Extensions of Backfitting to Mixed and Spatial Models

Master Thesis in Mathematics (MAT491)

by

Stefan Willi
16-739-120

supervised by

Prof. Dr. Reinhard Furrer

University of Zurich[UZH]

Zurich, April 2021

Extensions of Backfitting to Mixed and Spatial Models

Stefan Willi

Version April 15, 2021

Contents

Acknowledgment iii

Abstract v

1 Introduction 1

2 Backfitting for Additive Models 7
2.1 Frequentist Backfitting for Additive Models . . . 8
2.2 Bayesian Backfitting for Additive Models . . . 11
2.3 Posterior Realizations in Linear Regressions . . . 17

3 Backfitting for Mixed Models 23
3.1 Frequentist Backfitting for Linear Mixed Models . . . 26
3.2 Bayesian Backfitting for Linear Mixed Models . . . 27
3.3 Simulations . . . 30

4 Backfitting for Spatial Models 45
4.1 Introduction to Spatial Statistics . . . 45
4.2 Frequentist Backfitting for Spatial Models . . . 48
4.3 Bayesian Backfitting for Spatial Models . . . 50
4.4 Simulations . . . 52

5 Discussion 71

6 Outlook 75

A Appendix 77
A.1 R Code . . . 77

Bibliography 89

Acknowledgment

Working on this master thesis has been an incredibly rewarding experience that helped me to grow both personally and academically. I would like to thank my supervisor Prof. Dr. Reinhard Furrer for all the support and critics throughout this project. It was an honor to do my master thesis with such a renowned mathematician and statistician. I am also very grateful to Roman Flury for his time, advice and help. I have benefited tremendously from our meetings. You both receive my fullest admiration for your clear and precise thinking and your ability to see the big picture without having all the details. I also want to thank Michael for showing me the path into statistics and helping me to improve my knowledge and skills in this area. In addition, I would like to thank my fellow students Nicolas, Philipp, Jonas, Sarina, Stefan, Annina, Melena, Nathan, David, Roger and Carl for making the time at university exciting and diversified. We had a lot of fun and it would not have been the same without you. Furthermore, I thank my parents for supporting me during my whole studies. However, the biggest thanks go to Elena who gave me the motivation and strength to pursue my degree.

Stefan Willi
April 2021

Abstract

In spatial statistics one often faces a large amount of data. Doing computations with such large datasets is sometimes very time and storage consuming or even impossible. Especially, if the goal is to fit a spatial model with a large number of observations using a maximum likelihood approach, memory limits of the computer might be reached. For this reason, Furrer and Sain (2009) proposed the backfitting algorithm for spatial models. This algorithm fits a spatial model in an iterative way and can better handle a large number of data. Nevertheless, this procedure is not only used for computational reasons. With the backfitting algorithm one has the ability to estimate the linear trend and the spatial process separately.

However, the backfitting algorithm was not a new method at this time. Hastie and Tibshirani (1986) already used the backfitting algorithm to fit generalized additive models. Furthermore, Hastie and Tibshirani (2000) proposed a method called *Bayesian backfitting* for sampling from the posterior distribution belonging to a generalized additive model. Whereas the frequentist backfitting algorithm returns estimates for the functions, the Bayesian backfitting algorithm samples from a distribution, similar to the Gibbs sampling method.

In this work, the frequentist as well as the Bayesian backfitting algorithm are extended to the class of linear mixed-effects models. Additionally, an extension to the Bayesian setting of the backfitting algorithm for spatial models is presented. The goal of this master thesis is to compare both algorithms for some simulated setups and see which performs better in what situations.

All computations, visualizations as well as simulations are done with the statistical programming language R (R Core Team, 2020) and its additional packages `lme4` (Bates *et al.*, 2015), `spam` (Furrer and Sain, 2010) and `plot3D` (Soetaert, 2019).

Chapter 1

Introduction

Linear regression is widely used in statistics because it may yield the answer to the question how a target value of special interest depends on several other factors or causes. There are thousands of examples where linear regression (or regression analysis in general) is used. For instance:

- how fertilizer and soil quality affect the growth of plants,
- how size, location, furnishment and age affect apartment rents,
- how age, gender, experience and nationality affect car insurance premiums.

Regression techniques describe the relation between one or more predictor variables and a variable of special interest, called the response variable.

The linear regression model (also called linear model) assumes that the relationship between the response variable Y and the p regressors $x_1, \ldots, x_p$ is linear. For this reason, the unknown parameters $\beta_0, \beta_1, \ldots, \beta_p$ are introduced. The relationship is modeled through an error variable ε which is an unobserved random variable that adds noise to the linear relationship between the response variable and the regressors. Thus, the model takes the form

$$Y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \varepsilon.$$

Given the $(p+1)$-tuples $\{Y_i, x_{i1}, \ldots, x_{ip}\}_{i=1}^n$, we can write the linear model as

$$Y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \varepsilon_i, \quad i = 1, \ldots, n.$$

Often, it is more convenient to write these n equations in a matrix-vector representation as

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

- $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\top \in \mathbb{R}^n$,
- $\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \in \mathbb{R}^{n \times (p+1)}$,

- $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^\top \in \mathbb{R}^{p+1}$,
- $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)^\top \in \mathbb{R}^n$.

Remark 1.1.

1. $\boldsymbol{Y}$ is an n-dimensional vector of random variables Y_i for $i = 1, \ldots, n$ representing the response variables.
2. $\boldsymbol{X}$ is an $n \times (p+1)$-dimensional matrix, called the design matrix. It contains the fixed and known regressor variables x_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, p$.
3. $\boldsymbol{\beta}$ is a $(p+1)$-dimensional parameter vector where β_0 is the intercept term. Its elements are known as effects or regression coefficients, which are fixed but unknown. The goal is to estimate the β_j's for $j = 0, 1, \ldots, p$.
4. $\boldsymbol{\varepsilon}$ is an n-dimensional error vector. It is called the error term or noise. This variable captures all other unknown factors which influence the response variable Y other than the regressors $x_1, \ldots, x_p$. We assume that $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$.

Remark 1.2. We denote with $\boldsymbol{Y}$ the vector of random variables $Y_1, \ldots, Y_n$. The observations are then denoted with $y_1, \ldots, y_n$ and the observation vector with $\boldsymbol{y}$.

Linear regression models are often fitted by the method of ordinary least squares. In this case the estimate $\widehat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ will be the minimizer of the residual sum of squares, i.e.

$$\widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{\beta}}_{\text{OLS}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p+1}}{\arg\min} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p+1}}{\arg\min} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2.$$

This minimization problem has a unique solution which is given by

$$\widehat{\boldsymbol{\beta}}_{\text{OLS}} = \left(\boldsymbol{X}^\top \boldsymbol{X} \right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}. \tag{1.1}$$

Remark 1.3. We assume that the rank of $\boldsymbol{X}$ equals $p+1$ (i.e. $\text{rank}(\boldsymbol{X}) = p+1$). This assumption guarantees that the inverse of $\boldsymbol{X}^\top \boldsymbol{X}$ exists. In practical terms, this implies that we do not include twice the same predictor, or that a predictor has additional information on top of the already included predictors.

Fitting a linear model in R is done with the `lm()` function.

R-Code 1.1. Fitting a linear model

```
## model setup ####
set.seed(1)

n <- 100 # number of observations

# predictors
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
x4 <- rnorm(n)
```

```r
# true beta
beta <- c(0.7, 1, 2, 0.5, -3) # (beta_0, beta_1, ..., beta_4)

# design matrix X
X <- as.matrix(cbind(1, x1, x2, x3, x4))

# observed values
y <- X %*% beta + rnorm(n)

## fitting a linear model ####
lm_fit <- lm(y ~ x1 + x2 + x3 + x4)

# estimates of beta using least squares
lm_fit$coef
```

```
## (Intercept)          x1          x2          x3          x4
##   0.6430246   1.1933287   2.1696399   0.5337476  -2.9573178
```

Instead of having a linear relationship between response and predictors we might try a different approach where this relationship is nonlinear. Consider an additive model of the form

$$Y_i = \alpha + \sum_{j=1}^{p} f_j(x_{ij}) + \varepsilon_i, \quad i = 1, \ldots, n,$$

where the f_j's are smooth functions and $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. Note that the linear model is a special case of the additive model, where the functions f_j are linear, i.e. $f_j(x_{ij}) = \beta_j x_{ij}$.

Additive models are more flexible than linear models, but still interpretable since the functions f_j can be plotted to give a sense of the marginal relationship between one predictor and the response.

In contrast to the linear model one cannot find estimates for the functions using the method of least squares. However, the estimation of α, $f_1(\cdot), \ldots, f_p(\cdot)$ is done with an iterative procedure, called the *backfitting algorithm* (Hastie and Tibshirani, 1986).

The idea is that the multidimensional optimization problem is broken down into multiple one-dimensional optimizations. The procedure is the following:

- First, take some initial values (i.e. guesses) for the estimates of α and the functions f_j for $j = 1, \ldots, p$ denoted by $\widehat{\alpha}$ and $\widehat{f_j}$ respectively.
- Then fix the estimates $\widehat{f_2}, \ldots, \widehat{f_p}$ and fit the function f_1 in the new (simpler) model

$$y_i - \widehat{\alpha} - \sum_{k=2}^{p} \widehat{f_k}(x_{ik}) = f_1(x_{i1}) + \varepsilon_i, \quad i = 1, \ldots, n.$$

  Denote the fitted function again by $\widehat{f_1}$.

- Fix $\widehat{f}_1, \widehat{f}_3, \ldots, \widehat{f}_p$ and fit the function f_2 in the new (simpler) model

$$y_i - \widehat{\alpha} - \sum_{\substack{k=1 \\ k \neq 2}}^{p} \widehat{f}_k(x_{ik}) = f_2(x_{i2}) + \varepsilon_i, \quad i = 1, \ldots, n.$$

  Denote the fitted function again by $\widehat{f}_2$.

- Repeat this for every function f_j, $j = 3, \ldots, p$.

- After fitting the function $\widehat{f}_p$ repeat the same procedure until the function estimates $\widehat{f}_1, \ldots, \widehat{f}_p$ do not change much anymore.

Remark 1.4. In the sequel, we call this algorithm the *frequentist* backfitting algorithm to distinguish it from the *Bayesian* backfitting algorithm (see Section 2.2). Sometimes we just write "backfitting algorithm" to mention the frequentist backfitting algorithm.

The backfitting algorithm can also be applied to linear models in order to estimate the parameters β_0, β_1, …, β_p. Instead of using the closed formula for $\widehat{\boldsymbol{\beta}}$ given by the method of ordinary least squares in equation (1.1), the backfitting algorithm splits the optimization over all parameters into p different one-dimensional optimization problems.

Of course, in the linear case this makes practically not much of a difference because one can easily compute $\widehat{\boldsymbol{\beta}}$. However, for illustration reasons, the backfitting algorithm for linear regressions is the following:

Algorithm 1.1 (Backfitting algorithm for linear regression)**.**

[0] set $\widehat{\alpha} = \frac{1}{n}\sum_{i=1}^{n} y_i$, initialize $\widehat{\boldsymbol{f}}_1^{(0)} = \ldots = \widehat{\boldsymbol{f}}_p^{(0)} = \mathbf{0} \in \mathbb{R}^n$ and set $t = 0$

[1] $t = t + 1$

[2] do for $j = 1, 2, \ldots, p$:

- fit $\widehat{\mu}_j$ and $\widehat{\beta}_j$ in the model

$$\boldsymbol{y} - \widehat{\alpha}\mathbf{1} - \sum_{k=1}^{j-1} \widehat{\boldsymbol{f}}_k^{(t)} - \sum_{k=j+1}^{p} \widehat{\boldsymbol{f}}_k^{(t-1)} = \mu_j \mathbf{1} + \beta_j \boldsymbol{x}_j + \boldsymbol{\varepsilon}$$

- update $\widehat{\boldsymbol{f}}_j^{(t)} = \widehat{\mu}_j \mathbf{1} + \widehat{\beta}_j \boldsymbol{x}_j$

[3] repeat [1] and [2] until convergence

[4] output $\widehat{\beta}_0, \widehat{\beta}_1, \ldots, \widehat{\beta}_p$ as estimates for the regression coefficients where $\widehat{\beta}_0$ is the combined estimate of the intercepts, i.e. $\widehat{\beta}_0 = \widehat{\alpha} + \widehat{\mu}_1 + \cdots + \widehat{\mu}_p$

Remark 1.5.

1. In step [2] of Algorithm 1.1 a simple linear regression is fitted with

$$\boldsymbol{y} - \widehat{\alpha}\mathbf{1} - \sum_{k=1}^{j-1} \widehat{\boldsymbol{f}}_k^{(t)} - \sum_{k=j+1}^{p} \widehat{\boldsymbol{f}}_k^{(t-1)}$$

   as the response and $\boldsymbol{x}_j$ as the only predictor variable.

2. If the predictors $\boldsymbol{x}_j$ are centered, i.e. $\text{mean}(\boldsymbol{x}_j) = 0$ or $\sum_{i=1}^{n} x_{ij} = 0$ for $j = 1, \ldots, p$, then estimating $\widehat{\mu}_j$ in step [2] will not be necessary since it is always zero.

We implement the backfitting algorithm for a linear regression in R and compare the resulting output with the least squares estimates of $\beta_0, \beta_1, \ldots, \beta_p$. Note that the backfitting algorithm converges after 5 iterations and the estimates coincide.

R-Code 1.2. Backfitting algorithm for a linear regression

```
itermax <- 20 # maximum number of iterations allowed

# initialize hat{alpha} and hat{f_j} for j = 1,2,3,4
alpha_hat <- mean(y)
f1 <- f2 <- f3 <- f4 <- matrix(rep(0, n), nrow = n, ncol = itermax)

# parameters for convergence
eps <- 10^(-10)
conv <- 0
t <- 1

while (!conv) {
  t <- t + 1

  # for hat{f_1}
  r1 <- y - alpha_hat - f2[,t-1] - f3[,t-1] - f4[,t-1]
  fit1 <- lm(r1 ~ x1)
  f1[,t] <- fit1$fitted.values

  # for hat{f_2}
  r2 <- y - alpha_hat - f1[,t] - f3[,t-1] - f4[,t-1]
  fit2 <- lm(r2 ~ x2)
  f2[,t] <- fit2$fitted.values

  # for hat{f_3}
  r3 <- y - alpha_hat - f1[,t] - f2[,t] - f4[,t-1]
  fit3 <- lm(r3 ~ x3)
  f3[,t] <- fit3$fitted.values

  # for hat{f_4}
  r4 <- y - alpha_hat - f1[,t] - f2[,t] - f3[,t]
  fit4 <- lm(r4 ~ x4)
  f4[,t] <- fit4$fitted.values

  # check convergence
  maxerr <- max(sum((f1[,t]-f1[,t-1])^2) / sum(f1[,t-1]^2),
                sum((f2[,t]-f2[,t-1])^2) / sum(f2[,t-1]^2),
                sum((f3[,t]-f3[,t-1])^2) / sum(f3[,t-1]^2),
```

```
                sum((f4[,t]-f4[,t-1])^2) / sum(f4[,t-1]^2))
  conv <- maxerr < eps | t >= itermax
}

# number of iterations
(iter <- t-1)

## [1] 5

## Compare backfitting with least squares ####

# put all intercepts of the linear regression fits within
# the backfitting algorithm together
beta_0.hat <- alpha_hat + coef(fit1)[1] + coef(fit2)[1] + coef(fit3)[1] +
  coef(fit4)[1]

# estimated paramters hat{b_j} from backfitting algorithm
(beta.hat <- c(beta_0.hat, coef(fit1)[2], coef(fit2)[2], coef(fit3)[2],
               coef(fit4)[2]))

## (Intercept)          x1          x2          x3          x4
##   0.6430246   1.1933286   2.1696399   0.5337476  -2.9573178

# linear regression fit
lm_fit$coef

## (Intercept)          x1          x2          x3          x4
##   0.6430246   1.1933287   2.1696399   0.5337476  -2.9573178
```

After a first glimpse of the backfitting algorithm, we state in more detail the frequentist backfitting algorithm for additive models in Chapter 2. Then we extend this to a Bayesian setting and introduce the Bayesian backfitting algorithm based on Hastie and Tibshirani (2000). Bayesian backfitting is a generalization of the frequentist backfitting algorithm where the goal is to sample from the posterior distribution of an additive model.

In Chapter 3, we establish the backfitting algorithm for the class of linear mixed-effects models in the frequentist as well as in the Bayesian case. In this new theory, the linear coefficients and the random effects will be estimated individually and iteratively. We close that chapter with a bunch of simulations and investigate how the backfitting algorithm for linear mixed models performs.

In Chapter 4, we present yet another extension of the backfitting algorithm, namely to the class of spatial models, which represents the response as an additive combination of a linear trend and a spatial process. As for mixed models, this algorithm estimates the linear trend and the (random) spatial process separately. Whereas Furrer and Sain (2009) have already described the frequentist backfitting algorithm for spatial models, we also come up with the Bayesian backfitting algorithm.

The work is then concluded with further chapters looking at some results and an outlook of this methodology.

Chapter 2

Backfitting for Additive Models

The focus in this chapter lies on the backfitting algorithm applied to additive models.

Definition 2.1. An additive model is a model of the form

$$Y = \alpha + \sum_{j=1}^{p} f_j(x_j) + \varepsilon,$$

where the functions f_j for $j = 1, \ldots, p$ are arbitrary smooth functions and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

Given the $(p+1)$-tuples $\{Y_i, x_{i1}, ..., x_{ip}\}_{i=1}^{n}$, we can write the additive model in the form

$$Y_i = \alpha + \sum_{j=1}^{p} f_j(x_{ij}) + \varepsilon_i, \quad i = 1, \ldots, n, \tag{2.1}$$

where $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$.

Remark 2.1. For identifiability between α and the functions f_j for $j = 1, \ldots, p$ we assume that

$$\sum_{i=1}^{n} f_j(x_{ij}) = 0 \quad \text{for all } j = 1, \ldots, p.$$

This implies that $\alpha = \frac{1}{n}\sum_{i=1}^{n} Y_i$. To see why this constraint is needed, consider the simple case where $p = 2$. If we add constants c_1 to f_1 and c_2 to f_2 but subtract $c_1 + c_2$ from α then nothing observable has changed about the model.

There is also a vector representation of equation (2.1) given by

$$\boldsymbol{Y} = \alpha \mathbf{1} + \sum_{j=1}^{p} \boldsymbol{f}_j + \boldsymbol{\varepsilon}, \tag{2.2}$$

where

- $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\top \in \mathbb{R}^n$,
- $\mathbf{1} = (1, \ldots, 1)^\top \in \mathbb{R}^n$,
- $\boldsymbol{f}_j = (f_j(x_{1j}), \ldots, f_j(x_{nj}))^\top \in \mathbb{R}^n$ for $j = 1, \ldots, p$,
- $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)^\top \in \mathbb{R}^n$, $\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 \boldsymbol{I}_n)$.

2.1 Frequentist Backfitting for Additive Models

In Chapter 1, the frequentist backfitting algorithm for fitting additive models has already been encountered. The purpose of this section is to state in detail the backfitting algorithm for an additive model as in equation (2.1). The algorithm is the following:

Algorithm 2.1 (Backfitting algorithm for additive models)**.**

[0] set $\widehat{\alpha} = \frac{1}{n}\sum_{i=1}^{n} y_i$, initialize $\widehat{f}_j^{(0)}(\cdot) \equiv 0$ for $j = 1, \ldots, p$ and set $t = 0$

[1] $t = t + 1$

[2] do for $j = 1, 2, \ldots, p$:

$$\widehat{f}_j^{(t)} = \text{Smooth}\left[\left\{y_i - \widehat{\alpha} - \sum_{k=1}^{j-1} \widehat{f}_k^{(t)}(x_{ik}) - \sum_{k=j+1}^{p} \widehat{f}_k^{(t-1)}(x_{ik})\right\}_{i=1}^{n}\right] \text{ (backfitting step)}$$

$$\widehat{f}_j^{(t)} = \widehat{f}_j^{(t)} - \frac{1}{n}\sum_{i=1}^{n} \widehat{f}_j^{(t)}(x_{ij}) \text{ (mean centering)}$$

where Smooth$[\cdot]$ is a smoothing operator (e.g. cubic smoothing splines)

[3] repeat [1] and [2] until convergence

We might stop the iterations if for instance the individual functions $\widehat{f}_j(\cdot)$ do not change much anymore. The corresponding convergence criterion would then be

$$\frac{\sum_{i=1}^{n}\left[\widehat{f}_j^{(t)}(x_{ij}) - \widehat{f}_j^{(t-1)}(x_{ij})\right]^2}{\sum_{i=1}^{n}\left[\widehat{f}_j^{(t-1)}(x_{ij})\right]^2} < \epsilon$$

for some desirable $\epsilon > 0$.

The main interest of this work does not lie on the convergence of the algorithm. Therefore, we only verify the convergence of Algorithm 2.1 numerically. By experience it converges very quickly.

Remark 2.2. Using the vector representation of equation (2.2) the backfitting algorithm for additive models (Algorithm 2.1) can also be written in a vector form.

- The reformulated backfitting step is then

$$\widehat{\boldsymbol{f}}_j^{(t)} = \text{Smooth}\left[\boldsymbol{y} - \widehat{\alpha}\mathbf{1} - \sum_{k=1}^{j-1} \widehat{\boldsymbol{f}}_k^{(t)} - \sum_{k=j+1}^{p} \widehat{\boldsymbol{f}}_k^{(t-1)}\right].$$

- The centering of the $\widehat{\boldsymbol{f}}_j^{(t)}$'s is done via

$$\widehat{\boldsymbol{f}}_j^{(t)} = \widehat{\boldsymbol{f}}_j^{(t)} - \text{mean}\left(\widehat{\boldsymbol{f}}_j^{(t)}\right).$$

- The convergence criterion could be

$$\frac{\left\|\widehat{\boldsymbol{f}}_j^{(t)} - \widehat{\boldsymbol{f}}_j^{(t-1)}\right\|_2}{\left\|\widehat{\boldsymbol{f}}_j^{(t-1)}\right\|_2} < \epsilon$$

for some desirable $\epsilon > 0$.

Let us have a look at the backfitting algorithm for additive models in R. We use the *Prestige* data from the R package *car* (Fox and Weisberg, 2019) to illustrate the backfitting algorithm. Each row in the dataset is an observation that relates to an occupation. The columns relate to predictors such as average years of education for this occupation, average income in this occupation, etc. The response is the prestige score for the occupation. The goal is to understand how education and income influence the prestige score. We apply the backfitting algorithm to fit the additive model

$$\text{prestige}_i = \alpha + f_1(\text{income}_i) + f_2(\text{education}_i) + \varepsilon_i, \quad i = 1, \ldots, 102. \tag{2.3}$$

R-Code 2.1. Backfitting algorithm for additive models

```
# load the dataset
data("Prestige")
n <- dim(Prestige)[1] # n = 102

prestige <- Prestige$prestige
income <- Prestige$income
education <- Prestige$education


## backfitting algorithm ####

# maximum number of iterations allowed
itermax <- 20

# initialize hat{alpha} and hat{f_j} for j = 1,2
alpha_hat <- mean(prestige)
f1 <- matrix(rep(0, n), nrow = n, ncol = itermax)
f2 <- matrix(rep(0, n), nrow = n, ncol = itermax)

# parameters for convergence
eps <- 10^(-10)
conv <- 0
t <- 1

# degrees of freedom used in smoothing splines
this_df <- 4

while (!conv) {
  t <- t + 1

  # for hat{f_1}
  r1 <- prestige - alpha_hat - f2[,t-1]
  f1.new <- predict(smooth.spline(income, r1, df = this_df), income)$y
  f1[,t] <- f1.new - mean(f1.new)
```

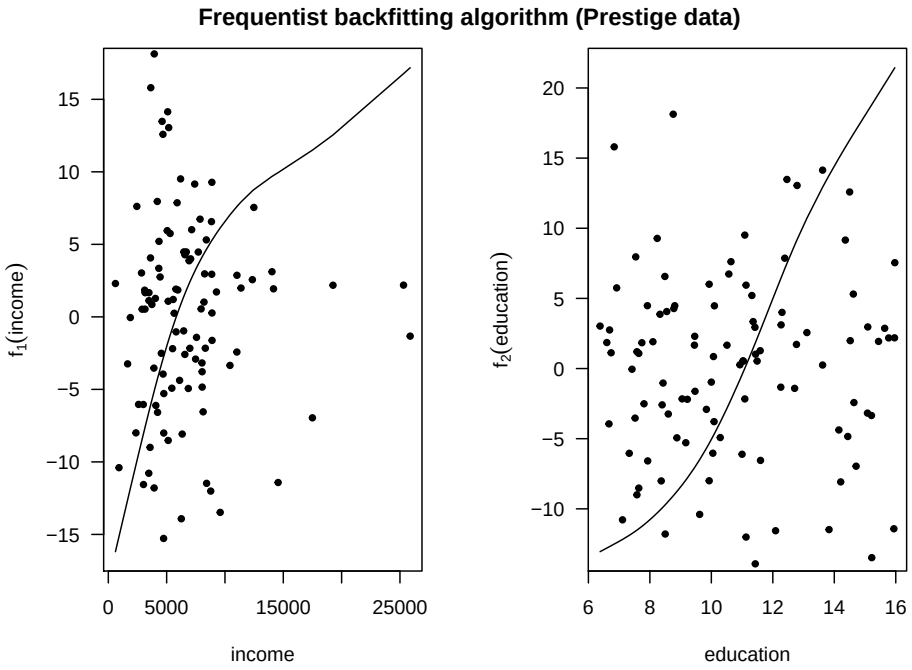

Figure 2.1: Estimated functions for income and education from the *Prestige* dataset using the frequentist backfitting algorithm. The displayed black dots represent the residuals (see R-Code 2.1).

```
  # for hat{f_2}
  r2 <- prestige - alpha_hat - f1[,t]
  f2.new <- predict(smooth.spline(education, r2, df = this_df), education)$y
  f2[,t] <- f2.new - mean(f2.new)

  # check convergence
  maxerr <- max(sum((f1[,t]-f1[,t-1])^2) / sum(f1[,t-1]^2),
                sum((f2[,t]-f2[,t-1])^2) / sum(f2[,t-1]^2))
  conv <- maxerr < eps | t >= itermax
}

# number of iterations
(iter <- t-1)

## [1] 17
```

The backfitting algorithm for the additive model given in equation (2.3) converges after 17 iterations with $\epsilon = 10^{-10}$. Figure 2.1 shows the estimated functions for income and education. As expected, the prestige score increases with respect to income and years of education. However, the linear relationship between these variables is not linear.

2.2 Bayesian Backfitting for Additive Models

Our next objective is to inspect a general procedure for posterior sampling from additive models. This procedure is a stochastic generalization of the frequentist backfitting algorithm as seen before and is called *Bayesian backfitting* (Hastie and Tibshirani, 2000). The main idea evolves from the close relationship between the backfitting algorithm for fitting additive models and the Gibbs sampler (Gamerman and Lopes, 2006, Chapter 5) for drawing realizations from a posterior distribution.

We start with a simple additive model. Consider the pairs $\{(x_1, Y_1), (x_2, Y_2), \ldots, (x_n, Y_n)\}$ where the Y_i's are the response values and the x_i's are the univariate predictor values. We postulate the model

$$Y_i = f(x_i) + \varepsilon_i, \quad i = 1, \ldots, n, \tag{2.4}$$

where f is an arbitrary smooth function and $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. From this we have observations $y_1, \ldots, y_n$.

We then take the cubic smoothing spline for estimating the function f. It is derived as the minimizer of the penalized sum of squares criterion

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int_{-\infty}^{+\infty} \left[f''(x)\right]^2 dx \tag{2.5}$$

over all functions f which are twice differentiable. The constant $\lambda \geq 0$ is a tuning parameter with larger values resulting in smoother curves.

The model in equation (2.4) can also be written in vector notation as

$$\boldsymbol{Y} = \boldsymbol{f} + \boldsymbol{\varepsilon},$$

where

- $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\top \in \mathbb{R}^n$,
- $\boldsymbol{f} = (f(x_1), \ldots, f(x_n))^\top \in \mathbb{R}^n$,
- $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)^\top \in \mathbb{R}^n$, $\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n)$.

The fitted values at the n input values x_i are then given by

$$\widehat{\boldsymbol{f}} = \boldsymbol{S}(\lambda)\boldsymbol{y},$$

where $\boldsymbol{S}(\lambda)$ is a symmetric $n \times n$-dimensional operator matrix, called the smoother matrix. It depends on the values x_i and λ, but not on $\boldsymbol{y}$. However, the cubic smoothing splines give a prediction for any value of x, not just the ones in the dataset, since they produce a function $\widehat{f}$.

Furthermore, there is also a Bayesian characterization of $\widehat{\boldsymbol{f}}$. We can impose an improper Gaussian prior for $\boldsymbol{f}$,

$$\boldsymbol{f} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{K}^- \tau^2),$$

where $\boldsymbol{K}^- = \lambda[\boldsymbol{S}(\lambda)^- - \boldsymbol{I}_n]^-$. Here, $\boldsymbol{A}^-$ denotes a generalized inverse of a matrix $\boldsymbol{A}$.

Remark 2.3. $\boldsymbol{K}$ determines the penalty in the penalized sum of squares criterion in (2.5), i.e.

$$\int_{-\infty}^{+\infty} \left[f''(x)\right]^2 dx = \boldsymbol{f}^\top \boldsymbol{K} \boldsymbol{f}.$$

The resulting posterior distribution of $\boldsymbol{f}$ has the form

$$\boldsymbol{f} \,|\, (\boldsymbol{Y} = \boldsymbol{y}) \sim \mathcal{N}_n(\boldsymbol{S}(\lambda)\boldsymbol{y}, \boldsymbol{S}(\lambda)\sigma^2) \tag{2.6}$$

with $\lambda = \sigma^2/\tau^2$ (see Hastie and Tibshirani, 2000). Hence, the cubic smoothing spline is the mean of the posterior distribution. However, the interest lies not only in the mean but in the entire posterior distribution of $\boldsymbol{f}$ given in (2.6).

Let $\boldsymbol{z} = (z_1, \ldots, z_n)^\top \in \mathbb{R}^n$ be a vector of independent $\mathcal{N}(0,1)$ random variables, i.e. $\boldsymbol{z} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{I}_n)$. Then

$$\boldsymbol{f} \,|\, (\boldsymbol{Y} = \boldsymbol{y}) \overset{\mathcal{D}}{=} \boldsymbol{S}\boldsymbol{y} + \sigma \boldsymbol{S}^{1/2} \boldsymbol{z}$$

has the same distribution as (2.6). Note that we have dropped the dependence on λ, i.e. $\boldsymbol{S} = \boldsymbol{S}(\lambda)$. Therefore, we can generate a posterior realization of $\boldsymbol{f}$ by adding the noise $\sigma \boldsymbol{S}^{1/2} \boldsymbol{z}$ to the fitted smoothing spline $\widehat{\boldsymbol{f}} = \boldsymbol{S}\boldsymbol{y}$. Remark that σ (or rather σ^2, the error variance) is considered to be fixed and not random in this setting.

Next, we consider an additive model of the form

$$Y_i = \alpha + \sum_{j=1}^{p} f_j(x_{ij}) + \varepsilon_i, \quad i = 1, \ldots, n,$$

where the f_j's are smooth functions and $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$.

Suppose that we define a cubic smoothing spline operator $\boldsymbol{S}_j$ for each $j = 1, \ldots, p$. Then the frequentist backfitting algorithm uses iterations of the form

$$\boldsymbol{f}_j = \boldsymbol{S}_j \left(\boldsymbol{y} - \bar{y}\boldsymbol{1} - \sum_{\substack{k=1 \\ k \neq j}}^{p} \boldsymbol{f}_k \right)$$

for estimating the $\boldsymbol{f}_j$'s for $j = 1, \ldots, p$.

We additionally define $\boldsymbol{f}_0 = \bar{y}\boldsymbol{1}$ where $\bar{y} = \frac{1}{n}\sum_{i=1}^n y_i$ and the associated operator matrix $\boldsymbol{S}_0 = \frac{1}{n}\boldsymbol{1}\boldsymbol{1}^\top$ where $\boldsymbol{1} = (1, \ldots, 1)^\top \in \mathbb{R}^n$. The variance σ^2 is considered to be fixed (i.e. non-random).

Rather than obtaining estimates for $\boldsymbol{f}_j$, which in the Bayesian language means to compute the maximum a posteriori (MAP) estimates (Shynk, 2012, Chapter 9.15) (here, the posterior means) we want to generate from their joint distribution.

For this reason, we define priors for each $\boldsymbol{f}_j$ by

$$\boldsymbol{f}_j \sim \mathcal{N}_n(\boldsymbol{0}, (\boldsymbol{S}_j^- - \boldsymbol{I}_n)^- \sigma^2).$$

Then the resulting posterior distribution of $\boldsymbol{f}_j$ for every $j = 0, 1, \ldots, p$ has the form

$$\boldsymbol{f}_j \,|\, \left(\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{f}_0, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_{j-1}, \boldsymbol{f}_{j+1}, \ldots, \boldsymbol{f}_p\right) \sim \mathcal{N}_n \left(\boldsymbol{S}_j \left(\boldsymbol{y} - \sum_{\substack{k=0 \\ k \neq j}}^{p} \boldsymbol{f}_k \right), \boldsymbol{S}_j \sigma^2 \right). \tag{2.7}$$

Let $\boldsymbol{z} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{I}_n)$. Then

$$\boldsymbol{f}_j \,|\, (\boldsymbol{Y} = \boldsymbol{y}) \overset{\mathcal{D}}{=} \boldsymbol{S}_j \left(\boldsymbol{y} - \sum_{\substack{k=0 \\ k \neq j}}^{p} \boldsymbol{f}_k \right) + \sigma \boldsymbol{S}_j^{1/2} \boldsymbol{z}$$

has the same distribution as (2.7). Therefore, we add the noise $\sigma \boldsymbol{S}_j^{1/2} \boldsymbol{z}$ to the estimate from the frequentist algorithm at each backfitting step. Hence, the Bayesian backfitting corresponds to sampling from the conditional distribution of each $\boldsymbol{f}_j$.

The Bayesian backfitting algorithm is defined as follows:

Algorithm 2.2 (Bayesian backfitting algorithm for additive models)**.**

[0] initialize $\boldsymbol{f}_j^{(0)} \in \mathbb{R}^n$ for $j = 1, \ldots, p$ and set $t = 0$

[1] $t = t + 1$

[2] do for $j = 0, 1, \ldots, p$:

- define the partial residual

$$\boldsymbol{r}_j^{(t)} = \boldsymbol{y} - \sum_{k=0}^{j-1} \boldsymbol{f}_k^{(t)} - \sum_{k=j+1}^{p} \boldsymbol{f}_k^{(t-1)}$$

- generate $\boldsymbol{z}_j^{(t)} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{I}_n)$ and update $\boldsymbol{f}_j^{(t)} = \boldsymbol{S}_j \boldsymbol{r}_j^{(t)} + \sigma \boldsymbol{S}_j^{1/2} \boldsymbol{z}_j^{(t)}$

[3] repeat [1] and [2] until the joint distribution of $\left(\boldsymbol{f}_0^{(t)}, \boldsymbol{f}_1^{(t)}, \ldots, \boldsymbol{f}_p^{(t)} \right)$ does not change

This algorithm refers to Algorithm 3.1 in Hastie and Tibshirani (2000).

Remark 2.4.

1. As initial values for $\boldsymbol{f}_j^{(0)}$ for $j = 1, \ldots, p$ we can choose for example the fitted functions from a standard additive model fit to the data (e.g. the estimated functions from the backfitting algorithm).
2. The variance σ^2 is usually unknown and has to be estimated. As an estimate for σ^2 we might choose the unbiased estimate $\widehat{\sigma}^2 = \frac{1}{n - \mathrm{df}} \sum_{i=1}^{n} (y_i - \widehat{y}_i)^2$ where $\widehat{y}_i = \widehat{\alpha} + \sum_{j=1}^{p} \widehat{f}_j(x_{ij})$ are the fitted values and df is the number of degrees of freedom which is used in the smoothing spline. In this case we would replace σ by $\widehat{\sigma}$ in step [2].
3. Observe that in contrast to the frequentist backfitting algorithm (Algorithm 2.1) we also update the function $\boldsymbol{f}_0$ which corresponds to the intercept term α.
4. By the results in Tierney (1994), the joint distribution of the iterates $\left(\boldsymbol{f}_0^{(t)}, \boldsymbol{f}_1^{(t)}, \ldots, \boldsymbol{f}_p^{(t)} \right)$ converges to that of the true distribution of $\left(\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_p \right) \,|\, (\boldsymbol{Y} = \boldsymbol{y})$.
5. To generate $\boldsymbol{S}_j^{1/2} \boldsymbol{z}_j^{(t)}$ we use the Algorithm 2.3 with $\boldsymbol{S} = \boldsymbol{S}_j$.

Instead of directly factorizing $\boldsymbol{S}_j$ we use an approach that works for any (symmetric) smoother $\boldsymbol{S}$ with eigenvalues in $[0, 1]$ and has essentially $\mathcal{O}(n^2)$ operation count. The algorithm is the following:

Algorithm 2.3 (General procedure for generating $\boldsymbol{S}^{1/2}\boldsymbol{z}$).

[1] generate $\boldsymbol{z} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{I}_n)$ and set $\boldsymbol{z}' = \boldsymbol{S}\boldsymbol{z}$, $\boldsymbol{z}'' = \boldsymbol{z}$ and $b = 1$

[2] $b = b + 1$

[3]
- put $\boldsymbol{z}'' = \frac{3/2-b}{b-1}(\boldsymbol{S}\boldsymbol{z}'' - \boldsymbol{z}'')$
- put $\boldsymbol{z}' = \boldsymbol{z}' + \boldsymbol{S}\boldsymbol{z}''$

[4] repeat [2] and [3] until $\|\boldsymbol{S}\boldsymbol{z}''\|$ is small enough

This algorithm refers to Algorithm A.1 in Hastie and Tibshirani (2000).

Remark 2.5. In Algorithm 2.3, $\boldsymbol{z}'$ converges to $\boldsymbol{S}^{1/2}\boldsymbol{z}$.

We use the following self-implemented R function to generate $\boldsymbol{S}^{1/2}\boldsymbol{z}$:

R-Code 2.2. Algorithm for generating $\boldsymbol{S}^{1/2}\boldsymbol{z}$

```
generateSqrtSz <- function(S) {
  # see Algorithm A.1 in Hastie and Tibshirani (2000)

  n <- dim(S)[1]

  # generate z ~ N(0,I_n)
  z <- rnorm(n, mean = 0, sd = 1)

  z1 <- S %*% z
  z2 <- z

  conv <- 0
  b <- 2

  while (!conv) {
    z2 <- (3/2-b)/(b-1) * (S %*% z2 - z2)
    z1 <- z1 + S %*% z2

    # check convergence
    conv <- max(abs(S %*% z2)) < 0.01

    # update b
    b <- b+1
  }

  # z1 should converge to S^(1/2)z
  return(z1)
}
```

Let us delve into the Bayesian backfitting algorithm in R. Consider again the *Prestige* dataset

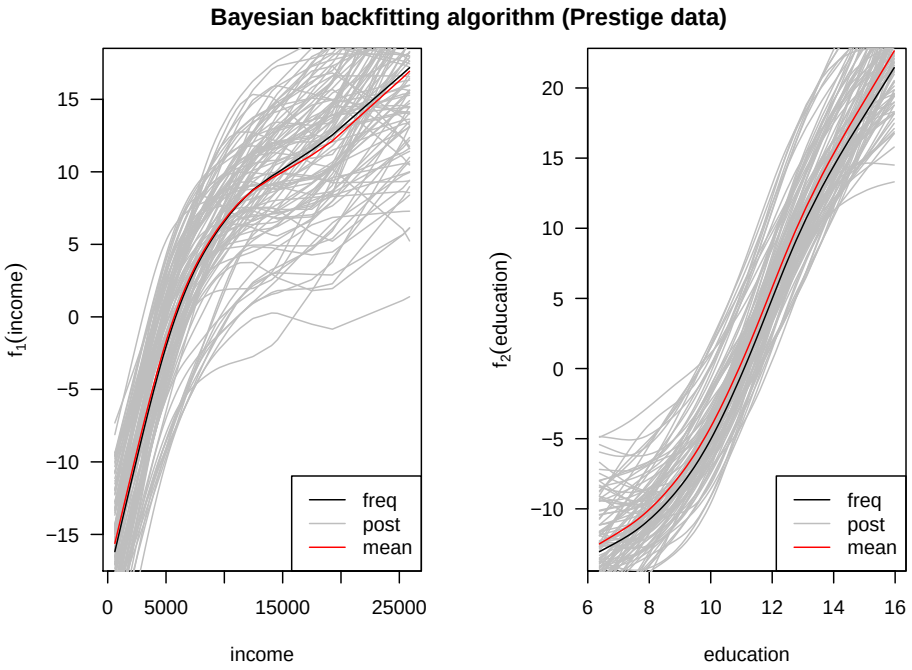

Figure 2.2: 100 posterior realizations (shown in grey) for income and education from the *Prestige* dataset using the Bayesian backfitting algorithm. The black lines are the estimated functions from the frequentist backfitting algorithm and the red lines are the posterior means (see R-Code 2.3).

from the R package *car* (Fox and Weisberg, 2019) and the additive model

$$\text{prestige}_i = \alpha + f_1(\text{income}_i) + f_2(\text{education}_i) + \varepsilon_i, \quad i = 1, \ldots, 102.$$

As the initial values for $\boldsymbol{f}_1^{(0)}$ and $\boldsymbol{f}_2^{(0)}$ we choose the estimated functions from the frequentist backfitting algorithm. Then 100 posterior realizations of the functions f_1 and f_2 are drawn. The results are shown in Figure 2.2.

R-Code 2.3. Bayesian backfitting algorithm for additive models

```
# smoother matrix for smoothing splines
smooth.matrix <- function(x, df) {
  sm <- matrix(0, n, n)
  for (i in 1:n) {
    y <- rep(0, n)
    y[i] <- 1
    sm[,i] <- predict(smooth.spline(x, y, df = df), x)$y
  }
  return(sm)
}

set.seed(6)
```

```r
# number of realizations from the posterior distribution
n_post <- 100

# initialize f_j for j = 0,1,2
f0 <- f1 <- f2 <- matrix(rep(0, n), nrow = n, ncol = n_post)
f0[,1] <- rep(mean(prestige), n)
f1[,1] <- freq_f1
f2[,1] <- freq_f2

# degrees of freedom used in smoothing splines
this_df <- 4

# smoother matrices associated to f_j for j = 0,1,2
S0 <- matrix(rep(1/n, n^2), nrow = n, ncol = n)
S1 <- smooth.matrix(income, df = this_df)
S2 <- smooth.matrix(education, df = this_df)

# parameters for convergence
eps <- 10^(-5)
conv <- 0
t <- 1

# hat{sigma} (estimated from the frequentist backfitting algorithm)
sigma <- sqrt(1/(n-this_df) * sum(res^2)) # res being the residuals

while (!conv) {
  t <- t + 1

  # for f_0
  r0 <- prestige - f1[,t-1] - f2[,t-1]
  f0[,t] <- S0 %*% r0 + sigma * generateSqrtSz(S = S0)

  # for f_1
  r1 <- prestige - f0[,t] - f2[,t-1]
  f1[,t] <- S1 %*% r1 + sigma * generateSqrtSz(S = S1)

  # for f_2
  r2 <- prestige - f0[,t] - f1[,t]
  f2[,t] <- S2 %*% r2 + sigma * generateSqrtSz(S = S2)

  # check convergence
  maxerr <- max(sum((f1[,t]-f1[,t-1])^2) / sum(f1[,t-1]^2),
                sum((f2[,t]-f2[,t-1])^2) / sum(f2[,t-1]^2))
  conv <- maxerr < eps | t >= n_post
}
```

If we compare the frequentist and the Bayesian backfitting algorithm for additive models, we see that in one approach we end up with function estimates and in the other approach we get realizations from the posterior distribution of the functions. So, if the main focus lies on answering the question "which algorithm better estimates the functions?" then we need to take the mean of the posterior draws from the Bayesian backfitting algorithm into account.

However, the frequentist backfitting algorithm results in function estimates. One might ask what the uncertainties of these estimates are. Usually, this uncertainty is shown in confidence intervals. Therefore, in order to measure the uncertainty of the function estimates we should compute the confidence intervals of $\widehat{f_j}$ for each $j = 1, \dots, p$.

On the other hand, since the Bayesian backfitting algorithm is a method to draw from a distribution, we do not have to deal with uncertainty. We directly see the realizations from this distribution and do not have to compute confidence intervals or deal with other measures of uncertainty.

2.3 Posterior Realizations in Linear Regressions

We have seen in (2.6) that in a simple additive model (i.e. $\boldsymbol{Y} = \boldsymbol{f} + \boldsymbol{\varepsilon}$), with an appropriate prior for $\boldsymbol{f}$, we have

$$\boldsymbol{f} \,|\, (\boldsymbol{Y} = \boldsymbol{y}) \sim \mathcal{N}_n(\boldsymbol{S}(\lambda)\boldsymbol{y}, \boldsymbol{S}(\lambda)\sigma^2).$$

We turn our attention one more time to multiple linear regressions of the form $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ and want to examine how we can derive the posterior distribution of $\boldsymbol{X}\boldsymbol{\beta} \,|\, (\boldsymbol{Y} = \boldsymbol{y})$.

For this reason, consider the multiple linear regression model

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

- $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ is the design matrix,
- $\boldsymbol{\beta} \in \mathbb{R}^p$,
- $\boldsymbol{\varepsilon} \in \mathbb{R}^n$, $\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n)$.

In order to view this in a Bayesian framework we choose a prior for $\boldsymbol{\beta}$,

$$\boldsymbol{\beta} \sim \mathcal{N}_p(\boldsymbol{0}, \tau^2 \boldsymbol{I}_p).$$

Thus, $\boldsymbol{X}\boldsymbol{\beta} \sim \mathcal{N}_n\left(\boldsymbol{0}, \tau^2 \boldsymbol{X}\boldsymbol{X}^\top\right)$. We also know from the model formulation that

$$\boldsymbol{Y} \,|\, \boldsymbol{X}\boldsymbol{\beta} \sim \mathcal{N}_n(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2 \boldsymbol{I}_n).$$

Recall the following lemma from Held and Bové (2013, Chapter B.1.5) about combining quadratic forms:

Lemma 2.1. Let $\boldsymbol{x}$, $\boldsymbol{a}$ and $\boldsymbol{b}$ be vectors in $\mathbb{R}^n$ and $\boldsymbol{A}$ and $\boldsymbol{B}$ be symmetric $n \times n$-dimensional matrices such that the inverse of $\boldsymbol{C} := \boldsymbol{A} + \boldsymbol{B}$ exists. Then

$$(\boldsymbol{x} - \boldsymbol{a})^\top \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{a}) + (\boldsymbol{x} - \boldsymbol{b})^\top \boldsymbol{B}(\boldsymbol{x} - \boldsymbol{b}) = (\boldsymbol{x} - \boldsymbol{c})^\top \boldsymbol{C}(\boldsymbol{x} - \boldsymbol{c}) + (\boldsymbol{a} - \boldsymbol{b})^\top \boldsymbol{A}\boldsymbol{C}^{-1}\boldsymbol{B}(\boldsymbol{a} - \boldsymbol{b})$$

where $\boldsymbol{c} = \boldsymbol{C}^{-1}(\boldsymbol{A}\boldsymbol{a} + \boldsymbol{B}\boldsymbol{b})$.

Denote by $\pi(\cdot \,|\, \cdot)$ the density of the first argument given the second argument. We want to compute $\pi(\boldsymbol{X\beta} \,|\, \boldsymbol{Y} = \boldsymbol{y})$, the posterior density of $\boldsymbol{X\beta} \,|\, (\boldsymbol{Y} = \boldsymbol{y})$:

$$\begin{aligned}
\pi(\boldsymbol{X\beta} \,|\, \boldsymbol{Y} = \boldsymbol{y}) &\propto \pi(\boldsymbol{y} \,|\, \boldsymbol{X\beta}) \cdot \pi(\boldsymbol{X\beta}) \\
&\propto \exp\left(-\frac{1}{2}\left[(\boldsymbol{y} - \boldsymbol{X\beta})^\top \frac{1}{\sigma^2}\boldsymbol{I}_n(\boldsymbol{y} - \boldsymbol{X\beta}) + (\boldsymbol{X\beta})^\top \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\boldsymbol{X\beta}\right]\right) \\
&\propto \exp\left(-\frac{1}{2}\left[(\boldsymbol{X\beta} - \boldsymbol{y})^\top \frac{1}{\sigma^2}\boldsymbol{I}_n(\boldsymbol{X\beta} - \boldsymbol{y}) + (\boldsymbol{X\beta} - \boldsymbol{0})^\top \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}(\boldsymbol{X\beta} - \boldsymbol{0})\right]\right) \\
&\propto \exp\left(-\frac{1}{2}\left[\left(\boldsymbol{X\beta} - \left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2}\boldsymbol{y}\right)^\top \right.\right. \\
&\qquad \left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right) \\
&\qquad \left.\left.\left(\boldsymbol{X\beta} - \left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2}\boldsymbol{y}\right)\right]\right).
\end{aligned}$$

Here, we have used Lemma 2.1 in the last step.

Therefore,

$$\boldsymbol{X\beta} \,|\, (\boldsymbol{Y} = \boldsymbol{y}) \sim \mathcal{N}_n\left(\underbrace{\left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2}}_{=:\boldsymbol{S}(\tau^2)}\boldsymbol{y}, \underbrace{\left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1}}_{=\boldsymbol{S}(\tau^2)\sigma^2}\right) \tag{2.8}$$

and hence, letting $\boldsymbol{z} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{I}_n)$, $\boldsymbol{X\beta} \,|\, (\boldsymbol{Y} = \boldsymbol{y}) \overset{\mathcal{D}}{=} \boldsymbol{S}(\tau^2)\boldsymbol{y} + \sigma\boldsymbol{S}(\tau^2)^{1/2}\boldsymbol{z}$ has the same distribution as (2.8).

Observe that

$$\begin{aligned}
\boldsymbol{S}(\tau^2) &= \left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2} \\
&= \left(\boldsymbol{I}_n + \frac{\sigma^2}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1} \\
&= \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X} + \frac{\sigma^2}{\tau^2}\boldsymbol{I}_p\right)^{-1}\boldsymbol{X}^\top.
\end{aligned}$$

However, taking $\tau \to \infty$ to represent prior ignorance yields

$$\boldsymbol{S}(\tau^2) \xrightarrow{\tau\to\infty} \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top =: \boldsymbol{H},$$

where $\boldsymbol{H}$ denotes the hat matrix in linear regression.

Lemma 2.2. The hat matrix $\boldsymbol{H} = \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top$ is idempotent, i.e. $\boldsymbol{H}^2 = \boldsymbol{H}$.

Proof.

$$\begin{aligned}
\boldsymbol{H}^2 &= \boldsymbol{X}\underbrace{\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top \cdot \boldsymbol{X}}_{=\boldsymbol{I}_p}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top \\
&= \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top \\
&= \boldsymbol{H}
\end{aligned}$$

□

Therefore, we can write the posterior realizations of $\boldsymbol{X\beta}\,|\,(\boldsymbol{Y}=\boldsymbol{y})$ in the form

$$\begin{aligned}\boldsymbol{X\beta}\,|\,(\boldsymbol{Y}=\boldsymbol{y}) &\overset{\mathcal{D}}{=} \boldsymbol{Hy}+\sigma\boldsymbol{H}^{1/2}\boldsymbol{z}\\ &= \boldsymbol{Hy}+\sigma\boldsymbol{Hz}\\ &= \underbrace{\boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}}_{\text{fitted values}}+\sigma\boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{z}.\end{aligned}$$

Finally,

$$\boldsymbol{\beta}\,|\,(\boldsymbol{Y}=\boldsymbol{y}) \overset{\mathcal{D}}{=} \underbrace{\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}}_{=\widehat{\boldsymbol{\beta}}_{\text{OLS}}}+\sigma\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{z}. \tag{2.9}$$

The first term in equation (2.9) is equal to the ordinary least squares estimate of $\boldsymbol{\beta}$. Therefore, the noise term $\sigma\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{z}$ is added to the estimate of $\boldsymbol{\beta}$ which we would compute using a frequentist approach.

We generate posterior realizations in linear regressions in R using the following code:

R-Code 2.4. Posterior realizations in linear regressions

```
## model setup ####
set.seed(1)

n <- 100 # number of observations

# predictors
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
x4 <- rnorm(n)

# true beta
beta <- c(1.4, 3.1, -0.8, -2.5, 1)

# design matrix X
X <- as.matrix(cbind(1, x1, x2, x3, x4))

# observed values
sigma <- 1.1
y <- X %*% beta + rnorm(n, mean = 0, sd = sigma)

## Posterior realizations of beta ####
n_post <- 1000
beta_post <- matrix(rep(0, 5), nrow = 5, ncol = n_post)
for (i in 1:n_post) {
  z <- rnorm(n)
  beta_post[,i] <- solve(t(X) %*% X, t(X) %*% y) +
```

```
    sigma * solve(t(X) %*% X, t(X) %*% z)
}
```

Figure 2.3 shows the effect of β_i on the response for $i = 1, \ldots, 4$ and Figure 2.4 shows the trace plots of the 1'000 posterior realizations of $\boldsymbol{\beta}$, which look as expected.

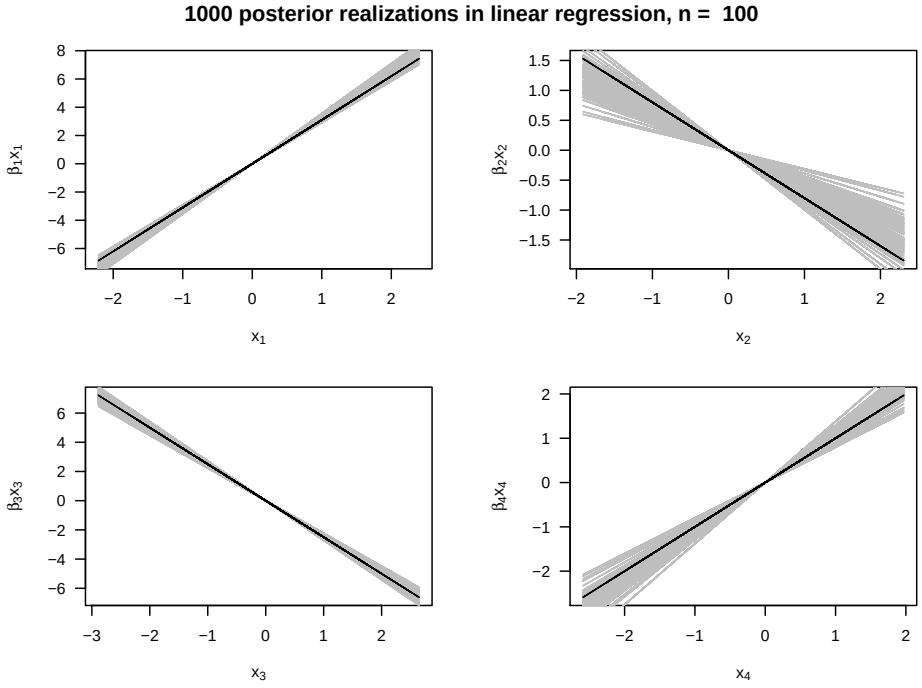

Figure 2.3: Visualization of 1'000 posterior realizations for $\beta_1, \beta_2, \beta_3, \beta_4$ in a linear regression. The grey lines show the posterior realizations $\beta_i x_i$ and the black lines correspond to the true functions $\beta_i x_i$ for $i = 1, 2, 3, 4$ (see R-Code 2.4).

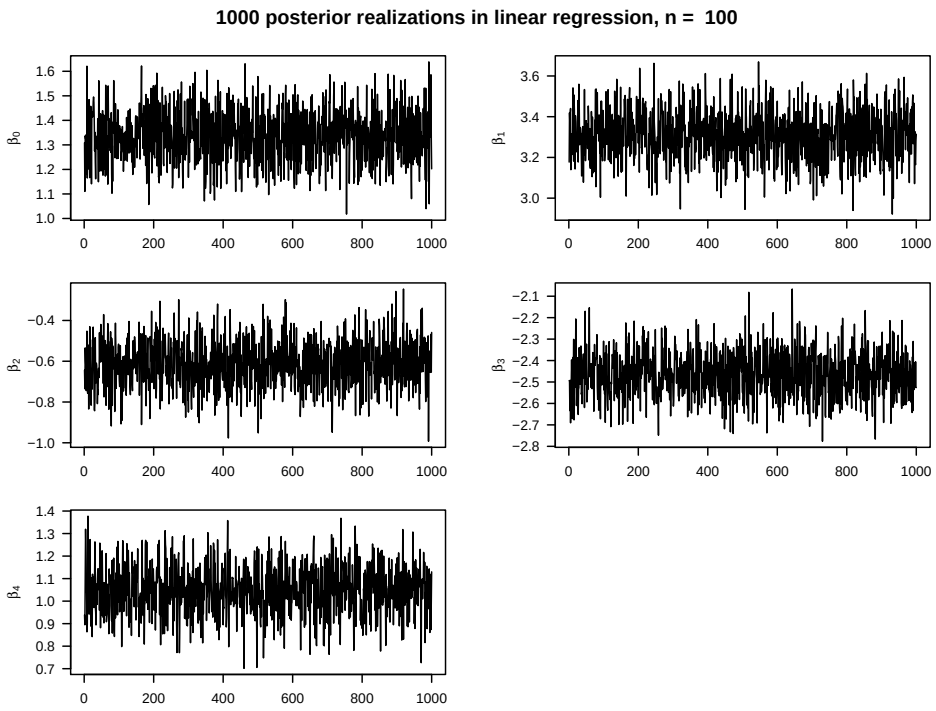

Figure 2.4: Trace plots of 1'000 posterior realizations for $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$ in a linear regression (see R-Code 2.4).

Chapter 3

Backfitting for Mixed Models

The next target is to extend the backfitting algorithm to the class of linear mixed-effects models.

Definition 3.1. A linear mixed-effects model (or simply a linear mixed model) is a model of the form

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon}, \tag{3.1}$$

where

- $\boldsymbol{Y}$ is an n-dimensional response vector,
- $\boldsymbol{\beta}$ is a p-dimensional vector of fixed effects,
- $\boldsymbol{u}$ is a q-dimensional vector of random effects,
- $\boldsymbol{X}$ is an $n \times p$-dimensional design matrix,
- $\boldsymbol{Z}$ is an $n \times q$-dimensional design matrix,
- $\boldsymbol{\varepsilon}$ is an n-dimensional error vector.

We usually assume that $\boldsymbol{u} \sim \mathcal{N}_q(\boldsymbol{0}, \boldsymbol{\Sigma}_u)$ and $\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma}_u$ is the $q \times q$ covariance matrix of $\boldsymbol{u}$ and $\boldsymbol{\Sigma}$ is the $n \times n$ covariance matrix of $\boldsymbol{\varepsilon}$. Additionally, we suppose that $\boldsymbol{u}$ and $\boldsymbol{\varepsilon}$ are independent. Therefore, we have

$$\begin{aligned}
\mathbb{E}[\boldsymbol{Y}] &= \boldsymbol{X}\boldsymbol{\beta}, \\
\operatorname{Var}(\boldsymbol{Y}) &= \operatorname{Var}(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon}) \\
&= \operatorname{Var}(\boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon}) \\
&= \operatorname{Var}(\boldsymbol{Z}\boldsymbol{u}) + \operatorname{Var}(\boldsymbol{\varepsilon}) \\
&= \boldsymbol{Z} \operatorname{Var}(\boldsymbol{u}) \boldsymbol{Z}^\top + \operatorname{Var}(\boldsymbol{\varepsilon}) \\
&= \boldsymbol{Z}\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top + \boldsymbol{\Sigma}.
\end{aligned}$$

Remark 3.1. We denote the variance of $\boldsymbol{Y}$ by $\boldsymbol{V}$, i.e. $\boldsymbol{V} := \operatorname{Var}(\boldsymbol{Y}) = \boldsymbol{Z}\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top + \boldsymbol{\Sigma}$. Thus, $\boldsymbol{Y} \sim \mathcal{N}_n(\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{V})$.

If the covariance matrices $\boldsymbol{\Sigma}_u$ and $\boldsymbol{\Sigma}$ are known, then estimates of $\boldsymbol{\beta}$ and predictions of the random effects $\boldsymbol{u}$ are given by

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{y}, \tag{3.2}$$

$$\tilde{\boldsymbol{u}} = \boldsymbol{\Sigma}_u \boldsymbol{Z}^\top \boldsymbol{V}^{-1} (\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}), \tag{3.3}$$

where $\widehat{\boldsymbol{\beta}}$ is the generalized least squares (GLS) estimate of $\boldsymbol{\beta}$ and $\tilde{\boldsymbol{u}}$ is the best linear unbiased predictor (BLUP) (Henderson, 1975). Note that $\boldsymbol{y}$ is a realization of $\boldsymbol{Y}$ and denotes the vector of observations.

Remark 3.2.

1. The estimator $\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{Y}$ is unbiased, i.e.

$$\begin{aligned}
\mathbb{E}\left[\widehat{\boldsymbol{\beta}}\right] &= \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \mathbb{E}[\boldsymbol{Y}] \\
&= \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\boldsymbol{\beta} \\
&= \boldsymbol{\beta}.
\end{aligned}$$

2. The variance of $\widehat{\boldsymbol{\beta}}$ is given by

$$\begin{aligned}
\operatorname{Var}\left(\widehat{\boldsymbol{\beta}}\right) &= \operatorname{Var}\left(\left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{Y}\right) \\
&= \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \operatorname{Var}(\boldsymbol{Y}) \left(\left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1}\right)^\top \\
&= \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{V} \boldsymbol{V}^{-1} \boldsymbol{X} \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \\
&= \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X} \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \\
&= \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1}.
\end{aligned}$$

3. $\tilde{\boldsymbol{u}}$ is the conditional mode of $\boldsymbol{u}$ given $\boldsymbol{Y} = \boldsymbol{y}$, i.e. $\tilde{\boldsymbol{u}} = \operatorname{Mod}(\boldsymbol{u} \,|\, \boldsymbol{Y} = \boldsymbol{y})$. However, since we have a Gaussian setting, the conditional mode equals the conditional mean, i.e. $\tilde{\boldsymbol{u}} = \mathbb{E}[\boldsymbol{u} \,|\, \boldsymbol{Y} = \boldsymbol{y}]$.

Let us investigate how the GLS estimator for $\boldsymbol{\beta}$ is derived. First, we can write

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}^*,$$

where $\boldsymbol{\varepsilon}^* := \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{V})$. Then we can use a Cholesky decomposition to write $\boldsymbol{V} = \boldsymbol{L}\boldsymbol{L}^\top$, where $\boldsymbol{L}$ is a lower triangular matrix.

Next, we pre-multiply both sides of the equation $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}^*$ by $\boldsymbol{L}^{-1}$ and get

$$\tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{X}}\boldsymbol{\beta} + \tilde{\boldsymbol{\varepsilon}},$$

where $\tilde{\boldsymbol{Y}} = \boldsymbol{L}^{-1}\boldsymbol{Y}$, $\tilde{\boldsymbol{X}} = \boldsymbol{L}^{-1}\boldsymbol{X}$ and $\tilde{\boldsymbol{\varepsilon}} = \boldsymbol{L}^{-1}\boldsymbol{\varepsilon}^*$.

Notice that

$$\begin{aligned}
\operatorname{Var}(\tilde{\boldsymbol{\varepsilon}}) &= \operatorname{Var}(\boldsymbol{L}^{-1}\boldsymbol{\varepsilon}^*) \\
&= \boldsymbol{L}^{-1} \operatorname{Var}(\boldsymbol{\varepsilon}^*) \left(\boldsymbol{L}^{-1}\right)^\top \\
&= \boldsymbol{L}^{-1} \boldsymbol{V} \left(\boldsymbol{L}^{-1}\right)^\top \\
&= \boldsymbol{L}^{-1} \left(\boldsymbol{L}\boldsymbol{L}^\top\right) \left(\boldsymbol{L}^{-1}\right)^\top \\
&= \boldsymbol{I}_n.
\end{aligned}$$

Therefore, since the residuals $\tilde{\boldsymbol{\varepsilon}}$ are uncorrelated, we can apply the ordinary least squares method to $\tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{X}}\boldsymbol{\beta} + \tilde{\boldsymbol{\varepsilon}}$ and get

$$\begin{aligned}
\widehat{\boldsymbol{\beta}} &= \left(\tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{X}}\right)^{-1} \tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{Y}} \\
&= \left(\boldsymbol{X}^\top \underbrace{\left(\boldsymbol{L}^{-1}\right)^\top \boldsymbol{L}^{-1}}_{=\boldsymbol{V}^{-1}} \boldsymbol{X} \right)^{-1} \boldsymbol{X}^\top \underbrace{\left(\boldsymbol{L}^{-1}\right)^\top \boldsymbol{L}^{-1}}_{=\boldsymbol{V}^{-1}} \boldsymbol{Y} \\
&= \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{Y}.
\end{aligned}$$

The corresponding estimate is then

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{y}.$$

To obtain the expression of $\tilde{\boldsymbol{u}}$ from equation (3.3) we first recall that

$$\begin{aligned}
\boldsymbol{Y} &\sim \mathcal{N}_n(\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{V}), \\
\boldsymbol{u} &\sim \mathcal{N}_q(\boldsymbol{0}, \boldsymbol{\Sigma}_u).
\end{aligned}$$

In addition, we have

$$\begin{aligned}
\operatorname{Cov}(\boldsymbol{Y}, \boldsymbol{u}) &= \operatorname{Cov}(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon}, \boldsymbol{u}) \\
&= \operatorname{Cov}(\boldsymbol{Z}\boldsymbol{u}, \boldsymbol{u}) + \underbrace{\operatorname{Cov}(\boldsymbol{\varepsilon}, \boldsymbol{u})}_{=0} \\
&= \boldsymbol{Z} \operatorname{Var}(\boldsymbol{u}) \\
&= \boldsymbol{Z}\boldsymbol{\Sigma}_u
\end{aligned}$$

and

$$\begin{aligned}
\operatorname{Cov}(\boldsymbol{u}, \boldsymbol{Y}) &= \operatorname{Cov}(\boldsymbol{u}, \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon}) \\
&= \operatorname{Cov}(\boldsymbol{u}, \boldsymbol{Z}\boldsymbol{u}) + \underbrace{\operatorname{Cov}(\boldsymbol{u}, \boldsymbol{\varepsilon})}_{=0} \\
&= \operatorname{Var}(\boldsymbol{u}) \boldsymbol{Z}^\top \\
&= \boldsymbol{\Sigma}_u \boldsymbol{Z}^\top.
\end{aligned}$$

We end up with the following joint distribution of $\boldsymbol{Y}$ and $\boldsymbol{u}$:

$$(\boldsymbol{Y}, \boldsymbol{u}) \sim \mathcal{N}_{n+q}\left(\begin{pmatrix} \boldsymbol{X}\boldsymbol{\beta} \\ \boldsymbol{0} \end{pmatrix}, \begin{pmatrix} \boldsymbol{V} & \boldsymbol{Z}\boldsymbol{\Sigma}_u \\ \boldsymbol{\Sigma}_u \boldsymbol{Z}^\top & \boldsymbol{\Sigma}_u \end{pmatrix} \right).$$

Then we get (see Rue and Held, 2005, Chapter 2.1.7)

$$\mathbb{E}[\boldsymbol{u} \,|\, \boldsymbol{Y} = \boldsymbol{y}] = \boldsymbol{\Sigma}_u \boldsymbol{Z}^\top \boldsymbol{V}^{-1} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}).$$

By replacing $\boldsymbol{\beta}$ with the GLS estimate $\widehat{\boldsymbol{\beta}}$ from equation (3.2) we obtain exactly the BLUP from equation (3.3).

For the remaining part of this chapter we assume that $\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n)$, i.e. $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_n$. Then $\boldsymbol{V} = \boldsymbol{Z}\boldsymbol{\Sigma}_u \boldsymbol{Z}^\top + \sigma^2 \boldsymbol{I}_n$.

3.1 Frequentist Backfitting for Linear Mixed Models

Usually in linear mixed models, $\boldsymbol{\beta}$ is estimated by the generalized least squares method, i.e. $\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{y}$ (as seen in equation (3.2)). However, there might be a problem while computing the inverse of the inverse of the matrix $\boldsymbol{V}$ if we are dealing with a huge amount of data.

In such a case, we can use the backfitting algorithm to avoid computing this double inverse of the matrix $\boldsymbol{V}$. The idea is that we iteratively estimate first the fixed effects $\boldsymbol{\beta}$ while keeping the random effects fixed. This estimation is done via the ordinary least squares method and does not even take the matrix $\boldsymbol{V}$ into account. In a second step within the iteration, we predict the random effect $\boldsymbol{u}$ with a similar formula as for the BLUP.

The backfitting algorithm for linear mixed models is defined as follows:

Algorithm 3.1 (Backfitting algorithm for linear mixed models)**.**

[0] initialize $\tilde{\boldsymbol{u}}^{(0)}$ and set $t = 0$

[1] $t = t + 1$

[2] put $\widehat{\boldsymbol{\beta}}^{(t)} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \left(\boldsymbol{y} - \boldsymbol{Z}\tilde{\boldsymbol{u}}^{(t-1)}\right)$

[3]
- estimate $\widehat{\boldsymbol{\Sigma}}_u^{(t)}$ and $\widehat{\sigma}^{(t)}$ to get $\widehat{\boldsymbol{V}}^{(t)} = \boldsymbol{Z}\widehat{\boldsymbol{\Sigma}}_u^{(t)} \boldsymbol{Z}^\top + \left(\widehat{\sigma}^{(t)}\right)^2 \boldsymbol{I}_n$
- put $\tilde{\boldsymbol{u}}^{(t)} = \widehat{\boldsymbol{\Sigma}}_u^{(t)} \boldsymbol{Z}^\top \left(\widehat{\boldsymbol{V}}^{(t)}\right)^{-1} \left(\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}^{(t)}\right)$

[4] repeat [1] to [3] until convergence

Remark 3.3.

1. In contrast to the backfitting algorithm for linear regressions (Algorithm 1.1) we fit in Algorithm 3.1 all parameters for the fixed effects β_i, $i = 1, \ldots, p$ in one step and all parameters for the random effects u_j, $j = 1, \ldots, q$ in a second step. This strategy speeds up the backfitting procedure.

2. In step [2] we observe that $\widehat{\boldsymbol{\beta}}^{(t)}$ has the form of an ordinary least squares estimate keeping the random effects fixed.

3. In step [3] we discover that $\tilde{\boldsymbol{u}}^{(t)}$ is similar to the BLUP from equation (3.3) with the true covariance matrices replaced by their estimates.

4. If $\boldsymbol{\Sigma}_u$ and σ are known then we can directly use these quantities and do not have to estimate them in step [3].

5. Similarly, if $\boldsymbol{\Sigma}_u$ and σ are unknown but the estimates $\widehat{\boldsymbol{\Sigma}}_u$ and $\widehat{\sigma}$ are available before starting the algorithm then these estimates can be used instead of $\widehat{\boldsymbol{\Sigma}}_u^{(t)}$ and $\widehat{\sigma}^{(t)}$ in step [3].

The various R codes for the frequentist backfitting algorithm are listed in the Appendix (R-Codes A.2 and A.4).

3.2 Bayesian Backfitting for Linear Mixed Models

Our next intention is to give a Bayesian interpretation of the backfitting algorithm for linear mixed models. Consider again a linear mixed model of the form

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon},$$

where $\boldsymbol{u} \sim \mathcal{N}_q(\boldsymbol{0}, \boldsymbol{\Sigma}_u)$, $\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n)$ and $\boldsymbol{u}$ and $\boldsymbol{\varepsilon}$ are independent.

Similar as in Section 2.3 we propose a prior for $\boldsymbol{\beta}$,

$$\boldsymbol{\beta} \sim \mathcal{N}_p(\boldsymbol{0}, \tau^2 \boldsymbol{I}_p).$$

We then have

$$\begin{aligned}
\boldsymbol{X}\boldsymbol{\beta} &\sim \mathcal{N}_n\left(\boldsymbol{0}, \tau^2 \boldsymbol{X}\boldsymbol{X}^\top\right), \\
\boldsymbol{Z}\boldsymbol{u} &\sim \mathcal{N}_n\left(\boldsymbol{0}, \boldsymbol{Z}\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top\right), \\
\boldsymbol{Y} \mid (\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{Z}\boldsymbol{u}) &\sim \mathcal{N}_n\left(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u}, \sigma^2 \boldsymbol{I}_n\right).
\end{aligned}$$

The derivation of the posterior distributions of $\boldsymbol{X}\boldsymbol{\beta} \mid (\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z}\boldsymbol{u})$ and $\boldsymbol{Z}\boldsymbol{u} \mid (\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{X}\boldsymbol{\beta})$ works similar as in Section 2.3.

We start with the posterior density of $\boldsymbol{X}\boldsymbol{\beta} \mid (\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z}\boldsymbol{u})$ denoted by $\pi(\boldsymbol{X}\boldsymbol{\beta} \mid \boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z}\boldsymbol{u})$:

$$\begin{aligned}
\pi(\boldsymbol{X}\boldsymbol{\beta} \mid \boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z}\boldsymbol{u}) &\propto \pi(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{Z}\boldsymbol{u}) \cdot \underbrace{\pi(\boldsymbol{X}\boldsymbol{\beta} \mid \boldsymbol{Z}\boldsymbol{u})}_{=\pi(\boldsymbol{X}\boldsymbol{\beta})} \\
&\propto \exp\left(-\frac{1}{2}\left[(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Z}\boldsymbol{u})^\top \frac{1}{\sigma^2}\boldsymbol{I}_n(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Z}\boldsymbol{u})\right.\right. \\
&\qquad \left.\left. + (\boldsymbol{X}\boldsymbol{\beta})^\top \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\boldsymbol{X}\boldsymbol{\beta}\right]\right) \\
&\propto \exp\left(-\frac{1}{2}\left[(\boldsymbol{X}\boldsymbol{\beta} - (\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{u}))^\top \frac{1}{\sigma^2}\boldsymbol{I}_n(\boldsymbol{X}\boldsymbol{\beta} - (\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{u}))\right.\right. \\
&\qquad \left.\left. + (\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{0})^\top \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{0})\right]\right) \\
&\propto \exp\left(-\frac{1}{2}\left[\left(\boldsymbol{X}\boldsymbol{\beta} - \left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2}(\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{u})\right)^\top\right.\right. \\
&\qquad \left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right) \\
&\qquad \left.\left.\left(\boldsymbol{X}\boldsymbol{\beta} - \left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2}(\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{u})\right)\right]\right).
\end{aligned}$$

Here, we have used in the first step that $\boldsymbol{Z}\boldsymbol{u}$ and $\boldsymbol{X}\boldsymbol{\beta}$ are independent and Lemma 2.1 in the last step.

Thus,

$$\boldsymbol{X}\boldsymbol{\beta} \mid (\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z}\boldsymbol{u}) \sim \mathcal{N}_n\left(\underbrace{\left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2}}_{=:\boldsymbol{S}_\beta(\tau^2)}(\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{u}), \underbrace{\left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1}}_{=\boldsymbol{S}_\beta(\tau^2)\sigma^2}\right).$$

Similar as for the posterior realizations in linear regressions we let $\tau \to \infty$ to represent prior ignorance. Then $\boldsymbol{S}_\beta(\tau^2) \xrightarrow{\tau\to\infty} \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top =: \boldsymbol{H}$, where $\boldsymbol{H}$ is the hat matrix.

Let $\boldsymbol{z} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{I}_n)$. Then we have

$$\begin{aligned}\boldsymbol{X\beta}\,|\,(\boldsymbol{Y}=\boldsymbol{y}) &\overset{\mathcal{D}}{=} \boldsymbol{H}(\boldsymbol{y}-\boldsymbol{Zu}) + \sigma\boldsymbol{H}^{1/2}\boldsymbol{z}\\ &= \boldsymbol{H}(\boldsymbol{y}-\boldsymbol{Zu}) + \sigma\boldsymbol{H}\boldsymbol{z}\\ &= \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top(\boldsymbol{y}-\boldsymbol{Zu}) + \sigma\boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{z},\end{aligned}$$

where we have used that $\boldsymbol{H} = \boldsymbol{H}^{1/2}$ (see Lemma 2.2).

Hence,

$$\boldsymbol{\beta}\,|\,(\boldsymbol{Y}=\boldsymbol{y}) \overset{\mathcal{D}}{=} \left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top(\boldsymbol{y}-\boldsymbol{Zu}) + \sigma\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{z}, \tag{3.4}$$

where $\boldsymbol{z} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{I}_n)$.

The first term in equation (3.4) refers to an ordinary least squares estimate if we keep the random effects fixed. Therefore, for the posterior realizations of $\boldsymbol{\beta}$ the noise term $\sigma\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{z}$ is added to the least squares estimate of $\boldsymbol{\beta}$.

Let us compute the posterior density of $\boldsymbol{Zu}\,|\,(\boldsymbol{Y}=\boldsymbol{y}, \boldsymbol{X\beta})$ denoted by $\pi(\boldsymbol{Zu}\,|\,\boldsymbol{Y}=\boldsymbol{y}, \boldsymbol{X\beta})$:

$$\begin{aligned}\pi(\boldsymbol{Zu}\,|\,\boldsymbol{Y}=\boldsymbol{y}, \boldsymbol{X\beta}) &\propto \pi(\boldsymbol{y}\,|\,\boldsymbol{X\beta}, \boldsymbol{Zu})\cdot\underbrace{\pi(\boldsymbol{Zu}\,|\,\boldsymbol{X\beta})}_{=\pi(\boldsymbol{Zu})}\\ &\propto \exp\left(-\frac{1}{2}\left[(\boldsymbol{y}-\boldsymbol{X\beta}-\boldsymbol{Zu})^\top\frac{1}{\sigma^2}\boldsymbol{I}_n(\boldsymbol{y}-\boldsymbol{X\beta}-\boldsymbol{Zu})\right.\right.\\ &\qquad\left.\left.+(\boldsymbol{Zu})^\top\left(\boldsymbol{Z\Sigma}_u\boldsymbol{Z}^\top\right)^{-1}\boldsymbol{Zu}\right]\right)\\ &\propto \exp\left(-\frac{1}{2}\left[(\boldsymbol{Zu}-(\boldsymbol{y}-\boldsymbol{X\beta}))^\top\frac{1}{\sigma^2}\boldsymbol{I}_n(\boldsymbol{Zu}-(\boldsymbol{y}-\boldsymbol{X\beta}))\right.\right.\\ &\qquad\left.\left.+(\boldsymbol{Zu}-\boldsymbol{0})^\top\left(\boldsymbol{Z\Sigma}_u\boldsymbol{Z}^\top\right)^{-1}(\boldsymbol{Zu}-\boldsymbol{0})\right]\right)\\ &\propto \exp\left(-\frac{1}{2}\left[\left(\boldsymbol{Zu}-\left(\frac{1}{\sigma^2}\boldsymbol{I}_n+\left(\boldsymbol{Z\Sigma}_u\boldsymbol{Z}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2}(\boldsymbol{y}-\boldsymbol{X\beta})\right)^\top\right.\right.\\ &\qquad\left(\frac{1}{\sigma^2}\boldsymbol{I}_n+\left(\boldsymbol{Z\Sigma}_u\boldsymbol{Z}^\top\right)^{-1}\right)\\ &\qquad\left.\left.\left(\boldsymbol{Zu}-\left(\frac{1}{\sigma^2}\boldsymbol{I}_n+\left(\boldsymbol{Z\Sigma}_u\boldsymbol{Z}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2}(\boldsymbol{y}-\boldsymbol{X\beta})\right)\right]\right).\end{aligned}$$

Again, we have used in the first step that $\boldsymbol{Zu}$ and $\boldsymbol{X\beta}$ are independent and Lemma 2.1 in the last step.

Therefore,

$$\boldsymbol{Zu}\,|\,(\boldsymbol{Y}=\boldsymbol{y}, \boldsymbol{X\beta}) \sim \mathcal{N}_n\left(\underbrace{\left(\frac{1}{\sigma^2}\boldsymbol{I}_n+\left(\boldsymbol{Z\Sigma}_u\boldsymbol{Z}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2}}_{=:\boldsymbol{S}_u}(\boldsymbol{y}-\boldsymbol{X\beta}), \underbrace{\left(\frac{1}{\sigma^2}\boldsymbol{I}_n+\left(\boldsymbol{Z\Sigma}_u\boldsymbol{Z}^\top\right)^{-1}\right)^{-1}}_{=\boldsymbol{S}_u\sigma^2}\right).$$

With some matrix manipulations we get

$$\begin{aligned}\boldsymbol{S}_u &= \left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \left(\boldsymbol{Z}\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2}\\ &= \left(\boldsymbol{I}_n + \sigma^2\left(\boldsymbol{Z}\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top\right)^{-1}\right)^{-1}\\ &= \boldsymbol{Z}\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top\left(\boldsymbol{Z}\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top + \sigma^2\boldsymbol{I}_n\right)^{-1}\\ &= \boldsymbol{Z}\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top\boldsymbol{V}^{-1}.\end{aligned}$$

Hence,

$$\begin{aligned}\boldsymbol{Z}\boldsymbol{u}\,|\,(\boldsymbol{Y}=\boldsymbol{y}) &\overset{\mathcal{D}}{=} \boldsymbol{S}_u(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) + \sigma\boldsymbol{S}_u^{1/2}\boldsymbol{z}\\ &= \boldsymbol{Z}\underbrace{\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top\boldsymbol{V}^{-1}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})}_{\text{similar to the BLUP}} + \sigma\left[\boldsymbol{Z}\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top\boldsymbol{V}^{-1}\right]^{1/2}\boldsymbol{z},\end{aligned} \tag{3.5}$$

where $\boldsymbol{z}\sim\mathcal{N}_n(\boldsymbol{0},\boldsymbol{I}_n)$.

Here, we almost obtain the BLUP as a part of the first term in equation (3.5). Recall that in the expression for the BLUP $\tilde{\boldsymbol{u}}$ (equation (3.3)) we have the GLS estimate $\widehat{\boldsymbol{\beta}}$ instead of the true $\boldsymbol{\beta}$.

However, since in general $\boldsymbol{Z}$ is not a square (or even invertible) matrix, we cannot simply pre-multiply the equation (3.5) by $\boldsymbol{Z}^{-1}$. Nevertheless, we can pre-multiply it by $\left(\boldsymbol{Z}^\top\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^\top$ and end up with just $\boldsymbol{u}$ on the left hand side of equation (3.5). We then arrive at

$$\boldsymbol{u}\,|\,(\boldsymbol{Y}=\boldsymbol{y}) \overset{\mathcal{D}}{=} \boldsymbol{\Sigma}_u\boldsymbol{Z}^\top\boldsymbol{V}^{-1}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) + \sigma\left(\boldsymbol{Z}^\top\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^\top\left[\boldsymbol{Z}\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top\boldsymbol{V}^{-1}\right]^{1/2}\boldsymbol{z}.$$

For the posterior realizations of $\boldsymbol{u}$ the noise term $\sigma\left(\boldsymbol{Z}^\top\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^\top\left[\boldsymbol{Z}\boldsymbol{\Sigma}_u\boldsymbol{Z}^\top\boldsymbol{V}^{-1}\right]^{1/2}\boldsymbol{z}$ is added to a term which resembles the BLUP $\tilde{\boldsymbol{u}}$.

With this in mind, we can state the Bayesian backfitting algorithm for linear mixed models.

Algorithm 3.2 (Bayesian backfitting algorithm for linear mixed models)**.**

[0] initialize $(\boldsymbol{Z}\tilde{\boldsymbol{u}})^{(0)}$ and $\widehat{\sigma}^{(0)}$ and set $t=0$

[1] $t=t+1$

[2]
- generate $\boldsymbol{z}\sim\mathcal{N}_n(\boldsymbol{0},\boldsymbol{I}_n)$
- put $\widehat{\boldsymbol{\beta}}^{(t)} = \left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\left(\boldsymbol{y}-(\boldsymbol{Z}\tilde{\boldsymbol{u}})^{(t-1)}\right) + \widehat{\sigma}^{(t-1)}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{z}$

[3]
- generate $\boldsymbol{w}\sim\mathcal{N}_n(\boldsymbol{0},\boldsymbol{I}_n)$
- estimate $\widehat{\boldsymbol{\Sigma}}_u^{(t)}$ and $\widehat{\sigma}^{(t)}$ to get $\widehat{\boldsymbol{V}}^{(t)} = \boldsymbol{Z}\widehat{\boldsymbol{\Sigma}}_u^{(t)}\boldsymbol{Z}^\top + \left(\widehat{\sigma}^{(t)}\right)^2\boldsymbol{I}_n$
- put $(\boldsymbol{Z}\tilde{\boldsymbol{u}})^{(t)} = \boldsymbol{Z}\widehat{\boldsymbol{\Sigma}}_u^{(t)}\boldsymbol{Z}^\top\left(\widehat{\boldsymbol{V}}^{(t)}\right)^{-1}\left(\boldsymbol{y}-\boldsymbol{X}\widehat{\boldsymbol{\beta}}^{(t)}\right) + \widehat{\sigma}^{(t)}\left[\boldsymbol{Z}\widehat{\boldsymbol{\Sigma}}_u^{(t)}\boldsymbol{Z}^\top\left(\widehat{\boldsymbol{V}}^{(t)}\right)^{-1}\right]^{1/2}\boldsymbol{w}$

[4] repeat [1] to [3] until convergence

Remark 3.4.

1. Compared to the frequentist backfitting algorithm for linear mixed models (Algorithm 3.1) we initialize $\boldsymbol{Z}\tilde{\boldsymbol{u}}$ instead of only $\tilde{\boldsymbol{u}}$. Additionally, we have to initialize $\widehat{\sigma}$ because it is already used in step [2].
2. To compute the matrix square root in step [3] we use Algorithm 2.3 with the matrix $\boldsymbol{S}$ being $\boldsymbol{Z}\widehat{\boldsymbol{\Sigma}}_u^{(t)}\boldsymbol{Z}^\top\left(\widehat{\boldsymbol{V}}^{(t)}\right)^{-1}$.
3. If $\boldsymbol{\Sigma}_u$ and σ are known then we can directly use these quantities and do not have to estimate them in step [3].
4. Similarly, if $\boldsymbol{\Sigma}_u$ and σ are unknown but the estimates $\widehat{\boldsymbol{\Sigma}}_u$ and $\widehat{\sigma}$ are available before starting the algorithm then these estimates can be used instead of $\widehat{\boldsymbol{\Sigma}}_u^{(t)}$ and $\widehat{\sigma}^{(t)}$ in step [3].
5. If we want the algorithm to result in $\tilde{\boldsymbol{u}}^{(t)}$ instead of $(\boldsymbol{Z}\tilde{\boldsymbol{u}})^{(t)}$ we can pre-multiply $(\boldsymbol{Z}\tilde{\boldsymbol{u}})^{(t)}$ by $\left(\boldsymbol{Z}^\top\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^\top$ to end up with only $\tilde{\boldsymbol{u}}^{(t)}$.

The various R codes for the Bayesian backfitting algorithm are listed in the Appendix (R-Codes A.3 and A.5)

Similar as for generalized additive models, the backfitting algorithms for linear mixed models have different outcomes. The frequentist backfitting algorithm returns an estimate for $\boldsymbol{\beta}$ and a prediction for $\boldsymbol{u}$. However, the Bayesian backfitting algorithm results in a whole sample for $\boldsymbol{\beta}$ and $\boldsymbol{u}$.

We can again ask about uncertainty of the estimates from the frequentist backfitting algorithm. If we want to have a look at the confidence intervals we might compute a Wald type confidence interval for $\widehat{\boldsymbol{\beta}}$ and calculate the standard error of $\widehat{\boldsymbol{\beta}}$. Though, we have already seen that the variance of $\widehat{\boldsymbol{\beta}}$ is given by

$$\operatorname{Var}\left(\widehat{\boldsymbol{\beta}}\right) = \left(\boldsymbol{X}^\top\boldsymbol{V}^{-1}\boldsymbol{X}\right)^{-1}.$$

Depending on the size of the data it might be challenging or even impossible to compute this variance.

On the other hand, considering the Bayesian backfitting algorithm, the question about uncertainty is eliminated because there we sample from a distribution.

3.3 Simulations

Since the Bayesian backfitting algorithm for linear mixed models is something we have never seen before, we want to explore how accurate it is and what the differences between frequentist and Bayesian backfitting are. On this account, consider the linear mixed model

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon},$$

where

- $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\top \in \mathbb{R}^n$, where n is the sample size,
- $\boldsymbol{X} \in \mathbb{R}^{n\times 7}$ containing the predictors,
- $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_6)^\top \in \mathbb{R}^7$,

- $\boldsymbol{Z} \in \mathbb{R}^{n \times q}$,
- $\boldsymbol{u} = (u_1, \ldots, u_q)^\top \in \mathbb{R}^q$.

3.3.1 Simulations with Known Variances

First, consider the case where the covariance matrices $\boldsymbol{\Sigma}_u$ and $\sigma^2 \boldsymbol{I}_n$ of $\boldsymbol{u}$ and $\boldsymbol{\varepsilon}$ respectively are known. Therefore, these quantities do not have to be estimated within the backfitting algorithm. We impose

- $\boldsymbol{\Sigma}_u = \sigma_u^2 \boldsymbol{I}_q$, where $\sigma_u^2 = 4$ and
- $\sigma^2 = 0.7$.

The simulations are executed using R-Code A.2 for the frequentist and R-Code A.3 for the Bayesian backfitting algorithm. We vary the sample size n, the number of levels of the random effect q and the trend $\boldsymbol{X}\boldsymbol{\beta}$. Table 3.1 shows the different cases which we are considering. In Cases 1 and 2 we look at a 20 level random effect with sample sizes 400 and 1'000. Cases 3 and 4 consider a more extreme trend $\boldsymbol{X}\boldsymbol{\beta}$ compared to Cases 1 and 2. Finally, Cases 5 and 6 look at a random effect with only 4 levels.

Table 3.1: Different settings for simulations with known variances.

Case	n	q	$\boldsymbol{\beta}^\top$
3.1	400	20	$(0.7, -1.2, 0.5, 3.1, -2.4, 1.1, 0.8)$
3.2	1'000	20	$(0.7, -1.2, 0.5, 3.1, -2.4, 1.1, 0.8)$
3.3	400	20	$(12.0, -7.5, 4.1, 18.9, -10.0, 3.5, 1.1)$
3.4	1'000	20	$(12.0, -7.5, 4.1, 18.9, -10.0, 3.5, 1.1)$
3.5	400	4	$(0.7, -1.2, 0.5, 3.1, -2.4, 1.1, 0.8)$
3.6	1'000	4	$(0.7, -1.2, 0.5, 3.1, -2.4, 1.1, 0.8)$

The procedure is the same in every case. We first choose some ϵ for the convergence criterion and fit $\boldsymbol{\beta}$ and $\boldsymbol{u}$ with the frequentist backfitting algorithm. Second, we draw 250 posterior realizations of $\boldsymbol{\beta}$ and $\boldsymbol{u}$ using the Bayesian backfitting algorithm. We consider the first 50 draws as the burn-in period and only deal with the last 200 draws for the inference. Since we work with random variables in the Bayesian setting we run this algorithm twice and check how robust it is. From the 200 posterior realizations of the Bayesian backfitting algorithm we take the mean, i.e. the posterior mean, and compare it with the true values and the estimated values from the frequentist backfitting algorithm. Additionally to the posterior means, the 2.5% and 97.5% quantiles are shown in small. The displayed numbers are always rounded up to three digits.

Case 3.1. $n = 400$, $q = 20$ and $\boldsymbol{\beta} = (0.7, -1.2, 0.5, 3.1, -2.4, 1.1, 0.8)^\top$
In this case, the matrix $\boldsymbol{Z} \in \mathbb{R}^{400 \times 20}$ looks as follows:

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{I}_{20} \\ \vdots \\ \boldsymbol{I}_{20} \end{pmatrix},$$

where $\boldsymbol{I}_{20}$ denotes the 20×20 identity matrix. Since we have a 20 level random effect $\boldsymbol{u}$ and $n = 400$, we have in total 20 observation per level.

The frequentist backfitting algorithm converges after 5 iterations with $\epsilon = 10^{-10}$. The Bayesian backfitting algorithm takes approximately 12 seconds to run once.

The resulting estimates and posterior means for $\boldsymbol{\beta}$ are shown in Table 3.2.

Table 3.2: Estimates of $\boldsymbol{\beta}$ in Case 3.1.

	true	`lm()`	freq.	Bayesian (1st)	Bayesian (2nd)
β_0	0.7	1.115	1.100	1.279 (0.812, 1.548)	1.035 (0.324, 1.588)
β_1	−1.2	−1.176	−1.179	−1.178 (−1.270, −1.092)	−1.183 (−1.271, −1.093)
β_2	0.5	0.490	0.544	0.546 (0.461, 0.628)	0.545 (0.461, 0.632)
β_3	3.1	3.101	3.016	3.015 (2.917, 3.099)	3.011 (2.921, 3.090)
β_4	−2.4	−2.492	−2.472	−2.475 (−2.560, −2.393)	−2.472 (−2.555, −2.388)
β_5	1.1	1.206	1.024	1.017 (0.931, 1.102)	1.023 (0.945, 1.112)
β_6	0.8	0.870	0.809	0.813 (0.723, 0.896)	0.811 (0.720, 0.890)

We observe that the estimates of $\boldsymbol{\beta}$ from both backfitting algorithms are very precise besides the estimate for the intercept β_0. There we see a slight deviation of roughly 0.3 to 0.4. We also see that the estimates from the frequentist backfitting algorithm and the posterior means from the Bayesian backfitting algorithm are quite close to each other. Also, there is only a slight difference in the posterior means from the two runs of the Bayesian backfitting algorithm. However, also a standard `lm()` fit results in a similar pattern.

If we look closer at the posterior realizations for β_0 we observe in the first run of the Bayesian backfitting algorithm values between 0.800 and 1.691 and in the second run between 0.275 and 1.632. See Figure 3.1 for histogram plots of the posterior realizations of β_0.

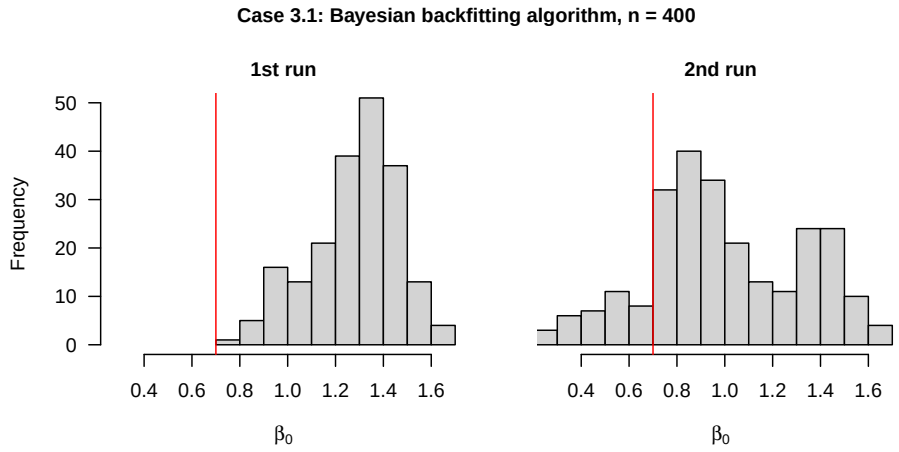

Figure 3.1: Histograms of 200 posterior realizations of β_0 from the Bayesian backfitting algorithm in a linear mixed model with sample size $n = 400$. The red vertical lines mark the true values.

However, the ranges for the posterior realizations of $\beta_1, \ldots, \beta_6$ are much smaller than for β_0. For example for β_1 we observe values between -1.282 and -1.070 in the first run of the Bayesian backfitting algorithm. See Figure 3.2 for the histograms of $\beta_1, \ldots, \beta_6$ from the first run of the Bayesian backfitting algorithm.

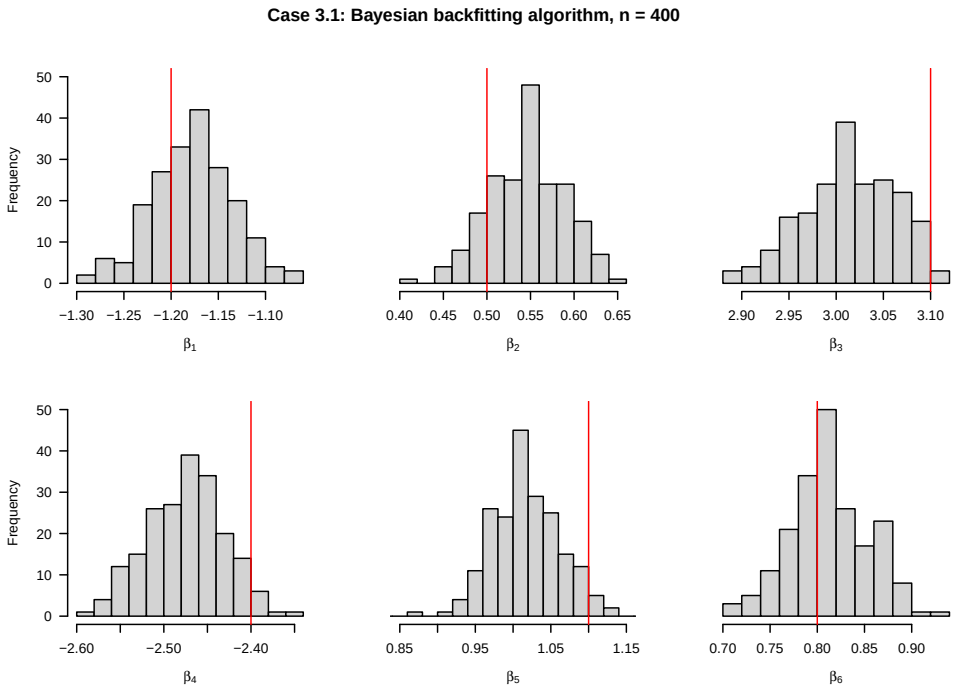

Figure 3.2: Histograms of 200 posterior realizations of $\beta_1, \ldots, \beta_6$ from the first run of the Bayesian backfitting algorithm in a linear mixed model with sample size $n = 400$. The red vertical lines mark the true values.

We focus on the 20 level random effect $\boldsymbol{u}$. Since it is overwhelming to compare 20 different values we only look at the first four elements of the predicted random effects, i.e. u_1, u_2, u_3, u_4. However, the pattern is similar for the remaining u_j's.

Table 3.3: Predictions of $\boldsymbol{u}$ in Case 3.1.

	true	freq.	Bayesian (1st)	Bayesian (2nd)
u_1	0.882	0.553	0.371 (−0.129, 1.006)	0.593 (−0.028, 1.325)
u_2	1.402	1.028	0.840 (0.302, 1.424)	1.060 (0.369, 1.786)
u_3	−1.625	−1.841	−2.032 (−2.540, −1.450)	−1.779 (−2.415, −1.012)
u_4	−0.091	−0.225	−0.394 (−0.857, 0.243)	−0.127 (−0.816, 0.586)
⋮	⋮	⋮	⋮	⋮

We obtain that the frequentist prediction of u_1, u_2, u_3, u_4 is not as close to the true values as we would like to have it. We see an absolute difference of around 0.2 to 0.4. However, the posterior means from the first run of the Bayesian backfitting algorithm are even more off compared to the frequentist fit. On the other hand, the posterior means from the second run are close to the frequentist predictions.

In this setting, we do not gain much if we choose the Bayesian backfitting algorithm in favor of the frequentist backfitting algorithm.

We can also analyze if an increased number of posterior draws does make a significant change. We keep everything unchanged but we modify the burn-in period to 250 and the number of draws afterwards to 1'000.

First, we look at the posterior means for $\boldsymbol{\beta}$ (Table 3.4 shows the estimates). Note that the frequentist estimates are exactly the same as in Table 3.2.

Table 3.4: Estimates of $\boldsymbol{\beta}$ in Case 3.1 with 1'000 posterior realizations.

	true	`lm()`	freq.	Bayesian (1st)	Bayesian (2nd)
β_0	0.7	1.115	1.100	0.993 (0.390, 1.644)	1.031 (0.447, 1.690)
β_1	−1.2	−1.176	−1.179	−1.181 (−1.270, −1.094)	−1.179 (−1.274, −1.092)
β_2	0.5	0.490	0.544	0.545 (0.456, 0.633)	0.545 (0.450, 0.636)
β_3	3.1	3.101	3.016	3.012 (2.925, 3.099)	3.017 (2.925, 3.112)
β_4	−2.4	−2.492	−2.472	−2.474 (−2.554, −2.392)	−2.472 (−2.553, −2.390)
β_5	1.1	1.206	1.024	1.025 (0.940, 1.115)	1.023 (0.938, 1.112)
β_6	0.8	0.870	0.809	0.809 (0.719, 0.896)	0.812 (0.726, 0.898)

We obtain a similar behaviour as for only 250 posterior draws. The posterior means for β_0 are again not as accurate as the other estimates. Figure 3.3 shows the trace plots for β_0 for both runs of the Bayesian backfitting algorithm. These trace plots do not look optimal, it seems that the posterior sampling procedure has not converged yet. However, if we look at the trace plots for β_3 for example (see Figure 3.4) then we see a much better picture. The trace plots for the other parameters β_i, $i = 1, 2, 4, 5, 6$ have the same style as the ones for β_3.

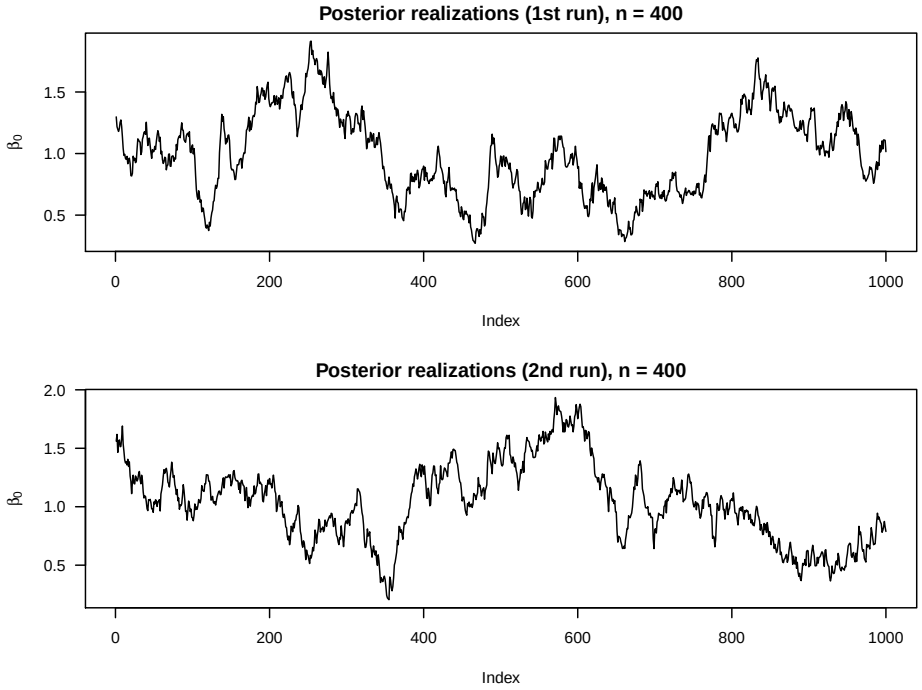

Figure 3.3: Trace plots of 1'000 posterior realizations for β_0 (after burn-in) from the the Bayesian backfitting algorithm in a linear mixed model with sample size $n = 400$.

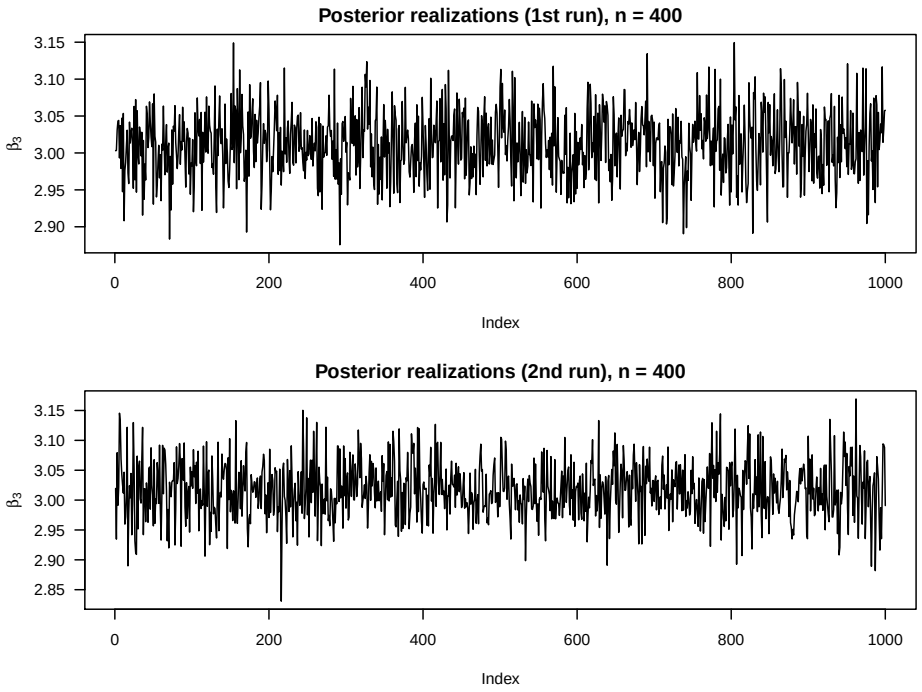

Figure 3.4: Trace plots of 1'000 posterior realizations for β_3 (after burn-in) from the the Bayesian backfitting algorithm in a linear mixed model with sample size $n = 400$.

For the random effect $\boldsymbol{u}$ we obtain a similar pattern. For readability reasons, we only show the first four elements of $\boldsymbol{u}$, i.e. u_1, u_2, u_3, u_4. Note that the frequentist predictions are the same as in Table 3.3 since we have only changed the number of posterior realizations.

Table 3.5: Predictions of $\boldsymbol{u}$ in Case 3.1 with 1'000 posterior realizations.

	true	freq.	Bayesian (1st)	Bayesian (2nd)
u_1	0.882	0.553	0.657 (−0.117, 1.341)	0.622 (−0.119, 1.339)
u_2	1.402	1.028	1.118 (0.333, 1.787)	1.107 (0.368, 1.779)
u_3	−1.625	−1.841	−1.723 (−2.502, −1.028)	−1.773 (−2.554 −1.081)
u_4	−0.091	−0.225	−0.121 (−0.878, 0.570)	−0.150 (−0.903, 0.492)
⋮	⋮	⋮	⋮	⋮

Compared to the predictions from the frequentist backfitting algorithm, the posterior means from both runs of the Bayesian backfitting algorithm are a little bit closer to the true values as before when only having 200 posterior realizations. However, if we look at the trace plots for u_1 for instance we do not obtain a perfect shape (see Figure 3.5). We see that there is some meandering over the indices compared to for example Figure 3.4. The trace plots for the other parameters u_j, $j \neq 1$, appear similar as the ones for u_1 and are therefore not shown.

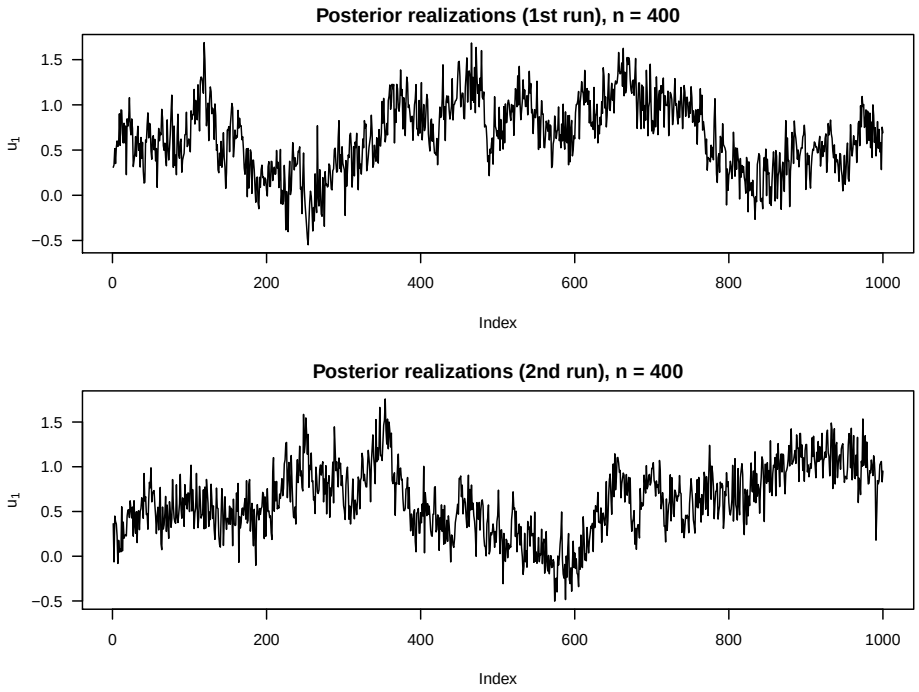

Figure 3.5: Trace plots of 1'000 posterior realizations for u_1 (after burn-in) from the Bayesian backfitting algorithm in a linear mixed model with sample size $n = 400$.

Case 3.2. $n = 1'000$, $q = 20$ and $\boldsymbol{\beta} = (0.7, -1.2, 0.5, 3.1, -2.4, 1.1, 0.8)^\top$
We want to know if the estimates get better if the sample size n is increased. The matrix $\boldsymbol{Z}$ has the same form as in Case 3.1 despite the fact that it is now 1000×20 dimensional. Therefore, we have 50 observations per level of the random effect.

In this setting, the frequentist backfitting algorithm converges after 5 iterations with $\epsilon = 10^{-10}$. The Bayesian backfitting algorithm takes approximately 2.5 minutes to run once.

The resulting estimates and posterior means for $\boldsymbol{\beta}$ are shown in Table 3.6.

Table 3.6: Estimates of $\boldsymbol{\beta}$ in Case 3.2.

	true	`lm()`	freq.	Bayesian (1st)	Bayesian (2nd)
β_0	0.7	1.187	1.185	0.903 (0.559, 1.171)	1.144 (0.910, 1.444)
β_1	−1.2	−1.175	−1.190	−1.193 (−1.247, −1.136)	−1.192 (−1.240, −1.138)
β_2	0.5	0.414	0.500	0.498 (0.441, 0.544)	0.500 (0.448, 0.557)
β_3	3.1	3.041	3.072	3.072 (3.020, 3.129)	3.071 (3.019, 3.115)
β_4	−2.4	−2.457	−2.396	−2.396 (−2.456, −2.340)	−2.395 (−2.443, −2.345)
β_5	1.1	1.182	1.100	1.102 (1.053, 1.152)	1.098 (1.043, 1.151)
β_6	0.8	0.811	0.817	0.819 (0.772, 0.869)	0.818 (0.760, 0.874)

Similar as in Table 3.2 of Case 3.1, we obtain good estimates for $\beta_1, \ldots, \beta_6$ in both the frequentist and the Bayesian backfitting algorithm. Also, the frequentist estimates and the posterior means are quite close to each other for $\beta_1, \ldots, \beta_6$. However, the estimation of β_0 has not improved compared to Case 3.1 with only 400 observations. Also with the `lm()` function we do not get a better estimation of β_0.

Figure 3.6 shows the histograms of the posterior realizations of $\beta_1, \ldots, \beta_6$ from the first run of the Bayesian backfitting algorithm. The histograms have a similar pattern as the ones in Figure 3.2 of Case 3.1.

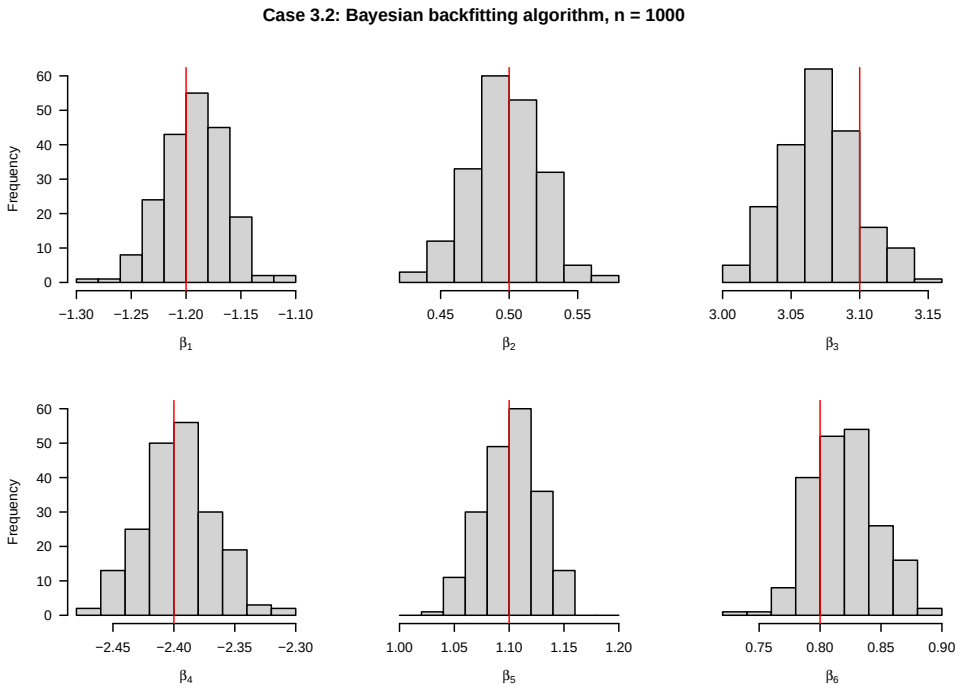

Figure 3.6: Histograms of 200 posterior realizations of $\beta_1, \ldots, \beta_6$ from the first run of the Bayesian backfitting algorithm in a linear mixed model with sample size $n = 1'000$. The red vertical lines mark the true values.

We also take a look at the first four entries of the predictions of the random effect $\boldsymbol{u}$.

Table 3.7: Predictions of $\boldsymbol{u}$ in Case 3.2.

	true	freq.	Bayesian (1st)	Bayesian (2nd)
u_1	2.764	2.049	2.327 (1.907, 2.789)	2.107 (1.699, 2.418)
u_2	−0.766	−1.317	−1.053 (−1.439, −0.598)	−1.293 (−1.668, −0.958)
u_3	−2.435	−3.077	−2.791 (−3.217, −2.338)	−3.023 (−3.404, −2.677)
u_4	2.876	2.393	2.676 (2.279, 3.092)	2.436 (2.048, 2.796)
⋮	⋮	⋮	⋮	⋮

Here, we obtain a better approximation of the true values of $\boldsymbol{u}$ with the Bayesian backfitting algorithm. However, the posterior means from the second run are almost the same as the predictions from the frequentist backfitting algorithm.

Case 3.3. $n = 400$, $q = 20$ and $\boldsymbol{\beta} = (12.0, -7.5, 4.1, 18.9, -10.0, 3.5, 1.1)^\top$
In constrast to Cases 3.1 and 3.2 we consider a stronger trend $\boldsymbol{X\beta}$, i.e. the values in $\boldsymbol{\beta}$ are more extreme. Nevertheless, the matrix $\boldsymbol{Z}$ has again the same form as in Case 3.1.

The frequentist backfitting algorithm converges after 4 iterations with $\epsilon = 10^{-10}$. The Bayesian backfitting algorithm takes approximately 12 seconds to run once.

The estimates of $\boldsymbol{\beta}$ are displayed in Table 3.8.

Table 3.8: Estimates of $\boldsymbol{\beta}$ in Case 3.3.

	true	`lm()`	freq.	Bayesian (1st)	Bayesian (2nd)
β_0	12.0	12.415	12.400	12.579 (12.111, 12.848)	12.335 (11.624, 12.888)
β_1	−7.5	−7.476	−7.479	−7.478 (−7.570, −7.392)	−7.483 (−7.571, −7.393)
β_2	4.1	4.090	4.144	4.146 (4.061, 4.228)	4.145 (4.061, 4.232)
β_3	18.9	18.901	18.816	18.815 (18.717, 18.899)	18.811 (18.721, 18.890)
β_4	−10.0	−10.092	−10.072	−10.075 (−10.160, −9.993)	−10.072 (−10.155, −9.988)
β_5	3.5	3.606	3.424	3.417 (3.331, 3.502)	3.423 (3.345, 3.512)
β_6	1.1	1.170	1.109	1.113 (1.023, 1.196)	1.111 (1.020, 1.190)

Similar as in Table 3.2 of Case 3.1, we obtain a good fit for the parameters $\beta_1, \dots, \beta_6$. We still have the problem with the intercept β_0. There we have an absolute difference of around 0.5 between the true and the estimated values. Nevertheless, we can already say that a stronger trend does not improve or disimprove the algorithm when focusing on $\boldsymbol{\beta}$.

The pattern for the 20 level random effect $\boldsymbol{u}$ is similar as in Case 3.1 and is therefore omitted.

Case 3.4. $n = 1'000$, $q = 20$ and $\boldsymbol{\beta} = (12.0, -7.5, 4.1, 18.9, -10.0, 3.5, 1.1)^\top$
We increase the sample size compared to Case 3.3 and keep the remaining parameters unchanged. The matrix $\boldsymbol{Z}$ has the same form as in Case 3.2.

The frequentist backfitting algorithm converges again after 4 iterations with $\epsilon = 10^{-10}$. The Bayesian backfitting algorithm takes approximately 2.5 minutes to run once.

The estimates of $\boldsymbol{\beta}$ are the following:

Table 3.9: Estimates of $\boldsymbol{\beta}$ in Case 3.4.

	true	`lm()`	freq.	Bayesian (1st)	Bayesian (2nd)
β_0	12.0	12.487	12.485	12.203 (11.859, 12.471)	12.444 (12.210, 12.744)
β_1	−7.5	−7.475	−7.490	−7.493 (−7.547, −7.436)	−7.492 (−7.540, −7.438)
β_2	4.1	4.014	4.100	4.098 (4.041, 4.144)	4.100 (4.048, 4.157)
β_3	18.9	18.841	18.872	18.872 (18.820, 18.929)	18.871 (18.819, 18.915)
β_4	−10.0	−10.057	−9.996	−9.996 (−10.056, −9.940)	−9.995 (−10.043, −9.945)
β_5	3.5	3.582	3.500	3.502 (3.453, 3.552)	3.498 (3.443, 3.551)
β_6	1.1	1.111	1.117	1.119 (1.072, 1.169)	1.118 (1.060, 1.174)

This estimation looks very similar as in Table 3.8 of Case 3.3.

The prediction of the random effect $\boldsymbol{u}$ is also comparable to Case 3.2 and is therefore omitted.

Case 3.5. $n = 400$, $q = 4$ and $\boldsymbol{\beta} = (0.7, -1.2, 0.5, 3.1, -2.4, 1.1, 0.8)^\top$
Next, we consider a random effect with only four levels, i.e. $q = 4$. Therefore, the matrix $\boldsymbol{Z} \in \mathbb{R}^{400 \times 4}$ takes the form

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{I}_4 \\ \vdots \\ \boldsymbol{I}_4 \end{pmatrix},$$

where $\boldsymbol{I}_4$ denotes the 4×4 identity matrix. This means that for each level of the random effect $\boldsymbol{u}$ we have 100 observations.

Unsurprisingly, the estimates for $\boldsymbol{\beta}$ from the frequentist and the Bayesian backfitting algorithm are almost identical to the ones from Case 3.1 and are therefore omitted. Let us take a closer look at the behaviour of the random effect $\boldsymbol{u}$.

Table 3.10: Predictions of $\boldsymbol{u}$ in Case 3.5.

	true	freq.	Bayesian (1st)	Bayesian (2nd)
u_1	0.882	0.584	0.335 (0.005, 0.920)	0.836 (0.266, 1.577)
u_2	1.402	1.280	1.033 (0.702, 1.634)	1.531 (0.926, 2.236)
u_3	−1.625	−1.670	−1.910 (−2.255, −1.328)	−1.406 (−1.977, −0.700)
u_4	−0.091	−0.194	−0.443 (−0.780, 0.139)	0.079 (−0.508, 0.746)

Here, we observe that the posterior means in the first run of the Bayesian backfitting algorithm are not as accurate as the frequentist values. However, in the second run two of the four posterior means are closer to the true values than the frequentist values. So, in this scenario we cannot conclude if the Bayesian backfitting algorithm performs better than the frequentist.

Figure 3.7 shows the histograms of the posterior realizations of u_1, u_2, u_3, u_4 from the second run of the Bayesian backfitting algorithm. There we actually see a good scatter around the true values.

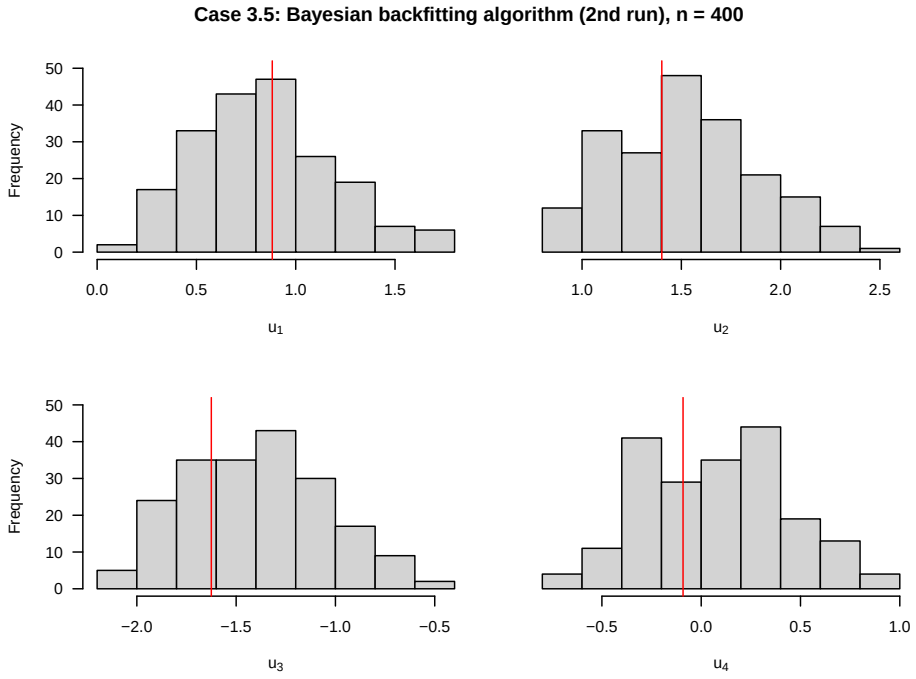

Figure 3.7: Histograms of 200 posterior realizations of u_1, u_2, u_3, u_4 from the second run of the Bayesian backfitting algorithm in a linear mixed model with sample size $n = 400$. The red vertical lines mark the true values.

Case 3.6. $n = 1'000$, $q = 4$ and $\boldsymbol{\beta} = (0.7, -1.2, 0.5, 3.1, -2.4, 1.1, 0.8)^\top$
Compared to Case 3.5 we increase the sample size n. The remaining quantities stay the same. Also the matrix $\boldsymbol{Z}$ has the same form as in Case 3.5, however, now $\boldsymbol{Z} \in \mathbb{R}^{1000 \times 4}$. Thus, we have 250 observations per level of the random effect.

We only focus on the behaviour of the random effects because the estimates for $\boldsymbol{\beta}$ are similar as in Case 3.2. (Note that we have different values for $\boldsymbol{u}$ than in Case 3.5.)

Table 3.11: Predictions of $\boldsymbol{u}$ in Case 3.6.

	true	freq.	Bayesian (1st)	Bayesian (2nd)
u_1	2.764	2.130	2.476 (2.111, 2.936)	2.174 (1.856, 2.427)
u_2	−0.766	−1.355	−1.012 (−1.347, −0.558)	−1.312 (−1.624, −1.060)
u_3	−2.435	−3.153	−2.803 (−3.153, −2.360)	−3.098 (−3.409, −2.844)
u_4	2.873	2.379	2.731 (2.368, 3.146)	2.425 (2.120, 2.687)

In both runs of the Bayesian backfitting algorithm we obtain posterior means that are (slightly) better then the frequentist predictions compared to the true values.

3.3.2 Simulations with Unknown Variances

Consider the same model as before. However, suppose now that the variance parameters σ_u^2 for the random effect and σ^2 for the error term are unknown and have to be estimated within the backfitting algorithm. For these simulations we exercise R-Code A.4 for the frequentist and R-Code A.5 for the Bayesian backfitting algorithm.

Case 3.7. $n = 400$, $q = 20$ and $\boldsymbol{\beta} = (0.7, -1.2, 0.5, 3.1, -2.4, 1.1, 0.8)^\top$
Compared to the scenario where the variances are known, the frequentist backfitting algorithm takes much longer to converge since the variances also have to be estimated in every iteration. In this case it needs 422 iterations to converge with $\epsilon = 10^{-8}$.

For the Bayesian backfitting algorithm we choose again a burn-in period of 50 iterations and draw 200 posterior realizations of $\boldsymbol{\beta}$ and $\boldsymbol{u}$ afterwards. One run takes approximately 13 seconds.

Additionally, we compare our estimates with a fit using the `lmer()` function from the R package `lme4` (Bates *et al.*, 2015). Table 3.12 shows the estimates of $\boldsymbol{\beta}$ in this case.

Table 3.12: Estimates of $\boldsymbol{\beta}$ in Case 3.7.

	true	`lmer()`	freq.	Bayesian (1st)	Bayesian (2nd)
β_0	0.7	1.100	1.078	1.072 (0.351, 1.376)	0.731 (−0.071, 1.437)
β_1	−1.2	−1.179	−1.179	−1.178 (−1.271, −1.092)	−1.183 (−1.273, −1.092)
β_2	0.5	0.544	0.544	0.546 (0.459, 0.628)	0.545 (0.461, 0.634)
β_3	3.1	3.016	3.016	3.015 (2.917, 3.101)	3.011 (2.920, 3.092)
β_4	−2.4	−2.472	−2.472	−2.475 (−2.561, −2.391)	−2.472 (−2.556, −2.386)
β_5	1.1	1.024	1.024	1.017 (0.929, 1.104)	1.023 (0.943, 1.113)
β_6	0.8	0.809	0.809	0.813 (0.723, 0.897)	0.811 (0.713, 0.895)

We observe that the frequentist estimates and the posterior means from the Bayesian backfitting

algorithm for $\beta_1, \ldots, \beta_6$ are quite close to the true values. However, similar to the cases where the variances are known, the estimates and posterior means for the intercept β_0 are a little bit off. Note that the frequentist estimates are almost the same as the estimates from the `lmer()` function.

If we focus on the 20 level random effect $\boldsymbol{u}$ we obtain the following predictions (we only show the first four elements, i.e. u_1, u_2, u_3, u_4):

Table 3.13: Predictions of $\boldsymbol{u}$ in Case 3.7.

	true	`lmer()`	freq.	Bayesian (1st)	Bayesian (2nd)
u_1	0.882	0.553	0.575	0.576 (0.084, 1.475)	0.895 (0.117, 1.826)
u_2	1.402	1.028	1.049	1.044 (0.526, 1.935)	1.362 (0.524, 2.283)
u_3	−1.625	−1.841	−1.819	−1.827 (−2.354, −1.034)	−1.476 (−2.266, −0.542)
u_4	−0.091	−0.225	−0.203	−0.189 (−0.645, 0.749)	0.175 (−0.657, 1.129)
⋮	⋮	⋮	⋮	⋮	⋮

We see that the posterior means from the first run of the Bayesian backfitting algorithm are almost the same as the frequentist predictions. Though, the posterior means from the second run of the Bayesian backfitting algorithm are closer to the true values. Moreover, the frequentist predictions are very similar as the ones from the `lmer()` fit.

What about the variance estimates? Since we perform a simulation, we actually know the true variances ($\sigma_u^2 = 4$ and $\sigma^2 = 0.7$). The estimated values from the frequentist backfitting algorithm and the posterior means from the Bayesian backfitting algorithm are shown in Table 3.14.

Table 3.14: Estimates of σ_u^2 and σ^2 in Case 3.7.

	true	`lmer()`	freq.	Bayesian (1st)	Bayesian (2nd)
σ_u^2	4.0	3.995	3.761	3.879 (3.458, 4.505)	4.124 (3.563, 5.227)
σ^2	0.7	0.698	0.665	0.712 (0.687, 0.748)	0.712 (0.692, 0.746)

In the first run of the Bayesian backfitting algorithm we observe values for σ^2 between 0.683 and 0.751 and in the second run between 0.689 and 0.761 (after the burn-in period). For σ_u^2 we observe values between 3.380 and 4.657 and between 3.445 and 5.337 for the first and the second run respectively (see Figure 3.8 for the histograms of σ^2 and σ_u^2). We also see that the estimated variances from the `lmer()` fit are very accurate and closer to the true values than the values from the frequentist and Bayesian backfitting algorithms. Nonetheless, we have to keep in mind that estimating a variance needs a large number of observations. Therefore, to get better estimates we need to further increase n.

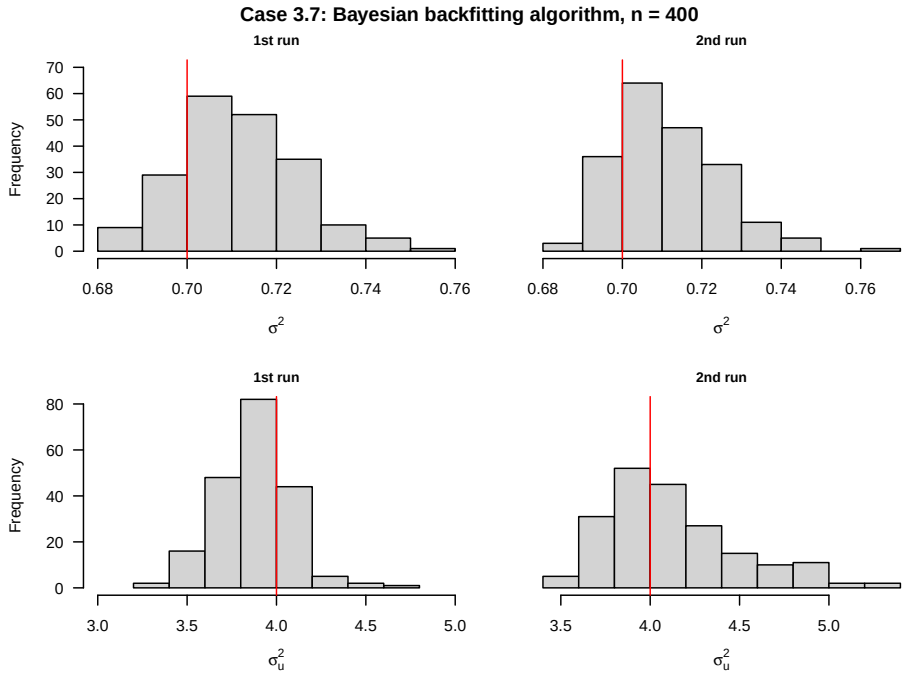

Figure 3.8: Histograms of 200 posterior realizations of σ^2 and σ_u^2 from the Bayesian backfitting algorithm in a linear mixed model with sample size $n = 400$. The red vertical lines mark the true values.

Case 3.8. $n = 1'000$, $q = 20$ and $\boldsymbol{\beta} = (0.7, -1.2, 0.5, 3.1, -2.4, 1.1, 0.8)^\top$
Let us increase the sample size and keep the remaining parameters the same. In this setting, the frequentist backfitting algorithm converges after 817 iteration with $\epsilon = 10^{-8}$. The Bayesian backfitting algorithm needs approximately 2.5 minutes to run once.

The behaviour of the estimates for $\boldsymbol{\beta}$ is similar as in Case 3.7 where $n = 400$. The same holds for the random effect $\boldsymbol{u}$ with a slight preference of the Bayesian backfitting algorithm for more precise predictions.

However, we want to know if the increased sample size improves the estimates of the variances. We obtain the following:

Table 3.15: Estimates of σ_u^2 and σ^2 in Case 3.8.

	true	`lmer()`	freq.	Bayesian (1st)	Bayesian (2nd)
σ_u^2	4.0	4.057	3.843	4.685 (4.215, 5.235)	4.300 (3.768, 4.815)
σ^2	0.7	0.705	0.692	0.711 (0.701, 0.722)	0.710 (0.701, 0.721)

Observe that the frequentist estimates of the variances are closer to the true values than in Case 3.7. This is a consequence of the larger sample size. The range of the realizations of σ^2 is very small and similar in both runs of the Bayesian backfitting algorithm. However, for σ_u^2 we obtain values between 4.188 and 5.353 and between 3.723 and 4.867 in the first and the second run resp. There is quite a difference between these ranges over the two runs.

As for the scenario where the variances are known, we could investigate more cases here. However,

changing the number of levels of the random effect $\boldsymbol{u}$ or increasing the trend $\boldsymbol{X\beta}$ has no major impact on the estimates of the variances. The estimates for $\boldsymbol{\beta}$ and $\boldsymbol{u}$ are similar as with known variances. For this reason, we will not show any other cases.

3.3.3 Conclusion

Both in the case with known and unknown variance parameters we see more or less the same pattern. The estimation of the parameters $\beta_1, \ldots, \beta_6$ is very precise whereas for β_0 we do not come as close to the true value as for the other parameters. However, we still get a reasonable estimate for β_0.

The prediction of the random effect tends to be slightly better with the Bayesian backfitting algorithm compared to the frequentist backfitting algorithm. And if we increase the number of posterior draws this will become even better. Further, we can also apply "thinning" (i.e. discard all but every k-th sampled value) to remove correlations between individual draws (see also Link and Eaton, 2012).

Compared to the `lmer()` fit the frequentist backfitting algorithm performs almost the same. This is already a positive sign because we know that the algorithm works and behaves quite well.

Nevertheless, we need to have in mind that the frequentist backfitting algorithm is in general very fast. Depending on how many posterior realizations we want to draw, the Bayesian backfitting algorithm could take some time to run. Surely, with a larger sample size this time increases as well.

Based on the simulations before, using the frequentist backfitting algorithm is highly recommended for every situation where estimates and predictions need to be accessible fast. If one aims for better estimates (especially for the random effect) then the Bayesian backfitting algorithm might be preferred.

Chapter 4

Backfitting for Spatial Models

In this chapter we take a closer look on the backfitting algorithm for spatial models. However, before doing this we introduce some definitions and notations about spatial statistics based on Furrer (2019) and Cressie (1993).

4.1 Introduction to Spatial Statistics

Definition 4.1. The stochastic process (or random field)

$$Z(\cdot) = \{Z(\boldsymbol{s}) : \boldsymbol{s} \in \mathcal{D} \subset \mathbb{R}^d, d \geq 1\}$$

is a spatial process. The set $\mathcal{D}$ is called the spatial domain and can be seen as an index set. It can be finite, discrete or continuous.

Remark 4.1.

1. In most applications, $\mathcal{D}$ has a bounded d-dimensional volume.
2. In geostatistics, d is almost always one, two or three.
3. In geostatistics, the possible values of the random function $Z(\cdot)$ are often in $\mathbb{R}$, $\mathbb{R}_+$, $[0, 1]$, or they are discrete.

The random process $Z(\cdot)$ is often defined by means of the joint distribution of an arbitrary, finite set of points $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m\}$, namely

$$F_{\boldsymbol{s}_1,\ldots,\boldsymbol{s}_m}(v_1, \ldots, v_m) = \mathbb{P}(Z(\boldsymbol{s}_1) \leq v_1, \ldots, Z(\boldsymbol{s}_m) \leq v_m), \tag{4.1}$$

satisfying the Kolmogorov's conditions of symmetry and consistency. These conditions basically impose that

- $F_{\cdot}(\cdot)$ is invariant under simultaneous permutation of $\boldsymbol{s}_i$ and v_i, i.e. let τ be a permutation of $\{1, \ldots, m\}$ then $F_{\boldsymbol{s}_1,\ldots,\boldsymbol{s}_m}(v_1, \ldots, v_m) = F_{\boldsymbol{s}_{\tau(1)},\ldots,\boldsymbol{s}_{\tau(m)}}\left(v_{\tau(1)}, \ldots, v_{\tau(m)}\right)$ and
- $F_{\boldsymbol{s}_1,\ldots,\boldsymbol{s}_m}(v_1, \ldots, v_l, \infty, \ldots, \infty) = F_{\boldsymbol{s}_1,\ldots,\boldsymbol{s}_l}(v_1, \ldots, v_l)$ for $l < m$.

If the joint distribution in (4.1) is a multivariate Gaussian distribution we will call $Z(\cdot)$ a Gaussian process or a Gaussian random field (GRF).

Definition 4.2. A realization of the random function $Z(\cdot)$, denoted by $z(\cdot)$, is called the sample surface or the sample path. In general, we actually observe a set of the following form

$$\{z(\boldsymbol{s}_i) : \boldsymbol{s}_1, \ldots, \boldsymbol{s}_n \in \mathcal{D}\}$$

denoting data measured at the locations $\boldsymbol{s}_1, ..., \boldsymbol{s}_n$ in a domain $\mathcal{D} \subset \mathbb{R}^d$, $d \geq 1$. The sample can be seen as a subset of a potentially infinite number of measurements $\{z(\boldsymbol{s}) : \boldsymbol{s} \in \mathcal{D}\}$.

To infer the statistical process from a single set of observations $\{z(\boldsymbol{s}_i) : \boldsymbol{s}_1, \ldots, \boldsymbol{s}_n \in \mathcal{D}\}$, we need some restrictions on the process $Z(\cdot)$.

Definition 4.3. A spatial process $Z(\cdot)$ is a strong stationary process if the equation

$$F_{\boldsymbol{s}_1,\ldots,\boldsymbol{s}_m}(v_1, \ldots, v_m) = F_{\boldsymbol{s}_1+\boldsymbol{h},\ldots,\boldsymbol{s}_m+\boldsymbol{h}}(v_1, \ldots, v_m)$$

holds for all m, $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m$, $v_1, \ldots, v_m$ and all $\boldsymbol{h}$.

Definition 4.4. A spatial process $Z(\cdot)$ is a second-order stationary process (or a weak stationary process) if the following two moments exist and satisfy

$$\mathbb{E}[Z(\boldsymbol{s})] \equiv \mu \text{ for all } \boldsymbol{s} \in \mathcal{D},$$
$$\text{Cov}(Z(\boldsymbol{s}_1), Z(\boldsymbol{s}_2)) = c(\boldsymbol{s}_1 - \boldsymbol{s}_2) \text{ for all } \boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathcal{D}.$$

The function $c(\cdot)$ is called the covariance function.

Definition 4.5. If $c(\cdot)$ is a function of the distance $\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|$ only, then the process $Z(\cdot)$ is called isotropic, otherwise it is called anisotropic.

Definition 4.6. The spatial lag $\boldsymbol{\ell}$ is defined as

$$\boldsymbol{\ell} := \boldsymbol{s}_1 - \boldsymbol{s}_2$$

for $\boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathcal{D}$. Moreover, we denote by $\ell = \|\boldsymbol{\ell}\|$.

Throughout this work we only use the Matérn covariance function given by

$$c(\ell; \theta_1, \theta_2, \theta_3, \sigma^2) = \begin{cases} \theta_2 + \sigma^2, & \text{if } \ell = 0 \\ \theta_2 (\ell/\theta_1)^{\theta_3} \mathcal{K}_{\theta_3}(\ell/\theta_1), & \text{otherwise} \end{cases}$$

where $\mathcal{K}_{\theta_3}$ is the modified Bessel function of the second kind of order $\theta_3 > 0$ (Abramowitz and Stegun, 1970).

In the Matérn covariance function the parameter θ_1 corresponds to the range, θ_2 to the partial sill, θ_3 to the smoothness and σ^2 to the nugget effect. The main reason for using the Matérn covariance function is its flexibility. There are also for example the spherical and exponential covariance function. However, these covariance function do not have a smoothness parameter.

R-Code 4.1. Different Matérn covariance functions

```
library(spam)

# Lag vector
l <- seq(0, 0.8, length.out = 100)
```

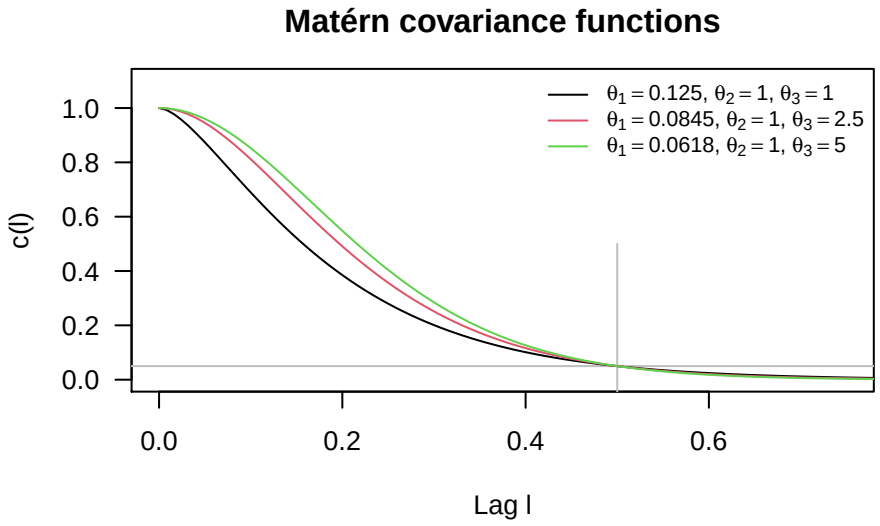

Figure 4.1: Different Matérn covariance functions with nugget effect 0, sill 1 and effective range 0.5 (see R-Code 4.1).

```
plot(0, xlim = c(0, 0.75), ylim = c(0, 1.1),
     main = "Matern covariance functions",
     type = "n", xlab = "Lag l", ylab = "c(l)")
abline(h = 0.05, col = "grey")
segments(0.5, -0.1, 0.5, 0.5, col = "grey")

# theta <- c(range, sill, smoothness, nugget)
lines(l, cov.mat(l, c(0.125, 1, 1, 0)), col = 1)
lines(l, cov.mat(l, c(0.0845, 1, 2.5, 0)), col = 2)
lines(l, cov.mat(l, c(0.0618, 1, 5, 0)), col = 3)
# where we have determined the ranges empirically

legend("topright", bty = "n", col = 1:3, lty = 1, cex = 0.8,
       legend = c(expression(paste(theta[1]==0.125, ", ", theta[2]==1, ", ",
                                   theta[3]==1)),
                  expression(paste(theta[1]==0.0845, ", ", theta[2]==1, ", ",
                                   theta[3]==2.5)),
                  expression(paste(theta[1]==0.0618, ", ", theta[2]==1, ", ",
                                   theta[3]==5)) ) )
```

When applying statistical procedures it is often necessary to decompose the process $Z(\cdot)$ into several parts in order to obtain stationary elements. We propose the following decomposition

$$Z(\boldsymbol{s}) = \mu(\boldsymbol{s}) + Y(\boldsymbol{s}) + \varepsilon(\boldsymbol{s}), \quad \boldsymbol{s} \in \mathcal{D},$$

where

- $\mu(\boldsymbol{s})$ is typically considered as the first moment or drift, i.e. $\mathbb{E}[Z(\boldsymbol{s})] = \mu(\boldsymbol{s})$,

- $Y(\boldsymbol{s})$ is a zero-mean stationary spatial process with continuous covariance function,
- $\varepsilon(\boldsymbol{s})$ is a zero-mean white-noise process, independent of $Z(\cdot)$ and is considered as measurement error. We often assume that $\varepsilon(\boldsymbol{s}) \sim \mathcal{N}(0, \sigma^2)$.

The drift (or also called the trend) $\mu(\cdot)$ can be modeled via a parametric approach. We typically write it in a "regression format", i.e.

$$\mu(\boldsymbol{s}) = \boldsymbol{x}(\boldsymbol{s})^\top \boldsymbol{\beta} = \beta_0 + \beta_1 x_1(\boldsymbol{s}) + \cdots + \beta_p x_p(\boldsymbol{s}).$$

4.2 Frequentist Backfitting for Spatial Models

The notations and concepts in this section are based on Furrer and Sain (2009).

Consider the following simple, additive (mixed-effects) spatial model for locations $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$:

$$Y(\boldsymbol{s}_i) = \boldsymbol{x}(\boldsymbol{s}_i)^\top \boldsymbol{\beta} + h(\boldsymbol{s}_i) + \varepsilon(\boldsymbol{s}_i), \quad i = 1, \ldots, n, \tag{4.2}$$

where $h(\cdot)$ represents a zero-mean spatial process and $\varepsilon(\cdot)$ represents a zero-mean error process independent of $h(\cdot)$.

The model (4.2) can also be written in a matrix-vector form:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{h} + \boldsymbol{\varepsilon}, \tag{4.3}$$

where

- $\boldsymbol{Y} = (Y(\boldsymbol{s}_1), \ldots, Y(\boldsymbol{s}_n))^\top \in \mathbb{R}^n$,
- $\boldsymbol{X} = \begin{pmatrix} 1 & x_1(\boldsymbol{s}_1) & x_2(\boldsymbol{s}_1) & \cdots & x_p(\boldsymbol{s}_1) \\ 1 & x_1(\boldsymbol{s}_2) & x_2(\boldsymbol{s}_2) & \cdots & x_p(\boldsymbol{s}_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1(\boldsymbol{s}_n) & x_2(\boldsymbol{s}_n) & \cdots & x_p(\boldsymbol{s}_n) \end{pmatrix} \in \mathbb{R}^{n \times (p+1)}$,
- $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^\top \in \mathbb{R}^{p+1}$,
- $\boldsymbol{h} = (h(\boldsymbol{s}_1), \ldots, h(\boldsymbol{s}_n))^\top \in \mathbb{R}^n$,
- $\boldsymbol{\varepsilon} = (\varepsilon(\boldsymbol{s}_1), \ldots, \varepsilon(\boldsymbol{s}_n))^\top \in \mathbb{R}^n$.

Therefore,

- $\boldsymbol{X}\boldsymbol{\beta}$ represents the fixed effects,
- $\boldsymbol{h} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{\Sigma}_h)$,
- $\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n)$,
- $\boldsymbol{V} := \mathrm{Var}(\boldsymbol{Y}) = \mathrm{Var}(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{h} + \boldsymbol{\varepsilon}) = \mathrm{Var}(\boldsymbol{h}) + \mathrm{Var}(\boldsymbol{\varepsilon}) = \boldsymbol{\Sigma}_h + \sigma^2 \boldsymbol{I}_n$.

Remark 4.2.

1. The covariance matrix of the spatial process $\boldsymbol{\Sigma}_h = \boldsymbol{\Sigma}_h(\boldsymbol{\theta})$ is based on a continuous covariance function parametrized by $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)^\top$. The parameters in $\boldsymbol{\theta}$ are called the covariance parameters and are related to the range (θ_1), sill (θ_2) and smoothness (θ_3).

2. σ^2 is a global variance related to the measurement error, called the nugget effect.

If we take a closer look at the spatial model (4.3) we observe that we can identify this as a linear mixed model which we have seen in Chapter 3. Recall the linear mixed model

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon}.$$

Therefore, the spatial model $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{h} + \boldsymbol{\varepsilon}$ can be seen as a mixed model with $\boldsymbol{Z} = \boldsymbol{I}_n$ and $\boldsymbol{u} = \boldsymbol{h}$. Hence, we can apply the same results as for mixed models.

Lemma 4.1. If the covariance parameters $\boldsymbol{\theta}$ and σ^2 are known, then estimates of $\boldsymbol{\beta}$ and predictions of the random effects $\boldsymbol{h}$ follow by

$$\begin{aligned}\widehat{\boldsymbol{\beta}} &= (\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{y},\\ \widehat{\boldsymbol{h}} &= \boldsymbol{\Sigma}_h \boldsymbol{V}^{-1} (\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}),\end{aligned}$$

where $\boldsymbol{V} = \mathrm{Var}(\boldsymbol{Y}) = \boldsymbol{\Sigma}_h + \sigma^2 \boldsymbol{I}_n$.

Remark 4.3. We can relate these quantities to $\widehat{\boldsymbol{\beta}}$ from equation (3.2) and $\tilde{\boldsymbol{u}}$ from equation (3.3) knowing that we can identify the spatial model as a linear mixed model. Hence, the derivation of $\widehat{\boldsymbol{\beta}}$ is done via the method of generalized least squares and $\widehat{\boldsymbol{h}}$ is the best linear unbiased predictor (BLUP).

However, the covariance parameters $\boldsymbol{\theta}$ and σ^2 are often unknown and have to be estimated via for instance method of moments. With an appropriate covariance estimate, the estimates of $\boldsymbol{\beta}$ and the predictions of the random effects $\boldsymbol{h}$ follow by

$$\begin{aligned}\widehat{\boldsymbol{\beta}} &= (\boldsymbol{X}^\top \widehat{\boldsymbol{V}}^{-1} \boldsymbol{X})^{-1} \boldsymbol{X}^\top \widehat{\boldsymbol{V}}^{-1} \boldsymbol{y},\\ \widehat{\boldsymbol{h}} &= \widehat{\boldsymbol{\Sigma}}_h \widehat{\boldsymbol{V}}^{-1} (\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}),\end{aligned}$$

where $\widehat{\boldsymbol{V}} = \widehat{\boldsymbol{\Sigma}}_h + \widehat{\sigma}^2 \boldsymbol{I}_n$.

Algorithm 4.1 (Backfitting algorithm for spatial models)**.**

[0] initialize $\widehat{\boldsymbol{h}}^{(0)}$ and set $t = 0$

[1] $t = t + 1$

[2] put $\widehat{\boldsymbol{\beta}}^{(t)} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \left(\boldsymbol{y} - \widehat{\boldsymbol{h}}^{(t-1)}\right)$

[3]
- estimate the covariance parameters $\widehat{\boldsymbol{\theta}}^{(t)}$ and $\left(\widehat{\sigma}^{(t)}\right)^2$ to get $\widehat{\boldsymbol{\Sigma}}_h^{(t)} = \boldsymbol{\Sigma}_h\left(\widehat{\boldsymbol{\theta}}^{(t)}\right)$ and $\widehat{\boldsymbol{V}}^{(t)} = \widehat{\boldsymbol{\Sigma}}_h^{(t)} + \left(\widehat{\sigma}^{(t)}\right)^2 \boldsymbol{I}_n$
- put $\widehat{\boldsymbol{h}}^{(t)} = \widehat{\boldsymbol{\Sigma}}_h^{(t)} \left(\widehat{\boldsymbol{V}}^{(t)}\right)^{-1} \left(\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}^{(t)}\right)$

[4] repeat [1] to [3] until convergence

This algorithm refers to Algorithm 1 in Furrer and Sain (2009).

Remark 4.4.

1. If the covariance parameters $\boldsymbol{\theta}$ and σ^2 are known we can directly use these quantities and do not have to estimate these in step [3].

2. Similar, if $\boldsymbol{\theta}$ and σ^2 are unknown but the estimates $\widehat{\boldsymbol{\theta}}$ and $\widehat{\sigma}^2$ are available before starting the algorithm then these estimates can be used instead of $\widehat{\boldsymbol{\theta}}^{(t)}$ and $\left(\widehat{\sigma}^{(t)}\right)^2$ in step [3].

The various R codes for the frequentist backfitting algorithm are listed in the Appendix (R-Codes A.7 and A.9).

4.3 Bayesian Backfitting for Spatial Models

Again, we want to give an interpretation for the backfitting algorithm in the Bayesian framework. Since we have identified spatial models as a special case of linear mixed models we can therefore use the same techniques as in Section 3.2.

We start by imposing a prior for $\boldsymbol{\beta}$,

$$\boldsymbol{\beta} \sim \mathcal{N}_p(\mathbf{0}, \tau^2 \boldsymbol{I}_p).$$

Then

$$\begin{aligned} \boldsymbol{X}\boldsymbol{\beta} &\sim \mathcal{N}_n\left(\mathbf{0}, \tau^2 \boldsymbol{X}\boldsymbol{X}^\top\right), \\ \boldsymbol{h} &\sim \mathcal{N}_n\left(\mathbf{0}, \boldsymbol{\Sigma}_h\right), \\ \boldsymbol{Y} \mid (\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{h}) &\sim \mathcal{N}_n\left(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{h}, \sigma^2 \boldsymbol{I}_n\right). \end{aligned}$$

The derivation of the posterior distributions of $\boldsymbol{X}\boldsymbol{\beta} \mid (\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{h})$ and $\boldsymbol{h} \mid (\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{X}\boldsymbol{\beta})$ is exactly the same as for mixed models. Therefore,

$$\boldsymbol{X}\boldsymbol{\beta} \mid (\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{h}) \sim \mathcal{N}_n\left(\underbrace{\left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1}\frac{1}{\sigma^2}}_{=:\boldsymbol{S}_\beta(\tau^2)}(\boldsymbol{y} - \boldsymbol{h}), \underbrace{\left(\frac{1}{\sigma^2}\boldsymbol{I}_n + \frac{1}{\tau^2}\left(\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\right)^{-1}}_{=\boldsymbol{S}_\beta(\tau^2)\sigma^2}\right),$$

$$\boldsymbol{h} \mid (\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{X}\boldsymbol{\beta}) \sim \mathcal{N}_n\left(\underbrace{\left(\frac{1}{\sigma^2}\boldsymbol{I}_n + (\boldsymbol{\Sigma}_h)^{-1}\right)^{-1}\frac{1}{\sigma^2}}_{=:\boldsymbol{S}_h}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}), \underbrace{\left(\frac{1}{\sigma^2}\boldsymbol{I}_n + (\boldsymbol{\Sigma}_h)^{-1}\right)^{-1}}_{=\boldsymbol{S}_h\sigma^2}\right).$$

We let $\tau \to \infty$ to represent prior ignorance. Then $\boldsymbol{S}_\beta(\tau^2) \xrightarrow{\tau\to\infty} \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top$.

Let $\boldsymbol{z} \sim \mathcal{N}_n(\mathbf{0}, \boldsymbol{I}_n)$. Then

$$\boldsymbol{X}\boldsymbol{\beta} \mid (\boldsymbol{Y} = \boldsymbol{y}) \overset{\mathcal{D}}{=} \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top(\boldsymbol{y} - \boldsymbol{h}) + \sigma\boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{z}$$

and hence,

$$\boldsymbol{\beta} \mid (\boldsymbol{Y} = \boldsymbol{y}) \overset{\mathcal{D}}{=} \left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top(\boldsymbol{y} - \boldsymbol{h}) + \sigma\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{z}. \tag{4.4}$$

With some matrix manipulations we get

$$\begin{aligned}\boldsymbol{S}_h &= \left(\frac{1}{\sigma^2}\boldsymbol{I}_n + (\boldsymbol{\Sigma}_h)^{-1}\right)^{-1}\frac{1}{\sigma^2}\\ &= \left(\boldsymbol{I}_n + \sigma^2\,(\boldsymbol{\Sigma}_h)^{-1}\right)^{-1}\\ &= \boldsymbol{\Sigma}_h\left(\boldsymbol{\Sigma}_h + \sigma^2\boldsymbol{I}_n\right)^{-1}\\ &= \boldsymbol{\Sigma}_h\boldsymbol{V}^{-1}.\end{aligned}$$

Therefore,

$$\begin{aligned}\boldsymbol{h}\,|\,(\boldsymbol{Y}=\boldsymbol{y}) &\overset{\mathcal{D}}{=} \boldsymbol{S}_h(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) + \sigma\boldsymbol{S}_h^{1/2}\boldsymbol{z}\\ &= \boldsymbol{\Sigma}_h\boldsymbol{V}^{-1}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) + \sigma\left[\boldsymbol{\Sigma}_u\boldsymbol{V}^{-1}\right]^{1/2}\boldsymbol{z}. \qquad (4.5)\end{aligned}$$

After computing the posterior distributions we can state the Bayesian backfitting algorithm for spatial models.

Algorithm 4.2 (Bayesian backfitting algorithm for spatial models)**.**

[0] initialize $\widehat{\boldsymbol{h}}^{(0)}$, $\widehat{\sigma}^{(0)}$ and set $t=0$

[1] $t = t+1$

[2]
- generate $\boldsymbol{z}\sim\mathcal{N}_n(\boldsymbol{0},\boldsymbol{I}_n)$
- put $\widehat{\boldsymbol{\beta}}^{(t)} = \left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\left(\boldsymbol{y}-\widehat{\boldsymbol{h}}^{(t-1)}\right) + \widehat{\sigma}^{(t-1)}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{z}$

[3]
- generate $\boldsymbol{w}\sim\mathcal{N}_n(\boldsymbol{0},\boldsymbol{I}_n)$
- estimate the covariance parameters $\widehat{\boldsymbol{\theta}}^{(t)}$ and $\left(\widehat{\sigma}^{(t)}\right)^2$ to get $\widehat{\boldsymbol{\Sigma}}_h^{(t)} = \boldsymbol{\Sigma}_h\left(\widehat{\boldsymbol{\theta}}^{(t)}\right)$ and $\widehat{\boldsymbol{V}}^{(t)} = \widehat{\boldsymbol{\Sigma}}_h^{(t)} + \left(\widehat{\sigma}^{(t)}\right)^2\boldsymbol{I}_n$
- put $\widehat{\boldsymbol{h}}^{(t)} = \widehat{\boldsymbol{\Sigma}}_h^{(t)}\left(\widehat{\boldsymbol{V}}^{(t)}\right)^{-1}\left(\boldsymbol{y}-\boldsymbol{X}\widehat{\boldsymbol{\beta}}^{(t)}\right) + \widehat{\sigma}^{(t)}\left[\widehat{\boldsymbol{\Sigma}}_h^{(t)}\left(\widehat{\boldsymbol{V}}^{(t)}\right)^{-1}\right]^{1/2}\boldsymbol{w}$

[4] repeat [1] to [3] until convergence

Remark 4.5.

1. To compute the matrix square root in step [3] we use Algorithm 2.3 with the matrix $\boldsymbol{S}$ being $\widehat{\boldsymbol{\Sigma}}_h^{(t)}\left(\widehat{\boldsymbol{V}}^{(t)}\right)^{-1}$.
2. If the covariance parameters $\boldsymbol{\theta}$ and σ^2 are known we can directly use these quantities and do not have to estimate these in step [3].
3. Similar, if $\boldsymbol{\theta}$ and σ^2 are unknown but the estimates $\widehat{\boldsymbol{\theta}}$ and $\widehat{\sigma}^2$ are available before starting the algorithm then these estimates can be used instead of $\widehat{\boldsymbol{\theta}}^{(t)}$ and $\left(\widehat{\sigma}^{(t)}\right)^2$ in step [3].

The various R codes for the frequentist backfitting algorithm are listed in the Appendix (R-Codes A.8 and A.10).

4.4 Simulations

With the definitions of the frequentist and Bayesian backfitting algorithm in mind, the focus of this section lies on testing these algorithms for a spatial model

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{h} + \boldsymbol{\varepsilon},$$

where

- $\boldsymbol{Y} = (Y_1, \dots, Y_n)^\top \in \mathbb{R}^n$,
- $\boldsymbol{\beta} = (1.0, 2.0, 0.2)^\top \in \mathbb{R}^3$,
- $\boldsymbol{h} \sim \mathcal{N}_n(\boldsymbol{0}, \boldsymbol{\Sigma}_h)$ where $\boldsymbol{\Sigma}_h = \boldsymbol{\Sigma}_h(\boldsymbol{\theta})$ with $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)^\top$ denoting the range, the sill and the smoothness respectively,
- $\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n)$ where σ^2 denotes the nugget effect.

We choose a Matérn covariance function and gridded data points on $[0, 1]^2$, i.e. we work with a regular grid. Of couse, one might also consider uniform data points or a perturbed grid. However, for the purpose of this work we only focus on gridded data points because we do not expect major differences in the results of the backfitting algorithms.

In total, three different settings (with different covariance parameters) are tested. We will vary the sample size n in each of these settings.

4.4.1 Simulations with Known Covariance

First, we want to focus on the case where the covariance matrices $\boldsymbol{\Sigma}_h$ and $\sigma^2 \boldsymbol{I}_n$ are known and do not have to be estimated. This means that the covariance parameters $\boldsymbol{\theta}$ and σ^2 are known. The simulations are done with R-Code A.7 for the frequentist and R-Code A.8 for the Bayesian backfitting algorithm.

In each of the following cases we will run the frequentist backfitting algorithm with $\epsilon = 10^{-8}$. Similar as in Section 3.3, the Bayesian backfitting algorithm is executed twice. We draw in total 1'250 posterior realizations in each run and consider the first 250 draws as the burn-in period.

Case 4.1. $\boldsymbol{\theta} = (0.05, 0.01, 1)^\top$ and $\sigma^2 = 0.1$
Figure 4.2 shows the covariance function of $\boldsymbol{h}$ in this case (the nugget is excluded since it is covered in $\boldsymbol{\varepsilon}$). Note that the effective range is around 0.2.

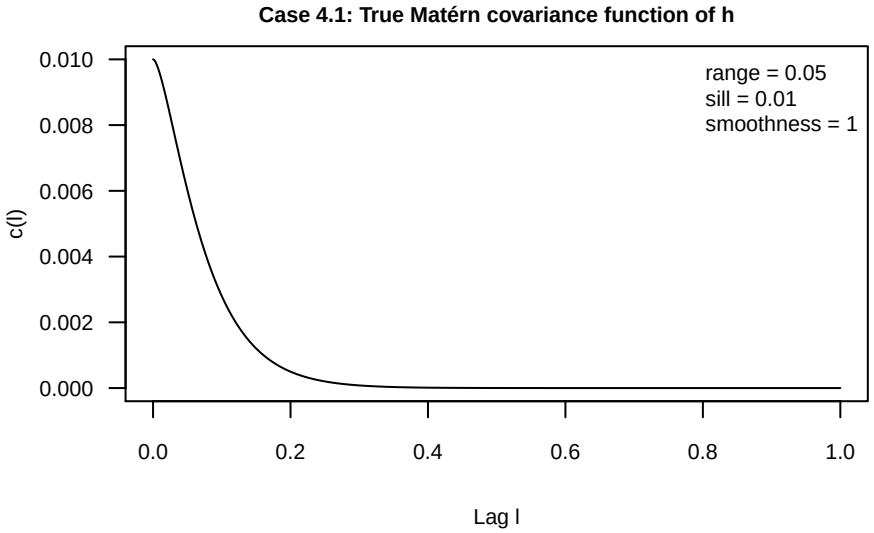

Figure 4.2: Covariance function of $\boldsymbol{h}$ with the true parameters for Case 4.1.

For $n = 1'600$ (i.e. a 40×40 grid), the frequentist backfitting algorithm needs 19 iterations to converge. On the other hand, the Bayesian backfitting algorithm takes around 3 minutes to draw the 1'250 posterior realizations.

The resulting estimates and posterior means for $\boldsymbol{\beta}$ are shown in Table 4.1. Note that we additionally give the 2.5% and 97.5% quantiles of the estimates in the Bayesian setting (in parentheses).

Table 4.1: Estimates of $\boldsymbol{\beta}$ in Case 4.1 for $n = 1'600$.

	true	freq.	Bayesian (1st)	Bayesian (2nd)
β_0	1.0	1.042	1.037 (0.963, 1.119)	1.030 (0.950, 1.111)
β_1	2.0	1.976	1.981 (1.878, 2.077)	1.986 (1.877, 2.098)
β_2	0.2	0.150	0.154 (0.043, 0.255)	0.158 (0.055, 0.268)

We see that both the frequentist estimates and the posterior means from the Bayesian backfitting algorithm are quite close to the true values. Moreover, the posterior means from the two runs of the Bayesian backfitting algorithm are almost the same.

The trace plots of $\boldsymbol{\beta}$ also look good (see Figure 4.3). Notice that since the trace plots of the two different runs look very similar, we only show the ones from the first run.

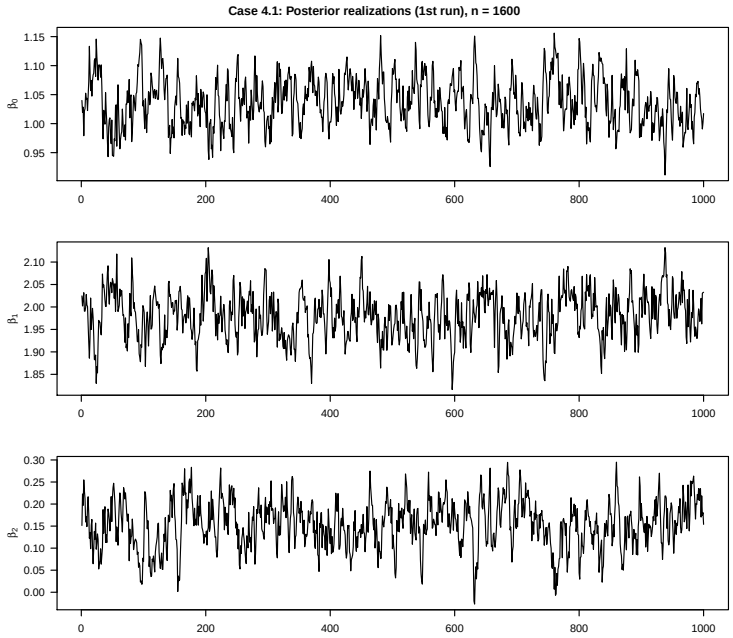

Figure 4.3: Trace plots of $\beta_0, \beta_1, \beta_2$ from the first run of the Bayesian backfitting algorithm in a spatial model with sample size $n = 1'600$.

We can also look at the spatial process $\boldsymbol{h}$. For this reason, we compare the true with the predicted random fields (see Figure 4.4).

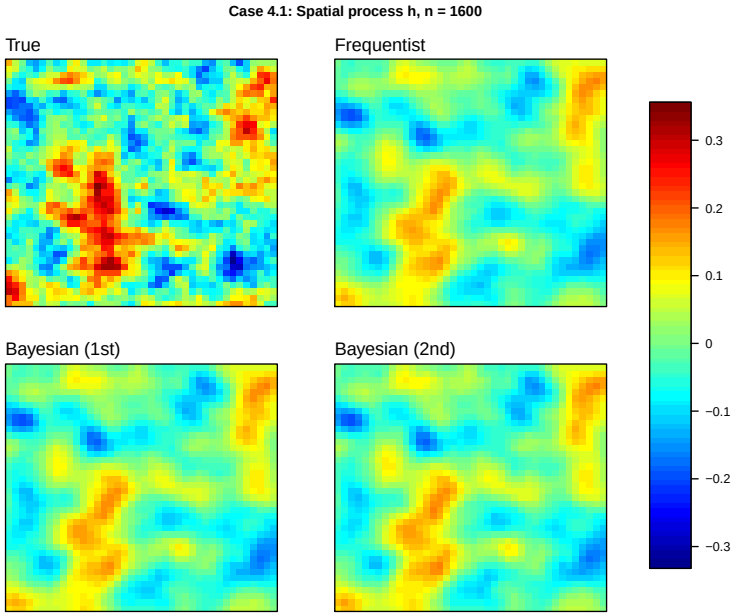

Figure 4.4: Image plots of the true and predicted spatial process $\boldsymbol{h}$ in a spatial model with sample size $n = 1'600$.

So, we see that the predictions are a little bit off compared to the true values. It seems that $\boldsymbol{h}$ would be underestimated in this case. Though, the frequentist and Bayesian fits are very similar to each other.

We can also look at the trace plots of $\boldsymbol{h}$. However, since $\boldsymbol{h} \in \mathbb{R}^{1600}$ we randomly pick three elements thereof. Figure 4.5 shows a good pattern of the traces.

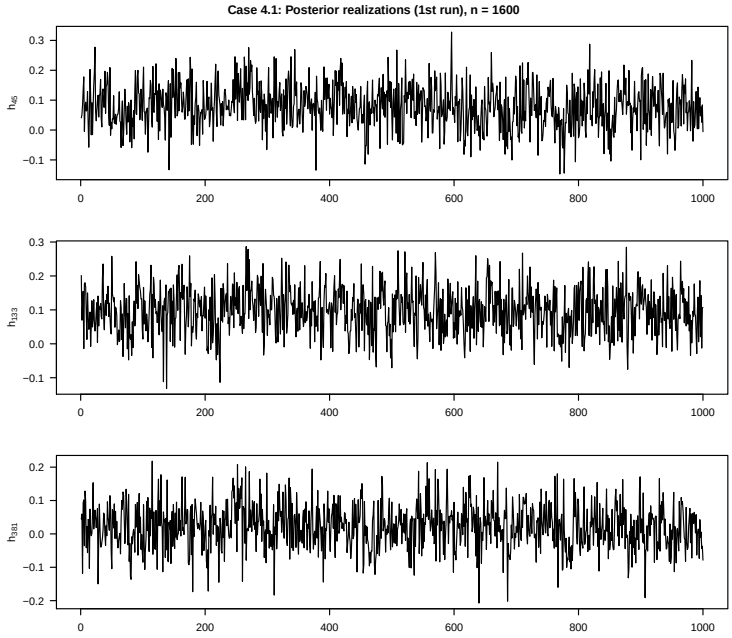

Figure 4.5: Trace plots of h_{45}, h_{133}, h_{381} from the first run of the Bayesian backfitting algorithm in a spatial model with sample size $n = 1'600$.

Finally, Figure 4.6 shows the observations and the fitted values from both the frequentist and the Bayesian backfitting algorithm. We see that the fitted values are smoother than the original observations. Moreover, observe that we do not catch the very extreme values on the grid.

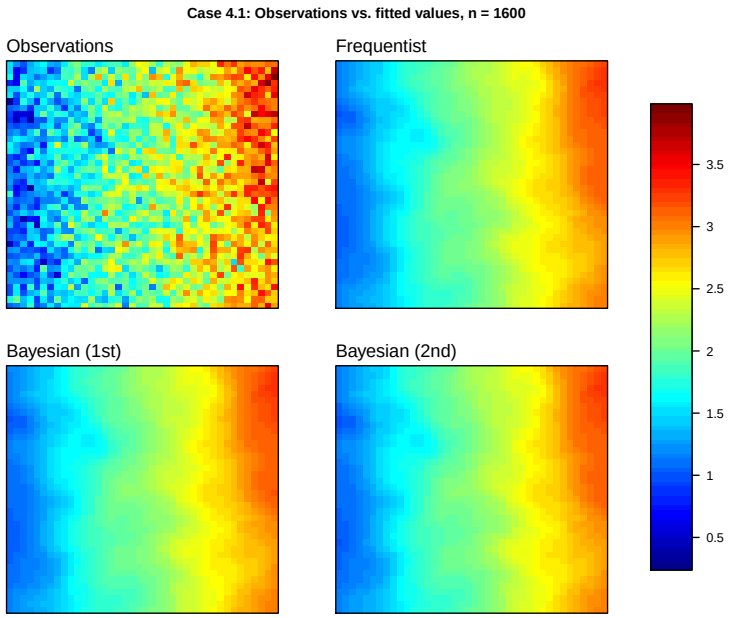

Figure 4.6: Image plots of the observations and the fitted values in a spatial model with sample size $n = 1'600$.

Using the function *persp3D* from the R package *plot3D* (Soetaert, 2019), we can compare the fitted values with the observations in a 3D plot. For simplicity, we only show the fitted values from the frequentist backfitting algorithm since the ones from the Bayesian backfitting algorithm are almost the same (see Figure 4.7).

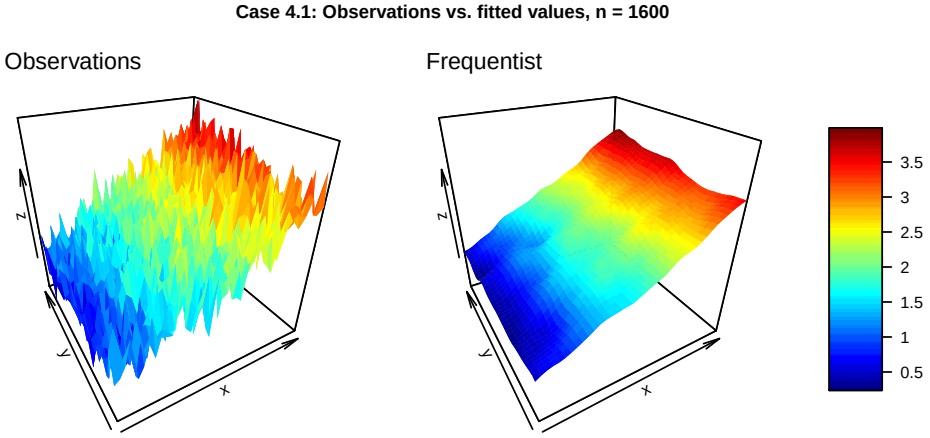

Figure 4.7: 3D plots of the observations and the fitted values from the frequentist backfitting algorithm in a spatial model with sample size $n = 1'600$.

Next, we increase the sample size n from 1'600 to 3'600, i.e. we consider a 60×60 grid in $[0,1]^2$. The resulting estimates of $\boldsymbol{\beta}$ are given in Table 4.2.

Table 4.2: Estimates of $\boldsymbol{\beta}$ in Case 4.1 for $n = 3'600$.

	true	freq.	Bayesian (1st)	Bayesian (2nd)
β_0	1.0	1.030	1.034 (0.954, 1.109)	1.036 (0.957, 1.117)
β_1	2.0	1.998	1.997 (1.906, 2.111)	1.991 (1.905, 2.077)
β_2	0.2	0.144	0.143 (0.057, 0.245)	0.146 (0.054, 0.237)

Compared to Table 4.1, we can see a slight improvement in estimating the parameters β_0 and β_1. On the other hand, β_2 has been closer to the true value when $n = 1'600$.

However, the trace plots of $\boldsymbol{\beta}$ (see Figure 4.8) do not look as optimal as the ones in Figure 4.5 for $n = 1'600$. For $n = 3'600$ the trace plots are not as regular as the ones for $n = 1'600$. When having more data points the draws fall in a smaller range with some peaks in the traces. However, in the case with 1'600 data points the range is more or less constant over all 1'000 draws.

The trace plots of $\boldsymbol{h}$ in this case show a similar pattern as the ones for $n = 1'600$ and are therefore omitted.

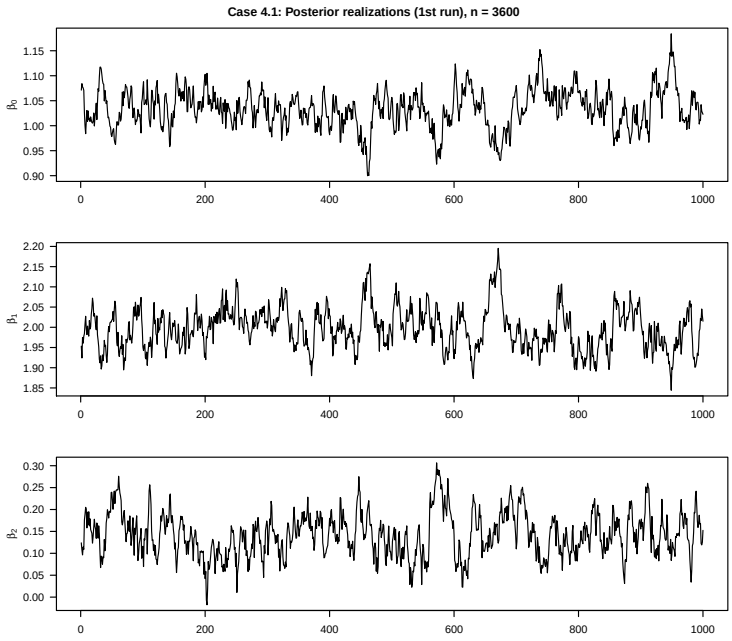

Figure 4.8: Trace plots of $\beta_0, \beta_1, \beta_2$ from the first run of the Bayesian backfitting algorithm in a spatial model with sample size $n = 3'600$.

Case 4.2. $\boldsymbol{\theta} = (0.08, 0.5, 0.6)^\top$ and $\sigma^2 = 0.3$
Figure 4.9 shows the covariance function of $\boldsymbol{h}$ in this case. Note that the effective range is around 0.3.

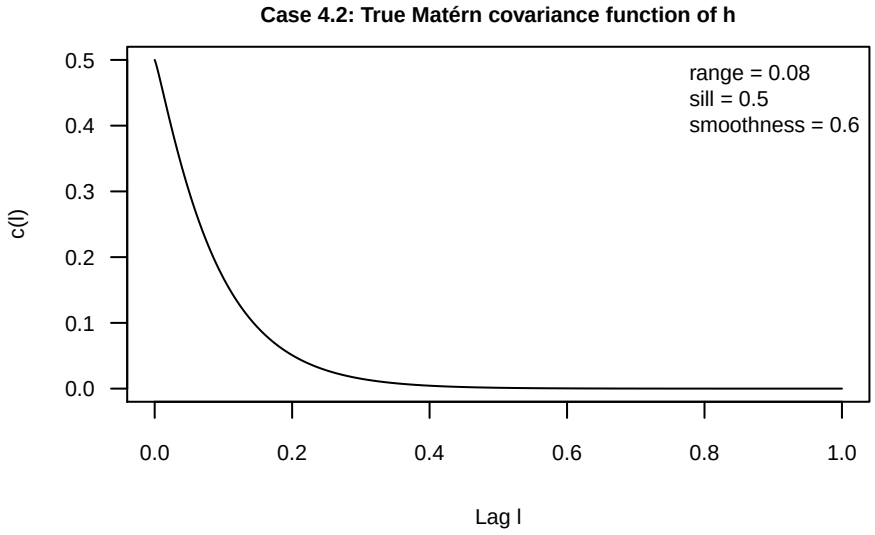

Figure 4.9: Covariance function of $\boldsymbol{h}$ with the true parameters for Case 4.2.

For n = 1'600, the frequentist backfitting algorithm needs 173 iterations to converge. On the other hand, the Bayesian backfitting algorithm needs around 3 minutes to draw the 1'250 posterior realizations.

The resulting estimates and posterior means of $\boldsymbol{\beta}$ are shown in Table 4.3.

Table 4.3: Estimates of $\boldsymbol{\beta}$ in Case 4.2 for n = 1'600.

	true	freq.	Bayesian (1st)	Bayesian (2nd)
β_0	1.0	1.449	1.382 (0.914, 1.825)	1.163 (0.476, 1.886)
β_1	2.0	1.789	1.906 (1.369, 2.481)	2.010 (1.125, 2.802)
β_2	0.2	−0.241	−0.234 (−1.007, 0.457)	−0.067 (−0.623, 0.823)

Compared to Table 4.1 of Case 4.1 we observe not quite precise estimates for $\boldsymbol{\beta}$. We see that the posterior means from the Bayesian backfitting algorithm are closer to the true values than the frequentist fits.

Figure 4.10 shows the trace plots of $\boldsymbol{\beta}$ of the first run of the Bayesian backfitting algorithm. There is not much of a difference between the two runs, it is therefore enough to show the trace plots for one run. Compared to Figure 4.3 of Case 4.1 there is less high-frequency oscillation in the traces. The jumps between the realizations are much smaller but there is more meandering throughout the indices from 1 to 1'000.

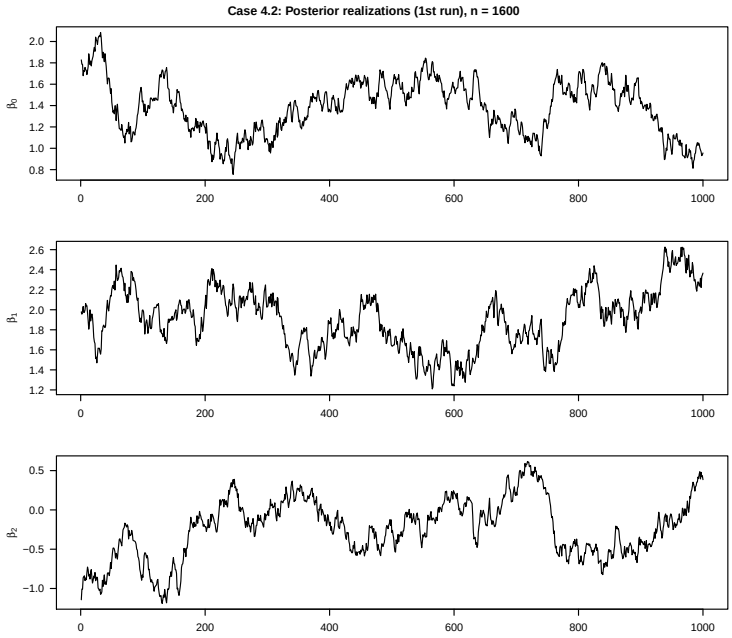

Figure 4.10: Trace plots of $\beta_0, \beta_1, \beta_2$ from the first run of the Bayesian backfitting algorithm in a spatial model with sample size $n = 1'600$.

Looking at the spatial process $\boldsymbol{h}$ gives a similar impression as in Case 4.1. Figure 4.11 shows the true and the estimated spatial processes. We again see that the backfitting algorithms smooth the field, not as pronounced as in Figure 4.4 of Case 4.1.

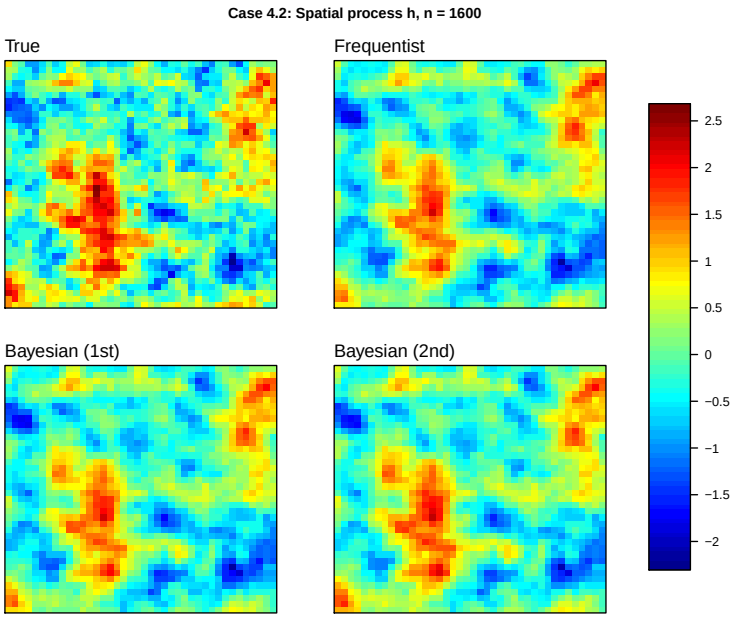

Figure 4.11: Image plots of the true and predicted spatial process $\boldsymbol{h}$ in a spatial model with sample size $n = 1'600$.

However, the trace plots of $\boldsymbol{h}$ have a better shape as the ones for $\boldsymbol{\beta}$. Figure 4.12 displays the traces of three randomly chosen elements of $\boldsymbol{h} \in \mathbb{R}^{1600}$.

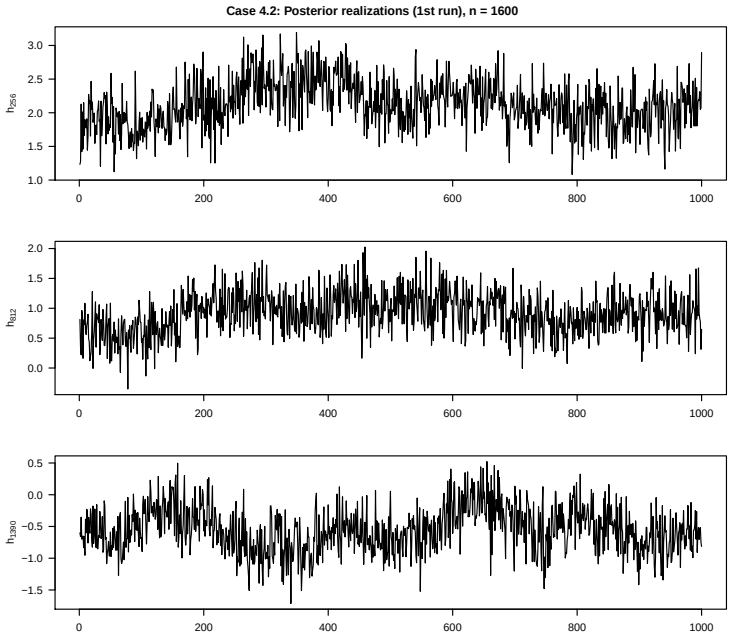

Figure 4.12: Trace plots of $h_{256}, h_{812}, h_{1390}$ from the first run of the Bayesian backfitting algorithm in a spatial model with sample size $n = 1'600$.

Nevertheless, if we look at the image plots of the observations and the fitted values from both backfitting algorithms we actually see a good fit. The fitted values result in a smoother field compared to the original observations (see Figure 4.13).

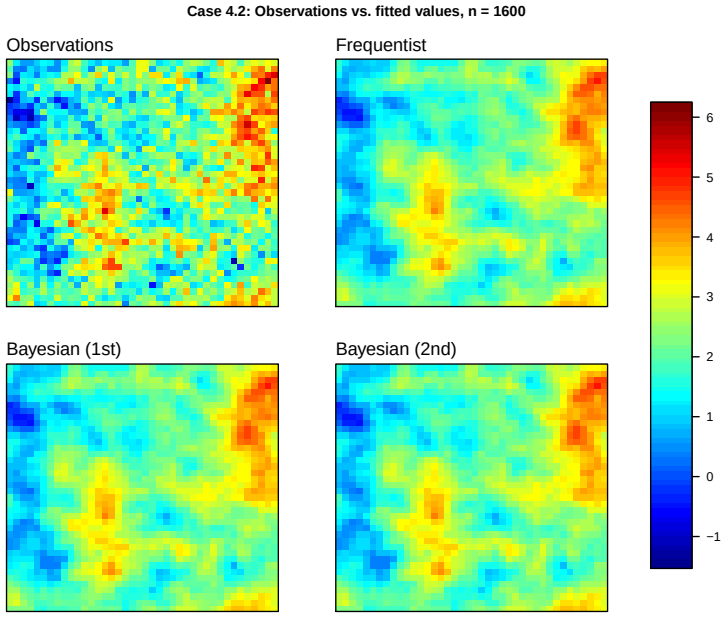

Figure 4.13: Image plots of the observations and the fitted values in a spatial model with sample size $n = 1'600$.

Finally, we can also look at the 3D plot and compare the observations with the fitted values one from the frequentist backfitting algorithm (see Figure 4.14).

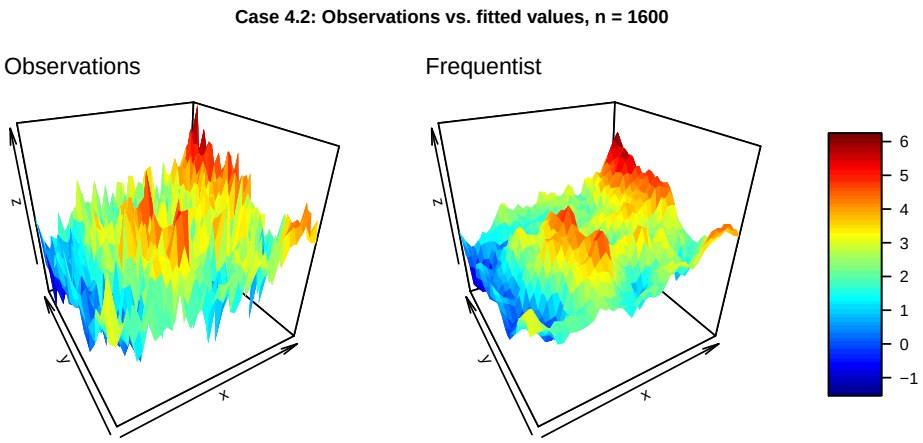

Figure 4.14: 3D plots of the observations and the fitted values from the frequentist backfitting algorithm in a spatial model with sample size $n = 1'600$.

What happens if we increase the sample size to 3'600? Table 4.4 shows the estimates and posterior means of $\boldsymbol{\beta}$ in this case. Compared to Table 4.3 where $n = 1'600$, the frequentist estimate for β_0 is similar, the one for β_1 is a bit better and the one for β_2 is worse.

Table 4.4: Estimates of $\boldsymbol{\beta}$ in Case 4.2 for $n = 3'600$.

	true	freq.	Bayesian (1st)	Bayesian (2nd)
β_0	1.0	1.429	1.472 (1.025, 1.864)	1.576 (1.205, 1.985)
β_1	2.0	1.905	1.956 (1.355, 2.440)	1.752 (1.278, 2.221)
β_2	0.2	−0.400	−0.419 (−0.991, 0.080)	−0.336 (−0.928, 0.267)

We can already say that an increased number of observations does not help much to estimate $\boldsymbol{\beta}$.

Case 4.3. $\boldsymbol{\theta} = (0.15, 1.5, 1.5)^\top$ and $\sigma^2 = 0.3$
Figure 4.15 shows the covariance function of $\boldsymbol{h}$ in this case. Note that the effective range is around 0.8.

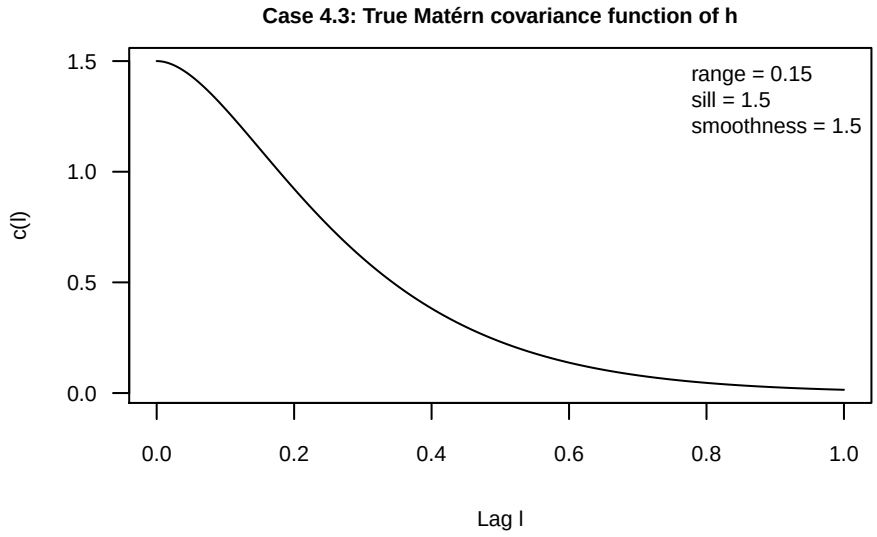

Figure 4.15: Covariance function of $\boldsymbol{h}$ with the true parameters for Case 4.3.

For $n = 1'600$, the frequentist backfitting algorithm needs 943 iterations to converge. On the other hand, the Bayesian backfitting algorithm takes around 3 minutes to draw the 1'250 posterior realizations.

The resulting estimates and posterior means for $\boldsymbol{\beta}$ are shown in Table 4.5.

Table 4.5: Estimates of $\boldsymbol{\beta}$ in Case 4.3 for $n = 1'600$.

	true	freq.	Bayesian (1st)	Bayesian (2nd)
β_0	1.0	3.166	2.781 (2.046, 3.764)	1.740 (0.407, 4.107)
β_1	2.0	0.744	1.131 (0.347, 2.269)	1.350 (−1.291, 3.067)
β_2	0.2	−1.394	−1.770 (−3.350, −0.257)	0.026 (−1.332, 1.442)

The estimates are now completely off. This is a consequence of the high effective range of around 0.8. Note that we are still on $[0, 1]^2$. Therefore, a spatial feature might be located on the whole grid. And since the sill is 1.5, this feature might get very big. Figure 4.16 shows the image plot and the 3D plot of the observations $\boldsymbol{y}$. If we look closer at the 3D plot then it makes sense that we cannot easily distinguish between the linear trend $\boldsymbol{X\beta}$ and the spatial process $\boldsymbol{h}$. Thus, the backfitting algorithm has problems to correctly estimate $\boldsymbol{\beta}$ and $\boldsymbol{h}$. However, this problem also occurs when we use a maximum likelihood estimation.

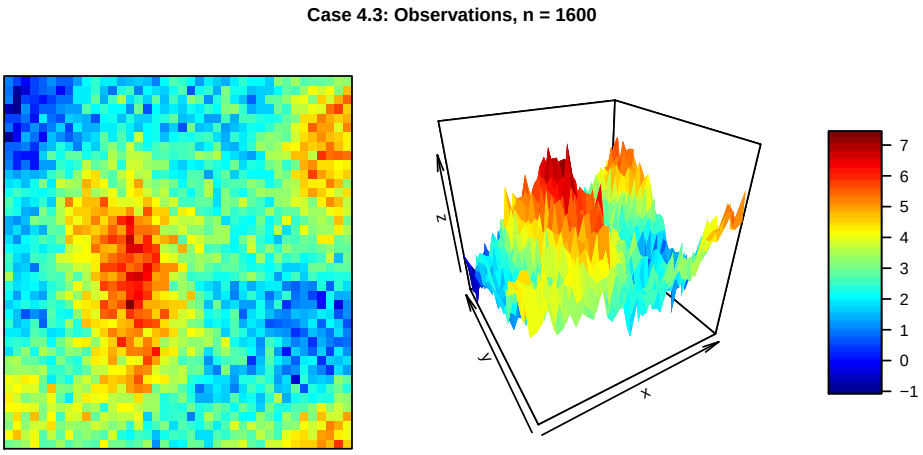

Figure 4.16: Image plot and 3D plot of the observations in Case 4.3 for $n = 1'600$.

The trace plots of $\boldsymbol{\beta}$ are shown in Figure 4.17. These traces do not look optimal, again a consequence of the large effective range.

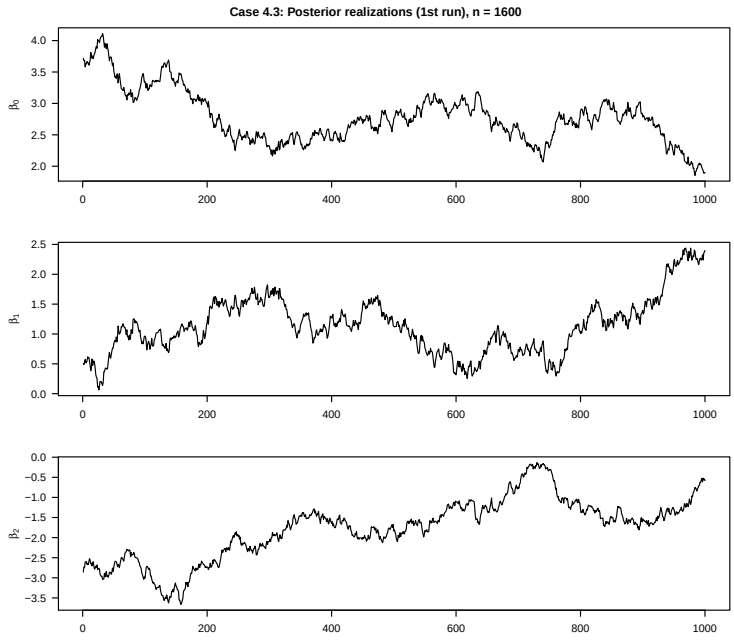

Figure 4.17: Trace plots of $\beta_0, \beta_1, \beta_2$ from the first run of the Bayesian backfitting algorithm in a spatial model with sample size $n = 1'600$.

Similar as for Cases 4.1 and 4.2 we see that the prediction of the spatial process $\boldsymbol{h}$ is a little bit off. However, different to the other cases, the predictions from the frequentist and Bayesian backfitting algorithm do not look the same. We see that the posterior mean of $\boldsymbol{h}$ from the second run of the Bayesian backfitting algorithm is closer to the true $\boldsymbol{h}$ (see Figure 4.18).

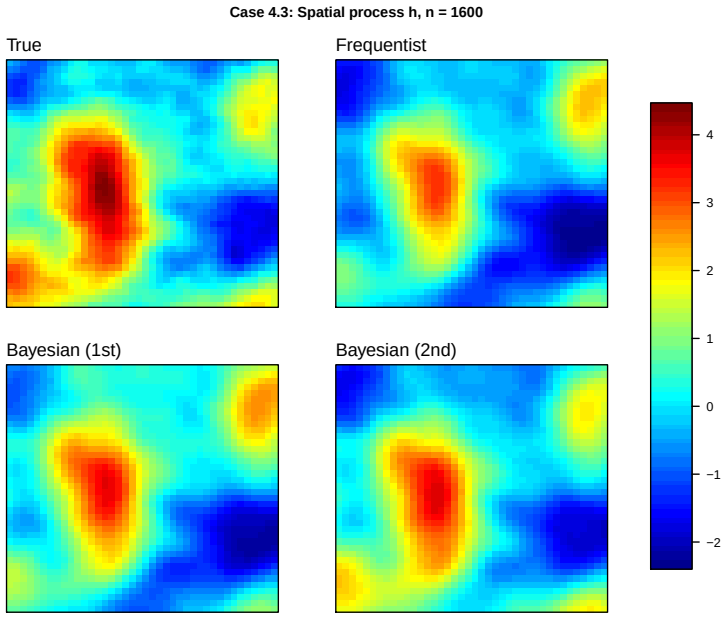

Figure 4.18: Image plots of the true and predicted spatial process $\boldsymbol{h}$ in a spatial model with sample size $n = 1'600$.

If we have a look at the trace plots of three randomly selected elements of $\boldsymbol{h}$ from the first run of the Bayesian backfitting algorithm in Figure 4.19 we see a slight trend in one direction. Compared to the trace plots of $\boldsymbol{\beta}$ we see more fluctuation in the traces of $\boldsymbol{h}$ but it does still not look optimal. Again as a consequence of the large effective range, these trace plots are highly correlated. We observe an upward trend of the traces of $\boldsymbol{h}$ whereas in Figure 4.17 we see a downward trend of the trace of β_0.

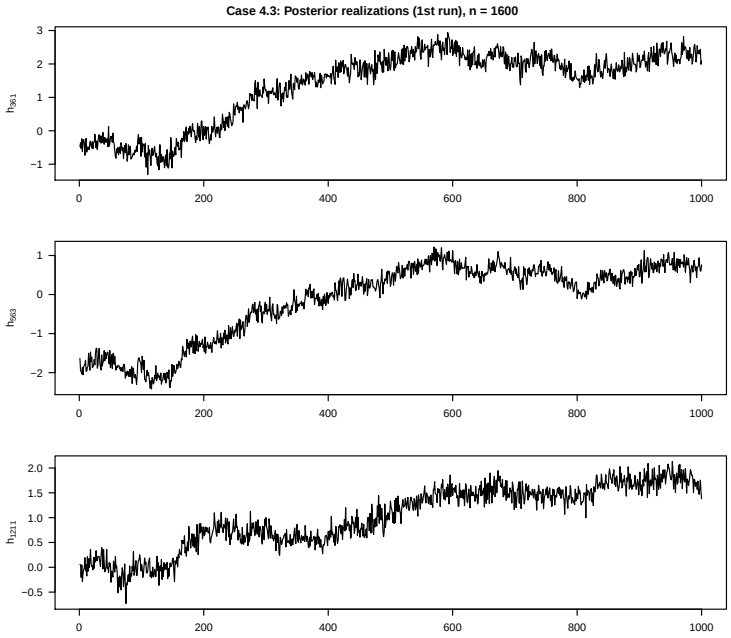

Figure 4.19: Trace plots of $h_{361}, h_{563}, h_{1211}$ from the first run of the Bayesian backfitting algorithm in a spatial model with sample size $n = 1'600$.

Nevertheless, the fitted values from both backfitting algorithms look quite equal. Similar to Cases 4.1 and 4.2 we obtain a smoother field compared to the observations (see Figure 4.20).

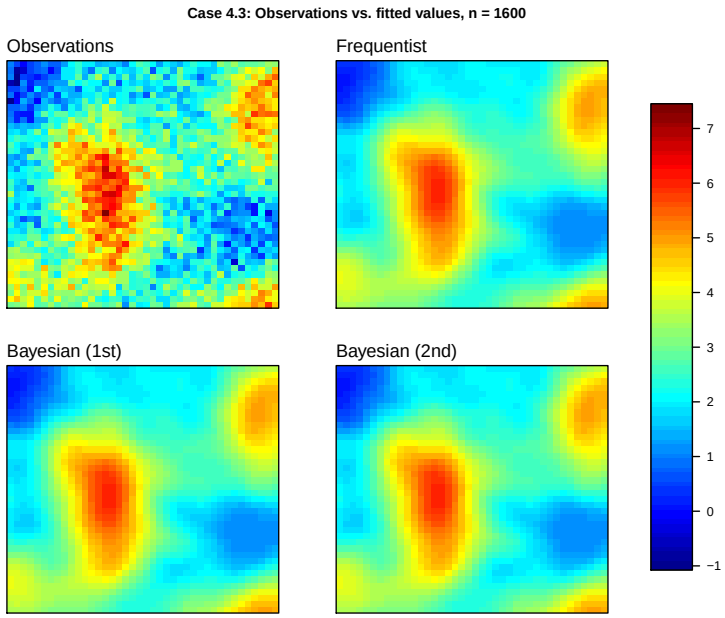

Figure 4.20: Image plots of the observations and the fitted values in a spatial model with sample size $n = 1'600$.

Finally, Figure 4.21 compares the 3D plots of the observations and the fitted values from the frequentist backfitting algorithm.

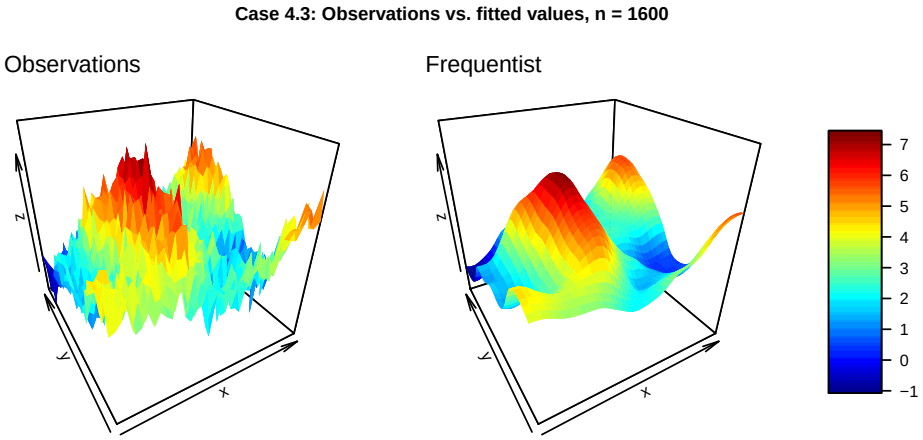

Figure 4.21: 3D plots of the observations and the fitted values from the frequentist backfitting algorithm in a spatial model with sample size $n = 1'600$.

4.4.2 Simulations with Unknown Covariance

Consider the case where the covariance matrices $\boldsymbol{\Sigma}_h$ and $\sigma^2 \boldsymbol{I}_n$ are unknown and have to be estimated. We work with $\boldsymbol{\beta} = (1.0, 2.0, 0.2)^\top$. The simulations in this case are executed with

R-Code A.9 for the frequentist and R-Code A.10 for the Bayesian backfitting algorithm.

Case 4.4. $\boldsymbol{\theta} = (0.05, 0.01, 1)^\top$ and $\sigma^2 = 0.1$, estimate all covariance parameters
We start with a regular 40×40 grid on $[0,1]^2$. Thus, we have $n = 1'600$. We actually know the true covariance parameters but we now want to see how the backfitting algorithm performs when estimating these parameters within the algorithm. For this reason, the function `mle.nomean.spam()` from the R package *spam* (Furrer and Sain, 2010) is used to estimate the covariance parameters in each iteration. The frequentist backfitting algorithm needs 16 iterations and approximately 45 minutes to converge with $\epsilon = 10^{-8}$. On the other hand, the Bayesian backfitting algorithm takes around 3 hours to draw 50 posterior realizations. The first 10 draws are considered to be the burn-in period. Notice that due to time reasons we choose a smaller number of posterior realizations as for the cases where the covariances are known.

Table 4.6 shows the estimates for $\boldsymbol{\beta}$ in this case. We see a good approximation of $\boldsymbol{\beta}$ with both backfitting algorithms.

Table 4.6: Estimates of $\boldsymbol{\beta}$ in Case 4.4 for $n = 1'600$.

	true	`mle.spam()`	frequentist	Bayesian
β_0	1.0	1.001	1.001	0.958 (0.910, 1.027)
β_1	2.0	1.963	1.963	1.975 (1.936, 2.021)
β_2	0.2	0.229	0.229	0.232 (0.184, 0.298)

Moreover, the estimates from the backfitting algorithms are very similar to the ones from the function `mle.spam()` from the R package *spam*. This function optimizes the likelihood as a function of all seven parameters $\beta_0, \beta_1, \beta_1, \theta_1, \theta_2, \theta_3, \sigma^2$.

We first compare the true spatial process $\boldsymbol{h}$ with the prediction from the frequentist backfitting algorithm. Figure 4.22 shows quite a good approximation. However, we still do not catch the most extreme values (as already seen in the cases with known covariance).

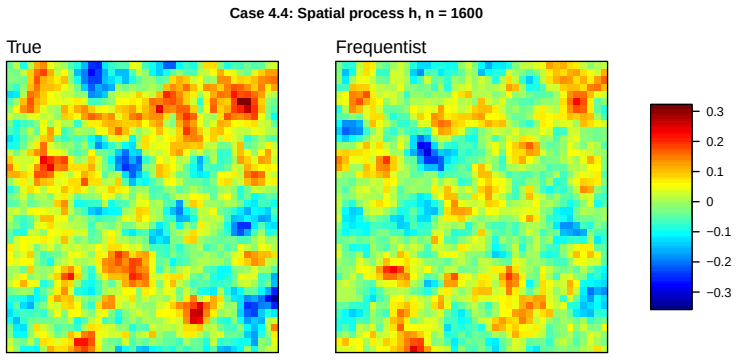

Figure 4.22: Image plots of the true and predicted spatial process $\boldsymbol{h}$ from the frequentist backfitting algorithm in a spatial model with sample size $n = 1'600$.

Figure 4.23 shows the observations and the fitted values from the frequentist backfitting algorithm. We see that fitted values are smoother than the observations.

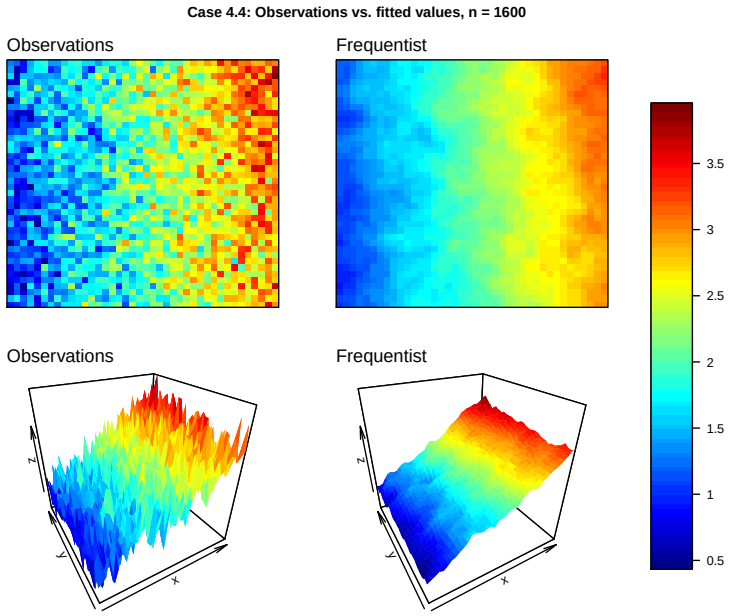

Figure 4.23: Image and 3D plots of the observations and the fitted values from the frequentist backfitting algorithm in a spatial model with sample size $n = 1'600$.

However, if we look at the fitted values from the Bayesian backfitting algorithm (i.e. the posterior mean of $\boldsymbol{X\beta} + \boldsymbol{h}$) we see a different pattern compared to the fitted values from the frequentist approach. Nonetheless, the fitted values in this case look more similar to the original observations (see Figure 4.24).

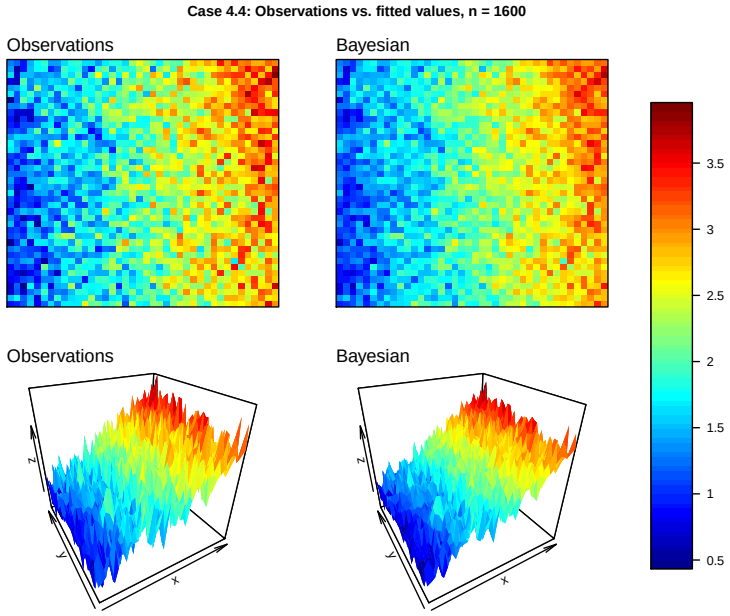

Figure 4.24: Image and 3D plots of the observations and the fitted values from the Bayesian backfitting algorithm in a spatial model with sample size $n = 1'600$.

Our main focus lies in estimating the covariance parameters. Table 4.7 shows the estimates of

the range, sill, smoothness and nugget.

Table 4.7: Estimates of the covariance parameters in Case 4.4 for $n = 1'600$.

	true	`mle.spam()`	frequentist	Bayesian
θ_1 = range	0.05	0.025	0.029	0.082 (0.032, 0.100)
θ_2 = sill	0.01	0.015	0.018	0.082 (0.015, 0.100)
θ_3 = smoothness	1.0	0.999	0.574	0.114 (0.036, 0.873)
σ^2 = nugget	0.1	0.095	0.091	0.029 (0.011, 0.094)

First of all, we see that the function `mle.spam()` estimates the covariance parameters really good in this setting. Moreover, the frequentist backfitting algorithm has problems to estimate the smoothness parameter.

Finally, we can compare the covariance functions of $\boldsymbol{h}$ with the estimated parameters. In Figure 4.25 all three covariance functions are displayed. However, all three have different shapes. Note that the covariance function from the Bayesian backfitting algorithm (green curve) would extend to 0.082 when the lag ℓ is tending to zero.

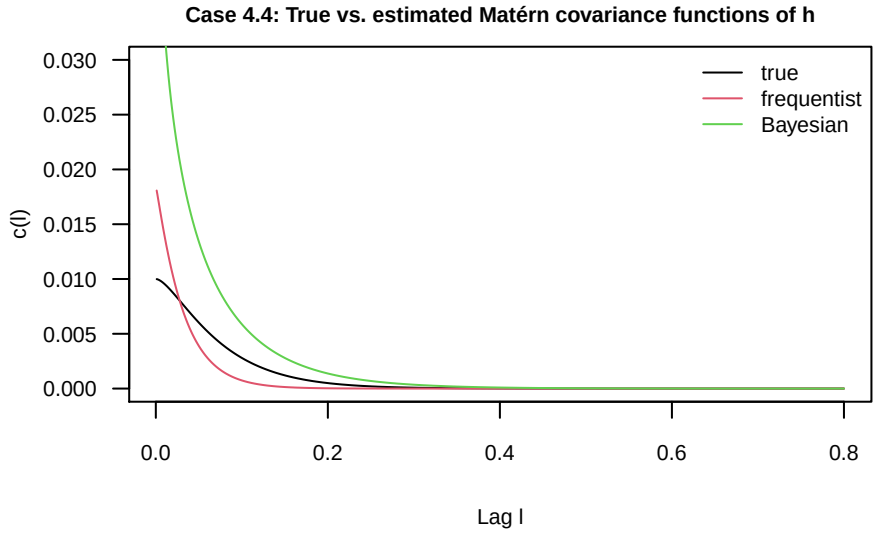

Figure 4.25: Covariance functions of $\boldsymbol{h}$ with the true and estimated parameters for Case 4.4.

Case 4.5. $\boldsymbol{\theta} = (0.05, 0.01, 1)^\top$ and $\sigma^2 = 0.1$, estimate only sill and nugget
We focus on the same setting as before in Case 4.4 but we are now only interested in estimating the sill and the nugget within the backfitting algorithm. We keep the range and the smoothness parameter fixed. This might be useful if one can estimate the range and the smoothness parameters beforehand.

Instead of optimizing all covariance parameters with the function `mle.nomean.spam()` we optimize the sill and the nugget parameter with the `optim()` function. Since in each iteration only two and not four parameters are estimted, this procedure should be significantly faster than the backfitting in Case 4.4.

We consider the same grid and field as in Case 4.4 with $n = 1'600$. The frequentist backfitting algorithm converges after 25 iterations with $\epsilon = 10^{-8}$. This takes approximately one hour. The time for drawing 50 realizations with the Bayesian backfitting algorithm is approximately 2.5 hours. Compared to Case 4.4 where all covariance parameters are estimated, the frequentist backfitting algorithm takes a bit longer but also has more iterations. The Bayesian backfitting algorithm needs approximately 30 minutes less than in Case 4.4 for the same number of posterior realizations.

Table 4.8 shows the estimates and posterior means of $\boldsymbol{\beta}$ as well as of the sill and the nugget in this case. We see that the parameters are estimated quite well and that the estimates of $\boldsymbol{\beta}$ are very similar to the ones in Table 4.6 of Case 4.4.

Table 4.8: Estimates of $\boldsymbol{\beta}$ and the covariance parameters in Case 4.5 for $n = 1'600$.

	true	frequentist	Bayesian
β_0	1.0	1.003	0.978 (0.913, 1.060)
β_1	2.0	1.960	1.961 (1.857, 2.065)
β_2	0.2	0.225	0.237 (0.153, 0.346)
$\theta_2 =$ sill	0.01	0.009	0.011 (0.009, 0.013)
$\sigma^2 =$ nugget	0.1	0.101	0.100 (0.100, 0.101)

Finally, we can compare the covariance functions with the estimated parameters. In Figure 4.26 all three covariance functions are displayed. The shapes of the covariance functions are very similar.

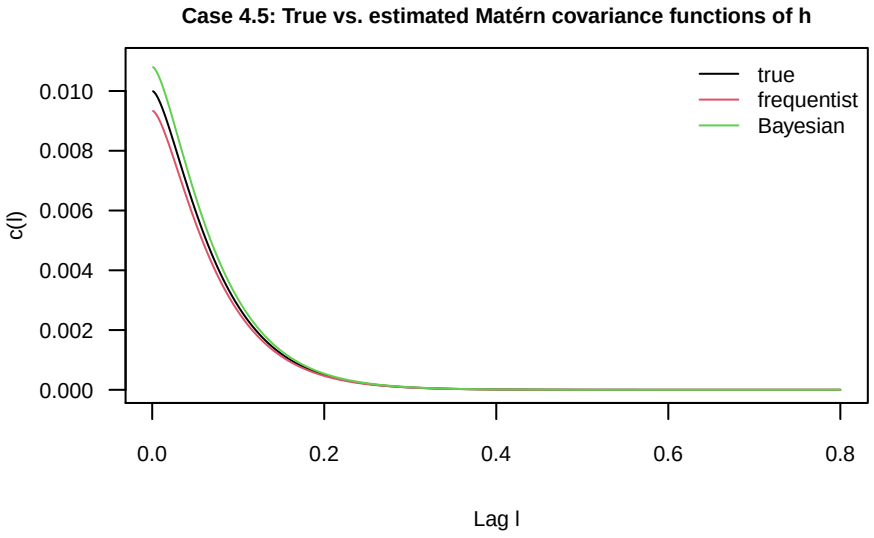

Figure 4.26: Covariance function with the true and estimated parameters for Case 4.5.

Chapter 5

Discussion

In this work we have investigated the backfitting algorithm for additive models and its extension to the Bayesian framework. The frequentist backfitting algorithm for additive models fits all the functions f_j whereas the Bayesian backfitting algorithm is a generalization of the frequentist one with the goal to sample from the posterior distribution of an additive model.

Whereas the frequentist backfitting algorithm returns some final estimates (after convergence), the Bayesian backfitting algorithm creates a whole sample from the posterior distribution.

The idea of backfitting has then been extended to the class of linear mixed-effects models. The idea is still the same as for additive models. In the frequentist case we end up with estimates for the parameters $\boldsymbol{\beta}$ and predictions for the random effect $\boldsymbol{u}$. With the Bayesian backfitting algorithm we generate a sample from the posterior distributions of $\boldsymbol{\beta}$ and $\boldsymbol{u}$. To compare these two algorithms in terms of accuracy of the estimates we can take the mean of our created sample from the posterior distribution (i.e. the posterior mean) and compare it to the estimates from the frequentist backfitting algorithm.

After seeing many simulations of the backfitting algorithm for linear mixed-effects models in Chapter 3 and spatial models in Chapter 4 we can summarize all the insights and results we have gained.

For mixed models the backfitting algorithm performs quite well. The main problem is the estimation of the intercept parameter β_0. This parameter has not been estimated correctly in both the frequentist and the Bayesian backfitting algorithm. The remaining parameters for the linear trend have been approximated very well. However, there are some differences between both algorithms for predicting the random effect $\boldsymbol{u}$. We have seen various settings but overall we cannot conclude which algorithm performs better in predicting the random effects. In most situations the Bayesian backfitting algorithm yields slightly closer predictions to the true values. However, since there are also differences between each run of the Bayesian backfitting algorithm we cannot guarantee this tendency.

In the case where the variance parameters σ_u^2 and σ^2 are unknown, we have seen that with the `lmer()` function from the R package `lme4` (Bates *et al.*, 2015) we get very accurate estimates of the variance components. Nonetheless, the variance estimates from the backfitting algorithms are also reasonable but not as close to the true values as the `lmer()` estimates. The global variance σ^2 is approximated quite well whereas the variance from the random effect σ_u^2 needs more replicates for a good estimation.

One of the major issues with the Bayesian backfitting algorithm is the time. Depending on the sample size n and how many posterior realizations one wants to draw, this algorithm can take very long. On the other hand, the frequentist backfitting algorithm converges already after a few iterations, no matter what the sample size is.

However, one has to have in mind that with the backfitting algorithm (frequentist or Bayesian) model misspecifications cannot be tracked. With the backfitting algorithm we do not know if we have too many or too few parameters in our model. Compared to an `lm()` or `lmer()` fit we do not get the significances or p-values of the parameters.

It is hard to argue whether one is better than the other in terms of estimating the parameters. However, with the Bayesian backfitting algorithm we do not have to deal with uncertainty of the estimates. In the frequentist backfitting algorithm, where we end up with one final estimate for each parameter, we can ask ourselves what the confidence interval for this specific parameter is. It could be difficult to calculate the standard error of the parameter $\boldsymbol{\beta}$ because the variance of its estimate $\widehat{\boldsymbol{\beta}}$ is given by

$$\text{Var}\left(\widehat{\boldsymbol{\beta}}\right) = \left(\boldsymbol{X}^\top \boldsymbol{V}^{-1} \boldsymbol{X}\right)^{-1}$$

and depending on the size of the data, computing this might be a challenge or even impossible. For this reason, computing the standard error and therefore the confidence interval might also be impossible. On the other hand, with the Bayesian backfitting algorithm we actually draw from the posterior distribution and can directly look at the quantiles of our samples. Therefore, in this case we do not have to calculate a variance.

In a next step, we have also extended the backfitting algorithm to the class of spatial models. We have used that a spatial model is a special case of a linear mixed-effects model and therefore, all computations and algorithms could be easily adapted to spatial models. In the simulations we have focused on a spatial field with a Matérn covariance function and have investigated different settings with different covariance parameters.

Overall, we can say that the larger the effective range is, the more problems the backfitting algorithm has. In the setting with a small effective range of around 0.2 we have obtained good estimates for $\boldsymbol{\beta}$ and also a reasonable fit for the spatial process $\boldsymbol{h}$. The fitted values look quite similar to the true observations which is a good sign.

For an effective range of around 0.3 we do not see as accurate estimates as for the first case. But still, the estimates are more or less reasonable and in a similar range as the true values.

Looking at an effective range of around 0.8 shows a different behavior. The estimates for $\boldsymbol{\beta}$ are completely off, the absolute difference to the true values is between 1 and 2. This is due to the shape of the random field. Looking at the 3D plot of the random field in this case (Figure 4.16) we cannot simply distinguish between the linear trend $\boldsymbol{X\beta}$ and the spatial process $\boldsymbol{h}$. Even though the estimation for the trend and the spatial process is bad, in the end the fitted values have a similar shape as the observations. Only the weighting of the linear trend and the spatial process is not correct.

In the case where the covariance parameters are unknown, we have seen that in the frequentist case the estimation of $\boldsymbol{\beta}$ and $\boldsymbol{h}$ are very similar to the case where the covariance parameters are known. Additionally, the covariance parameters are estimated well except from the smoothness parameter, which has been underestimated with the frequentist backfitting algorithm.

On the other hand, the Bayesian backfitting algorithm has problems with predicting the spatial process correctly. In the end, we see that the fitted values are almost exactly the same as the true observations (see Figure 4.24). However, the estimated covariance parameters (i.e. the posterior means of all drawn covariance parameters) are not as accurate as the frequentist estimates.

This problem can be relaxed if we only estimate the sill and nugget parameters with the backfitting algorithm. Then these two parameters are estimated very precisely with both the frequentist and Bayesian backfitting algorithm.

Overall we can argue that the performance of the backfitting algorithm for fitting spatial models is satisfying. It depends on the actual setting and what the final goal is. The fitted values look quite similar to the observations in all cases. As already mentioned, there are some problems in differentiating between the linear trend and the effect of the spatial process.

If the covariance parameters are known, or can be estimated beforehand, the backfitting algorithm is also fast. However, the duration increases heavily if we also want to estimate the covariance parameters within the backfitting algorithm.

One of the main challenges in spatial statistics is to deal with very large datasets. In Chapter 4 we have used simulated datasets with 1'600 or 3'600 data points. However, in practice one has to handle much larger datasets. 100'000 data points are not a coincidence. Also the size of the grid can be much larger than just $[0, 1]^2$. And since the Matérn covariance function is not compactly supported also points with a large distance between results in a positive value of the covariance function.

Of couse, we could simply apply our backfitting algorithm for this kind of data but the time consumption will be immense. A possible solution for this problem is to taper the covariance function (Furrer *et al.*, 2006). The main idea is that we multiply the Matérn covariance function with a covariance function of compact support. The resulting function is then also compactly supported. This procedure saves time in executing the backfitting algorithm which makes it also valuable for large datasets.

Chapter 6

Outlook

Throughout this work we have seen that the backfitting algorithm performs well under different settings. However, we can ask ourselves how we can further improve the algorithm to obtain even better results. There are various options so that we can extend or modify the frequentist and Bayesian backfitting algorithm.

We will shortly explain some possible options.

1. *Convergence of the Bayesian backfitting algorithm*
   In this work we have not talked much about the convergence of the Bayesian backfitting algorithm. In the Bayesian setting the last step in the algorithm for mixed and spatial models is always "repeat [1] to [3] until convergence". In the simulations we have always chosen the number of posterior draws and stopped the algorithm as soon as the sample is large enough. The choice of the number of draws is completely subjective and can be modified as desired. Nonetheless, there are also measures of convergence for posterior sampling. One can investigate how convergence is really achieved in the Bayesian backfitting algorithm.

2. *Full Bayesian approach of the backfitting algorithm*
   In the Bayesian backfitting algorithm for mixed models we impose a prior on $\boldsymbol{\beta}$ but keep the variance parameters σ_u^2 and σ^2 fixed and view these as non-random. Nevertheless, to have a full Bayesian framework one can also choose priors for both σ_u^2 and σ^2. In the case of spatial models one can also impose priors for the covariance parameters $\boldsymbol{\theta}$ and the nugget effect σ^2.

3. *Estimation of the covariance parameters in mixed and spatial models*
   As mentioned in the different simulations, known variances can save a massive amount of time to run the backfitting algorithm. Nonetheless, it is often the case that the variance parameters are unknown. It might still be possible to estimate these with the data before actually executing the backfitting algorithm. If this is not possible then the variance parameters have to be estimated within the backfitting algorithm. However, there are different methods for estimating the variances. One should investigate if there are some estimation methods under which the backfitting algorithm performs better.

4. *Backfitting algorithm for spatial models with two spatial processes*
   Consider the spatial model
   $$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{h}_1 + \boldsymbol{h}_2 + \boldsymbol{\varepsilon},$$

where $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ are two different spatial processes one with a small range and the other with a large range. One can analyze if the backfitting algorithm is able to identify the two spatial processes $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$.

Appendix A

Appendix

A.1 R Code

In this section we list some R codes which are used for simulations or other purposes.

R-Code A.1. Model setup for a linear mixed-effect model

```
n <- 400 # sample size

# predictor variables
{ x1 <- rnorm(n)
  x2 <- rnorm(n)
  x3 <- rnorm(n)
  x4 <- rnorm(n)
  x5 <- rnorm(n)
  x6 <- rnorm(n) }

truebeta <- c(0.7, -1.2, 0.5, 3.1, -2.4, 1.1, 0.8)
p <- length(truebeta) # dimension of fixed effect

# design matrix for the fixed effects
X <- as.matrix(cbind(1, x1, x2, x3, x4, x5, x6))

# 20 level random effect
q <- 20 # dimension of random effect
fac <- as.factor(rep(1:q, n/q))
sigma2_u <- 4 # true variance for the random effect
Z <- model.matrix(~ fac - 1)
true_u <- rnorm(q, mean = 0, sd = sqrt(sigma2_u))

# observations
sigma2 <- 0.7 # true error variance
y <- X %*% truebeta + Z %*% true_u + rnorm(n, mean = 0, sd = sqrt(sigma2))
```

```r
# matrices Sigma_u, Sigma and V
Sigma_u <- diag(rep(sigma2_u, q))
Sigma <- diag(rep(sigma2, n))
V <- sigma2_u * Z %*% t(Z) + Sigma # = Z %*% Sigma_u %*% t(Z) + Sigma
```

R-Code A.2. Frequentist backfitting algorithm for linear mixed models with known variances based on Algorithm 3.1

```r
# maximum number of iterations allowed
itermax <- 20

# hat{beta} and hat{u}
beta_hat <- matrix(rep(0, p), nrow = p, ncol = itermax)
u_hat <- matrix(rep(0, q), nrow = q, ncol = itermax)

# parameters for convergence
freq_eps <- 10^(-10)
conv <- 0
t <- 1

while (!conv) {
  t <- t + 1

  beta_hat[,t] <- solve(t(X) %*% X, (t(X) %*% (y - Z %*% u_hat[,t-1])))

  u_hat[,t] <- Sigma_u %*% t(Z) %*% solve(V, (y - X %*% beta_hat[,t]))

  # check convergence
  maxerr <- max(sum((beta_hat[,t]-beta_hat[,t-1])^2) / sum(beta_hat[,t-1]^2),
                sum((u_hat[,t]-u_hat[,t-1])^2) / sum(u_hat[,t-1]^2))
  conv <- maxerr < freq_eps | t >= itermax
}

# number of iterations
iter <- t-1

freq_beta <- beta_hat[,t] # estimated betas
freq_u <- u_hat[,t] # predictions of the random effect

# fitted values from the backfitting algorithm
bf_fitted_values <- X %*% freq_beta + Z %*% freq_u
```

R-Code A.3. Bayesian backfitting algorithm for linear mixed models with known variances based on Algorithm 3.2

```
# initializations
burnin <- 50
n_iter <- 200
n_total <- burnin + n_iter

# hat{beta} and hat{Zu}
beta_hat <- matrix(rep(0, p), nrow = p, ncol = n_total)
Zu_hat <- matrix(rep(0, n), nrow = n, ncol = n_total) # we need to estimate
                                                      # Zu instead of only u

# parameters for convergence
Bayesian_eps <- 10^(-8)
conv <- 0
t <- 1

while (!conv) {
  t <- t + 1

  z <- rnorm(n)
  beta_hat[,t] <- solve(t(X) %*% X, (t(X) %*% (y - Zu_hat[,t-1]))) +
    sqrt(sigma2) * solve(t(X) %*% X, (t(X) %*% z))

  Zu_hat[,t] <- Z %*% Sigma_u %*% t(Z) %*% solve(V, (y - X %*% beta_hat[,t]))
    + sqrt(sigma2) * generateSqrtSz(S = Z %*% Sigma_u %*% t(Z) %*% solve(V))

  # check convergence
  maxerr <- max(sum((beta_hat[,t]-beta_hat[,t-1])^2) / sum(beta_hat[,t-1]^2),
                sum((Zu_hat[,t]-Zu_hat[,t-1])^2) / sum(Zu_hat[,t-1]^2))
  conv <- maxerr < Bayesian_eps | t >= n_total
}

# compute the posterior means (after burn-in)
beta_mean <- apply(beta_hat[,-(1:burnin)], 1, mean)
Zu_mean <- apply(Zu_hat[,-(1:burnin)], 1, mean)

# predictions of the random effect (= posterior means)
u_mean <- solve(t(Z) %*% Z) %*% t(Z) %*% Zu_mean

# fitted values from the Bayesian backfitting algorithm
Bayesian_bf_fitted_values <- X %*% beta_mean + Zu_mean
```

R-Code A.4. Frequentist backfitting algorithm for linear mixed models with unknown variances based on Algorithm 3.1

```
# maximum number of iterations allowed
itermax <- 500

# hat{beta}, hat{u}, hat{sigma2} and hat{sigma2_u}
beta_hat <- matrix(rep(0, p), nrow = p, ncol = itermax)
u_hat <- matrix(rep(0, q), nrow = q, ncol = itermax)
sigma2_hat <- rep(0, itermax)
sigma2_u_hat <- rep(0, itermax)

# parameters for convergence
freq_eps <- 10^(-8)
conv <- 0
t <- 1

# initial guess for hat{u^(0)}
u_hat[,1] <- rep(1, q)

while (!conv) {
  t <- t + 1

  beta_hat[,t] <- solve(t(X) %*% X, (t(X) %*% (y - Z %*% u_hat[,t-1])))

  # estimate hat{sigma2} and hat{sigma2_u}
  sigma2_hat[t] <- 1/(n-p) * sum((y - X %*% beta_hat[,t] -
                                    Z %*% u_hat[,t-1])^2)
  sigma2_u_hat[t] <- 1/q * t(u_hat[,t-1]) %*% u_hat[,t-1]

  # matrices Sigma_u^{(t)}, Sigma^{(t)} and V^{(t)}
  Sigma_u <- diag(sigma2_u_hat[t], nrow = q)
  Sigma <- diag(sigma2_hat[t], nrow = n)
  V <- sigma2_u_hat[t] * Z %*% t(Z) + Sigma # = Z %*% Sigma_u %*% t(Z) + Sigma

  u_hat[,t] <- Sigma_u %*% t(Z) %*% solve(V, (y - X %*% beta_hat[,t]))

  # check convergence
  maxerr <- max(sum((beta_hat[,t]-beta_hat[,t-1])^2) / sum(beta_hat[,t-1]^2),
                sum((u_hat[,t]-u_hat[,t-1])^2) / sum(u_hat[,t-1]^2))
  conv <- maxerr < freq_eps | t >= itermax
}

# number of iterations
iter <- t-1
```

```r
freq_beta <- beta_hat[,t] # estimated betas
freq_u <- u_hat[,t] # predictions of the random effects
freq_sigma2 <- sigma2_hat[t] # estimated sigma^2
freq_sigma2_u <- sigma2_u_hat[t] # estimated sigma_u^2

# fitted values from the backfitting algorithm
bf_fitted_values <- X %*% freq_beta + Z %*% freq_u
```

R-Code A.5. Bayesian backfitting algorithm for linear mixed models with unknown variances based on Algorithm 3.2

```r
# initializations
burnin <- 50
n_iter <- 200
n_total <- burnin + n_iter

# hat{beta}, hat{Zu}, hat{u}, hat{sigma2} and hat{sigma2_u}
beta_hat <- matrix(rep(0, p), nrow = p, ncol = n_total)
Zu_hat <- matrix(rep(0, n), nrow = n, ncol = n_total)
u_hat <- matrix(rep(0, q), nrow = q, ncol = n_total)
sigma2_hat <- rep(0, n_total)
sigma2_u_hat <- rep(0, n_total)

# parameters for convergence
Bayesian_eps <- 10^(-8)
conv <- 0
t <- 1

# initial guess for hat{u^(0)}
Zu_hat[,1] <- rep(1, n)

while (!conv) {
  t <- t + 1

  z <- rnorm(n)
  beta_hat[,t] <- solve(t(X) %*% X, (t(X) %*% (y - Zu_hat[,t-1]))) +
    sqrt(sigma2_hat[t-1]) * solve(t(X) %*% X, (t(X) %*% z))

  # estimate hat{sigma2} and hat{sigma2_u}
  sigma2_hat[t] <- 1/(n-p) * sum((y - X %*% beta_hat[,t] - Zu_hat[,t-1])^2)
  u_hat[,t-1] <- solve(t(Z) %*% Z) %*% t(Z) %*% Zu_hat[,t-1]
  sigma2_u_hat[t] <- 1/q * t(u_hat[,t-1]) %*% u_hat[,t-1]

  # matrices Sigma_u, Sigma and V
  Sigma_u <- diag(sigma2_u_hat[t], nrow = q)
```

```
  Sigma <- diag(sigma2_hat[t], nrow = n)
  V <- sigma2_u_hat[t] * Z %*% t(Z) + Sigma # = Z %*% Sigma_u %*% t(Z) + Sigma

  Zu_hat[,t] <- Z %*% Sigma_u %*% t(Z) %*% solve(V, (y - X %*% beta_hat[,t]))
    + sqrt(sigma2_hat[t]) *
        generateSqrtSz(S = Z %*% Sigma_u %*% t(Z) %*% solve(V))

  # check convergence
  maxerr <- max(sum((beta_hat[,t]-beta_hat[,t-1])^2) / sum(beta_hat[,t-1]^2),
                sum((Zu_hat[,t]-Zu_hat[,t-1])^2) / sum(Zu_hat[,t-1]^2))
  conv <- maxerr < Bayesian_eps | t >= n_total
}

# compute the posterior means (after burn-in)
beta_mean <- apply(beta_hat[,-(1:burnin)], 1, mean)
Zu_mean <- apply(Zu_hat[,-(1:burnin)], 1, mean)

# predictions of the random effect (= posterior means)
u_mean <- solve(t(Z) %*% Z) %*% t(Z) %*% Zu_mean

# compute the posterior means of the variances
sigma2_mean <- mean(sigma2_hat[-(1:burnin)])
sigma2_u_mean <- mean(sigma2_u_hat[-(1:burnin)])

# fitted values from the Bayesian backfitting algorithm
Bayesian_bf_fitted_values <- X %*% beta_mean + Zu_mean
```

R-Code A.6. Model setup for a spatial model

```
library(spam)

# true parameters for beta
truebeta <- c(1, 2, 0.2) # beta = (intercept, linear in x, linear in y)
p <- length(truebeta)

# true covariance parameters
truetheta <- c(0.05, 0.01, 1) # range, sill, smoothness
sigma2 <- 0.1 # nugget

# locations (= grid)
l <- 40
x <- seq(0, 1, length.out = l)
locs <- expand.grid(x, x)
n <- dim(locs)[1] # = number of points
```

```r
# design matrix
X <- as.matrix(cbind(1, locs))

# distance matrix
distmat <- nearest.dist(locs, delta = 1.5, upper = NULL)

# true covariance matrices
Sigma_h <- cov.mat(distmat, truetheta)
Sigma <- as.spam(diag(sigma2, nrow = n, ncol = n))

V <- Sigma_h + Sigma

# sample
set.seed(15)

true_h <- c(rmvnorm.spam(1, rep.int(0, dim(Sigma_h)[1]), Sigma_h))
true_eps <- c(rmvnorm.spam(1, rep.int(0, dim(Sigma)[1]), Sigma))

y <- X %*% truebeta + true_h + true_eps
```

R-Code A.7. Frequentist backfitting algorithm for spatial models with known covariance parameters based on Algorithm 4.1

```r
# precompute some matrices which are used in the backfitting algorithm
V_inv <- solve.spam(V)
S_beta <- solve(t(X) %*% X, t(X))
S_h <- Sigma_h %*% V_inv

# maximum number of iterations allowed
itermax <- 1000

# initialize hat{beta} and hat{h}
beta_hat <- matrix(rep(0, p), nrow = p, ncol = itermax)
h_hat <- matrix(rep(0, n), nrow = n, ncol = itermax)

# parameters for convergence
freq_eps <- 10^(-8)
conv <- 0
t <- 1

while (!conv) {
  t <- t + 1

  beta_hat[,t] <- S_beta %*% (y - h_hat[,t-1])
```

```
  h_hat[,t] <- S_h %*% (y - X %*% beta_hat[,t])

  # check convergence
  maxerr <- max(sum((beta_hat[,t]-beta_hat[,t-1])^2) / sum(beta_hat[,t-1]^2),
                sum((h_hat[,t]-h_hat[,t-1])^2) / sum(h_hat[,t-1]^2))
  conv <- maxerr < freq_eps | t >= itermax
}

# number of iterations
iter <- t-1

# final estimates
freq_beta <- beta_hat[,t]
freq_h <- h_hat[,t]

# fitted values from the backfitting algorithm
bf_fitted_values <- X %*% freq_beta + freq_h
```

R-Code A.8. Bayesian backfitting algorithm for spatial models with known covariance parameters based on Algorithm 4.2

```
# precompute some matrices which are used in the backfitting algorithm
V_inv <- solve.spam(V)
S_beta <- solve(t(X) %*% X, t(X))
S_h <- Sigma_h %*% V_inv

# initializations
burnin <- 250
n_iter <- 1000
n_total <- burnin + n_iter

# initialize hat{beta} and hat{h}
beta_hat <- matrix(rep(0, p), nrow = p, ncol = n_total)
h_hat <- matrix(rep(0, n), nrow = n, ncol = n_total)

# parameters for convergence
Bayesian_eps <- 10^(-8)
conv <- 0
t <- 1

while (!conv) {
  t <- t + 1

  z <- rnorm(n)
  beta_hat[,t] <- S_beta %*% (y - h_hat[,t-1]) + sqrt(sigma2) * S_beta %*% z
```

```
  h_hat[,t] <- S_h %*% (y - X %*% beta_hat[,t]) +
    sqrt(sigma2) * generateSqrtSz(S = S_h)

  # check convergence
  maxerr <- max(sum((beta_hat[,t]-beta_hat[,t-1])^2) / sum(beta_hat[,t-1]^2),
                sum((h_hat[,t]-h_hat[,t-1])^2) / sum(h_hat[,t-1]^2))
  conv <- maxerr < Bayesian_eps | t >= n_total
  Bayesian_maxerr[t] <- maxerr
}

# compute the posterior means (after burn-in)
beta_mean <- apply(beta_hat[,-(1:burnin)], 1, mean)
h_mean <- apply(h_hat[,-(1:burnin)], 1, mean)

# fitted values from the Bayesian backfitting algorithm
Bayesian_bf_fitted_values <- X %*% beta_mean + h_mean
```

When also estimating the covariance parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)^\top$ and σ^2 is makes more sense to gather these quantities in one vector. Since we apply the R function *mle.nomean.spam()* which calls the *optim()* function, we need to commit also an upper and lower bound for the optimization. Therefore, we use the variables in the following R code additionally to the model setup given in R-Code A.6.

```
# true covariance parameters
truetheta <- c(0.05, 0.01, 1, 0.1) # range, sill, smoothness, nugget

thetalower = c(0.001, 0.001, 0.01, 0.001)
thetaupper = c(0.1, 0.1, 2, 0.5)
```

R-Code A.9. Frequentist backfitting algorithm for spatial models with unknown covariance parameters based on Algorithm 4.1

```
# maximum number of iterations allowed
itermax <- 1000

# initialize hat{beta} and hat{h}
beta_hat <- matrix(rep(0, p), nrow = p, ncol = itermax)
h_hat <- matrix(rep(0, n), nrow = n, ncol = itermax)

# initialize the covariance parameters
theta_hat <- matrix(rep(0, 4), nrow = 4, ncol = itermax)

# parameters for convergence
freq_eps <- 10^(-8)
```

```
conv <- 0
t <- 1

# initial guess for theta_hat[,1] and h_hat[,1]
theta_hat[,1] <- truetheta
h_hat[,1] <- true_h

while (!conv) {
  t <- t + 1

  beta_hat[,t] <- solve(t(X) %*% X, (t(X) %*% (y - h_hat[,t-1])))

  # estimate the covariance parameters
  res <- mle.nomean.spam(y - X %*% beta_hat[,t], distmat, cov.mat,
                         theta_hat[,t-1],
                         thetalower = thetalower, thetaupper = thetaupper)
  theta_hat[,t] <- res$par

  # matrices Sigma_h, Sigma and V
  Sigma_h <- cov.mat(distmat, theta_hat[1:3,t])
  Sigma <- diag(theta_hat[4,t], nrow = n)
  V <- Sigma_h + Sigma

  h_hat[,t] <- Sigma_h %*% solve(V, (y - X %*% beta_hat[,t]))

  # check convergence
  maxerr <- max(sum((beta_hat[,t]-beta_hat[,t-1])^2) / sum(beta_hat[,t-1]^2),
                sum((h_hat[,t]-h_hat[,t-1])^2) / sum(h_hat[,t-1]^2))
  conv <- maxerr < freq_eps | t >= itermax
}

# number of iterations
iter <- t-1

# final estimates
freq_beta <- beta_hat[,t]
freq_h <- h_hat[,t]
freq_theta <- theta_hat[,t]

# fitted values from the backfitting algorithm
bf_fitted_values <- X %*% freq_beta + freq_h
```

R-Code A.10. Bayesian backfitting algorithm for spatial models with unknown covariance parameters based on Algorithm 4.2

```
# initializations
burnin <- 10
n_iter <- 40
n_total <- burnin + n_iter

# initialize hat{beta} and hat{h}
beta_hat <- matrix(rep(0, p), nrow = p, ncol = n_total)
h_hat <- matrix(rep(0, n), nrow = n, ncol = n_total)

# initialize the covariance parameters
theta_hat <- matrix(rep(0, 4), nrow = 4, ncol = n_total)

# parameters for convergence
Bayesian_eps <- 10^(-8)
conv <- 0
t <- 1

# initial guess for theta_hat[,1] and h_hat[,1]
theta_hat[,1] <- truetheta
h_hat[,1] <- true_h

while (!conv) {
  t <- t + 1

  z <- rnorm(n)
  beta_hat[,t] <- solve(t(X) %*% X, (t(X) %*% (y - h_hat[,t-1]))) +
    sqrt(theta_hat[4,t-1]) * solve(t(X) %*% X, t(X) %*% z)

  # estimate the covariance parameters
  res <- mle.nomean.spam(y - X %*% beta_hat[,t], distmat, cov.mat,
                         theta_hat[,t-1],
                         thetalower = thetalower, thetaupper = thetaupper)
  theta_hat[,t] <- res$par

  # matrices Sigma_h, Sigma and V
  Sigma_h <- cov.mat(distmat, theta_hat[1:3,t])
  Sigma <- diag(theta_hat[4,t], nrow = n)
  V <- Sigma_h + Sigma

  h_hat[,t] <- Sigma_h %*% solve(V, (y - X %*% beta_hat[,t])) +
    sqrt(theta_hat[4,t]) * generateSqrtSz(S = Sigma_h %*% solve(V))

  # check convergence
```

```
  maxerr <- max(sum((beta_hat[,t]-beta_hat[,t-1])^2) / sum(beta_hat[,t-1]^2),
                sum((h_hat[,t]-h_hat[,t-1])^2) / sum(h_hat[,t-1]^2))
  conv <- maxerr < Bayesian_eps | t >= n_total
}

# compute the posterior means (after burnin)
beta_mean <- apply(beta_hat[,(burnin+1):n_total], 1, mean)
h_mean <- apply(h_hat[,(burnin+1):n_total], 1, mean)
theta_mean <- apply(theta_hat[,(burnin+1):n_total], 1, mean)

# fitted values from the Bayesian backfitting algorithm
Bayesian_bf_fitted_values <- X %*% beta_mean + h_mean
```

If we only want to optimize the sill and the nugget parameter, then we need to change some of the lines in R-Codes A.9 and A.10. Instead of calling the function *mle.nomean.spam()* in the algorithm we use the following lines of code.

```
# estimate the covariance parameters
Sigma_new <- cov.mat(distmat, theta_hat[,t-1])
Rstruct_new <- chol.spam(Sigma_new)

y_res <- y - X %*% beta_hat[,t]
neg2loglikelihood_new <- function(theta) {
  range <- truetheta[1] # fix range
  sill <- theta[1]
  smoothness <- truetheta[3] # fix smoothness
  nugget <- theta[2]
  new_theta <- c(range, sill, smoothness, nugget)
  Sigma <- do.call(cov.mat, list(distmat, new_theta))
  cholS <- chol.spam(Rstruct = Rstruct_new, Sigma)

  return( n * log(2*pi) +
            2*c(determinant.spam.chol.NgPeyton(cholS)$modulus) +
            sum(y_res * solve.spam( cholS, y_res))
  )
}

res <- optim(theta_hat[,t-1][c(2, 4)], neg2loglikelihood_new,
             method = "L-BFGS-B", lower = thetalower[c(2, 4)],
             upper = thetaupper[c(2, 4)], hessian = TRUE)
theta_hat[,t] <- c(truetheta[1], res$par[1], truetheta[3], res$par[2])
```

Bibliography

Abramowitz, M. and Stegun, I. A. (1970). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover. 46

Bates, D., Mächler, M., Bolker, B., and Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, **67**, 1–48. v, 40, 71

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. Wiley Series in Probability and Mathematical Statistics. 45

Fox, J. and Weisberg, S. (2019). *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, third edition. 9, 15

Furrer, R. (2019). Lecture notes in Modeling Dependent Data. Department of Mathematics, University of Zurich. Available at `www.math.uzh.ch/furrer/download/sta330`. 45

Furrer, R., Genton, M. G., and Nychka, D. (2006). Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, **15**, 502–523. 73

Furrer, R. and Sain, S. R. (2009). Spatial model fitting for large datasets with applications to climate and microarray problems. *Statistics and Computing*, **19**, 113–128. v, 6, 48, 49

Furrer, R. and Sain, S. R. (2010). spam: A sparse matrix R package with emphasis on MCMC methods for Gaussian Markov random fields. *Journal of Statistical Software*, **36**, 1–25. v, 66

Gamerman, D. and Lopes, H. F. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman & Hall/CRC, second edition. 11

Hastie, T. and Tibshirani, R. (1986). Generalized additive models. *Statistical Science*, **1**, 297–310. v, 3

Hastie, T. and Tibshirani, R. (2000). Bayesian backfitting. *Statistical Science*, **15**, 196–213. v, 6, 11, 12, 13, 14

Held, L. and Bové, D. S. (2013). *Applied Statistical Inference: Likelihood and Bayes*. Springer Publishing Company, Incorporated. 17

Henderson, C. R. (1975). Best linear unbiased estimation and prediction under a selection model. *Biometrics*, **31**, 423–447. 24

Link, W. A. and Eaton, M. J. (2012). On thinning of chains in MCMC. *Methods in Ecology and Evolution*, **3**, 112–115. 43

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. v

Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields: Theory and Applications*. Chapman & Hall/CRC. 25

Shynk, J. J. (2012). *Probability, Random Variables, and Random Processes: Theory and Signal Processing Applications*. John Wiley & Sons. 12

Soetaert, K. (2019). *plot3D: Plotting Multi-Dimensional Data*. R package version 1.3. v, 56

Tierney, L. (1994). Markov Chains for Exploring Posterior Distributions. *The Annals of Statistics*, **22**, 1701–1728. 13