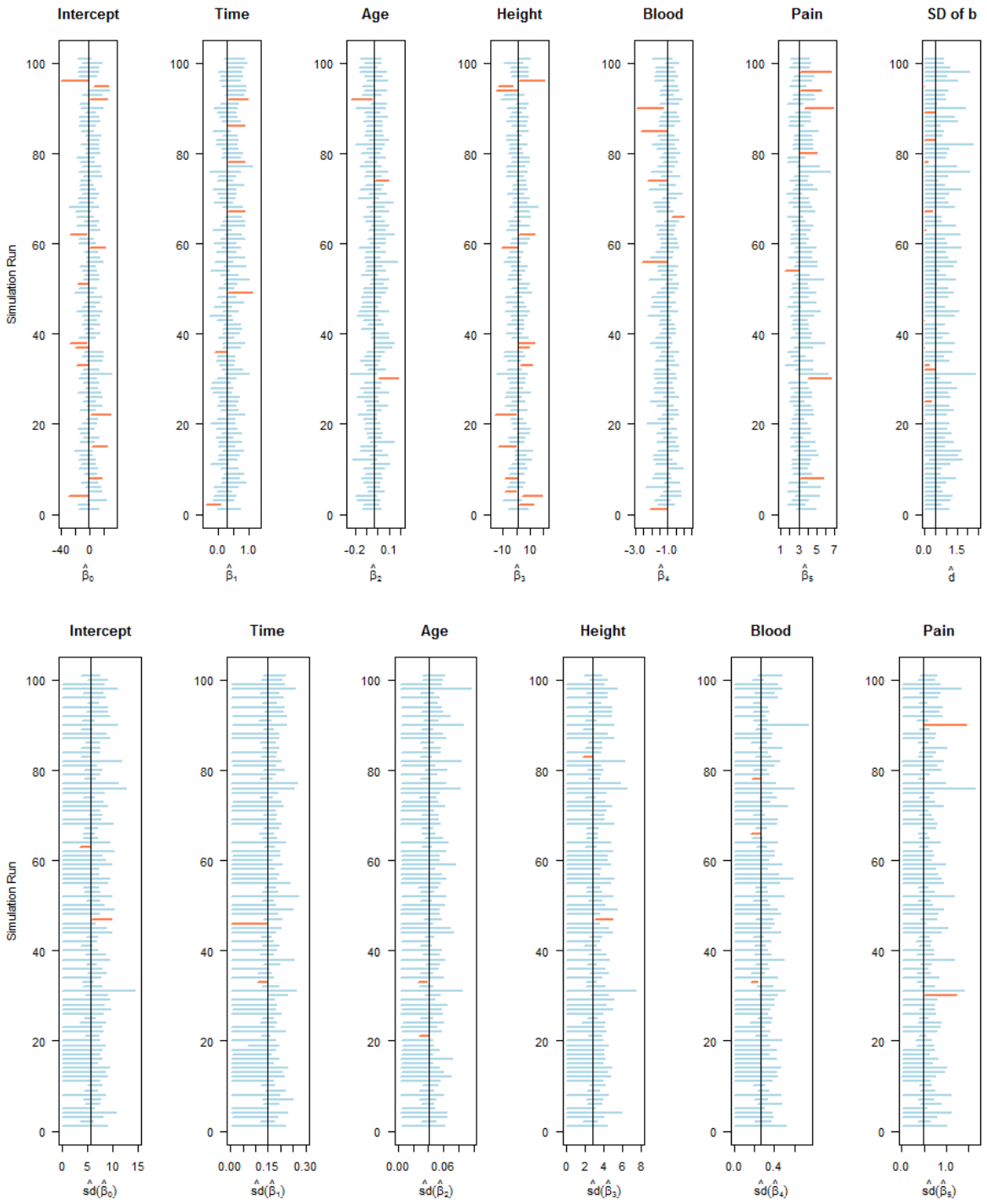


Figure 4.14.: Complete Case Simulation: 95% bootstrap percentile intervals of all parameters of interest for 100 simulation runs. Intervals are colored blue if they hold the true value (black line), and red if they don't.

4.7. Simulation III: Multiple Imputation (Amelia II)

The initial situation of the third simulation was again an incomplete data set. This time however, multiple imputation by Amelia II was performed, to approach the problem of missing values. The main goal was to examine the quality of GLMM estimations in this framework, with additional regard to the combination method of the random intercept standard deviation in the multiple imputation.

4.7.1. Overview

Analogously to the previous simulations, the empirical bootstrap distribution for the parameters of interest were analyzed. However, the computational complexity increased immensely here, due to the fact, that GLMMs needed to be fitted on every imputed data set.

An overview on the structure of one simulation run is given by the following steps:

(1) Build a data set using the true parameters $\boldsymbol{\theta}$.

(2) Induce missing values to this data set according to section 4.2.

(3) Create 200 bootstrap samples of the incomplete data set.

(4) Perform multiple imputation on every bootstrap sample to obtain 5 completed data sets each.

(5) Fit a GLMM on every imputed data set and extract the estimators of interest

(6) Combine the estimators of a bootstrap sample by a suitable procedure

(7) Use the combined estimators to create 95% bootstrap percentile intervals

(8) Examine, if the true value of the estimators is held by the respective interval

A schematic illustration of one simulation run is given by figure 4.15

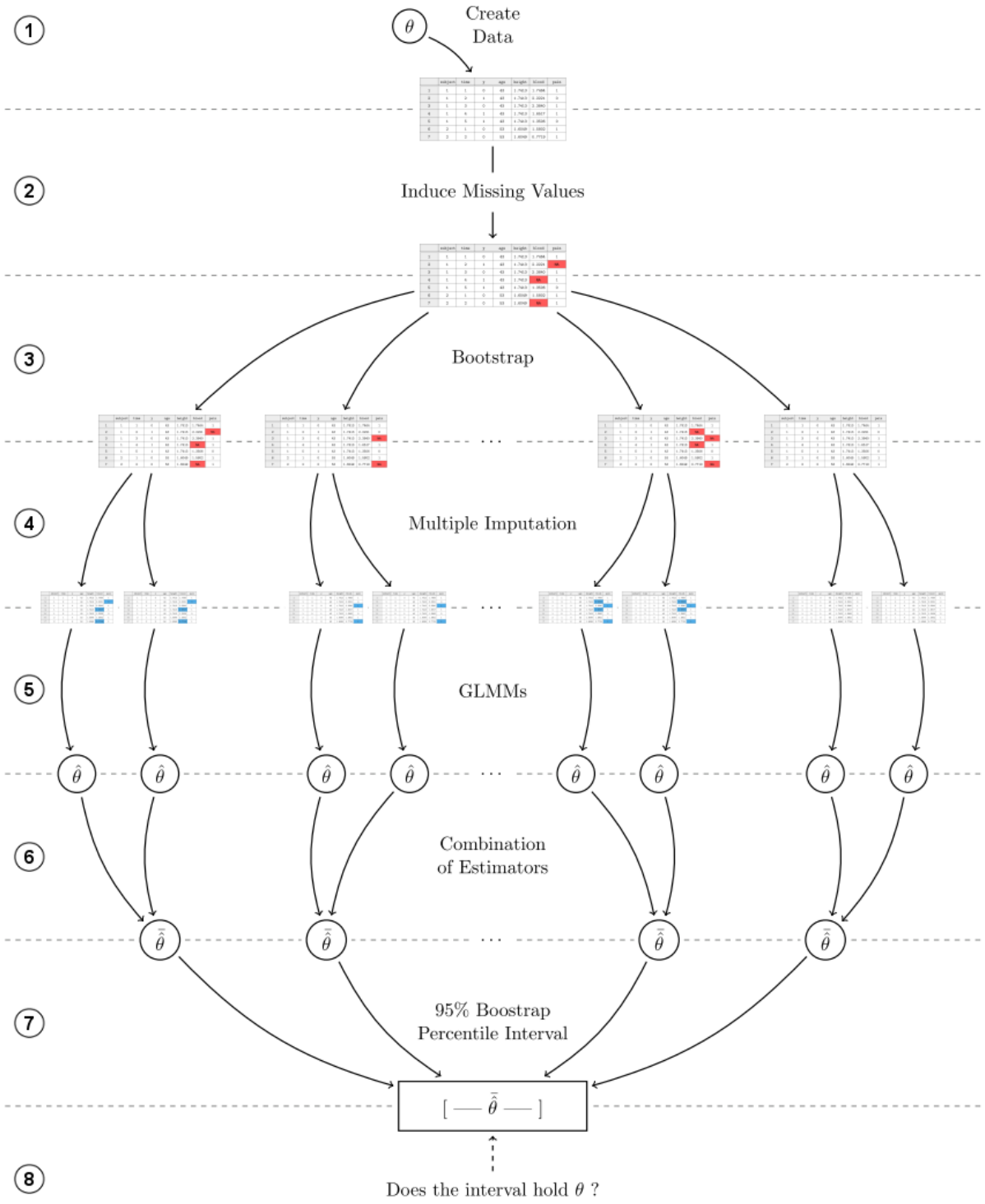


Figure 4.15.: Schematic illustration of one run of Simulation III. Descriptions on the circled numbers can be found in the text.

4.7.2. Implementation

The code to implement one multiple imputation run was subdivided, by outsourcing the model estimation to a function called `ameliaglmm()`.
It computes GLMMs for a list of completed datasets and extracts the parameters of interest. These are then stored in a matrix named `Imp`. The function ends by returning the parameter matrix.

```
1  ameliaglmm = function(ImpList){
2      Imp = matrix(nrow=length(Imp), ncol=13 )
3      for(m in 1:length(Imp)){
4        glmm = glmer(y~time+age+height+blood+pain+(1|subject),
5                     data = ImpList[[m]], family = binomial())
6        Imp[m,] = c( fixef(glmm),
7                     sqrt(as.numeric(VarCorr(glmm)$subject)),
8                     summary(glmm)$coefficients[,2] )
9      }
10     return(Imp)
11 }
```

The main simulation function `step()` is widely similar to the previously described ones. Yet, the return value has to differ here. In simulations I and II, one GLMM was fitted for a bootstrap sample, resulting in a vector of estimations. Now, after the multiple imputation procedure, 5 models are calculated per bootstrap. Their estimations are stored in a matrix. Because of that, the return object of the `step()` function was chosen to be a parameter list, containing all matrices returned by the `ameliaglmm()` function.

```
1  step = function(B){
2      Pl = list(length=B)
3      ...
```

The `step()` function proceeds by creating a data set and incorporating missing values. Variable 7 (`pain`), with dependency on variable 6 (`blood`) and variable 6, with dependency on variable 4 (`age`) were set to be the targets for the function `create.miss()`.

```
1    ...
2    Dat = create.miss( create.D(), 0.3, 7, 6 )
3    Dat = create.miss( Dat, 0.4, 6, 4 )
4    ...
```

The incomplete data set is now bootstrapped. The samples are stored in the list `DB`.

```
1    ...
2    DB = bootstrap( Dat, B )
3    ...
```

Multiple imputation is performed on every element of `DB`, by invoking `amelia()` inside an `lapply()` call. Parameters of `amelia()` are described in detail in section 3.6. The imputed datasets for each bootstrap sample are stored in a list named `AM`.

```
1    ...
2    AM = lapply( DB, function(x)
3                        amelia(x, m=5, ts="time", cs="subject",
4                               p2s=0, polytime=1)$imputations
5                )
6    ...
```

The final step is to fit GLMMs on every data set and extract their parameter estimations. This was realized with the function `ameliaglmm()`, surrounded by a `for()` loop, which iterates through the bootstrap samples. All results are stored in the parameter list `Pl`,

which is returned after the completion of the loop.

```
    ...
    for(i in 1:B) Pl[[i]] = ameliaglmm(AM[[i]])
    return(Pl)
}
```

The full simulation was performed, by iterating the `step()` function. Results of every run were saved to the list `MI`.

```
MI = list()
for(j in 1:1000) MI[[j]] = step(200)
```

The deeply nested composition makes the output value `MI` complex to oversee at first sight. Because of that, the structure of the returned list is illustrated in figure 4.16.

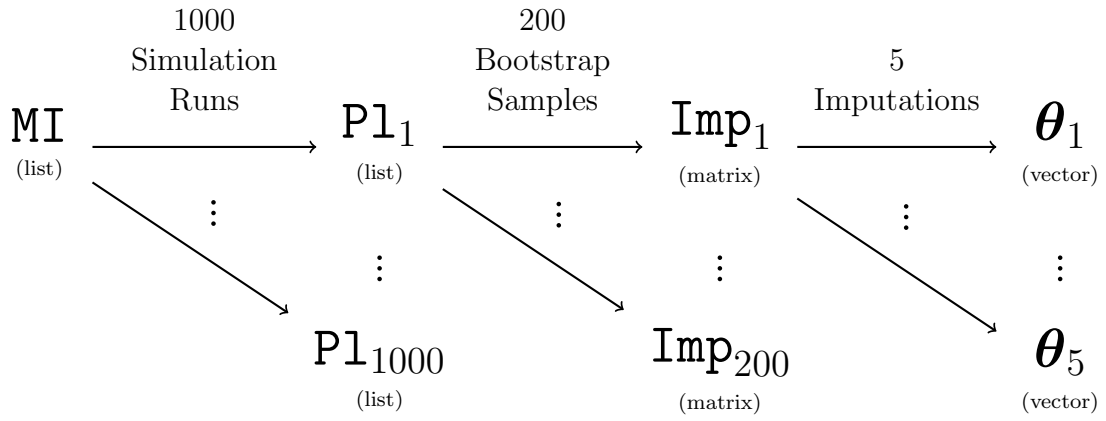


Figure 4.16.: Structure of the list `MI` returned by simulation III. The vector $\boldsymbol{\theta}_i$ contains estimations for $\boldsymbol{\beta}$, d and $sd(\boldsymbol{\beta})$.

The CPU-time of this simulation was approximately 26 hours.

4.7.3. Results

The first result to be analyzed was the empirical distribution of the estimated parameters. For the fixed effect β_1 and the corresponding standard error estimate, they are illustrated in figure 4.17.

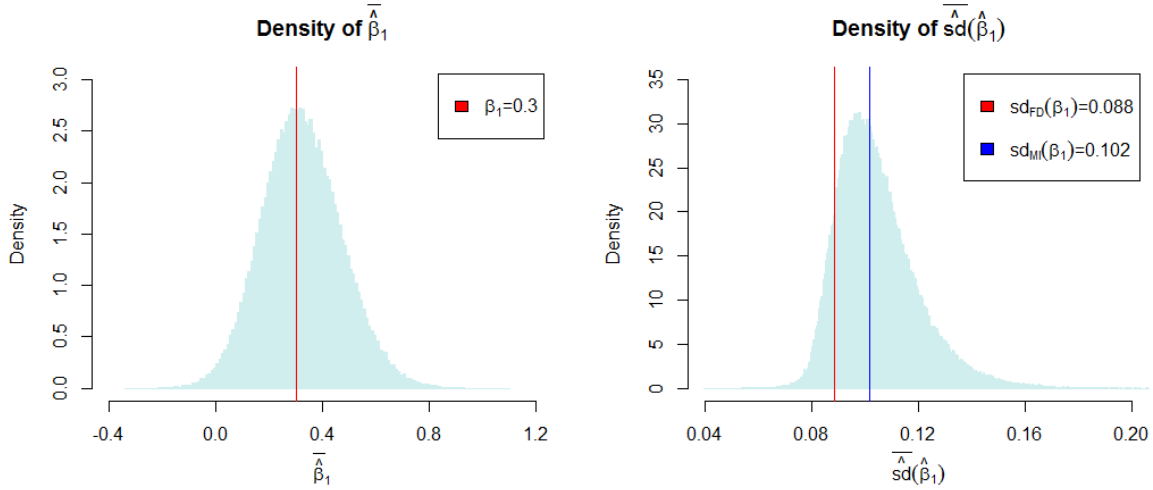


Figure 4.17.: Multiple Imputation Simulation: Empirical distribution of $\hat{\beta}_1$ and $\widehat{sd}(\hat{\beta}_1)$ using the parameter estimations of all 1000 simulation runs.

The density of the fixed effect seemed to be centered around the true value. This might also apply to the standard error density of $\hat{\beta}_1$, when comparing it to the computed theoretical value of the multiple imputation framework (blue line). The true value of the full data situation (red line) was approximately 0.014 smaller and lay clearly under the mean of the density in visual inspection. Nevertheless, to make sound statements on the quality of the estimation, the number of bootstrap percentile intervals holding the respective true value needed to be examined.

In this simulation, the estimator for the standard deviation of the random intercept was of special interest. Whereas combining methods for fixed effects and their variances exist as described in section 3.2, the rules on how to properly combine random effects

variance estimates after multiple imputation have not been thoroughly researched up to this point. Here, the main methods to be compared were the mean and median of the estimators. Other options, that were analyzed in this context can be found in appendix A.4.

The empirical densities of both methods are illustrated in figure 4.18.

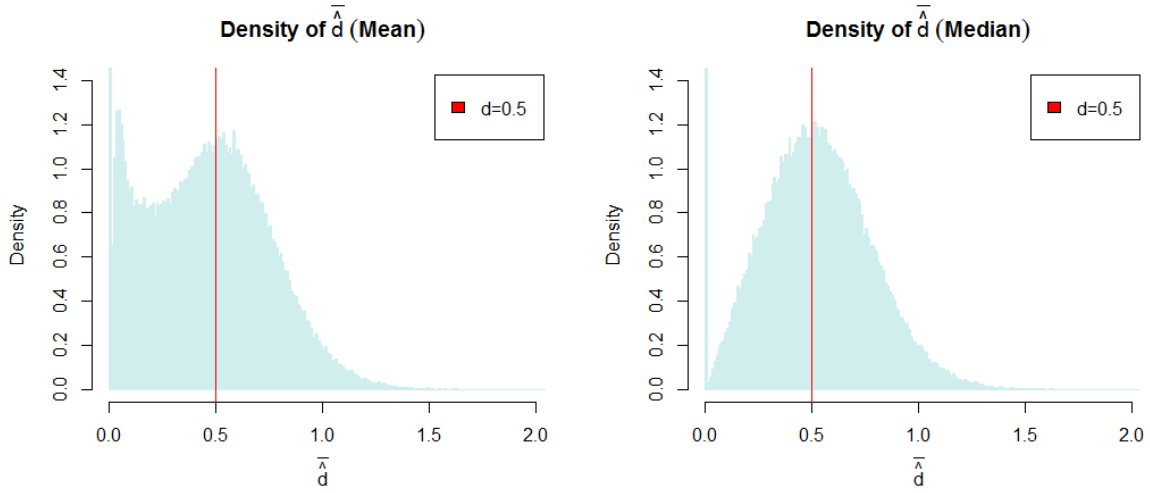


Figure 4.18.: Multiple Imputation Simulation: Comparison of the empirical densities for $\bar{\hat{d}}$ using the mean and median to combine the estimators of the imputations.

Both densities looked very similar for values > 0.3. Yet, the density of smaller values differed widely between the methods. When using the mean, these estimations appeared notably more often, while values of 0[7]occurred only half as frequent.

The misrepresentation of small values arises from not taking the underlying mixture distribution of d into account. Estimators can either be zero, or originate from a χ^2-distribution. Taking the mean of this mixture resulted in shifting proportions of the probability mass for zero onto the χ^2-distribution. Typically, this would distort the whole χ^2-distribution closer to zero. But due to the fact, that the estimators to be averaged over were obtained from very similar data sets, small effects were far more likely

[7]The y-axis of 4.18 was truncated to illustrate the structure of the density. Actual density values of 0 reached up to approx 70 here.

to occur alongside zero estimations. Consequently, they got affected more often by this issue. This problem seemed to resolve when using the median. The density was now able to resemble the true underlying mixture of a χ^2-distribution and a point mass at 0 much better.

To verify the validity of the combined model estimations after multiple imputation, the number of bootstrap percentile intervals holding the respective true values were examined. Table 4.6 shows the percentage of holds for the fixed effects, their standard errors and the standard deviation of the random intercept for the analyzed combination methods after 1000 simulation runs.

	(Intercept) $\bar{\hat{\beta}}_0$	time $\bar{\hat{\beta}}_1$	age $\bar{\hat{\beta}}_2$	height $\bar{\hat{\beta}}_3$	blood $\bar{\hat{\beta}}_4$	pain $\bar{\hat{\beta}}_5$	(subject) $\bar{\hat{d}}_{\text{mean}}$
$\theta \in$ [-$\hat{\theta}$-]	95.4	93.1	93.6	95.7	93.3	88.4	92.4

	(Intercept) $\bar{\widehat{sd}}(\hat{\beta}_0)$	time $\bar{\widehat{sd}}(\hat{\beta}_1)$	age $\bar{\widehat{sd}}(\hat{\beta}_2)$	height $\bar{\widehat{sd}}(\hat{\beta}_3)$	blood $\bar{\widehat{sd}}(\hat{\beta}_4)$	pain $\bar{\widehat{sd}}(\hat{\beta}_5)$	(subject) $\bar{\hat{d}}_{\text{median}}$
$\theta \in$ [-$\hat{\theta}$-]	96.8	99.3	98.6	96.7	99.8	98.8	93.4

Table 4.6.: Simulation III: Percentage of 95% bootstrap percentile intervals holding the respective true value.

The number of holds after multiple imputation was similar to the previous simulations. The fixed effects typically tended to have a percentage lower than 95%, whereas the values for the variances were higher. Comparing the covariables, again the binary variable `pain` had the lowest hold count.
Looking at the estimates for the standard deviation of the random intercept, the percentage of holds for the median was 1% higher than for the mean. Yet, both methods yielded good results and did not indicate errors in the combination process.

To further analyze the structure of the estimated intervals, the first 100 were plotted with the respective true value in figure 4.19.

The plots for $\bar{\hat{\beta}}_0$ to $\bar{\hat{\beta}}_3$ showed no irregularities or structural errors. However, the estimators of the variable `blood` seemed to underestimate the true value, while the variable `pain` got overestimated in a similar amount of times. For both variables, dependent missing values (MAR) were incorporated. The imputation algorithm thus could not thoroughly enable an unbiased estimation. Nevertheless, the number of bootstrap percentile intervals holding the true value did not indicate a severe estimation error here. Furthermore, having a binary variable, which is intrinsically holding very few information, additionally contaminated with missing values exacerbated a precise estimation even more. Small biases of the model parameters are thus not uncommon. They subsequently cause other variables to compensate the error, by inducing an opposing bias. This seemed to have occurred for β_4 and β_5.
Analyzing the plots for the standard errors of the fixed effects, no general irregularities could be found. However, compared to the previous simulations, the intervals differed in their structure. The variation in size between different runs had reduced drastically and the intervals did not include estimations close to 0 anymore. This mainly resulted from the combination method. By generating the final estimate from multiple values, the robustness against extreme values was higher. This applied to both, the size of the interval and the occurrence of near 0 values.
The last part of the illustration shows $\bar{\hat{d}}$ for using the mean and median. The plots of both methods look nearly identical. Yet, when applying the median, slightly more intervals held the true value. Also, the averaged size of the median intervals was larger (0.787) compared to the mean (0.774). Overall, both methods underestimated the true d more often. This incident however occurred in both of the previous simulations as well.

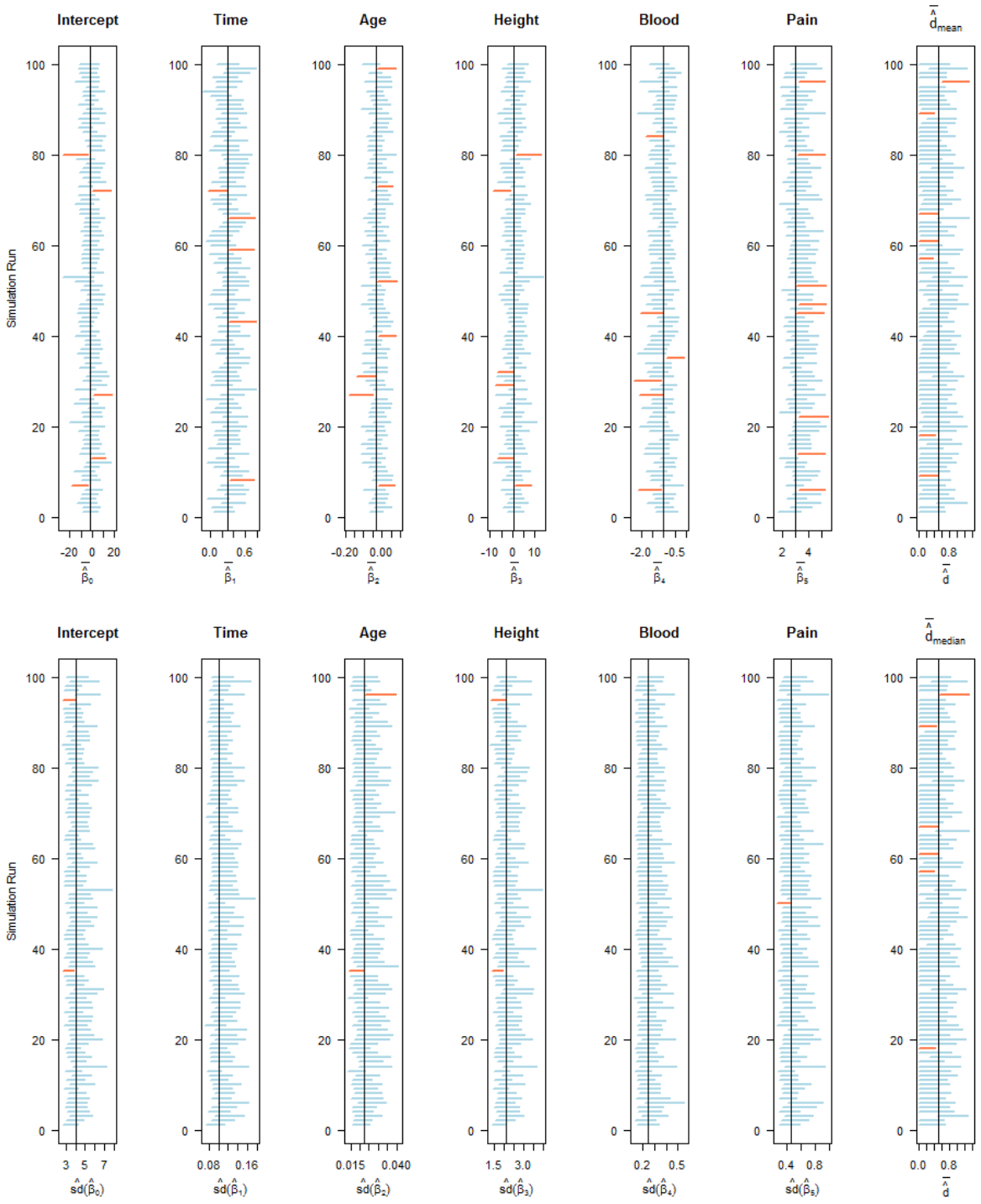


Figure 4.19.: Multiple Imputation Simulation: 95% bootstrap percentile intervals of all parameters of interest for 100 simulation runs. Intervals are colored blue if they hold the true value (black line), and red if they don't.

4.8. Results of Simulation I, II and III in comparison

The last section of this chapter compares the results of simulation I, II and III. Similar to the previous sections, the first part analyzes the empirical densities of the parameters. Figure 4.20 illustrates the obtained curves for $\hat{\beta}_1$ of all three simulations.

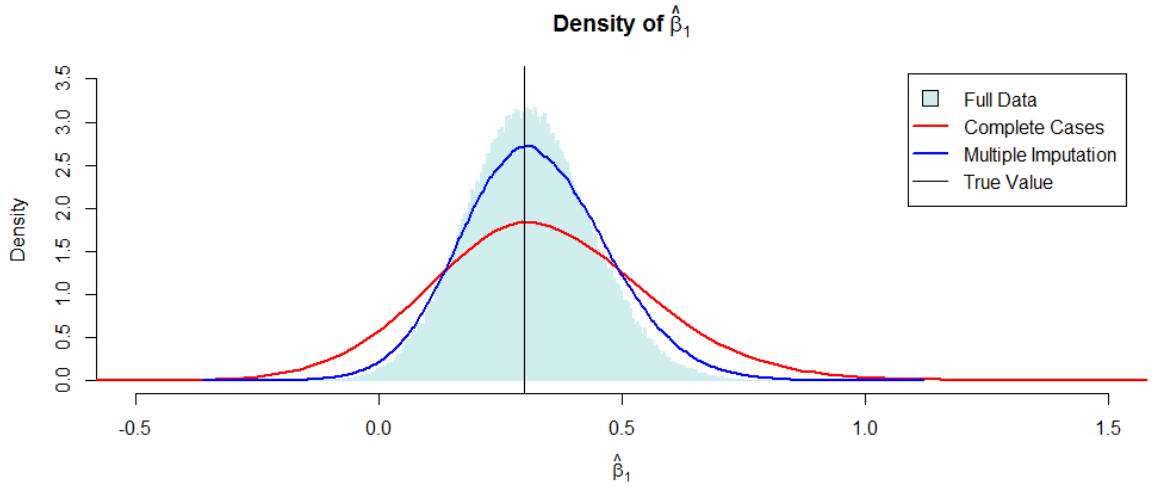


Figure 4.20.: Comparison of the empirical densities for $\hat{\beta}_1$ obtained from simulations I, II and III together with the true value $\beta_1 = 0.3$.

Comparative density plots for the other covariables are given at appendix B.
Like remarked in the individual analyses, all densities seemed to be centered around the true parameters. Yet, multiple differences could be spotted comparing the curves. To further illustrate these, table 4.21 presents basic measures of the densities for $\hat{\beta}_1$ alongside the individual boxplots.

The means of the empirical densities lay slightly over the true value of 0.3 for all simulations. Here, the highest deviation occurred in the complete case simulation with 0.0349. However, more considerable differences could be found in the measures of statistical dispersion. The interquartile range of the complete case simulation had increased by

	Mean	IQR	Range
FD	0.3095	0.1711	1.2201
CC	0.3349	0.2918	14.456
MI	0.3172	0.1998	1.422

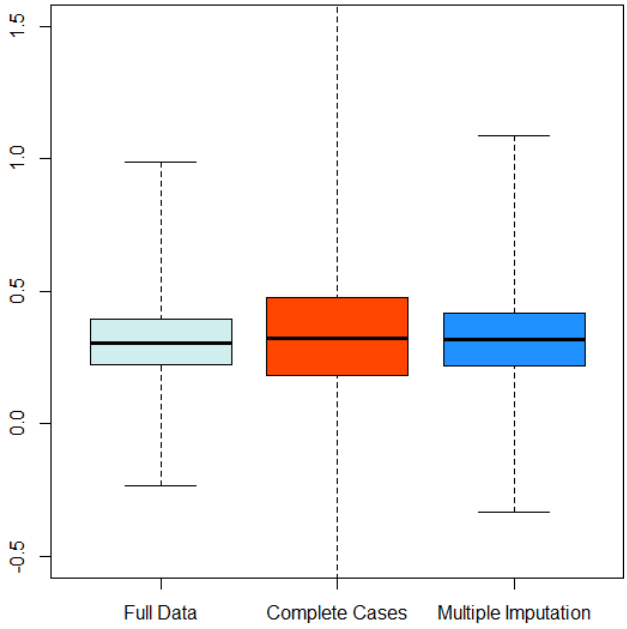


Figure 4.21.: Mean, interquartile range and range alongside the boxplots for the empirical densities of $\hat{\beta}_1$ from the simulations of this thesis.

70% compared to the full data, whereas the multiple imputation only generated a small increment of 17%. Even more discrepancy could be identified, when examining the range of the densities. Here, the value of the complete case analysis stood out. It was more than 12 times higher, than the respective value of the full data situation. The multiple imputation on the other hand only increased the range by 20%.

This analysis showed one of the main advantages of (multiple) imputation. Instead of dropping known information, missing values are imputed, which overall enables parameters to be estimated on more observations. This results in a higher amount of certainty compared to the complete case procedure.

The next part of the analysis compared the estimations for the standard deviations of the random intercepts and additionally went into the influence of the combination method being applied in the multiple imputation case. Figure 4.22 shows all corresponding empirical densities of the simulations.

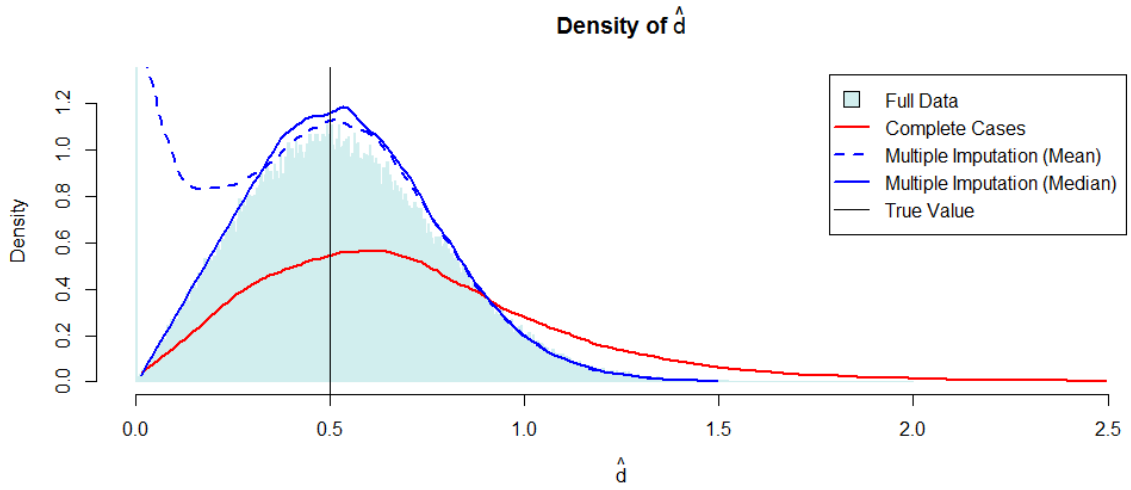


Figure 4.22.: Comparison of the empirical densities for d obtained from simulations I, II and both combination methods of simulation III together with the true value $d = 0.5$.

The negative impact on the precision of the estimation when performing a complete case analysis could be observed in a similar fashion for the parameter d. Furthermore, differences occurred between the combination methods of the multiple imputation. The empirical density of the mean did not match the one obtained from analyzing the full data set. Small values appeared notably more often here. This misrepresentation could be prevented, by using the median to combine the imputed estimators. The corresponding density seemed to match the full data curve in an adequate manner for this method.

The final comparative analysis examined the sizes of the computed bootstrap percentile intervals. For the estimations of β_1, a corresponding histogram is given in figure 4.23.

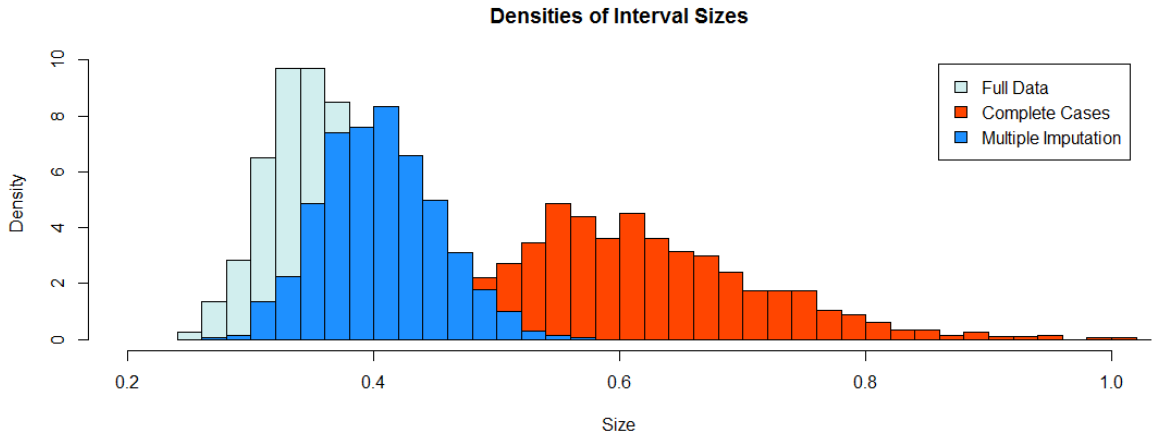


Figure 4.23.: Comparison of the densities for the bootstrap percentile interval sizes in simulations I, II and III for the parameter $\hat{\beta}_1$.

The bootstrap percentile interval sizes correlated with the variances of the respective estimators. While the multiple imputation produced on average only slightly bigger intervals than the full data, the averaged sizes for the complete case analysis were nearly doubled.

In conclusion: With the number of holds being in an acceptable range for all simulations, each method seemed suitable to address the missingness in a GLMM framework. However, due to the fact, that estimators of the multiple imputation were obtained with a much lower variance, this approach could be perceived as superior, compared to the complete case estimation.

5. Summary

The aim of this thesis was to analyze the estimation quality of generalized linear mixed models after multiple imputation. Special notice was given to the combination of the mixed model parameters. Here, various methods were compared and evaluated.

To thoroughly analyze the multiple imputation framework, three simulations on artificial data sets were realized. The first one examined a complete data situation. It served the purpose of obtaining reference values for all measures of the GLMM estimation. The second simulation was performed on incomplete data. Here, the missingness problem was eliminated by using listwise deletion (complete case analysis). The final simulation addressed the incomplete data situation by applying multiple imputation. To combine the random effects parameters of the GLMM, the mean and median method were compared.

None of the three simulations showed structural errors or biased estimations of fixed effects. Consequently, all examined methods are valid approaches to deal with the missingness problem. However, remarkable differences were found for the obtained variance estimates. As expected, the smallest variances occurred at the full data simulation. Only a slightly higher spread ($\approx$ 1.3-fold) of the effects estimates was observed at the multiple imputation simulation. The highest variance of the estimators was found at the complete case analysis ($\approx$ 2-fold). Therefore, the multiple imputation approach has revealed to be superior to the complete case analysis in terms of estimation precision.

The final part of the analysis was to compare the combination methods for the random effect estimates. Here, the mean has failed to resemble the true distribution obtained from the full data analysis. An adequate alternative was given by the median, which implicitly took the underlying mixture distribution of the parameter into account. With no structural errors, or biased estimations, the median has shown to be a suitable method to combine a random effects variance parameter in GLMMs.

A. R Code and Further Descriptions

Appendix A introduces R implementations, which were not explained thoroughly in the main thesis. In addition, further methods and used R packages are described.

A.1. Theoretical variance of β: `theosd()`

The R code of the function to estimate the true variance of $\boldsymbol{\beta}$, which was introduced in section 4.4 will be described in the following.

The function was named `theosd()`. It's parameters are the number of runs to be performed and an integer from 1 to 3 indicating, which data situation to use (Simulation 1, 2 or 3).
First, a parameter matrix is initialized to store the obtained estimations for $\boldsymbol{\beta}$ in.

```
1 theosd = function(Runs,simu){
2    Pm <- matrix(nrow = Runs, ncol = 6 )
3    ...
```

The next part differs, according to which data situation is analyzed. However, each of the code fragments starts with creating a for()-loop to iterate the procedure as many times, as specified by the `Runs` parameter.
For the full data analysis, the function estimates a GLMM on a newly created data set, and stores the obtained β-estimators is the parameter matrix.

```
...
if(simu==1) # Full Data
  for(i in 1:Runs){
    glmmi = glmer(y~time+age+height+blood+pain+(1|subject),
                  create.D(), binomial())
    Pm[i, 1:6] <- fixef(glmmi)
  }
...
```

If the complete case data situation is chosen, the function creates missing values in the data, similar to the second simulation. After that, listwise deletion is applied and a GLMM is fitted. The estimated parameters are then stored in the parameter matrix.

```
...
else if(simu==2) # Complete Case
  for(i in 1:Runs){
    Dat = create.miss(create.D(), 0.3, 7, 6)
    Dat = create.miss(Dat, 0.4, 6, 4)
    CC = Dat[apply(is.na(Dat), 1, sum)==0,]
    glmmi = glmer(y~time+age+height+blood+pain+(1|subject),
                  CC, binomial(), control=glmerControl(
                  optimizer="bobyqa",
                  optCtrl=list(maxfun = 100000)))
    Pm[i, 1:6] <- fixef(glmmi)
  }
  ...
```

The multiple imputation simulation starts by creating missing values. They are afterwards imputed multiple times and combined using the mean function. The final

estimators are stored in the matrix Pm.

```
    ...
    else # Multiple Imputation
      for(i in 1:Runs){
        Dat = create.miss(create.D(),0.3,7,6)
        Dat = create.miss(Dat,0.4,6,4)
        AM = amelia(Dat, p2s=0, m=5 ,ts="time", cs="subject",
                    polytime = 1)$imputations
        imp= matrix(nrow=5,ncol=6)
        for(k in 1:5){
          glmmi = glmer(y~time+age+height+blood+pain+
                        (1|subject), AM[[k]], binomial())
          imp[k,] = fixef(glmmi)
        }
        Pm[i,] = apply(imp,2,mean)
      }
    return(Pm)
}
```

The function ends with returning the filled parameter matrix.
For the analyses of this thesis, 1000 runs of each simulation were performed.

To obtain estimations of the standard errors from the returned β-matrix of truesd() (here named betamatrix), the function sd() needs to be applied for all individual fixed effects estimates.

```
apply(betamatrix, 2, sd)
```

A.2. Parallel computing using the package snow

In this section, a short introduction to the function `clusterCall()` of the `R` package `snow` is given. The basic structure of the parallel computing approach was similar throughout all applications of this thesis.

After loading the package, a cluster object is created, which initializes slave `R` processes. The parallel computing was performed on the servers of the Department of Statistics of the LMU Munich. 10 cores were available on one server, which is why 10 slave processes were initialized.

```
1 cl = makeCluster(10, type="SOCK")
```

To apply a generated function on multiple cores, the call `clusterCall()` is used. Its parameters are the previously created cluster object `cl` and a function to be applied on all defined cores. The essential steps of the inner function are outlined in green.

```
1 X <- clusterCall(cl, function(){
2                         # 1. Load all packages and functions
3                         # 2. Perform the desired task
4                         # 3. Return the results
5                       } )
```

`clusterCall()` returns a list, which contains the return values of each slave process. This list can then be combined into one result for further analyses.
After completing the parallel operation, the function `stopCluster()` is called to shut down cluster processes and clean up remaining connections.

```
1 stopCluster(cl)
```

A.3. Bootstrap Percentil Interval `bpi()`

The simulation functions themselves did not compute bootstrap percentile intervals, but returned the obtained estimations to maintain all information of the estimation process. The evaluation of these estimators was performed afterwards using the function `bpi()`.

The `R` implementation of `bpi()` can be found in the digital supplement. Bootstrap percentile intervals are computed for the estimators of each bootstrap sample. They are consecutively tested for holding the respective true value. In addition to the number of holds, the function returns an estimated interval of expected holds (see equation 4.26) and the average size of the computed bootstrap percentile intervals.

An exemplary output of `bpi()` for the full data simulation results is shown below.

```
$PercentageHolds
   b0    b1    b2    b3    b4    b5     d   sd0   sd1   sd2   sd3   sd4   sd5
0.930 0.935 0.932 0.934 0.937 0.940 0.897 0.950 0.966 0.947 0.953 0.967 0.966

$AbsoluteHolds
 b0  b1  b2  b3  b4  b5   d sd0 sd1 sd2 sd3 sd4 sd5
930 935 932 934 937 940 897 950 966 947 953 967 966

$ExpectedHolds
   Lower    Upper
936.4919 963.5081

$AvgIntervalSizes
         b0          b1          b2          b3          b4          b5           d
13.86595422  0.35066546  0.06910792  7.18559160  0.62174568  1.23108429  0.78744082
        sd0         sd1         sd2         sd3         sd4         sd5
 2.53981287  0.04587011  0.01240716  1.31802747  0.10528110  0.21088747
```

All findings on bootstrap percentile interval holds of chapter 4 can be reconstructed, by applying the `bpi()` function on the simulation results. The prerequisites for this can be found on the digital supplement CD.

A.4. Other discussed combination methods for $\hat{d}$

The main combination methods for the standard deviation parameter of the random effects, that were analyzed in this thesis are the mean and the median. In addition to them, a two-stage design and a probability weighting approach were examined.

A.4.1. Two-Stage Design

To address the underlying mixture distribution of the parameter, the first stage of the combination was to decide, which distribution the final value should belong to. This was realized, by prefixing a query, which returns 0 if more than 50% of the values were 0 (i.e. originate from the point-mass population). Else, all observations greater than 0 (χ^2-population) were combined using the mean.

A code sample for 5 values in the vector `d` would be

```
1 ifelse( sum(d>0)>2 , mean(d[d>0]) , 0 )
```

The resulting density of the combined estimators is shown in figure A.1.

One can see, that compared to the mean, the drastic overestimation of small values got reduced. The curve of the two-stage design was able to better resemble the real density. However, the probabilities for small values were still estimated too high. This approach of combining the estimates hence did not provide a fully satisfying result.

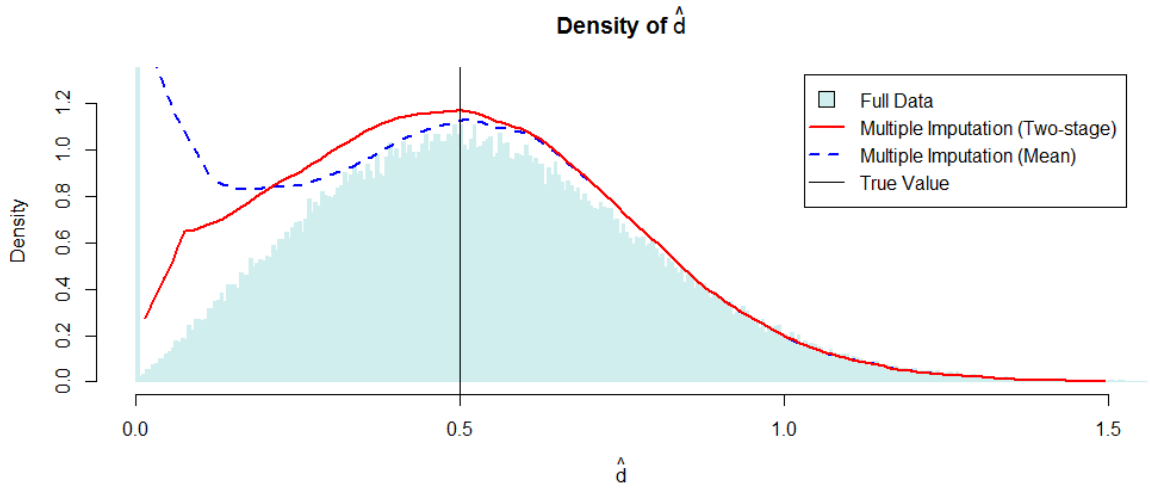


Figure A.1.: Empirical densities of $\hat{d}$ for the two-stage combination method alongside with the previous results.

A.4.2. Probability Weighting Approach

Another approach to imply the mixture distribution, was to add a preliminary stage, which decides the combined value's affiliation to a population by a binary random draw. The probability of this draw is chosen according to the share of the population in the vector, which is averaged over. If e.g. 2 out of 5 values were 0, the probability for the combined value to originate from the population of 0 should be 40%. In 60% of the cases, the combination would be obtained as the mean of the values > 0 (χ^2-population). To implement this procedure in `R`, the following code snippet was used.

```
1 ifelse( rbinom(1,1,mean(d!=0))>0 , mean(d[d>0]) , 0 )
```

The density estimation for the GLMM parameter $\hat{d}$ using the probability weighting approach is illustrated is figure A.2.

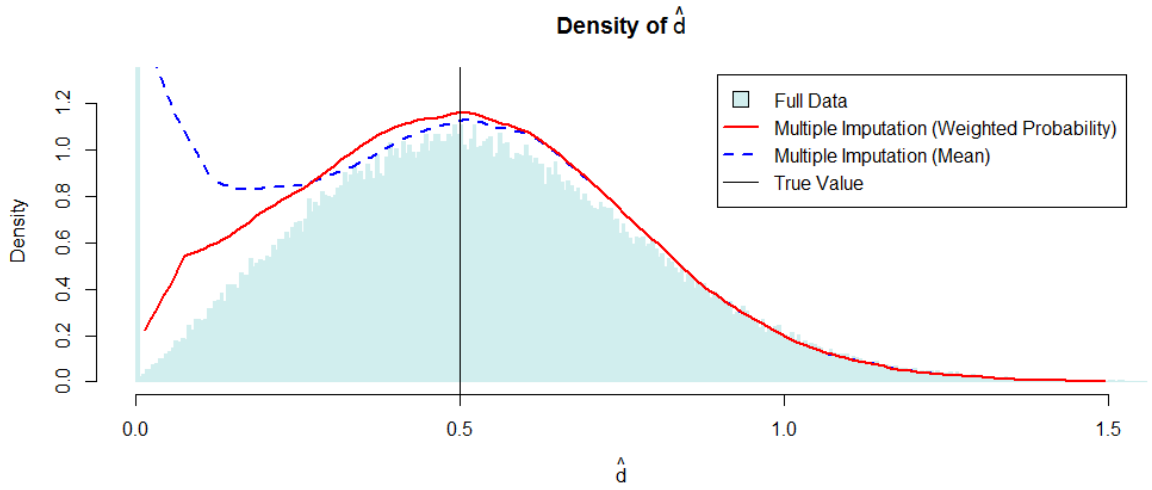
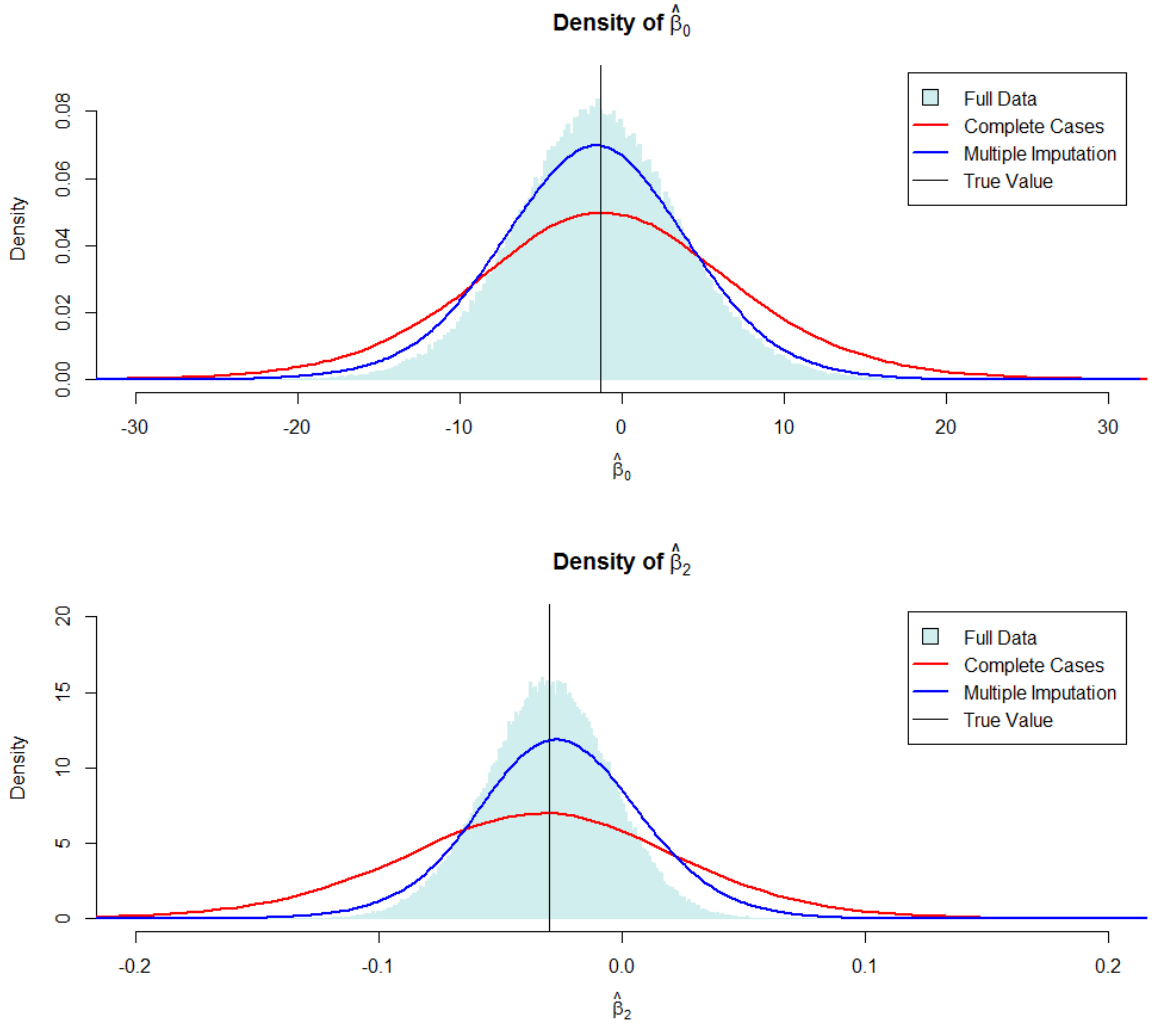


Figure A.2.: Empirical densities of $\hat{d}$ for the weighted probability combination method alongside with the previous results.

This approach again resembled the true density better compared to the combination method using the mean. Especially for small values, the curve was clearly closer to the theoretical probability of $\hat{d}$. The path of the curve was similar to the two-stage method. Yet, it was slightly closer to the full data density, which would make this approach the preferable one of the two supplementary methods. However, none of these additionally analyzed combination strategies could match up to the median, which was able to nearly perfectly resemble the theoretical density.

B. Additional Plots

B.1. Densities of the Fixed Effects Estimators



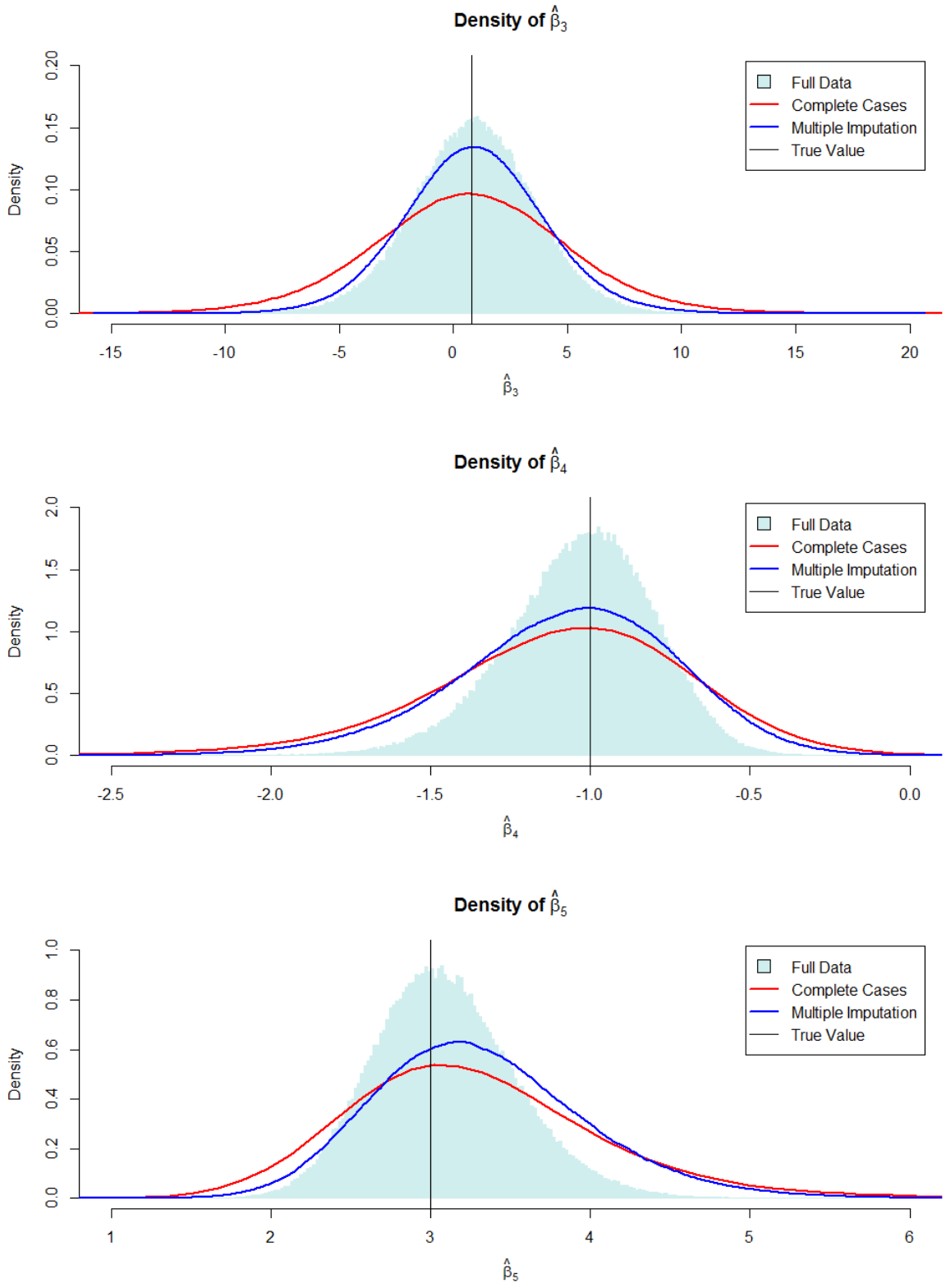
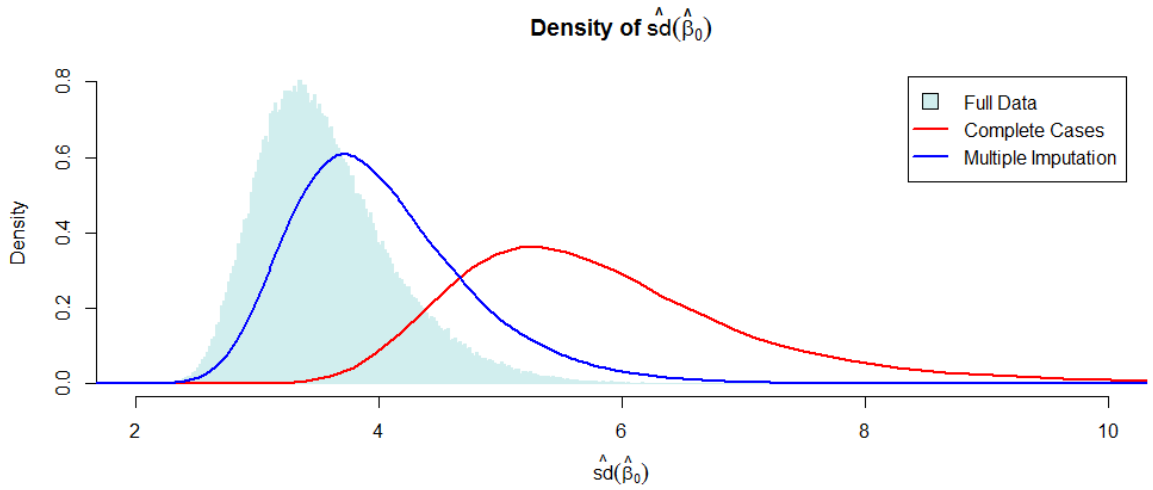
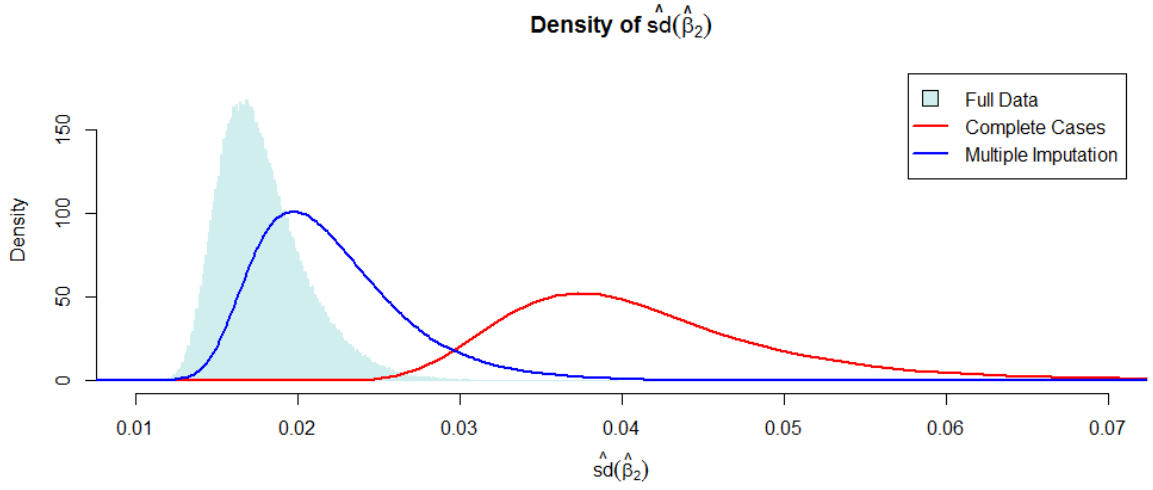
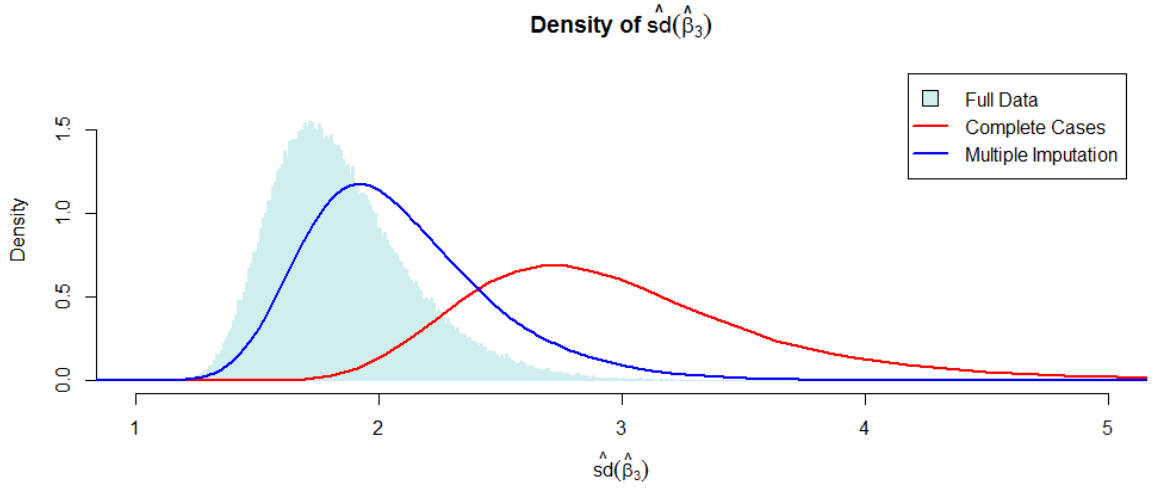


Figure B.1.: Empirical densities for all $\hat{\beta}$ of Simulation I, II and III.


Density of $\hat{sd}(\hat{\beta}_0)$
Full Data
Complete Cases
Multiple Imputation
Density
0.0
0.2
0.4
0.6
0.8
2
4
6
8
10
$\hat{sd}(\hat{\beta}_0)$


Density of $\hat{sd}(\hat{\beta}_2)$
Full Data
Complete Cases
Multiple Imputation
Density
0
50
100
150
0.01
0.02
0.03
0.04
0.05
0.06
0.07
$\hat{sd}(\hat{\beta}_2)$


Density of $\hat{sd}(\hat{\beta}_3)$
Full Data
Complete Cases
Multiple Imputation
Density
0.0
0.5
1.0
1.5
1
2
3
4
5
$\hat{sd}(\hat{\beta}_3)$

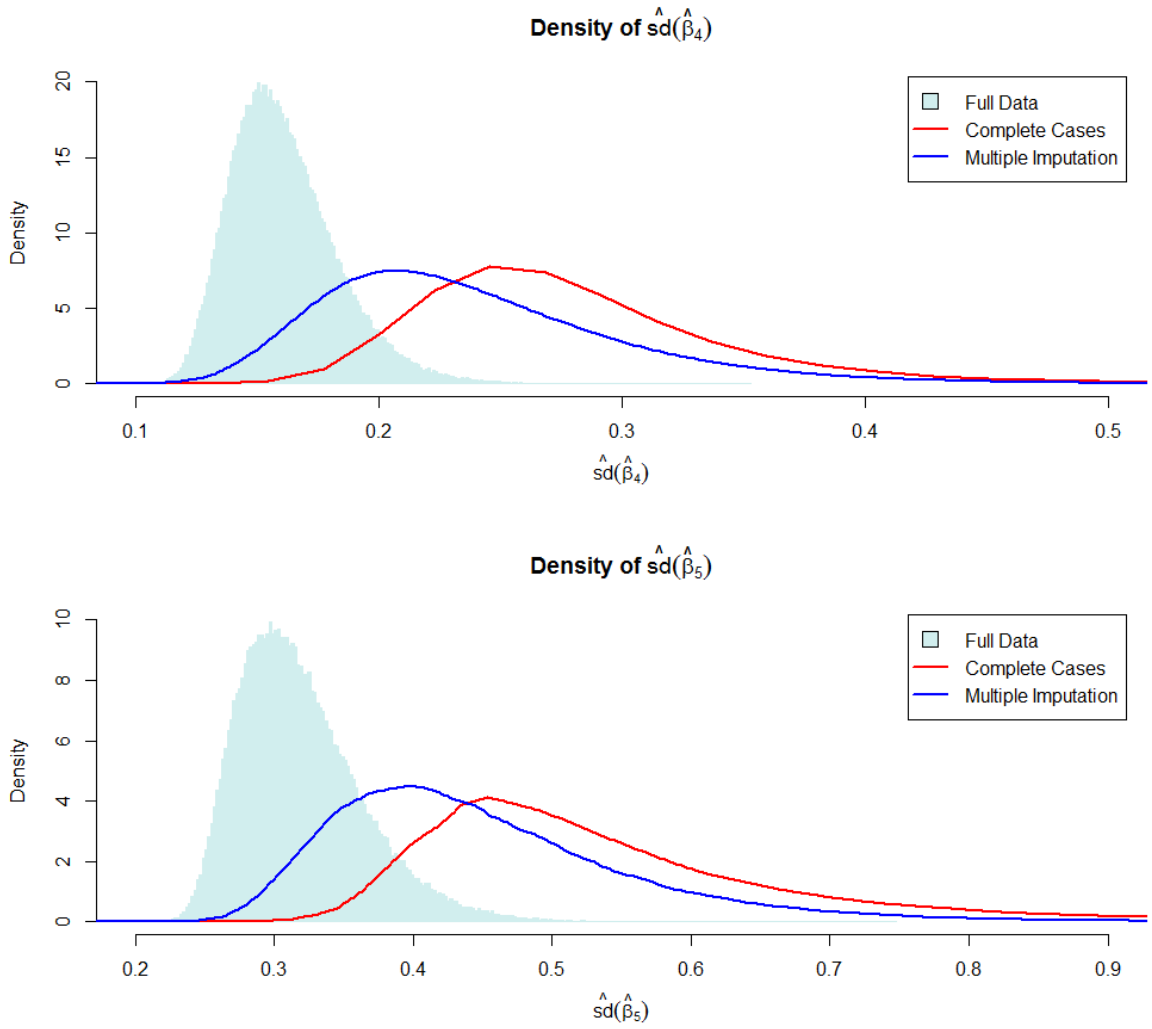


Figure B.2.: Empirical densities for all $\widehat{sd}(\hat{\beta})$ of Simulation I, II and III.

B.2. Bootstrap Percentile Intervals for 1000 Runs

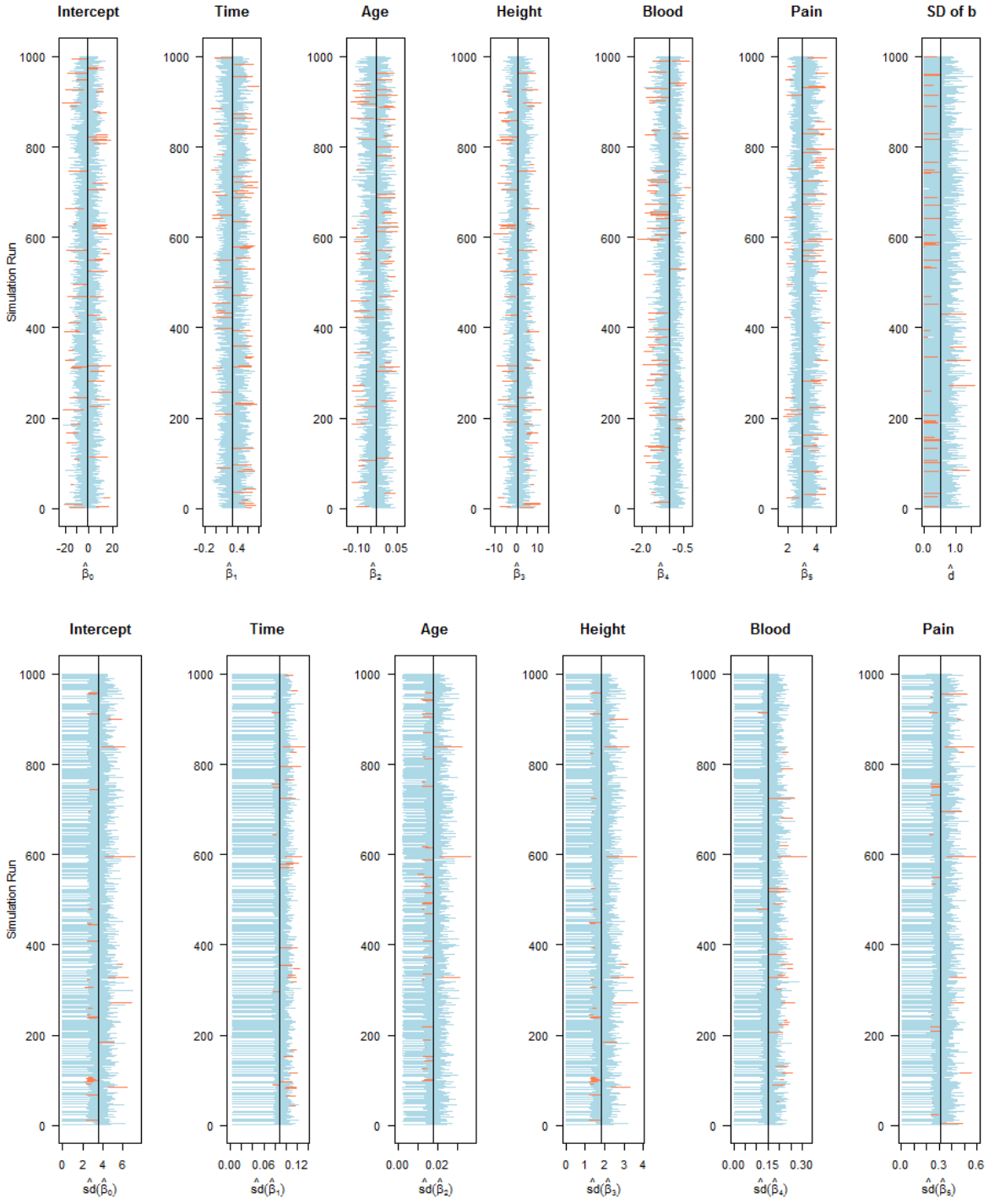


Figure B.3.: Full Data Simulation: 95% bootstrap percentile intervals of all parameters of interest for 1000 simulation runs. Intervals are colored blue if they hold the true value (black line), and red if they don't.

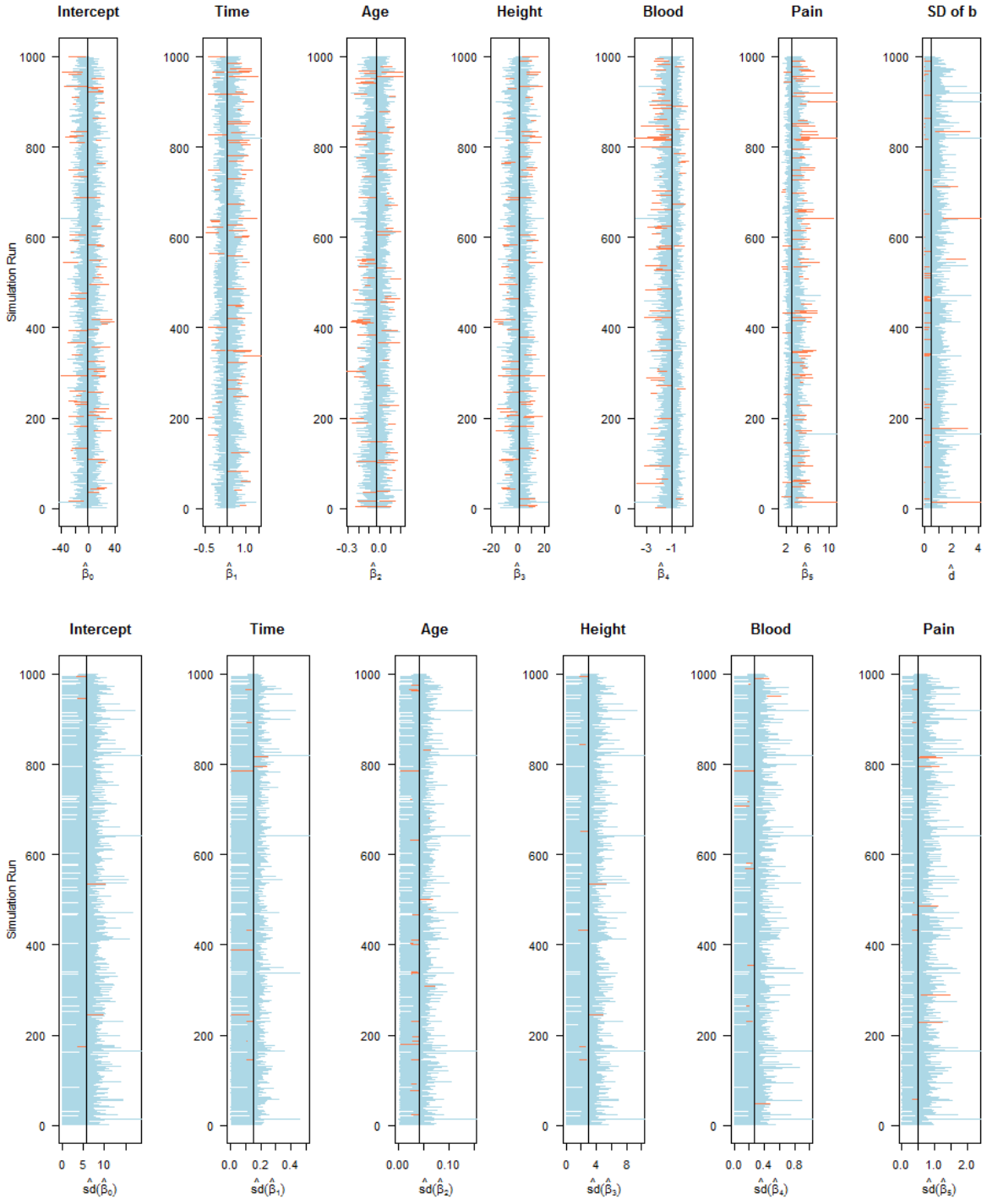


Figure B.4.: Complete Case Simulation: 95% bootstrap percentile intervals of all parameters of interest for 1000 simulation runs. Intervals are colored blue if they hold the true value (black line), and red if they don't.

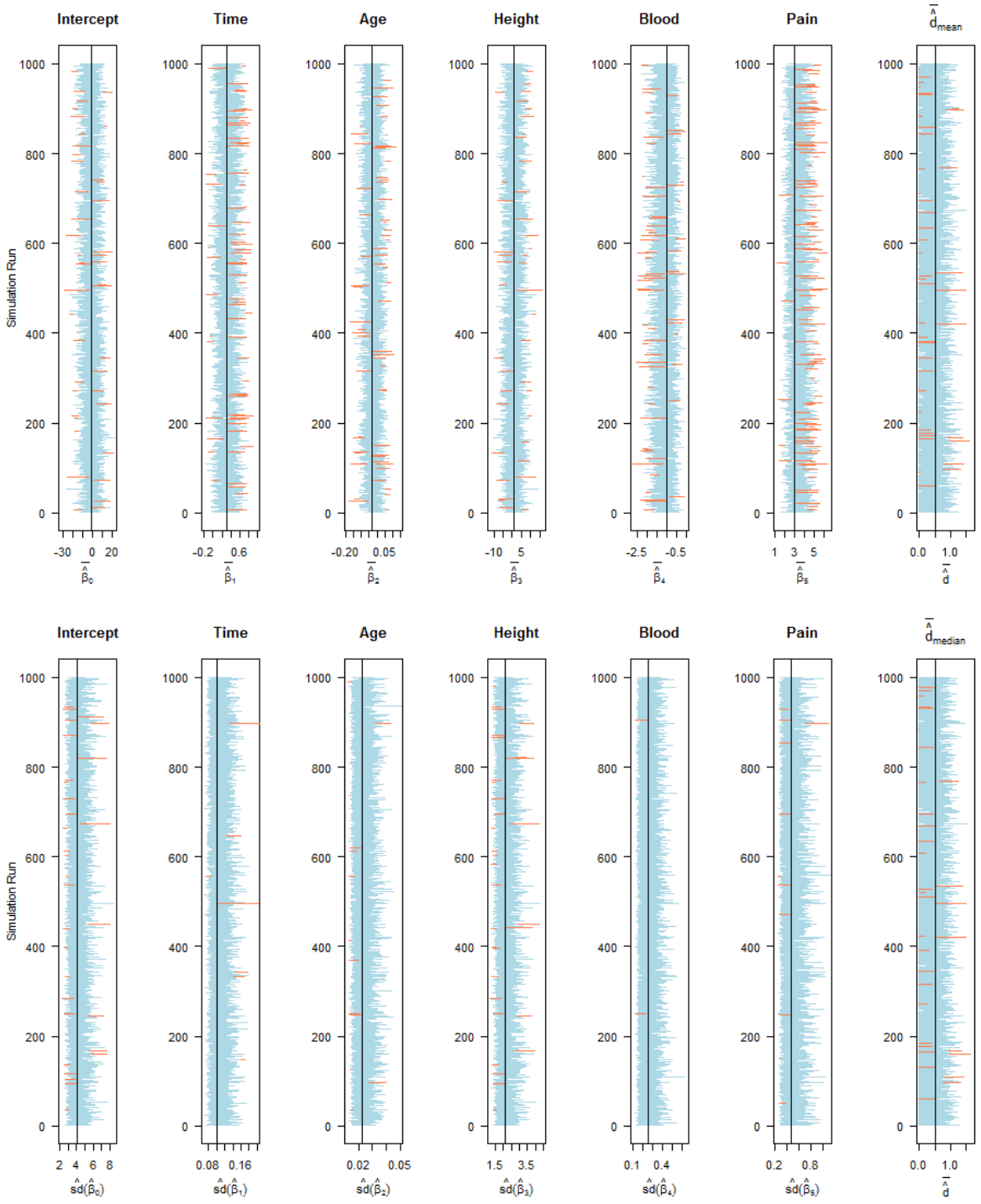


Figure B.5.: Multiple Imputation Simulation: 95% bootstrap percentile intervals of all parameters of interest for 1000 simulation runs. Intervals are colored blue if they hold the true value (black line), and red if they don't.

C. Digital Supplement on CD

Attached to this thesis is a supplementary CD containing the R-code and results obtained from the simulations. An overview of it's folder structure is given in figure C.1.

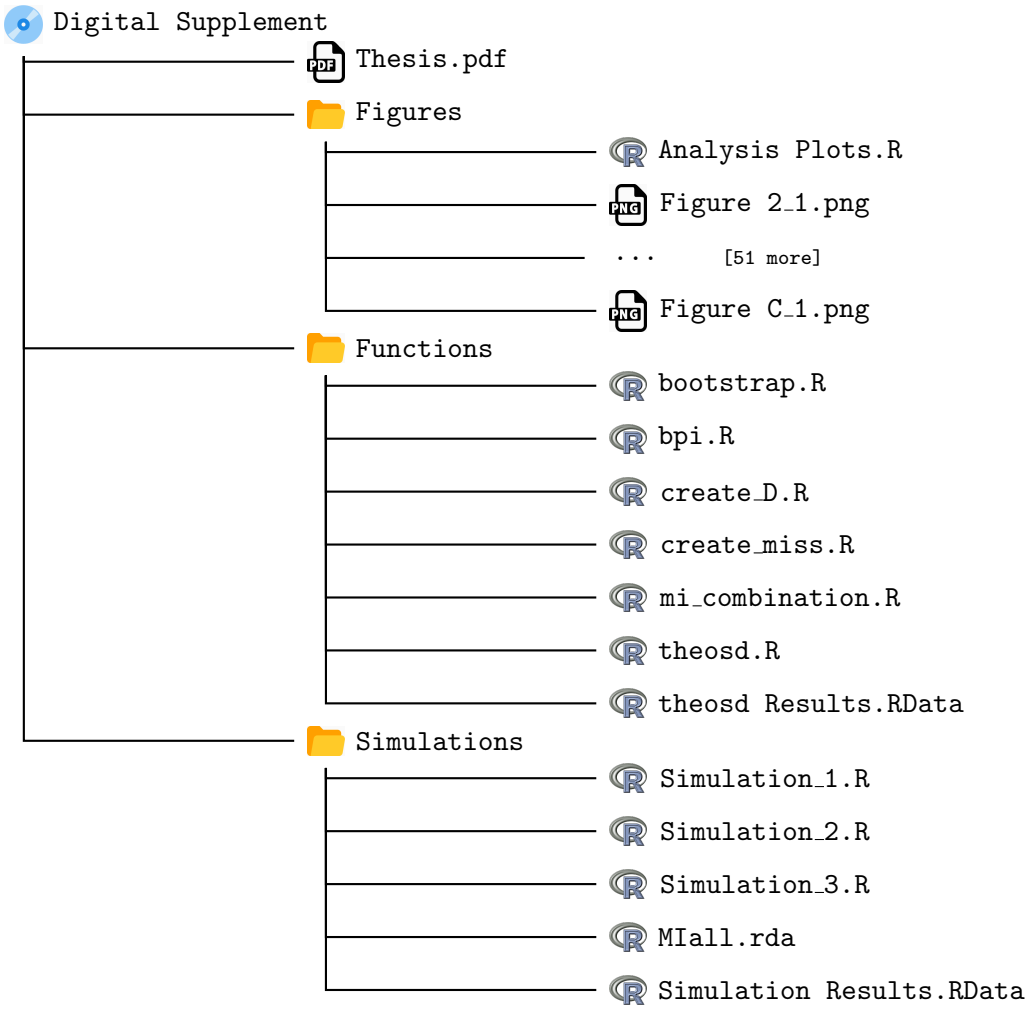


Figure C.1.: Folder Structure of the digital supplement on CD.

The CD comprises of the thesis in digital form and three folders.

The folder **Figures** contains all graphics, that were used in thesis. They are named by their caption number. Asides from that, an `R` script named "`Analysis Plots.R`" can be found. With this file, all analysis plots of section 4 can be recreated.

The folder **Functions** contains all functions, which are required to perform the simulations and analyze the results. In addition, the file "`theosd Results.RData`" is enclosed. It includes the results from the function "`theosd.R`" for all simulation situations.

The last folder is named **Simulations**. It contains the runnable `R` codes for all three simulations, which were executed on the servers of the Department of Statistics (LMU Munich). The obtained results (combined for simulation III) are enclosed in the file "`Simulation results.RData`". Lastly, the uncombined estimations of simulation III are given by the file "`MIall.rda`".

List of Figures

2.1. Illustration of a simple time series cross section data set. 8
2.2. The Random Intercept Model. 8
2.3. Tscs data, on which the random slope model appears to be suited. 9
2.4. Comparison of $\mathbb{P}(y_{ij} = 1|b_i)$, $\mathbb{P}(y_{ij} = 1)$ and $\mathbb{P}(y_{ij} = 1|b_i = 0)$. 17
2.5. Laplace approximation of a gamma density. 20
2.6. Scheme of creating an empirical bootstrap density. 27

3.1. Photo of Amelia Earhart. 30
3.2. Scheme of a multiple imputation analysis approach. 32

4.1. Comparison of the $\mathcal{B}(42, \frac{1}{2})$ and $\mathcal{B}\mathfrak{e}(3, 3)$ distribution. 50
4.2. Empirical distribution of the variable `height`. 53
4.3. The Autocorrelation function of the AR(1) for different values of ρ. . . . 55
4.4. Five realizations of the `ar1()` function. 56
4.5. Sample of the variable `pain` for 15 subjects. 58
4.6. Overview of the covariable dependencies. 59
4.7. $\mathbb{P}(y_{ij} = 1)$ for different values of β, c.p. 61
4.8. Sample of the response variable `y` for 15 subjects. 64
4.9. Simulation I: Overview of one run. 70
4.10. Simulation I: Empirical distribution of $\hat{\beta}_1$, $\widehat{\mathrm{sd}}(\hat{\beta}_1)$ and $\hat{d}$. 73
4.11. Simulation I: 95% bootstrap percentile intervals of 100 runs. 75
4.12. Simulation II: Overview of one run. 77
4.13. Simulation II: Empirical distribution of $\hat{\beta}_1$, $\widehat{\mathrm{sd}}(\hat{\beta}_1)$ and $\hat{d}$. 80

4.14. Simulation II: 95% bootstrap percentile intervals of 100 runs. 82
4.15. Simulation III: Overview of one run. 84
4.16. Simulation III: Structure of the return value. 87
4.17. Simulation III: Empirical densities of $\hat{\beta}_1$ and $\widehat{\mathrm{sd}}(\hat{\beta}_1)$. 88
4.18. Simulation III: Empirical densities for $\bar{\hat{d}}$ using the mean and median. . . 89
4.19. Simulation III: 95% bootstrap percentile intervals of 100 runs. 92
4.20. Comparison of the empirical densities of $\hat{\beta}_1$ for all simulations. 93
4.21. Boxplots of the empirical densities of $\hat{\beta}_1$. 94
4.22. Comparison of the empirical densities of d for all simulations. 95
4.23. Comparison of the densities for the bootstrap percentile interval sizes. . . 96

A.1. Empirical density of $\hat{d}$ for the two-stage combination method. 105
A.2. Empirical densities of $\hat{d}$ for the weighted probability combination method. 106

B.1. Empirical densities for all $\hat{\beta}$ of Simulation I, II and III. 108
B.2. Empirical densities for all $\widehat{sd}(\hat{\beta})$ of Simulation I, II and III. 110
B.3. Simulation I: 95% bootstrap percentile intervals of 1000 runs. 111
B.4. Simulation II: 95% bootstrap percentile intervals of 1000 runs. 112
B.5. Simulation III: 95% bootstrap percentile intervals of 1000 runs. 113

C.1. Folder Structure of the digital supplement on CD. 114

List of Tables

2.1. Excerpt from an exemplary TSCS data set. 24

4.1. Excerpt of a data set, created by the methods of section 4.2.1 and 4.2.3. . 64
4.2. Overview of the missingness structure. 66
4.3. Theoretical standard errors of the fixed effects for the three simulations. . 68
4.4. Simulation I: Percentage of 95% bootstrap percentile intervals holds. . . . 73
4.5. Simulation II: Percentage of 95% bootstrap percentile intervals holds. . . 81
4.6. Simulation III: Percentage of 95% bootstrap percentile intervals holds. . . 90

Bibliography

Bates, Douglas M. (2010). lme4: Mixed-effects modeling with R. Springer. URL: http://lme4.r-forge.r-project.org/lMMwR/lrgprt.pdf.

Bates, Douglas M. (2015). R Documentation of Package 'lme4'. URL: https://cran.r-project.org/web/packages/lme4/lme4.pdf.

Bishop, Christopher M. (2006). Pattern Recognition and Machine Learning. Springer. URL: http://www.springer.com/us/book/9780387310732.

Blay, Sigal. Snow Simplified. Simon Fraser University. URL: http://www.sfu.ca/~sblay/R/snow.html.

Brendel, Markus and Sara Wadle (2008). Gemischte (hierarchische) lineare Modelle bei Longitudinaldaten. LMU Munich. URL: http://www.statistik.lmu.de/~helmut/seminar_0809/H8.pdf.

Daniels, Michael and Joseph Hogan (2000). Missing Data in Longitudinal Studies. Chapman & Hall. URL: https://www.crcpress.com/Missing-Data-in-Longitudinal-Studies-Strategies-for-Bayesian-Modeling/Daniels-Hogan/9781584886099.

Dargatz, C., L. Fahrmeir, and C. Heumann (2012). Schätzen und Testen. LMU München. URL: https://www.elab.moodle.elearning.lmu.de/course/view.php?id=881.

Drechsler, Jörg (2011). Missing Data and Imputation. LMU München. URL: http://www.statistik.lmu.de/~fkreuter/imputation_sose2011/downloads/Imputationsvorlesung_5_slides.pdf.

Fahrmeir, Ludwig, Thomas Kneib, and Stefan Lang (2007). Regression. Springer. URL: http://www.springer.com/de/book/9783642018367.

Greven, Sonja (2015). Gemischte Modelle. LMU München. URL: http://www.statistik.lmu.de/institut/ag/fda/mixedmodels_2015/material.html.

Greven, Sonja and Jona Cederbaum (2015). Analyse Longitudinaler Daten. LMU München. URL: http://www.statistik.lmu.de/institut/ag/fda/ALD_2015/material.html.

Heumann, Christian and Michael Schomaker (2013). Model selection and model averaging after multiple imputation. In: Computational Statistics and Data Analysis. URL: http://www.sciencedirect.com/science/article/pii/S016794731300073X.

Honaker, James, Gary King, and Matthew Blackwell (2011). Amelia II: A Program for Missing Data. In: Journal of Statistical Software. URL: http://cran.r-project.org/web/packages/Amelia/vignettes/amelia.pdf.

Honaker, James, Gary King, and Matthew Blackwell (2012). Amelia II: A Program for Missing Data. URL: http://r.iq.harvard.edu/docs/amelia/amelia.pdf.

Honaker, James, Gary King, and Matthew Blackwell (2015). Package 'Amelia'. URL: http://r.iq.harvard.edu/docs/amelia/amelia.pdf.

Jacobs, Robert (2008). Bayesian Statistics: Normal-Normal Model. University of Rochester. URL: https://www.bcs.rochester.edu/people/robbie/jacobslab/cheat_sheet/bayes_Normal_Normal.pdf.

Jiang, Jiming (2007). Linear and Generalized Linear Mixed Models and Their Applications. Springer. URL: http://www.springer.com/us/book/9780387479415.

Jones, Douglas (1999). The Sweep Operator. University of New Jersey. URL: http://www-rci.rutgers.edu/~dhjones/APPLIED_LINEAR_STATISTICAL_MODELS%28PHD%29/LECTURES/LECTURE06/3-The%20sweep%20operator.pdf.

King, Gary and James Honaker (2008). What to do about Missing Values in Time Series Cross-Section Data. In: American Journal of Political Science. URL: http://gking.harvard.edu/files/pr.pdf.

King, Gary, James Honaker, Anne Joseph, and Kenneth Scheve (2001). Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation.

In: American Political Science Review. URL: http://gking.harvard.edu/files/gking/files/evil.pdf.

Schafer, Joseph (1997). Analysis of Incomplete Multivariate Data. Chapman & Hall. URL: http://onlinelibrary.wiley.com/wol1/doi/10.1002/%28SICI%291097-0258%2820000415%2919:7%3C1006::AID-SIM384%3E3.0.CO;2-T/abstract.

Tutz, Gerhard (2011). Generalisierte Regression. LMU Munich. URL: http://www.statistik.lmu.de/~fachsch/vorlesungsdateien/GeneralisierteRegression.pdf.

Tutz, Gerhard (2012). Regression for Categorical Data. Cambridge University Press. URL: http://www.cambridge.org/de/academic/subjects/statistics-probability/statistical-theory-and-methods/regression-categorical-data.

Verbeke, Geert (2000). Linear Mixed Models for Longitudinal Data. Springer. URL: http://www.springer.com/us/book/9781441902993.

Verteilung des Merkmals Körpergröße. URL: http://wiki.small-and-tall.com/index.php?title=Verteilung_des_Merkmals_Krpergre.