Bayesian Ridge Regression - Final Report

Dominik Strache, Nicolai Bäuerle & Joel Beck

Contents

Introduction

1 Mathematical Model

This chapter focuses on the underlying theoretical model of Bayesian Ridge Regression in the context of the Gaussian Location-Scale Regression model considered in the `lmls` package. First, the distributional assumptions for the parameter vectors $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ as well as the scalar quantities τ^2 and ξ^2 are clearly stated. Based on these prior distributions, the full conditional distributions of each parameter given all of the remaining model components are derived. Throughout this rather theoretical exposition we will build connections from the derived equations to practical consequences that have to be considered in the code implementation discussed in sections 2.2.1 and 2.2.2.

1.1 Prior Distributions

Assuming conditional independence among the regression coefficients $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ as well as flat priors for the intercept parameters

$$f(\beta_0) \propto const \qquad \text{and} \qquad f(\gamma_0) \propto const,$$

first note that

$$\begin{aligned} f(\boldsymbol{\beta} \mid \tau^2) &= f(\beta_0) f(\tilde{\boldsymbol{\beta}} \mid \tau^2) \propto f(\tilde{\boldsymbol{\beta}} \mid \tau^2) \quad \text{and} \\ f(\boldsymbol{\gamma} \mid \xi^2) &= f(\gamma_0) f(\tilde{\boldsymbol{\gamma}} \mid \xi^2) \propto f(\tilde{\boldsymbol{\gamma}} \mid \xi^2), \end{aligned}$$

where the notation $\boldsymbol{\beta} = (\beta_0, ..., \beta_K) \in \mathbb{R}^{K+1}$, $\tilde{\boldsymbol{\beta}} = (\beta_1, ..., \beta_K) \in \mathbb{R}^K$ and, analogously, $\boldsymbol{\gamma} = (\gamma_0, ..., \gamma_J) \in \mathbb{R}^{J+1}$, $\tilde{\boldsymbol{\gamma}} = (\gamma_1, ..., \gamma_J) \in \mathbb{R}^J$ is used.

Thus, the Prior distributions of the parameters in the Bayesian Ridge Regression model are given by

- $\tilde{\boldsymbol{\beta}} \mid \tau^2 \sim \mathcal{N}\left(\mathbf{0},\, \tau^2 \cdot \mathbf{I}_K\right)$,
- $\tilde{\boldsymbol{\gamma}} \mid \xi^2 \sim \mathcal{N}\left(\mathbf{0},\, \xi^2 \cdot \mathbf{I}_J\right)$,
- $\tau^2 \sim IG(a_\tau,\, b_\tau)$, with fixed hyperparameters a_τ and b_τ,
- $\xi^2 \sim IG(a_\xi,\, b_\xi)$, with fixed hyperparameters a_ξ and b_ξ.

1.2 Full Posterior Distribution

The starting point for deriving the Full Conditional distributions, which will majorly impact the implementation of the Metropolis-Hastings sampling process, is always the Full Posterior distribution. As usual in the Bayesian literature, we write the Full Posterior $f(\boldsymbol{\beta}, \boldsymbol{\gamma}, \tau^2, \xi^2 \mid \mathbf{y})$ in terms of the Likelihood function / observation model $f(\mathbf{y} \mid \boldsymbol{\beta}, \boldsymbol{\gamma}, \tau^2, \xi^2)$ and the *joint* Prior distribution $f(\boldsymbol{\beta}, \boldsymbol{\gamma}, \tau^2, \xi^2)$, i.e.

$$f(\boldsymbol{\beta}, \boldsymbol{\gamma}, \tau^2, \xi^2 \mid \mathbf{y}) \propto f(\mathbf{y} \mid \boldsymbol{\beta}, \boldsymbol{\gamma}, \tau^2, \xi^2) \cdot f(\boldsymbol{\beta}, \boldsymbol{\gamma}, \tau^2, \xi^2).$$

The specific form of the Likelihood function is given by the Location-Scale Regression model that the `lmls` package is built upon:

$$y_i \mid \boldsymbol{\beta}, \boldsymbol{\gamma}, \tau^2, \xi^2 = y_i \mid \boldsymbol{\beta}, \boldsymbol{\gamma} \sim \mathcal{N}\left(\mathbf{x}_i^T \boldsymbol{\beta},\, \exp\left(\mathbf{z}_i^T \boldsymbol{\gamma}\right)^2\right).$$

Taking the independence structure into account, the joint Prior distribution can be written as

$$\begin{aligned} f(\boldsymbol{\beta}, \boldsymbol{\gamma}, \tau^2, \xi^2) &= f(\boldsymbol{\beta} \mid \boldsymbol{\gamma}, \tau^2, \xi^2) \cdot f(\boldsymbol{\gamma}, \tau^2, \xi^2) \\ &= f(\boldsymbol{\beta} \mid \boldsymbol{\gamma}, \tau^2, \xi^2) \cdot f(\boldsymbol{\gamma} \mid \tau^2, \xi^2) \cdot f(\tau^2, \xi^2) \\ &= f(\boldsymbol{\beta} \mid \tau^2) \cdot f(\boldsymbol{\gamma} \mid \xi^2) \cdot f(\tau^2) \cdot f(\xi^2). \end{aligned}$$

Combining these results yields for the Full Posterior distribution the general form

$$\begin{aligned} f(\boldsymbol{\beta}, \boldsymbol{\gamma}, \tau^2, \xi^2 \mid \mathbf{y}) &\propto f(\mathbf{y} \mid \boldsymbol{\beta}, \boldsymbol{\gamma}, \tau^2, \xi^2) \cdot f(\boldsymbol{\beta} \mid \tau^2) \cdot f(\boldsymbol{\gamma} \mid \xi^2) \cdot f(\tau^2) \cdot f(\xi^2) \\ &\propto f(\mathbf{y} \mid \boldsymbol{\beta}, \boldsymbol{\gamma}) \cdot f(\tilde{\boldsymbol{\beta}} \mid \tau^2) \cdot f(\tilde{\boldsymbol{\gamma}} \mid \xi^2) \cdot f(\tau^2) \cdot f(\xi^2), \end{aligned}$$

in which the corresponding densities for the observation model and the individual prior distributions can be inserted. These are given by

- $f(\mathbf{y} \mid \boldsymbol{\beta}, \boldsymbol{\gamma}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi \exp\left(\mathbf{z}_i^T \boldsymbol{\gamma}\right)^2}} \cdot \exp\left(-\frac{1}{2\exp\left(\mathbf{z}_i^T \boldsymbol{\gamma}\right)^2} \cdot \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta}\right)^2\right)$,
- $f(\boldsymbol{\beta} \mid \tau^2) \propto \prod_{k=1}^{K} \frac{1}{\sqrt{2\pi\tau^2}} \cdot \exp\left(-\frac{1}{2\tau^2} \cdot \beta_k^2\right) = (2\pi)^{-\frac{K}{2}} \tau^{-K} \exp\left(-\frac{1}{2\tau^2} \cdot \tilde{\boldsymbol{\beta}}^T \tilde{\boldsymbol{\beta}}\right)$,
- $f(\boldsymbol{\gamma} \mid \xi^2) \propto \prod_{j=1}^{J} \frac{1}{\sqrt{2\pi\xi^2}} \cdot \exp\left(-\frac{1}{2\xi^2} \cdot \gamma_j^2\right) = (2\pi)^{-\frac{J}{2}} \xi^{-J} \exp\left(-\frac{1}{2\xi^2} \cdot \tilde{\boldsymbol{\gamma}}^T \tilde{\boldsymbol{\gamma}}\right)$,
- $f(\tau^2) = \frac{b_\tau}{\Gamma(a_\tau)} \left(\frac{1}{\tau^2}\right)^{a_\tau + 1} \exp\left(-\frac{b_\tau}{\tau^2}\right)$,
- $f(\xi^2) = \frac{b_\xi}{\Gamma(a_\xi)} \left(\frac{1}{\xi^2}\right)^{a_\xi + 1} \exp\left(-\frac{b_\xi}{\xi^2}\right)$,

leading to the Full Posterior

$$\begin{aligned} f(\boldsymbol{\beta}, \boldsymbol{\gamma}, \tau^2, \xi^2 \mid \mathbf{y}) \propto \prod_{i=1}^{n} \frac{1}{\exp\left(\mathbf{z}_i^T \boldsymbol{\gamma}\right)} \cdot \exp\left(-\frac{1}{2\exp\left(\mathbf{z}_i^T \boldsymbol{\gamma}\right)^2} \cdot \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta}\right)^2\right) \\ \cdot \tau^{-K} \exp\left(-\frac{1}{2\tau^2} \tilde{\boldsymbol{\beta}}^T \tilde{\boldsymbol{\beta}}\right) \cdot \xi^{-J} \exp\left(-\frac{1}{2\xi^2} \tilde{\boldsymbol{\gamma}}^T \tilde{\boldsymbol{\gamma}}\right) \\ \cdot \left(\frac{1}{\tau^2}\right)^{a_\tau + 1} \exp\left(-\frac{b_\tau}{\tau^2}\right) \cdot \left(\frac{1}{\xi^2}\right)^{a_\xi + 1} \exp\left(-\frac{b_\xi}{\xi^2}\right). \end{aligned}$$

1.3 Full Conditional Distributions

The Full Posterior distribution contains complete information about the statistical model. For our purposes, we are mostly interested in the Full Conditional Distribution of each model parameter. These can be obtained by simply neglecting all factors of the Full Posterior that do not depend on the parameter in consideration.

The Full Conditional distribution can then be recovered by the resulting density kernel, either by recognizing a known distribution or by adding a normalization constant (which, however, is not needed for Markov Chain Monte Carlo sampling).

1.3.1 Full Conditional of τ^2:

$$\begin{aligned} f(\tau^2 \mid \cdot) &\propto \tau^{-K} \cdot \exp\left(-\frac{1}{2\tau^2} \cdot \tilde{\boldsymbol{\beta}}^T \tilde{\boldsymbol{\beta}}\right) \cdot \left(\frac{1}{\tau^2}\right)^{a_\tau + 1} \exp\left(-\frac{b_\tau}{\tau^2}\right) \\ &\propto \left(\frac{1}{\tau^2}\right)^{a_\tau + \frac{K}{2} + 1} \cdot \exp\left(-\frac{1}{\tau^2}\left(b_\tau + \frac{1}{2}\tilde{\boldsymbol{\beta}}^T \tilde{\boldsymbol{\beta}}\right)\right). \end{aligned}$$

This is the kernel of an Inverse Gamma distribution parameterized by

$$\tau^2 \mid \cdot \sim IG(a_\tau + \frac{K}{2},\, b_\tau + \frac{1}{2}\tilde{\boldsymbol{\beta}}^T \tilde{\boldsymbol{\beta}}).$$

1.3.2 Full Conditional of ξ^2:

$$\begin{aligned} f(\xi^2 \mid \cdot) &\propto \xi^{-J} \cdot \exp\left(-\frac{1}{2\xi^2} \cdot \tilde{\boldsymbol{\gamma}}^T \tilde{\boldsymbol{\gamma}}\right) \cdot \left(\frac{1}{\xi^2}\right)^{a_\xi + 1} \exp\left(-\frac{b_\xi}{\xi^2}\right) \\ &\propto \left(\frac{1}{\xi^2}\right)^{a_\xi + \frac{J}{2} + 1} \cdot \exp\left(-\frac{1}{\xi^2}\left(b_\xi + \frac{1}{2}\tilde{\boldsymbol{\gamma}}^T \tilde{\boldsymbol{\gamma}}\right)\right). \end{aligned}$$

Thus, the Full Conditional of ξ^2 follows an Inverse Gamma distribution as well:

$$\xi^2 \mid \cdot \sim IG(a_\xi + \frac{J}{2},\, b_\xi + \frac{1}{2}\tilde{\boldsymbol{\gamma}}^T \tilde{\boldsymbol{\gamma}}).$$

1.3.3 Full Conditional of $\boldsymbol{\beta}$:

Here, the derivation is more involved. In order to keep the calculations structured, we introduce the following notation:

$$\mathbf{w}_i := \frac{\mathbf{x}_i}{\exp\left(\mathbf{z}_i^T\boldsymbol{\gamma}\right)} \in \mathbb{R}^{K+1}, \qquad \mathbf{W} := \begin{pmatrix} \mathbf{w}_1^T \\ \vdots \\ \mathbf{w}_n^T \end{pmatrix} \in \mathbb{R}^{n\times(K+1)}$$

and

$$u_i := \frac{y_i}{\exp\left(\mathbf{z}_i^T\boldsymbol{\gamma}\right)} \in \mathbb{R}, \qquad \mathbf{u} := \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \in \mathbb{R}^n,$$

yielding $\sum_{i=1}^n u_i\mathbf{w}_i = \mathbf{W}^T\mathbf{u}$ and $\sum_{i=1}^n \mathbf{w}_i\mathbf{w}_i^T = \mathbf{W}^T\mathbf{W}$.

Therefore the Full Conditional distribution of $\boldsymbol{\beta}$ can be written as

$$\begin{aligned}
f(\boldsymbol{\beta}\mid\cdot) &\propto \exp\left(-\frac{1}{2}\cdot\left[\frac{1}{\tau^2}\tilde{\boldsymbol{\beta}}^T\tilde{\boldsymbol{\beta}} + \sum_{i=1}^n \frac{1}{\exp\left(\mathbf{z}_i^T\boldsymbol{\gamma}\right)^2}\left(y_i - \mathbf{x}_i^T\boldsymbol{\beta}\right)^2\right]\right) \\
&= \exp\left(-\frac{1}{2}\cdot\left[\frac{1}{\tau^2}\tilde{\boldsymbol{\beta}}^T\tilde{\boldsymbol{\beta}} + \sum_{i=1}^n\left(\frac{y_i^2}{\exp\left(\mathbf{z}_i^T\boldsymbol{\gamma}\right)^2} - \frac{2y_i\mathbf{x}_i^T}{\exp\left(\mathbf{z}_i^T\boldsymbol{\gamma}\right)^2}\boldsymbol{\beta} + \boldsymbol{\beta}^T\frac{\mathbf{x}_i}{\exp\left(\mathbf{z}_i^T\boldsymbol{\gamma}\right)}\frac{\mathbf{x}_i^T}{\exp\left(\mathbf{z}_i^T\boldsymbol{\gamma}\right)}\boldsymbol{\beta}\right)\right]\right) \\
&\propto \exp\left(-\frac{1}{2}\cdot\left[\frac{1}{\tau^2}\tilde{\boldsymbol{\beta}}^T\tilde{\boldsymbol{\beta}} - 2\cdot\sum_{i=1}^n u_i\mathbf{w}_i^T\boldsymbol{\beta} + \sum_{i=1}^n\boldsymbol{\beta}^T\mathbf{w}_i\mathbf{w}_i^T\boldsymbol{\beta}\right]\right) \\
&= \exp\left(-\frac{1}{2}\cdot\left[\boldsymbol{\beta}^T\left(\sum_{i=1}^n\mathbf{w}_i\mathbf{w}_i^T + \frac{1}{\tau^2}\begin{pmatrix}0 & 0\\ 0 & \mathbf{I}_K\end{pmatrix}\right)\boldsymbol{\beta} - 2\cdot\sum_{i=1}^n\boldsymbol{\beta}^T u_i\mathbf{w}_i\right]\right) \\
&= \exp\left(-\frac{1}{2}\cdot\left[\boldsymbol{\beta}^T\left(\mathbf{W}^T\mathbf{W} + \frac{1}{\tau^2}\begin{pmatrix}0 & 0\\ 0 & \mathbf{I}_K\end{pmatrix}\right)\boldsymbol{\beta} - 2\cdot\boldsymbol{\beta}^T\mathbf{W}^T\mathbf{u}\right]\right).
\end{aligned}$$

Comparing this representation with the kernel of a multivariate normal distribution leads to the conclusion

$$\boldsymbol{\beta}\mid\cdot \sim \mathcal{N}_{K+1}\left(\boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_\beta\right)$$

with the parameters

$$\boldsymbol{\Sigma}_\beta = \left(\mathbf{W}^T\mathbf{W} + \frac{1}{\tau^2}\begin{pmatrix}0 & 0\\ 0 & \mathbf{I}_K\end{pmatrix}\right)^{-1} \qquad \text{and} \qquad \boldsymbol{\mu}_\beta = \boldsymbol{\Sigma}_\beta\mathbf{W}^T\mathbf{u}.$$

1.3.4 Full Conditional of $\boldsymbol{\gamma}$:

Using the notation

$$\mathbf{z}_i \in \mathbb{R}^{J+1}, \qquad \mathbf{Z} = \begin{pmatrix}\mathbf{z}_1^T \\ \vdots \\ \mathbf{z}_n^T\end{pmatrix} \in \mathbb{R}^{n\times(J+1)} \qquad \text{and} \qquad \boldsymbol{\gamma}\in\mathbb{R}^{J+1},$$

the Full Conditional distribution of $\boldsymbol{\gamma}$ is given by

$$\begin{aligned}
f(\boldsymbol{\gamma}\mid\cdot) &\propto \exp\left(-\frac{1}{2}\cdot\left[\frac{1}{\xi^2}\tilde{\boldsymbol{\gamma}}^T\tilde{\boldsymbol{\gamma}} + \sum_{i=1}^n\left(\frac{1}{\exp\left(\mathbf{z}_i^T\boldsymbol{\gamma}\right)^2}\left(y_i - \mathbf{x}_i^T\boldsymbol{\beta}\right)^2 + 2\cdot\mathbf{z}_i^T\boldsymbol{\gamma}\right)\right]\right) \\
&= \exp\left(-\frac{1}{2}\cdot\left[\frac{1}{\xi^2}\tilde{\boldsymbol{\gamma}}^T\tilde{\boldsymbol{\gamma}} + 2\cdot\mathbf{1}_n^T\mathbf{Z}\boldsymbol{\gamma} + \sum_{i=1}^n\left(\frac{y_i - \mathbf{x}_i^T\boldsymbol{\beta}}{\exp\left(\mathbf{z}_i^T\boldsymbol{\gamma}\right)}\right)^2\right]\right).
\end{aligned}$$

In contrast to the Full Conditionals for $\boldsymbol{\beta}$, τ^2 and ξ^2, this kernel cannot be assigned to a known distribution. Thus, for sampling from the Full Posterior distribution, it is not feasible to use a Gibbs Sampler in its purest form. More specifically, we will include a Metropolis Hastings step for sampling the $\boldsymbol{\gamma}$ parameter vector.

Although this 'inconvenience' is not required for $\boldsymbol{\beta}$ (since we can use independent samples from a multivariate normal distribution), we will briefly explore and analyze the convergence properties of the modified version of sampling both $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ via the Metropolis-Hastings procedure.

2 The `asp21bridge` Package

This chapter introduces numerous facets of the `asp21bridge` package.

Section 2.1 explains how to use and combine the functions that are contained in the package most efficiently.

Section 2.2 focuses on the implementation of the Markov Chain Monte Carlo Sampler with Ridge Penalty, links the source code to the mathematical model from chapter 1 and explains how the main function `mcmc_ridge()` as well as the helper functions implementing the Metropolis-Hastings algorithm are structured.

Finally, some additional components of the package development process and ideas, that the package is based upon, are discussed in section 2.3.

2.1 User Guide

This section aims to provide guidance for new users of the `asp21bridge` package. Although all exported functions are fully documented such that function arguments and brief examples can be looked up at the corresponding help page, the following tutorial extends the documentation by illustrating a typical workflow of simulation via the penalized MCMC Sampler, extracting meaningful statistical quantities from the samples and visually analyzing the results.

The `asp21bridge` package inherits all functions from the `lmls` package and exports 9 additional functions, which can be grouped into three categories:

- **Sampling:** The whole sampling process is covered by the very flexible and robust `mcmc_ridge()` function that is explained in detail in section 2.2.1.
- **Graphical Analysis:** The `trace_plot()`, `density_plot()` and `acf_plot()` functions provide the three most common visualizations of a *single* chain's development over time, its distribution and the autocorrelation between the samples. Since all of these are *diagnostic* tools, they are combined in the high-level `diagnostic_plots()` function, which simply collects all three plots in a grid and will be used more often than the separate building blocks.

  For visualizing *multiple* chains together, the `mult_plot()` function can be used. Several arguments for customizing the graphical output exist. In the most basic form, trace plots of all selected chains are displayed in the upper panel while the corresponding density plots are integrated into the lower panel.
- **Statistical Analysis:** Here, the `summary_complete()` function represents the main tool and provides a more thorough statistical summary than the generic `summary()` function. In addition, the `burnin()` and `thinning()` functions are convenient in the context of Markov Chains and in particular for extracting more meaningful features of the samples from the posterior distributions.

2.1.1 Sampling with the `mcmc_ridge()` function

As explained in section 2.2.1, there are two valid input types for simulating with the `mcmc_ridge()` function. Most commonly, the sampling procedure will be built upon an existing model that was initialized by the `lmls()` function. In this case, only the name of the model object is required, although many further arguments can be specified such as the number of simulations `nsim` which is set to 1000 by default.

The `mcmc_ridge()` function can, however, be used completely independent from the underlying `lmls` package. Therefore, the outcome vector $\mathbf{y}$ as well as the design matrices $\mathbf{X}$ and $\mathbf{Z}$, corresponding to the $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ coefficient vectors respectively (as explained in chapter 1), must be manually specified.

In order to illustrate both options, we construct two separate models:

- The first model uses the built-in `toy_data`, a data set which is specifically designed for introductory tutorials, documentation and unit tests. It consists of a column `y` representing a vector of observed values and the explanatory variables `x1`, `x2`, `z1` and `z2`.

The data is simulated according to the correctly specified location-scale regression model from chapter 1, where all explanatory variables predict the mean of $\mathbf{y}$ and only the latter two model the variance. This example will be used for model evaluation, since the true data generating values $\boldsymbol{\beta} = \begin{pmatrix} 0 & -2 & -1 & 1 & 2 \end{pmatrix}^T$ and $\boldsymbol{\gamma} = \begin{pmatrix} 0 & -1 & 1 \end{pmatrix}^T$ are known.

For the simulations, we use the model object that is created by the `lmls()` function as input and use most of the `mcmc_ridge()` default settings, except for the number of simulations, which we increase to 10000.

- The second model uses the more realistic `abdom` data set from the `lmls` package.

  In this case, we have to provide the data input as well as starting values for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ explicitly. Note that the dimension of `beta_start` and `gamma_start` have to match the number of columns in `X` and `Z` in the model *input*. If necessary, the `mcmc_ridge()` functions adds intercept columns to both design matrices, such that the dimension of the coefficient vectors might have increased (as in the case illustrated here) in the model *output*.

  This example differs from the first one with regards to the conclusions that can be drawn: Since the true data generating values of $\boldsymbol{\beta} = \begin{pmatrix} \beta_0 & \beta_1 \end{pmatrix}^T$ and $\boldsymbol{\gamma} = \begin{pmatrix} \gamma_0 & \gamma_1 \end{pmatrix}^T$ are *not* known, we have to rely more heavily on the model estimates. The number of simulations is again increased to 10000 such that statistically valid estimates of posterior characteristics are possible, even if preceding 'burnin' and 'thinning' steps might be required.

```
library(asp21bridge)
set.seed(1234)

toy_fit <- lmls(
  location = y ~ x1 + x2 + z1 + z2, scale = ~ z1 + z2,
  data = toy_data, light = FALSE
) %>%
  mcmc_ridge(num_sim = 10000)

y <- abdom$y
X <- as.matrix(abdom$x)
Z <- as.matrix(abdom$x)

abdom_fit <- mcmc_ridge(
  y = y, X = X, Z = Z, beta_start = 1, gamma_start = 1,
  num_sim = 10000
)
```

The different forms of data input cause different structures in the resulting model objects: `toy_fit` inherits the model structure as well as the S3 Class `lmls` from the `lmls()` function and adds a list entry `mcmc_ridge` containing the sampling results.

In contrast, `abdom_fit` only contains matrices filled with the simulations and the acceptance probability of the Metropolis-Hastings step for sampling $\boldsymbol{\gamma}$:

```
str(abdom_fit, max.level = 1)
## List of 2
##  $ sampling_matrices:List of 4
##  $ acceptance_rate  : num 0.431
```

2.1.2 Graphical Analysis

Most statistical analyses start with an exploratory phase. To obtain a graphical overview of the simulations, the `mult_plot()` function is convenient to display trace plots and/or density plots for all model coefficients.

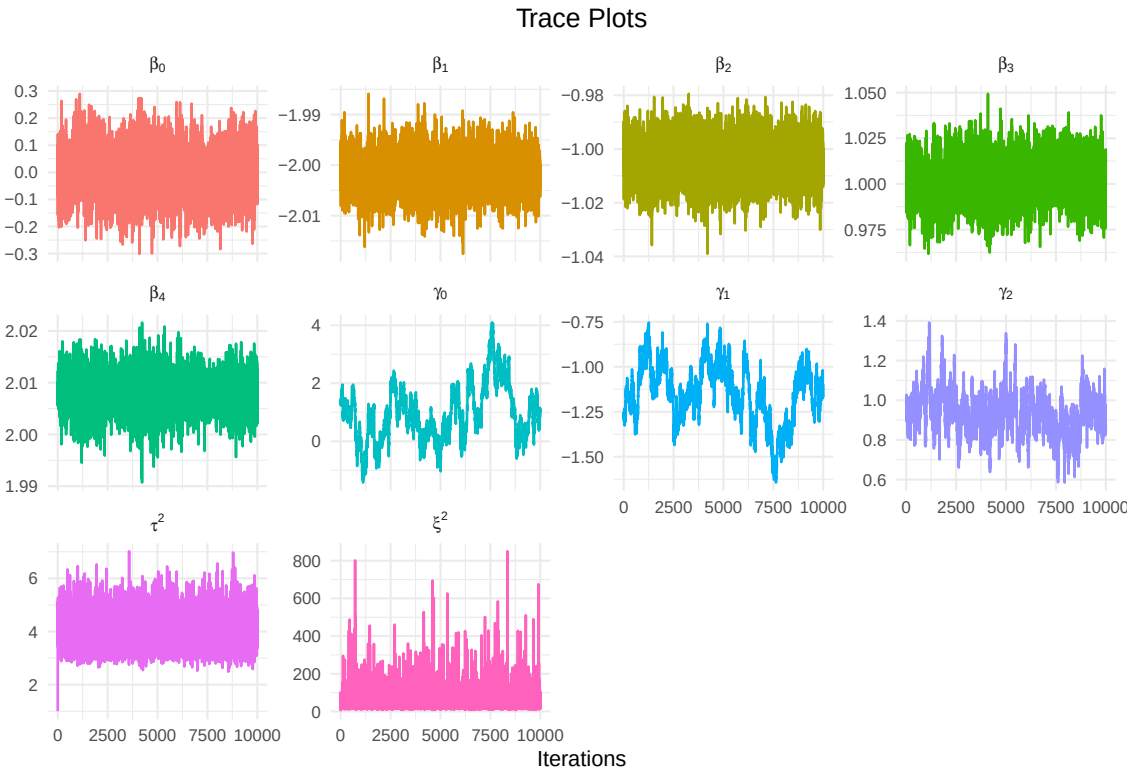

Figure 1: Trace Plots for all Coefficients, data: toy_data

The `free_scale` argument is often useful to obtain a meaningful graphical output, if the parameters are on different numerical scales. Setting `latex = TRUE` transforms the coefficient names to their corresponding greek symbols, which, although being a purely aesthetic feature, required a surprisingly nontrivial implementation.

```
mult_plot(samples = toy_fit, type = "trace", free_scale = TRUE, latex = TRUE)
```

Figure 1 shows trace plots for all coefficients of the `toy_fit` model. Due to the high number of simulations, the plot facets appear a bit overloaded. Considering the y-axis scales, the samples for the $\boldsymbol{\beta}$ vector show a small variance and fast convergence, whereas the samples for γ_0 and γ_1 indicate a significant autocorrelation and no sign of convergence.

The `log` argument can be useful for displaying variance parameters such as τ^2 and ξ^2, which are strictly positive. Figure 2 illustrates the effect of this option on both, the y-axis of the trace plots as well as the x-axis of the density plots.

```
variance_samples <- cbind(
  abdom_fit$sampling_matrices$tau_samples,
  abdom_fit$sampling_matrices$xi_samples
)

mult_plot(
  samples = variance_samples, type = "both", free_scale = TRUE,
  log = TRUE, latex = TRUE
)
```

Note that the `mult_plot()` function accepts various kinds of input data, such as a complete `lmls` model in the first case or simply one or multiple sampling matrices in the latter case, and correctly extracts the corresponding samples.

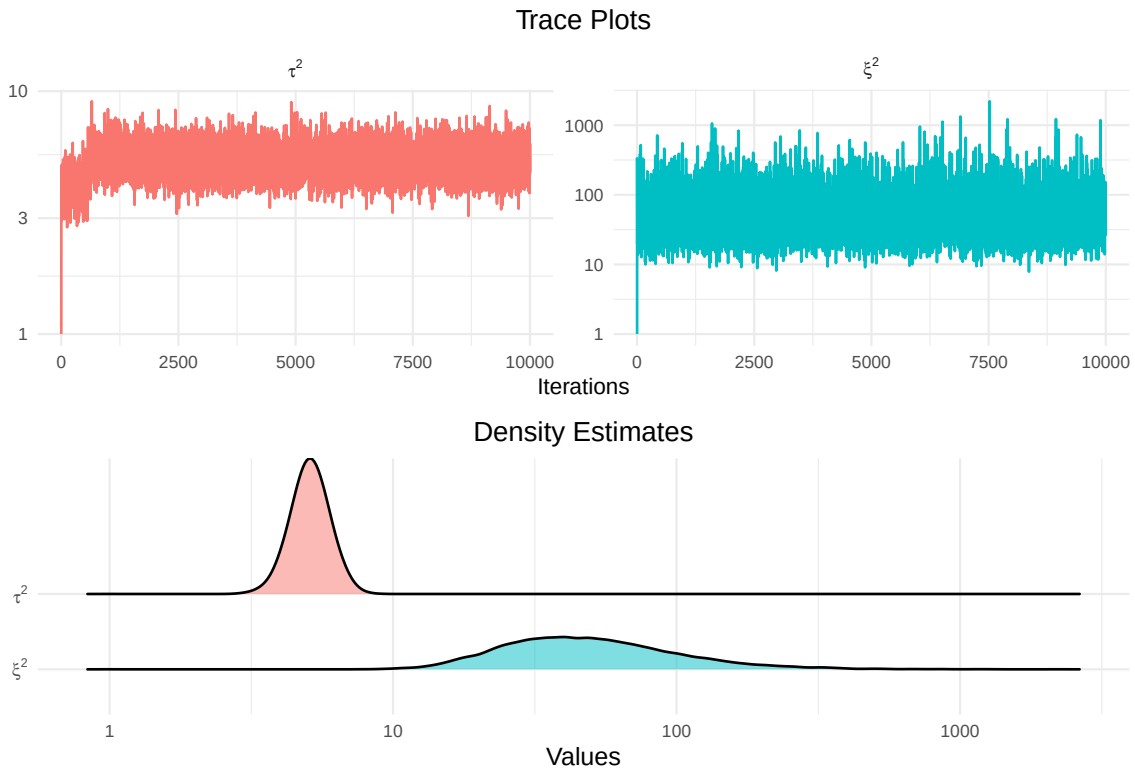

Figure 2: Trace and Density Plots for the Prior Variances, data: abdom

To focus on a single Markov chain, the `diagnostic_plots()` function is useful:

```
diagnostic_plots(
  samples = abdom_fit$sampling_matrices$beta_samples[, "beta_1", drop = FALSE],
  lag_max = 100, latex = TRUE
)
```

Figure 3 clearly shows the need for a burnin and thinning step for β_1, since the chain seems to converge after roughly 2500 iterations and the samples are strongly autocorrelated. The `lag_max` argument controls the number of lags that are included in the autocorrelation plot.

In order to confront these issues, we remove the first 2500 samples and additionally only keep every 10th iteration:

```
abdom_fit$sampling_matrices$beta_samples[, "beta_1", drop = FALSE] %>%
  burnin(num_burn = 2500) %>%
  thinning(freq = 10) %>%
  diagnostic_plots(lag_max = 100, latex = TRUE)
```

The diagnostic plots for β_1 in Figure 4 look much better. The chain has converged and there is little residual autocorrelation left after thinning out the samples. Additionally, the posterior density approximately follows a normal distribution.

Although removing posterior estimates in this way (and arguably losing valuable information) is not uncontroversial in the Bayesian community, statistical estimates from the remaining, well behaved chain are usually more stable and reliable. In any case, the above procedure clearly emphasizes the usefulness of a graphical diagnosis of the sampling results as a preliminary step before drawing any far reaching conclusions.

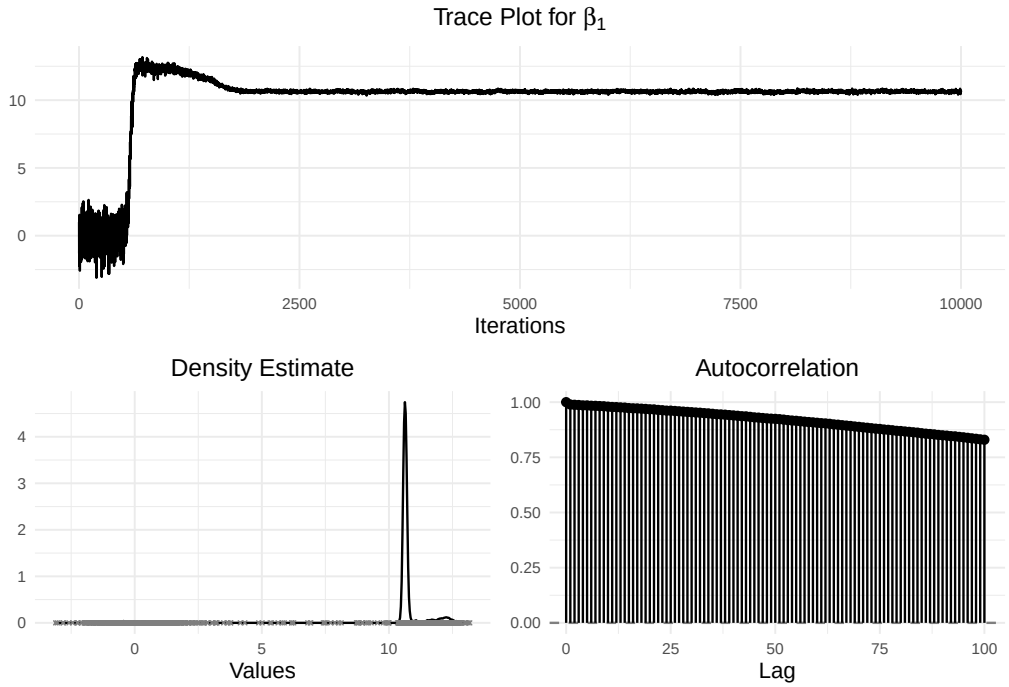

Figure 3: Diagnostic Plots for β_1 before thinning, data: abdom

2.1.3 Statistical Analysis

At the end of a Bayesian statistical analysis, summary statistics of the posterior distribution are of major interest. These include estimates for *centrality*, such as the Posterior Mean or the Posterior Median, estimates for the *spread*, e.g. the Posterior Variance, and quantile estimates in the tails of the posterior distribution as bounds for credible intervals.

The `asp21bridge` package offers two options to quickly access this information: If only a quick look at the numerical results without further investigation is desired, the `summary()` function can be used. The `lmls` package adapts this generic S3 method to the `lmls` class with an additional `type` argument. Specifying `mcmc_ridge` displays the estimates of the `mcmc_ridge()` function. Note, that this option is only applicable for the `toy_fit` model, which is based on the `lslm` class:

```
summary(toy_fit, type = "mcmc_ridge")
##
## Call:
## lmls(location = y ~ x1 + x2 + z1 + z2, scale = ~z1 + z2, data = toy_data,
##     light = FALSE)
##
## Pearson residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.29800 -0.38660  0.11770 -0.01354  0.57410  2.54600
##
## Location coefficients (identity link function):
##              Mean       2.5%        50%  97.5%
## beta_0  0.0003161 -0.1488290 -0.0013337  0.156
## beta_1 -2.0015679 -2.0082464 -2.0015799 -1.995
## beta_2 -1.0044943 -1.0179012 -1.0044382 -0.991
## beta_3  1.0014607  0.9802516  1.0017075  1.022
## beta_4  2.0080638  2.0015876  2.0081264  2.014
```

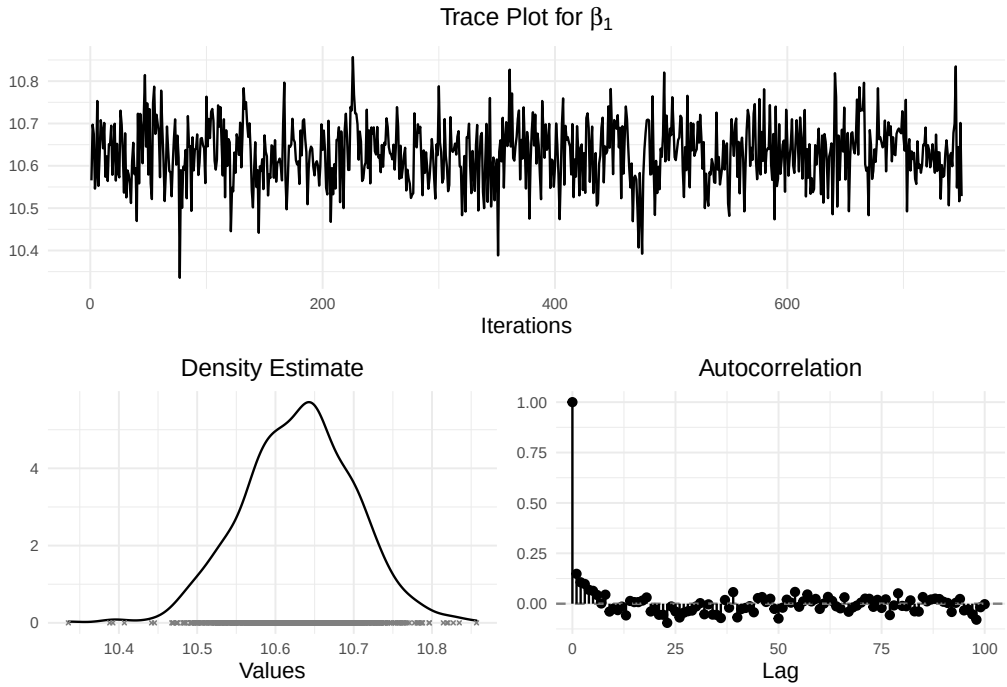

Figure 4: Diagnostic Plots for β_1 after thinning, data: abdom

```
##
## Scale coefficients (log link function):
##             Mean    2.5%     50%  97.5%
## gamma_0   0.9373 -0.7053  0.8277  3.074
## gamma_1 -1.1471 -1.5038 -1.1357 -0.862
## gamma_2   0.9325  0.7238  0.9305  1.176
##
## Residual degrees of freedom: 42
## Log-likelihood: 32.28
## AIC: -48.57
## BIC: -33.27
```

One downside of this approach is that the displayed values are not saved anywhere by default and, thus, cannot be immediately accessed. This issue is solved by the more informative `summary_complete()` function, which conveniently saves all relevant quantities in a data frame, the central object for data analysis in `R`.

In the following section we focus on the `toy_data` model, since there the true coefficient values are known, and use the popular `dplyr` package for further data frame manipulations. We are primarily interested in Posterior Mean estimates for all coefficients, such that we only select a subset of the output columns and add a new column containing the true data generating values:

```
library(dplyr)

true_beta <- c(0, -2, -1, 1, 2)
true_gamma <- c(0, -1, 1)

summary_complete(samples = toy_fit) %>%
  filter(stringr::str_detect(Parameter, pattern = "beta|gamma")) %>%
  mutate(Truth = c(true_beta, true_gamma)) %>%
  select(Parameter, `Posterior Mean`, Truth, `Standard Deviation`)
```

```
## # A tibble: 8 x 4
##   Parameter `Posterior Mean` Truth `Standard Deviation`
##   <chr>                <dbl> <dbl>                <dbl>
## 1 beta_0            0.000316     0               0.0769
## 2 beta_1           -2.00        -2               0.00339
## 3 beta_2           -1.00        -1               0.00676
## 4 beta_3            1.00         1               0.0104
## 5 beta_4            2.01         2               0.00315
## 6 gamma_0           0.937        0               0.967
## 7 gamma_1          -1.15        -1               0.163
## 8 gamma_2           0.933        1               0.109
```

Except for γ_0, all coefficients are estimated very accurately with a slightly larger deviation form the true values for the remaining $\boldsymbol{\gamma}$ vector, which is sampled by the Metropolis-Hastings algorithm, compared to the $\boldsymbol{\beta}$ vector, which is drawn directly from a multivariate normal distribution.

Similar to the plotting functions introduced in section 2.1.2, the `summary_complete()` function is very robust with respect to its data input: Besides the whole model object as in the example above, just the `toy_fit$mcmc_ridge` list entry or even simply the sampling matrices `toy_fit$mcmc_ridge$sampling_matrices` are valid inputs, all leading to the same output.

Further, the function has the optional `include_plot` argument. If set to `TRUE`, one additional `Plot` column is added to the data frame, which leverages the `diagnostic_plots()` function with some default settings and, thus, contains diagnostic plot objects for all coefficients. This option comes in handy in an interactive workflow: Particularly interesting findings from the `summary_complete()` output can be quickly extracted from the data frame without interrupting the current thought process by using the `diagnostic_plots()` function separately.

In the example above, the `gamma_0` row shows a large posterior variance in addition to the large deviation of the Posterior Mean estimate from the true value. In order to investigate, if the chain suffers from a lack of convergence or a significant autocorrelation, we could create the desired (yet here omitted) output with the following simple command:

```
summary_complete(toy_fit, include_plot = TRUE) %>%
  filter(Parameter == "gamma_0") %>%
  pull(Plot)
```

This introductory tutorial is not intended to underline the sampler's performance or validity, but rather provide an overview of the various applications, where the `asp21bridge` package turns out to be useful. A more in depth analysis of the `mcmc_ridge()` results, also in comparison to alternatives like the `lmls()` and `mcmc()` functions of the underlying `lmls` package, can be found in chapter 3 after explaining some of the internal implementations in section 2.2.

2.2 Implementation of the Markov Chain Monte Carlo Sampler

2.2.1 Iterative Parameter Sampling

2.2.2 Metropolis Hastings Step

2.3 Package Development

This section is dedicated to more niche aspects, that come along with the package development process. Part 2.3.1 focuses on more formal components of the package, whereas some of the ideas behind the `asp21bridge` functions are discussed in section 2.3.2.

2.3.1 Code Coverage

Arguably the most important component of a package for new users is the **documentation**. All exported functions of the `asp21bridge` package as well as the `toy_data` data set are fully documented according to common standards for `R` packages.

Specifically, we put effort into precise, yet not excessively long descriptions of the overall function and their parameters with highlighted default settings. The help pages are designed uniformly across all functions. The `examples` section usually starts with the most basic applications signaling the function's *intent* and proceeds with more complex use cases that cover many of the function's optional arguments.

A more elaborate, publicly accessible tutorial similar to section 2.1 of this report can be found in the `README` file or the front page of the `asp21bridge` GitLab website.

A second major aspect, that contributes to the quality of a package, are **unit tests**. Up to this point, a total of 251 unit tests have been written for the `lmls` and the `asp21bridge` packages combined. The number of implemented tests alone is, of course, not a meaningful metric, since the tests might be highly redundant and capture only a small fraction of the entire package's functionality. Thus, we prioritized both *width* and *depth* of the test coverage.

More specifically, unit tests are written for every single exported function with a larger focus on the more complex and more central functions such as the main `mcmc_ridge()` function.

Further, not only simple input checks (which are important nonetheless!), but also more esoteric edge cases and in particular very common use cases of combining inputs and outputs of multiple `asp21bridge` functions are covered. Whenever discovering and fixing an unexpected error during the development phase, we implemented corresponding unit tests, such that this specific error will not reoccur in the future. Tests are continuously written and updated to ensure a stable and enjoyable user experience.

At a larger scope, the `R CMD CHECK`, or equivalently the `devtools::check()` command, produces 0 errors, 0 warnings and 0 messages. Besides working unit tests, this includes correctly specified `DESCRIPTION` and `NAMESPACE` files for all imported and exported functions as well as functioning examples in the documentation. The master branch of the `asp21bridge` project will maintain this standard in the foreseeable future; possible new features that could involve breaking changes will first be implemented on separate Git branches and merged into the master branch at a later stage after a thorough testing procedure.

2.3.2 Design Choices

Throughout the development process, we tried to adhere to some 'best practices' for general software development. These include several different aspects:

Default Settings:

For many `asp21bridge` functions, the user has to specify very few inputs manually, since many input options are set to sensible default arguments. These inputs are chosen in a *neutral* way, such that the output aligns as best as possible with the user's expectation.

As an example, the default starting values for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are the Maximum Likelihood estimates from the `lmls()` function, if a model object is provided as input to the `mcmc_ridge()` function. This serves as guidance for users with little prior knowledge about their model. It is worth noting, however, that there are multiple input parameters which *can* be set manually if desired, enabling fine control over the sampling procedure.

Input Flexibility / Defensive Programming:

As already mentioned in section 2.1, all exported functions are designed to be as flexible as possible with respect to their input arguments. Thus, many functions accept multiple data structures like (nested) lists, data frames, matrices or even simple vectors as data input and internally transforms the input into the desired format.

These transformations, however, are only performed as long as the user's *intention* is clear, e.g. providing the samples as input to a plotting functions regardless of the exact data structure, where the samples are stored.

Whenever there is ambiguity or the input is simply not valid / incomplete, we have put effort into writing informative error messages.

To illustrate this last point, assume that we had forgotten to provide the second design matrix `Z` as input to our second application example from section 2.1 using the `abdom` data set. This might happen quite frequently, especially if the user is not particularly familiar with the structure of location-scale regression models:

```
y <- abdom$y
X <- as.matrix(abdom$x)

abdom_fit <- mcmc_ridge(
  y = y, X = X, beta_start = 1, gamma_start = 1, num_sim = 10000
)
## Error in validate_input(m = m, X = X, Z = Z, y = y, beta_start = beta_start, :
## At least either all model matrices (X, Z, y) and coefficients
## (beta_start, gamma_start) or a model object (m) must be given.
```

From the error message, it is immediately obvious which part of the input is missing.

One further quite neat feature is implemented for the generic `summary()` function. Since the possible values for the `type` argument are specific to the `lmls` class and therefore a potential reason for confusion, the error message provides some guidance in case of spelling mistakes, suggesting the closest valid input option:

```
summary(toy_fit, type = "mcmc-ridge")
## Error: `type` must be one of "ml", "boot", "mcmc", or "mcmc_ridge".
## Did you mean "mcmc_ridge"?
```

Modularity:

The `lmls` package served as a fantastic example how to design and compose functions in a modular way, leading to independent applications of single small functions in various contexts, easier debugging and arguably better code transparency and readability.

The `asp21bridge` package similarly splits large functions into multiple pieces according to the well known premise, that functions should be doing one thing only, but one thing well. Therefore the `mcmc_ridge_helpers` file contains various helper functions for the main `mcmc_ridge()` function, which perform tasks like input validation or inclusion of the Metropolis-Hastings step for the γ vector.

However, there is certainly a trade off to keep in mind: Since naming functions and variables is notoriously challenging, excessive modularity with poor naming choices can hinder code comprehension by inducing a wrong mental model of the function's task. Moreover, it can make sense to keep related functions close together in a physical sense and avoid spreading them across multiple files across the project.

Coding Style:

Before starting our work on different parts of the code base, we agreed to a consistent coding style. This includes using only lowercase letters and underscores in function and variable names and a limit of 80 characters per line. Here, we mostly conformed to the recommendations from the tidyverse style guide by Hadley Wickham. Moreover, we leveraged the styler package for consistent and aesthetically pleasing code formatting.

To extend consistent naming schemes even further, we chose the same argument names for all exported functions whenever possible. For instance, the data input for all functions that expect the MCMC simulations is called `samples` and all shared arguments of the 5 plotting functions are assigned the same name and the same functionality. Hence, it often suffices to be familiar with one function of a certain family, since that knowledge can be easily transferred to all related functions.

These conventions allow both new and experienced users of the package easier file navigation and orientation as well as a more enjoyable programming experience overall. Finally, there are two design choices, that are

very specific to the `asp21bridge` context.

Operation Chaining:

First, we aimed to allow for frictionless function composition via the popular `%>%` operator. As illustrated in section 2.1, this comes in particularly handy when including `burnin()` and `thinning()` operations in a larger pipeline without the need to define dummy variables for temporary objects:

```
lmls(
  location = y ~ x1 + x2 + z1 + z2, scale = ~ z1 + z2,
  data = toy_data, light = FALSE
) %>%
  mcmc_ridge(num_sim = 1000) %>%
  purrr::pluck("mcmc_ridge", "sampling_matrices") %>%
  burnin(num_burn = 100) %>%
  thinning(freq = 5) %>%
  mult_plot(type = "both", free_scale = TRUE, latex = TRUE)
```

This workflow proved to be so useful, that we provided access to the pipe operator directly by loading the `asp21bridge` package, i.e. we reexported `%>%` from the `magrittr` package. Alternatively, one could use the native `|>` pipe introduced in `R 4.1`. However, since the release of this `R` version was so recent, it might induce a stronger dependency than just relying on the `%>%` operator, which is ubiquitous at least in the `R` community that is related to the data science field.

There are two conditions that functions must fulfill to chain multiple operations:

- The first argument must be chosen uniformly for all (except the first) involved commands. In case of the `asp21bridge` package, *all* exported functions share the `samples` argument at the first position.
- Functions that serve as intermediary steps inside a chain should return the same object type as their input. When given a `lmls` model object, the `mcmc_ridge()` function returns a modified (copy of the same) `lmls` model object. Even more flexible are the `burnin()` and `thinning()` functions. They can take a list of matrices, a single matrix or a numeric vector as input and always return the same provided input data structure.

Object-Oriented Programming:

Secondly, we thought about initializing our own S3 class when calling the `mcmc_ridge()` function, but finally decided against it for two reasons: We always considered the `asp21bridge` package as a strict extension to the `lmls` package. Thus, we did not want to make the underlying `lmls()` results less accessible after using `mcmc_ridge()`, i.e. generic functions like `coef()`, `plot()` or `print()` should in our mind create the same output before and after extending the original model.

One exception is the `summary()` function, as we have already mentioned. Here, we added one more option to the `type` argument, which has to be specified anyway, as soon as the `boot()` and/or `mcmc()` function of the `lmls` package are used. Apart from that, we could not see much additional value in implementing wrapper functions, which adapt e.g. the `print()` output to the MCMC results with penalty, beyond the already existing tools, that were introduced in section 2.1.

3 Simulation Studies

This chapter investigates the effect of manipulating some of the `mcmc_ridge()` inputs and thereby sampling parameters in a controlled environment, such that changes in the estimation outcome can be directly linked to respective changes in the model inputs. After an introductory exploration phase, we decided for a subset of all possible model variations that indicated the greatest potential for interesting and relevant findings.

As a result, the simulation studies discussed in this chapter will be conducted on the following components:

- The **data** input (sections 3.1 and 3.2), which is captured by the function argument `m` or alternatively the combination of `X`, `Z` and `y`. Here, the sampler's *robustness* is tested across various scenarios.
- The **sample size** `n` (section 3.3) and the **number of simulations** `num_sim` (section 3.4). Here, we are looking for a possible stabilization process with increasing either of these two input parameters hinting at *asymptotic/convergence* properties.
- The **hyperparameters** `a_tau`, `b_tau`, `a_xi` and `b_xi` (section 3.5) of the Inverse Gamma prior distribution of the variance parameters τ^2 and ξ^2, as specified in the first report. The effect of hyperparameters in a hierarchical Bayesian model can be difficult to predict based on pure logical reasoning. Therefore simulations are a useful tool to either confirm prior assumptions or discover unexpected behaviour.

Since the resulting simulation studies serve different purposes (e.g. diagnostic vs. explorative), they demand for different approaches in the simulation settings, the implementation as well as the analysis and presentation of the results. For that reason, we decided against forcing all of the following sections into one common rigid framework. Instead, each section individually motivates, explains and interprets the methods chosen for its particular use case.

In order to keep the analysis compact and succinct, there will be almost no code included. It it worth noting though that the `R Markdown` document generating this report as well as all `R Scripts` used for the simulations are contained in the `simulation-studies` folder of the `asp21bridge` package. Thus, each figure as well as all numerical results are fully reproducible and can be repeated and extended by the reader.

3.1 Correlated Predictor Variables

Up to this point, we have often illustrated the usage and results of the `mcmc_ridge()` sampler with simulated data from the built-in `toy_data` set. There, each regressor variable is independently sampled from a normal distribution and the outcome variable is simulated based on a correctly specified location-scale regression model $y_i \sim \mathcal{N}\left(\mathbf{x}_i^T\boldsymbol{\beta}, \exp\left(\mathbf{z}_i^T\boldsymbol{\gamma}\right)^2\right)$. All these conditions lead to an excellent performance of the `mcmc_ridge()` sampler, but might arguably not represent the most challenging task.

Sections 3.1 and 3.2 analyze the sampler's performance on simulated data, which might be closer to data found in the real world. First, we will induce correlation among the predictor variables, whereas in the following section the distributional assumptions are considerably changed. Further, the `mcmc_ridge()` performance is compared to the Maximum Likelihood based `lmls()` estimates and the Markov Chain Monte Carlo `mcmc()` sampler without penalty from the `lmls` package.

3.1.1 Simulation Setting

- The design matrix $\mathbf{X} = \begin{pmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3\end{pmatrix}$ is simulated from a three dimensional normal distribution $\mathcal{N}_3\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ with mean vector $\boldsymbol{\mu} = \begin{pmatrix}-5 & 2 & 0\end{pmatrix}^T$ and covariance matrix $\begin{pmatrix}1 & \rho & \rho \\ \rho & 3 & \rho \\ \rho & \rho & 5\end{pmatrix}$. Hence, the dependence among the regressors is fully determined by the parameter ρ.
- The design matrix $\mathbf{Z} = \begin{pmatrix}\mathbf{z}_1 & \mathbf{z}_2\end{pmatrix}$ consists of linear combinations of the regressors $\mathbf{x}_1$ up to $\mathbf{x}_3$, more specifically $\mathbf{z}_1 = 0.8 \cdot \mathbf{x}_1 + 0.2 \cdot \mathbf{x}_2$ and $\mathbf{z}_2 = \mathbf{x}_2 - 0.5 \cdot \mathbf{x}_3$.

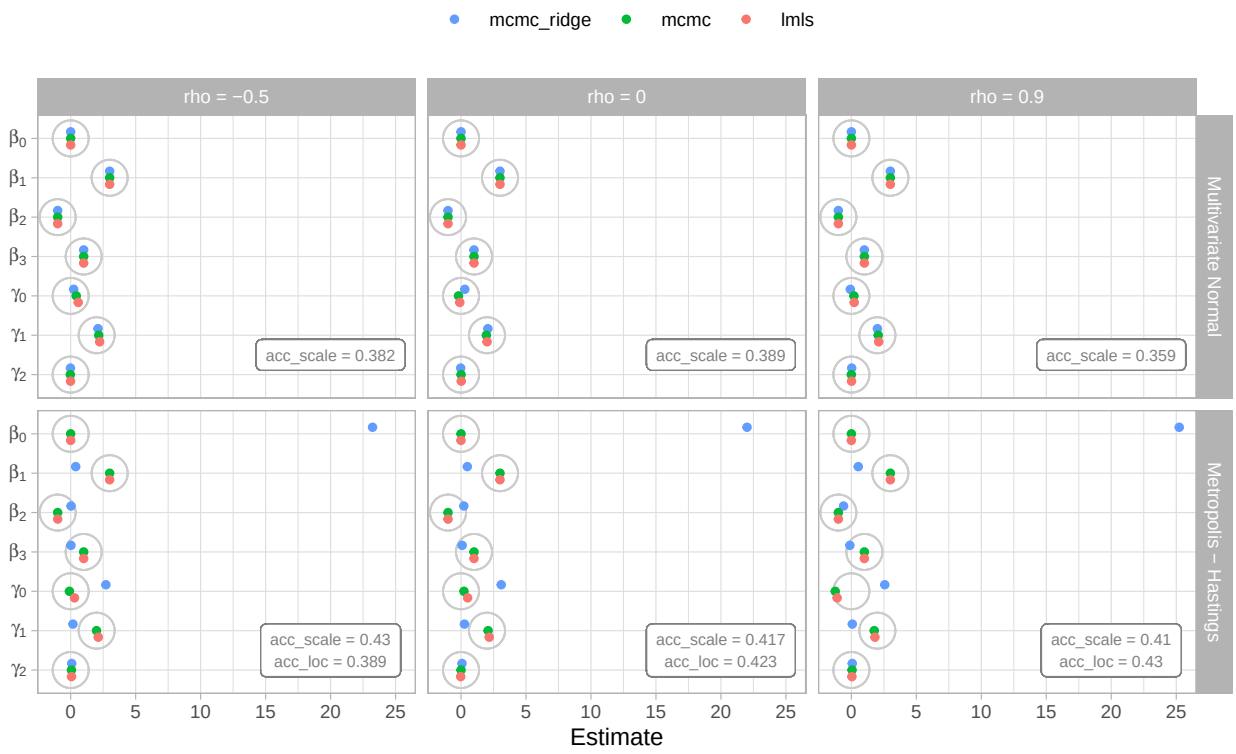

Figure 5: Comparison of Correlation Structures - One Simulation Cycle

- In both design matrices intercept columns are added for estimation purposes. The true coefficient vectors are given by $\boldsymbol{\beta} = \begin{pmatrix}\beta_0 & \beta_1 & \beta_2 & \beta_3\end{pmatrix}^T = \begin{pmatrix}0 & 3 & -1 & 1\end{pmatrix}^T$ and $\boldsymbol{\gamma} = \begin{pmatrix}\gamma_0 & \gamma_1 & \gamma_2\end{pmatrix}^T = \begin{pmatrix}0 & 2 & 0\end{pmatrix}^T$.
- Three different values were chosen for $\rho \in \{0, -0.5, 0.9\}$ to compare the 'nice' case of uncorrelated predictors with the performance for negative and positive dependence. For each covariance structure the three models `mcmc_ridge()`, `mcmc()` and `lmls()` were fitted, where each Posterior Mean estimate from both of the Markov Chain Monte Carlo samplers is based on 1000 samples.
- Moreover, we compared the performance of the usual `mcmc_ridge()` implementation, which draws $\boldsymbol{\beta}$ from the closed form full conditional (multivariate normal) distribution, with an alternative sampling process that uses a Metropolis-Hastings approach for both, the location parameter $\boldsymbol{\beta}$ as well as the scale parameter $\boldsymbol{\gamma}$. The latter is initiated by the `mcmc_ridge()` argument `mh_location = TRUE`. The variance of the corresponding proposal distribution is set to a carefully chosen default value, but can be manually changed by means of the `prop_var_loc` argument.

3.1.2 Simulation Results

Figure 5 displays the posterior mean estimates for both MCMC samplers and the Maximum Likelihood estimates for the `lmls()` function of one complete simulation cycle. For a better visual comparison the true values for each coefficient are indicated by grey circles, whereas the acceptance rate(s) of the Metropolis-Hastings algorithm used in the sampling process are provided in grey boxes.

The scaling of the x - axis is dominated by one outlier in the lower panel for each correlation structure. While the Metropolis-Hastings approach for $\boldsymbol{\beta}$ performs moderately well for most of the coefficients, it massively overestimates the intercept β_0.

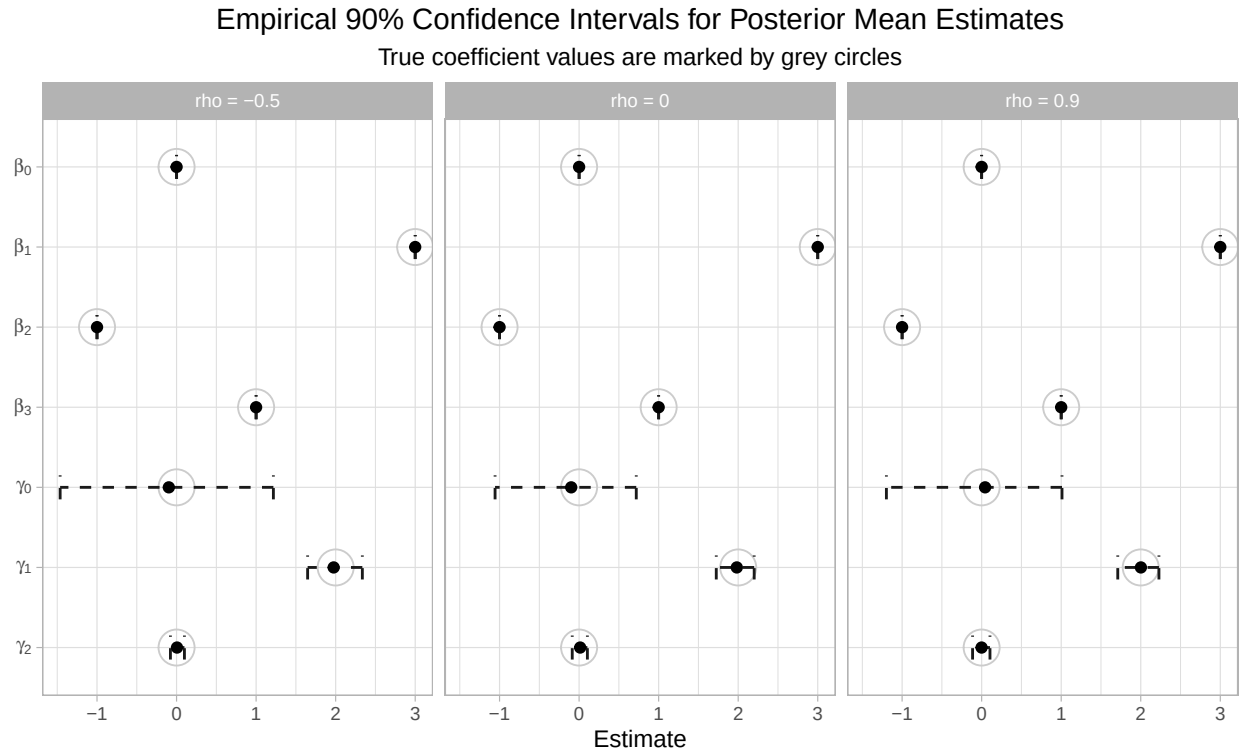

Figure 6: Comparison of Correlation Structures - 50 Simulation Cycles

This observation can be made across many different data sets: In some special cases the performance is close to (but never better) than sampling directly and independently from a multivariate normal distribution. However, most of the time the performance is significantly worse and the samples show (obviously) much larger correlation requiring a higher number of simulations for stable estimation. For that reason, we limit the Metropolis-Hastings sampling process for $\boldsymbol{\beta}$ to this one illustration and will focus on the classical `mcmc_ridge()` implementation in the remaining parts of the report.

The upper panel in Figure 5 indicates a very good performance by all three estimation procedures in consideration. Further, all acceptance probabilities are in a reasonable range supporting a fast convergence of all Markov Chains.

It is important to remember, that each point in the plot only represents exactly one measurement. In order to make any conclusions about bias and variance of the different estimation models, the above procedure is repeated 50 times. The black points in Figure 6 represent the mean of these 50 Posterior Mean estimates. Since we cannot rely on distributional theory for the standard errors, the variability of the estimates is displayed by nonparametric 'confidence' intervals, which are simply given by the range from the empirical 0.05 quantile to the 0.95 quantile of the 50 estimated values.

Further investigations have shown that the `mcmc_ridge()`, the `mcmc()` and the `lmls()` functions perform very similar for each correlation structure. For that reason only the results of the `mcmc_ridge()` sampler are shown in Figure 6.

There are three conclusions from this first simulation study:

1. The correlation structure does not have a significant impact of the sampling results. The three plot facets look almost identical.
2. The `mcmc_ridge()` sampler (as well as the `mcmc()` and `lmls()` functions) are very robust towards correlated data and perform extremely well. In particular, all three approaches (visually and numerically) provide close to unbiased estimates.

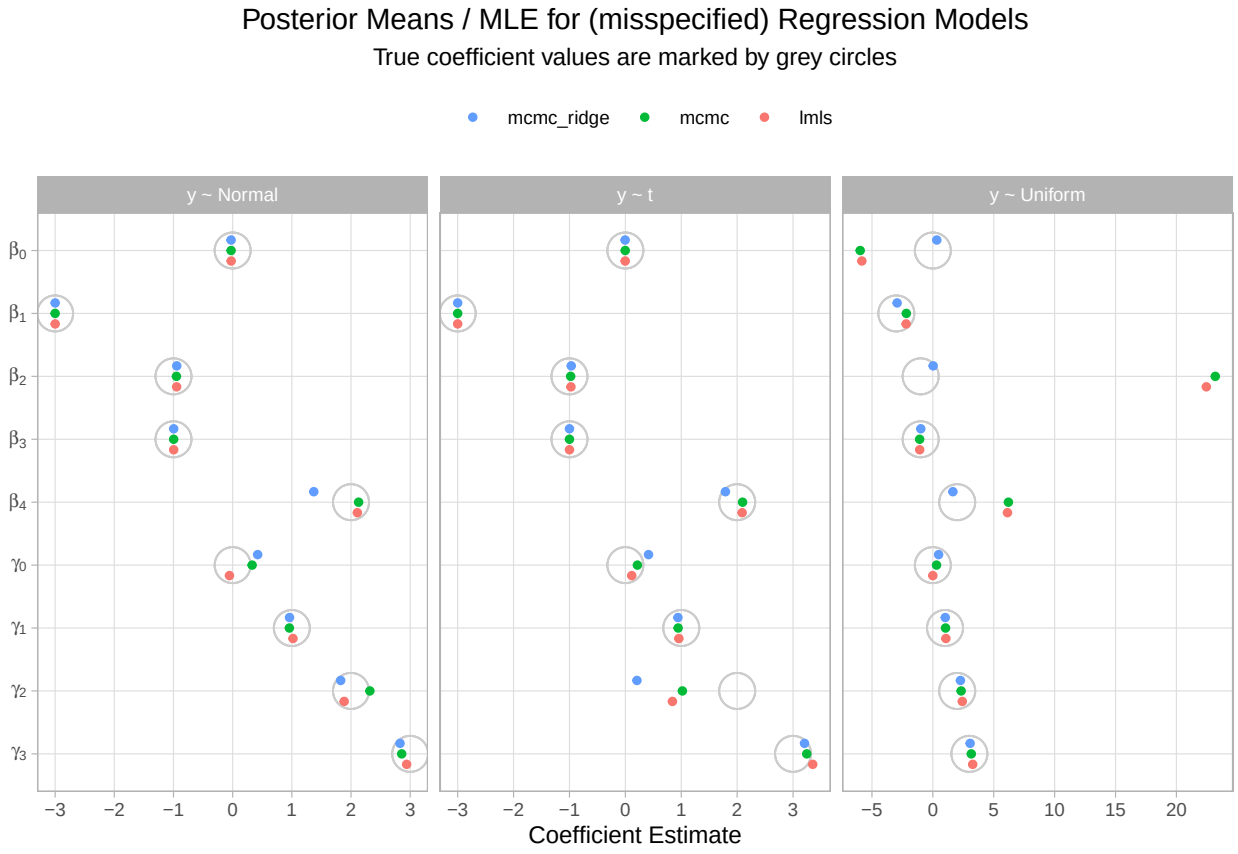

Figure 7: Comparison of Outcome Distributions - One Simulation Cycle

3. The variability among the $\boldsymbol{\beta}$ estimates is almost nonexistent, such that results from a single simulation cycle are already reliable and representative. While the estimates for the $\boldsymbol{\gamma}$ vector are still correct on average, the variability across different simulations is significant (particularly for γ_0). Thus, averaging the results from multiple repetitions of the sampling process is advisable.

3.2 Challenging the Model Assumptions

This simulation study is structured in a very similar way to the study considered in section 3.1. Instead of varying the correlation structure among the regressors in the underlying data set, both the regressors and the outcome variable y are sampled from distributions that are more challenging for estimation than the normal distribution.

3.2.1 Simulation Setting

- The design matrix $\mathbf{X} = \begin{pmatrix} \mathbf{1}_n & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \end{pmatrix}$ contains four independently sampled regressor variables plus one intercept column:
  - $\mathbf{x}_1 \overset{iid}{\sim} \mathcal{N}(5, 16)$,
  - $\mathbf{x}_2 \overset{iid}{\sim} \text{Exp}(5)$,
  - $\mathbf{x}_3 \overset{iid}{\sim} \mathcal{U}([-2,\ 12])$,
  - $\mathbf{x}_4 \overset{iid}{\sim} \text{Ber}(0.3)$.
- The design matrix $\mathbf{Z} = \begin{pmatrix} \mathbf{1}_n & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{z}_3 \end{pmatrix}$ contains the additional regressor variable $\mathbf{z}_3 \overset{iid}{\sim} t_{10}$, which is independently sampled from all other columns.

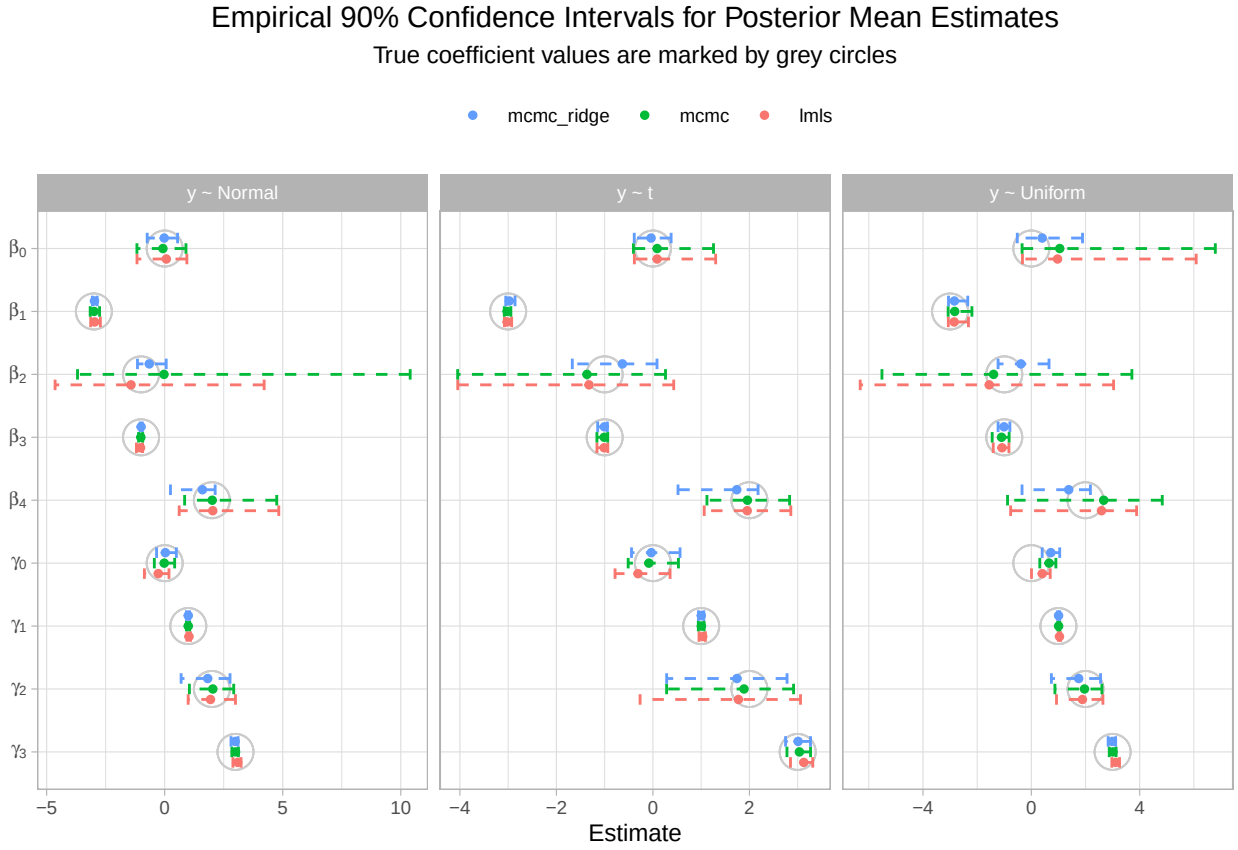

Figure 8: Comparison of Outcome Distributions - 50 Simulation Cycles

- The true coefficient vectors are given by $\boldsymbol{\beta} = \begin{pmatrix} \beta_0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 \end{pmatrix}^T = \begin{pmatrix} 0 & -3 & -1 & -1 & 2 \end{pmatrix}^T$ and $\boldsymbol{\gamma} = \begin{pmatrix} \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 \end{pmatrix}^T = \begin{pmatrix} 0 & 1 & 2 & 3 \end{pmatrix}^T$.
- Three different specifications for the outcome distribution were chosen:
  - $y_i \sim \mathcal{N}\left(\mu, \sigma^2\right)$,
  - $y_i \sim \mu + \left(\sigma \cdot \sqrt{\frac{3}{5}}\right) T$, where $T \sim t_5$,
  - $y_i \sim \mu + \sigma \cdot U$, where $U \sim \mathcal{U}\left([0,\ 1]\right)$.

In order to isolate the impact of the different shapes of the three probability distributions, the mean $\mu = \mathbf{x}_i^T \boldsymbol{\beta}$ and the variance $\sigma^2 = \exp\left(\mathbf{z}_i^T \boldsymbol{\gamma}\right)^2$ are held constant across the models.

Note that the `lmls()`, `mcmc()` and `mcmc_ridge()` models are built upon the assumption $y_i \sim \mathcal{N}\left(\mu, \sigma^2\right)$. Hence, we expect all three estimation procedures to perform well under the first outcome specification, which they were designed for. The remaining two cases analyze the performance in presence of a mild (t distribution) and a moderately strong (uniform distribution) violation of this model assumption.

3.2.2 Simulation Results

Just as in section 3.1 the results of one complete simulation cycle (each of the $3 \cdot 3 = 9$ models was fitted once / each data point represents one estimate) are displayed in Figure 7. Note that the second facet is labeled by $y \sim t$, although it is formally sampled from an affine transformation of a t-distributed random variable, which does not follow an exact t distribution.

The differences within each facet as well as between the facets are significant. All three models seem to estimate the $\boldsymbol{\beta}$ vector well, when there are no or only mild violations of the normal assumption for y. If y

Table 1: Bias of Coefficient Estimates

	Normal			t			Uniform		
	lmls	mcmc	mcmc_ridge	lmls	mcmc	mcmc_ridge	lmls	mcmc	mcmc_ridge
β_0	0.07	-0.07	-0.02	0.09	0.09	-0.04	0.97	1.05	0.40
β_2	-0.43	0.97	0.35	-0.32	-0.37	0.37	-0.55	-0.39	0.61
β_4	0.04	0.01	-0.40	-0.05	-0.04	-0.26	0.59	0.67	-0.62
γ_0	-0.27	-0.02	0.03	-0.31	-0.08	-0.03	0.40	0.66	0.72
γ_2	-0.06	0.04	-0.18	-0.23	-0.11	-0.26	-0.11	-0.04	-0.25

Table 2: Standard Errors of Coefficient Estimates

	Normal			t			Uniform		
	lmls	mcmc	mcmc_ridge	lmls	mcmc	mcmc_ridge	lmls	mcmc	mcmc_ridge
β_0	0.83	0.69	0.47	0.62	0.62	0.49	2.15	2.37	1.47
β_2	7.11	4.41	0.40	2.54	2.58	0.54	6.48	6.48	0.67
β_4	1.32	1.33	0.68	0.57	0.56	0.53	4.10	4.30	0.81
γ_0	0.33	0.28	0.27	0.35	0.33	0.34	0.22	0.19	0.21
γ_2	0.78	0.65	0.62	1.01	0.93	0.89	0.55	0.54	0.56

is sampled from a uniform distribution, there are major differences for β_0, β_2 and β_4 (notice the extended x-scale in the third facet). Interestingly, the $\boldsymbol{\gamma}$ vector is estimated very well in the latter case with more deviations in the setting, where y is based on the t distribution.

To gain insights beyond this single simulation cycle, which could well be disturbed by random noise, we repeat the sampling process 50 times. The resulting means as well as empirical confidence intervals (analogous to section 3.1) are plotted in Figure 8.

This plot (literally) paints a drastically different picture, emphasizing the necessity of repeating experiments multiple times whenever possible. Across all 50 simulations the deviation of the estimates for β_2 (corresponding to the regressor variable from the exponential distribution) is huge for all three distributional specifications of y.

The small bias induced by all three models is negligible compared to the wide confidence intervals, which is particularly interesting, when y stems from a normal distribution. In this case all models should perform well, however the `lmls()` and the `mcmc()` Posterior Mean / Maximum Likelihood estimates vary wildly across the simulation cycles. A similar effect can be observed for β_0 in case of the uniform distribution. Here, all models overestimate the true value on average, while the `mcmc_ridge()` function again shows the smallest variability. In contrast, the estimates for γ_0 in the right facet are fairly stable across simulation cycles, but also consistently wrong at the same time.

Estimates of the bias and the standard error of the Posterior Mean / Maximum Likelihood estimates can be more distinctly compared by their numerical values provided in Tables 1 and 2. In order to emphasize the interesting/differing entries, both tables only include a subset of the estimated coefficients.

Considering the bias estimates in Table 1 first, there are no obvious patterns that would suggest the superiority of one model. Further, none of the three models tend to only over- or underestimate the true coefficient values. The most interesting entries are the bias estimates for β_0 and γ_0 in the uniform case, where all three models agree to significantly overestimate. However, the intercept coefficients are often of minor interest.

The standard error estimates displayed in Table 2 clearly indicate the worst performance of the `lmls()` and the `mcmc()` model for β_2. In almost all cases (and sometimes very significantly), the `mcmc_ridge()` sampler has the smallest standard error. This finding nicely confirms the underlying mathematical theory: The

present prior specifications in the Bayesian setting, which induces the equivalent form of a frequentist Ridge penalty, can lead to biased estimation.

However, this loss in accuracy can be (as it is in this case) dominated by the gain in precision by the shrinkage effect of the penalty. Note that (except for γ_0 in the most right facet in Figure 8) the `mcmc_ridge()` sampler slightly *overestimates* coefficients with true *negative* values and *underestimates* those with true *positive* values. This again is caused by the Ridge penalty leading to estimated coefficients close to zero.

In summary, the following conclusions can be drawn:

1. All three models are affected by changes in the regressor and/or outcome distributions. In the former case the regressor variables sampled from the Exponential and the Bernoulli distributions were the greatest challenge, in the latter case the outcome variable from the Uniform distribution. This is generally not surprising, since these distributions deviate most from the nicely behaved normally distributed case.
2. As expected, the `lmls()` function is affected strongly by violating the model assumptions, since its estimation process is based on the normal (log) Likelihood. Surprisingly, the `mcmc()` sampler without penalty often did not perform much better.
3. While the `mcmc_ridge()` function does not excel at estimation accuracy, it does lead to the most stable estimation with smallest standard errors in the vast majority of cases. As emphasized above, this behaviour nicely agrees with the mathematical theory of Ridge penalization.
4. Had we not conducted repeated experiments, our conclusions would have been quite different. Simulation results are therefore always worth repeating many times to consolidate the correct interpretations.

3.2.3 Technical Aspects

As outlined in the previous paragraph, a total of $50 \cdot 3 \cdot 3 = 450$ models were fitted to analyze the performance differences. In order to speed up the involved computations of this specific and some of the other simulation studies in this report, we used the *parallel computing* capabilities of `R`.

There are many options from various packages to choose from. We decided to use the furrr package, which is built on top of the `future` package specialized on parallel processing. As the name suggests, `furrr` provides a convenient way to use many functions from the popular `purrr` package, while using multiple cores at the same time. This *functional programming* based approach (similar to the `apply()` family in 'base `R`') is particularly well suited for simulation studies and provides some structural as well as minor performance advantages compared to the classical `for`-loop approach.

The following (slightly modified) code snippet provides a brief insight into the implementation:

```
plan(multisession, workers = 8)

full_results <- tibble(id = 1:50) %>%
  mutate(samples = future_map(
    .x = id,
    .f = ~ show_results(n = 50, num_sim = 1000),
    .options = furrr_options(seed = 1)
  ))
```

The `plan()` function borrowed from the `future` package initializes the parallel computing process and determines the number of cores/workers available for computation. The `show_results()` helper function fits all three models `mcmc_ridge()`, `mcmc()` and `lmls()` for each outcome distribution in a single simulation cycle.

This entire procedure is repeated 50 times in parallel using the `future_map()` function from the `furrr` package, where the results of all 450 models are saved in a well organized structure inside of a list column. This new column of the data frame contains complete information about all simulations, such that any required element for the further analysis can be easily extracted and post processed.

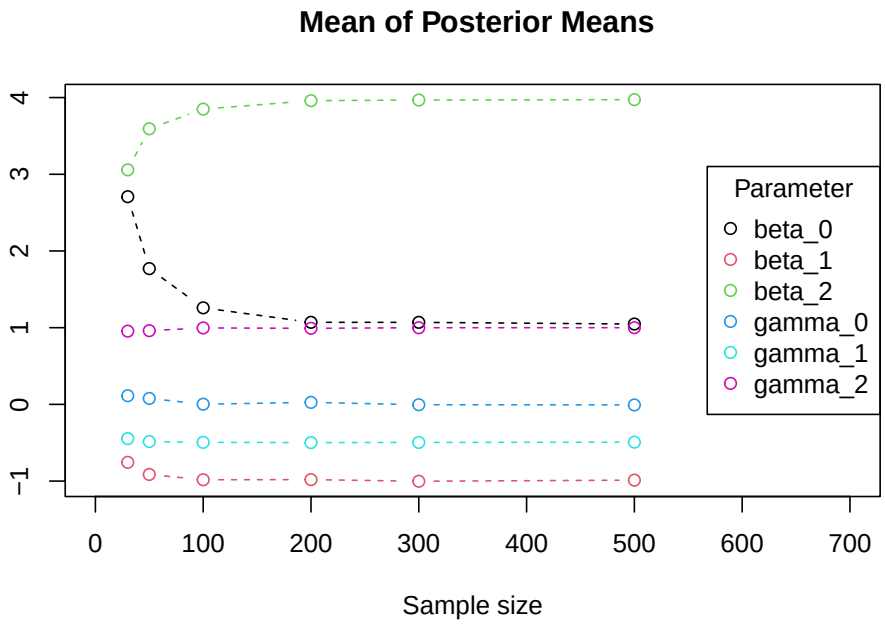

Figure 9: Mean value of 100 Posterior Mean Estimates

Finally, the `.options()` argument allows the specification of a random seed. Random number generation in the context of parallel computing is slightly more involved compared to the sequential approach. This additional complexity is automatically handled by the `future_map()` function, such that all results are sampled in a statistically valid and fully reproducible manner.

3.3 Sample Size

This simulation study analyzes the effect of the sample size `n` on the means of the posterior distribution for the coefficients of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. There are two main goals of this simulation study: On the one hand, we want to investigate whether the posterior means of large samples are closer to the true values than the posterior means of small samples. On the other hand, we want to analyze whether the `mcmc_ridge()` penalty affects the location of the posterior means.

3.3.1 Simulation Setting

- The design matrix $\mathbf{X} = \begin{pmatrix} \mathbf{1}_n & \mathbf{x}_1 & \mathbf{x}_2 \end{pmatrix}$ contains two independently sampled regressor variables plus one intercept column:
  - $\mathbf{x}_1 \overset{iid}{\sim} \mathcal{N}(1, 1)$,
  - $\mathbf{x}_2 \overset{iid}{\sim} \mathcal{N}(2, 1)$.
- The design matrix $\mathbf{Z} = \begin{pmatrix} \mathbf{1}_n & \mathbf{z}_1 & \mathbf{z}_2 \end{pmatrix}$ is structured in the same way with the regressor variables:
  - $\mathbf{z}_1 \overset{iid}{\sim} \mathcal{N}(1, 1)$,
  - $\mathbf{z}_2 \overset{iid}{\sim} \mathcal{N}(2, 1)$.
- The true coefficient vectors are given by $\boldsymbol{\beta} = \begin{pmatrix} \beta_0 & \beta_1 & \beta_2 \end{pmatrix}^T = \begin{pmatrix} 1 & -1 & 4 \end{pmatrix}^T$ and $\boldsymbol{\gamma} = \begin{pmatrix} \gamma_0 & \gamma_1 & \gamma_2 \end{pmatrix}^T = \begin{pmatrix} 0 & -0.5 & 1 \end{pmatrix}^T$.
- The posterior means are analyzed with respect to 6 different sample sizes: $\mathtt{n} \in \{0, 50, 100, 200, 300, 500\}$.
- In the next step, the outcome vector $y \in \mathbb{R}^n$ is simulated and passed to the `mcmc_ridge()` function with $\mathtt{nsim} = 500$ simulations.

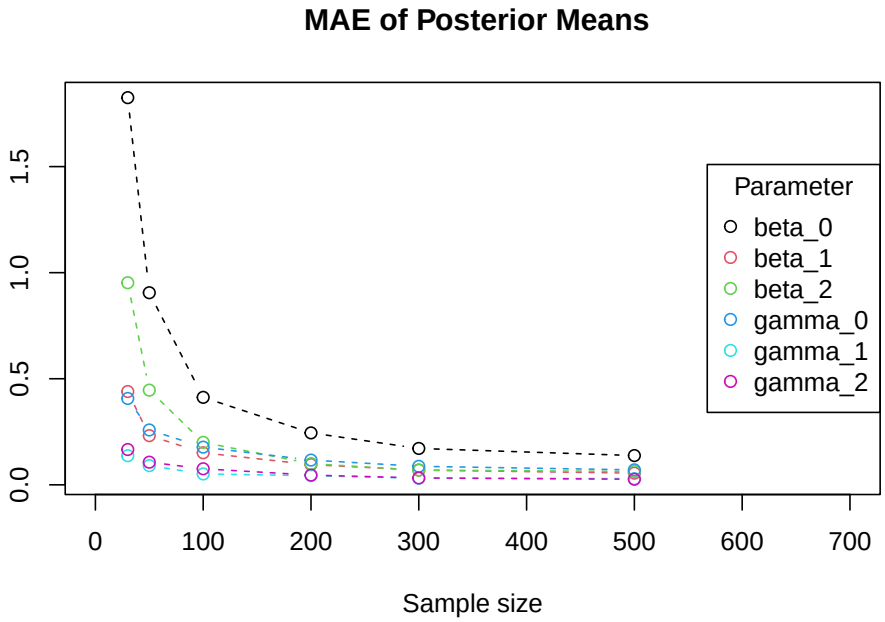

Figure 10: Mean Absolute Error Estimates

- To make the results more stable, the above procedure is repeated 100 times. For each coefficient, the mean value of the Posterior Mean estimates of each coefficient is calculated as well as the Mean Absolute Error (MAE) with respect to the true values of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$.

3.3.2 Simulation Results

The means of the Posterior Mean estimates are displayed in Figure 9. For larger sample sizes ($\texttt{n} \geq 200$) none of the six parameters are extremely biased.

Moreover, for $\texttt{n} = 30$, β_0 and β_2 are significantly biased, which might be caused by the high `mcmc_ridge()` penalty for $\beta_2 = 4$. The significant bias of β_0 might be explained by a counteract of the β_2 bias.

After getting an impression about empirical biases of the coefficients, we now focus on the variability of the posterior means of the coefficients, which are measured by the MAE based on the results of the 100 repetitions. Figure 10 points out that the posterior means of β_0 have significantly larger errors than the posterior means of β_2 for $\texttt{n} = 30$. However, this might also be explained by the fact that for $\texttt{n} = 30$, β_0 has a greater empirical bias than β_2 as could be observed in Figure 9.

In addition, for increasing sample sizes, the MAE of the Posterior Means tend to zero for all coefficients except β_0. Nevertheless, also the errors of β_0 seem to become smaller with increasing sample size.

3.4 Number of Simulations

Another important model input is `nsim`, the number of Markov Chain Monte Carlo simulations within the `mcmc_ridge()` function.

As explained in our first report, we chose the proposal density for the Metropolis-Hastings algorithm of $\boldsymbol{\gamma}$ in order to achieve acceptance rates between 35% und 50%. A higher number of `mcmc_ridge()` simulations increases the absolute number of accepted proposals and might deliver more precise approximations of the true posterior distributions, but also increases the computational cost such that there is a trade-off between precision and speed.

This simulation study analyzes the effect of the `nsim` parameter on the posterior distribution.

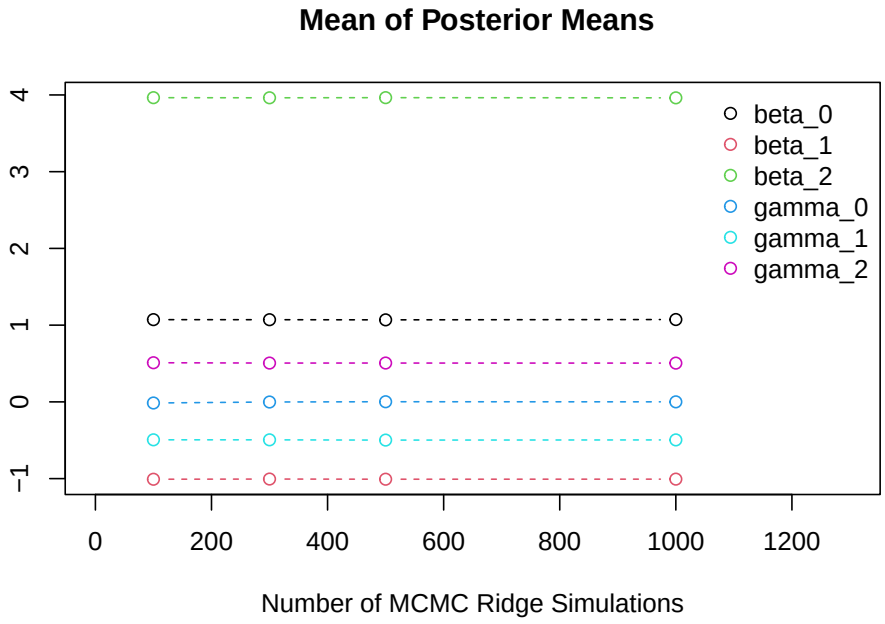

Figure 11: Mean value of 100 Posterior Mean Estimates

3.4.1 Simulation Setting

- The design matrix $\mathbf{X} = \begin{pmatrix} \mathbf{1}_n & \mathbf{x}_1 & \mathbf{x}_2 \end{pmatrix}$ contains two independently sampled regressor variables plus one intercept column. The sample size `n` is chosen to be 100.
  - $\mathbf{x}_1 \overset{iid}{\sim} \mathcal{N}(1, 1)$.
  - $\mathbf{x}_2 \overset{iid}{\sim} \mathcal{N}(2, 1)$.
- The design matrix $\mathbf{Z} = \begin{pmatrix} \mathbf{1}_n & \mathbf{z}_1 & \mathbf{z}_2 \end{pmatrix}$ similarly contains two independently sampled regressor variables plus one intercept column. The sample size `n` is chosen to be 100 as well.
  - $\mathbf{z}_1 \overset{iid}{\sim} \mathcal{N}(1, 1)$.
  - $\mathbf{z}_2 \overset{iid}{\sim} \mathcal{N}(2, 1)$.
- The true coefficient vectors are given by $\boldsymbol{\beta} = \begin{pmatrix} \beta_0 & \beta_1 & \beta_2 \end{pmatrix}^T = \begin{pmatrix} 1 & -1 & 4 \end{pmatrix}^T$ and $\boldsymbol{\gamma} = \begin{pmatrix} \gamma_0 & \gamma_1 & \gamma_2 \end{pmatrix}^T = \begin{pmatrix} 0 & -0.5 & 0.5 \end{pmatrix}^T$.
- In the next step, the outcome vector $y \in \mathbb{R}^n$ is simulated according to the correct normality assumption and passed to the `mcmc_ridge()` function.
- The posterior means are analyzed with respect to 4 different values for `nsim` $\in \{100, 300, 500, 1000\}$.
- Similar as in the simulation study concerning the sample size, the above procedure is repeated 100 times to make the results more stable. For each coefficient, the mean value of the 100 Posterior Means is calculated as well as the mean absolute error (MAE) with respect to the true values of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. In addition, the variances *within* the Markov Chains are analyzed.

3.4.2 Simulation Results

Similar as in the sample size simulation study, Figure 11 contains the mean values of the Posterior Mean estimates for the different `nsim` specifications. The result is quite clear: In the range between `nsim` $= 100$ and `nsim` $= 1000$, the `nsim` parameter has no significant impact on the mean of the posterior means.

Furthermore, for the chosen sample size `n` $= 100$, none of the 6 coefficients has a large bias. However, the

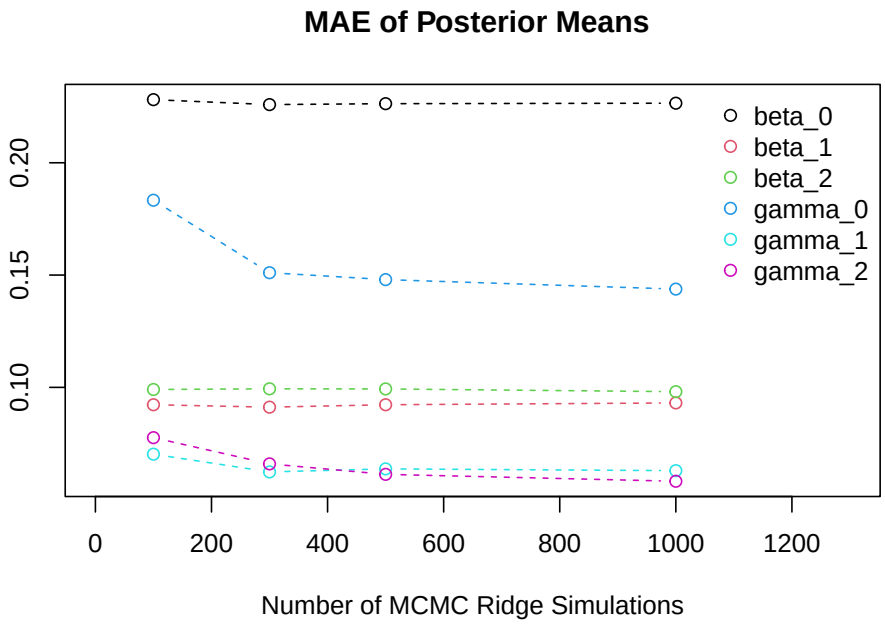

Figure 12: Mean Absolute Error Estimates

estimates of β_0 are all close to 1.07, whereas the estimates of β_2 are all close to 3.96. Thus, we obtain small biases for β_0 and β_2, which again might be a result of the high `mcmc_ridge()` penalty for $\beta_2 = 4$ (similar as discussed in section 3.3).

We also analyzed the *MAE* for different values of `nsim`. Figure 12 provides interesting results for the errors of the $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ coefficients: On the one hand, `nsim` seems not to have any impact on the errors of the $\boldsymbol{\beta}$ coefficients. But on the other hand, an increase of `nsim` leads to more precise $\boldsymbol{\gamma}$ estimates and apparently lower *MAE* values for all $\boldsymbol{\gamma}$ coefficients.

In the last step, we analyzed whether the value of `nsim` has any impact on the posterior variance within the Markov Chains. The results are displayed in Figure 13. Again, it can be observed that the `nsim` parameter has no impact on the variances within the $\boldsymbol{\beta}$ samples of the Markov chain, whereas the variance within the $\boldsymbol{\gamma}$ samples increases with increasing value of `nsim`.

This result is not really surprising, since the Markov Chain typically jumps to points which are quite close to the current location. As a consequence, if the number of jumps increases, also the variance of the samples might increase.

In combination with the lower *MAE* for the $\boldsymbol{\gamma}$ parameters, one can conclude that the approximated posterior distributions of the $\boldsymbol{\gamma}$ coefficients might be squeezed for small values of `nsim` and converge to the true posterior distributions if the number of simulations increases.

3.5 Hyperparameters

In the past, we have been sampling data with the `mcmc_ridge()` function without having a closer look on the effect of the hyperparameters and model inputs `a_tau`, `b_tau`, `a_xi` and `b_xi`. However, they affect the Full Conditional Distributions of τ^2 and ξ^2, as stated in sections 1.3.1 and 1.3.2 in chapter 1.

Moreover, the mean vector $\boldsymbol{\mu}_{beta}$ and covariance matrix $\boldsymbol{\Sigma}_{beta}$ of the $\boldsymbol{\beta}$ vector both depend on τ^2 and, thus, implicitly on the hyperparameters a_τ and b_τ (see section 1.3.3). Analogously, section 1.3.4 illustrates the direct effect of the Full Conditional distribution of $\boldsymbol{\gamma}$ on ξ^2, which in turn depends on the hyperparameters a_ξ and b_ξ.

Finally, cross effects can be observed, since $f(\boldsymbol{\beta} \mid \cdot)$ depends on $\boldsymbol{\gamma}$ through the quantities $\mathbf{W}$ and $\mathbf{u}$ as defined

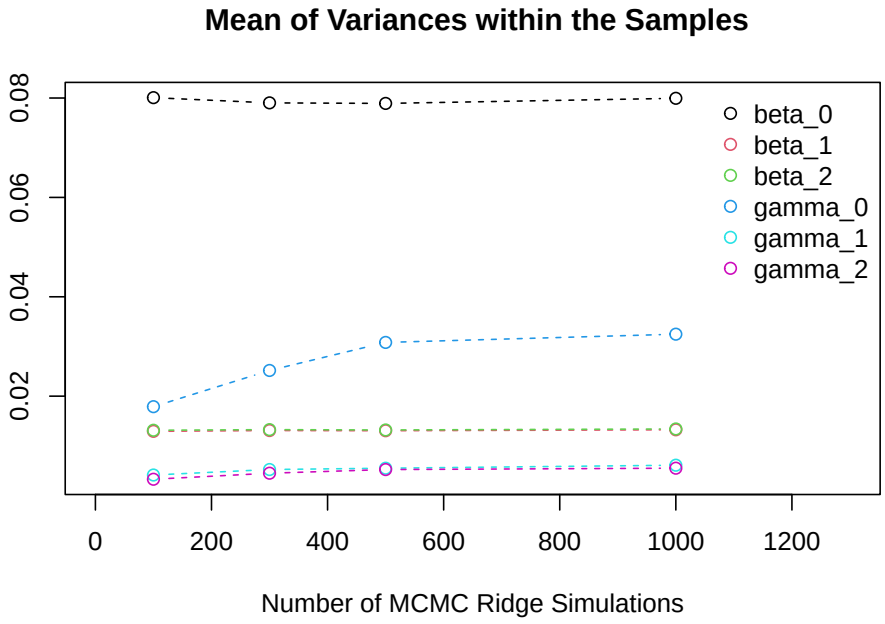

Figure 13: Mean value of 100 Posterior Variance Estimates

in chapter 1 and $f(\boldsymbol{\gamma} \mid \cdot)$ directly depends on $\boldsymbol{\beta}$. These dependencies are reflected in the `mcmc_ridge()` sampler by the iterative sampling procedure which is discussed in great detail in the previous sections 2.2.1 and 2.2.2.

Thus, the hyperparameter choice of a_τ, b_τ, a_ξ and b_ξ inevitably impacts the result of *all* coefficient estimates contained in the model in a nontrivial way, such that pure analytical reasoning might be misleading. For this reason, this section investigates these effects based on a simulation approach.

3.5.1 Simulation Setting

- The design matrix $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 \end{pmatrix}$ is simulated from a two dimensional normal distribution $\mathcal{N}_2\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ with mean vector $\boldsymbol{\mu} = \begin{pmatrix} 1 & 2 \end{pmatrix}^T$ and identity covariance matrix $\boldsymbol{\Sigma} = \mathbf{I}_2$. The same holds true for the design matrix $\mathbf{Z} = \begin{pmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{pmatrix}$ with mean vector $\boldsymbol{\mu} = \begin{pmatrix} 5 & 3 \end{pmatrix}^T$ and identity covariance matrix.
- In both design matrices intercept columns are added for estimation purposes. The true coefficient vectors are given by $\boldsymbol{\beta} = \begin{pmatrix} \beta_0 & \beta_1 & \beta_2 \end{pmatrix}^T = \begin{pmatrix} 0 & -1 & 4 \end{pmatrix}^T$ and $\boldsymbol{\gamma} = \begin{pmatrix} \gamma_0 & \gamma_1 & \gamma_2 \end{pmatrix}^T = \begin{pmatrix} 0 & -2 & 1 \end{pmatrix}^T$.
- For sampling the location parameter, the full conditional multivariate normal distribution of $\boldsymbol{\beta}$ is chosen, i.e. `mcmc_ridge(..., mh_location = FALSE)` is used. Therefore, the location estimate is directly affected by the hyperparameters.
- For simulating the influence of the hyperparameters, nine different values are chosen: $a_\tau, b_\tau, a_\xi, b_\xi \in \{-1, 0, 0.5, 1, 2, 10, 50, 100, 200\}$. Since for statistical properties like the mean of an Inverse Gamma distribution $\frac{b}{a-1}$ the condition $a > 1$ is required, particular attention is given to larger values. However, it is an aim to inspect the performance of the sampler for smaller hyperparameter values than 1 as well.

3.5.2 Simulation Results

The first two plots of Figures 14 and 15 display the absolute deviations of the Posterior Mean estimates from the true parameters with the stated different values for a_τ and b_τ. For each estimate, the Posterior Mean averages over 1000 simulations of the `mcmc_ridge()` sampler. Note, that location and scale parameters are plotted separately, according to the relationship mentioned above. For a better overview, the dotted line displays the linear trend of all estimate deviations.

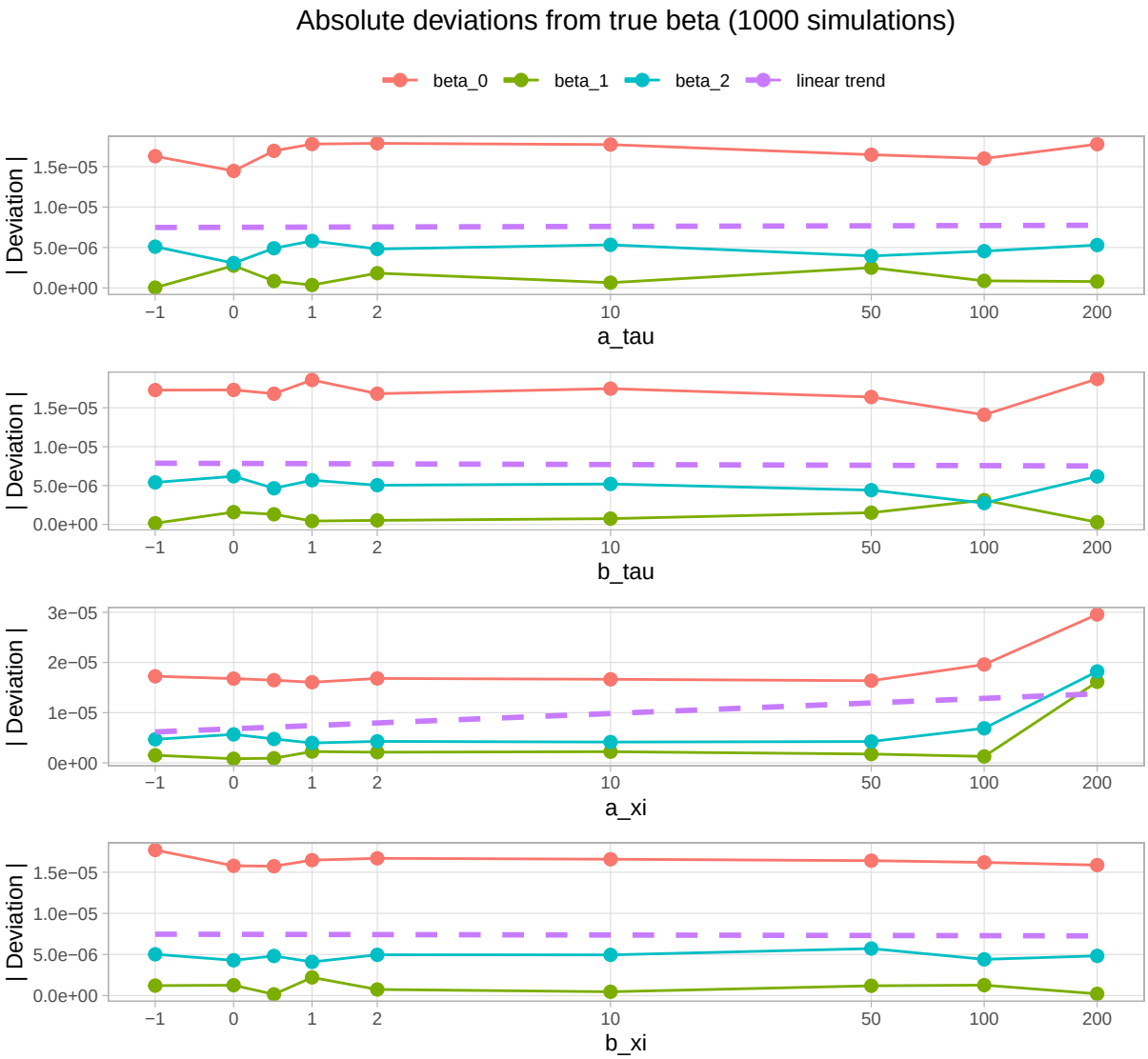

Figure 14: Comparison of the absolute deviations of beta parameters

The x - axis is transformed by a pseudo logarithm in order to clearly visualize the deviations in the range of -1 to 10, which would not be possible on original scales. Since -1 and 0 are also part of the hyperparameter values, the `pseudo_log_trans()` function of the `scales` package is applied, log-transforming positive values only.

It can be observed, that the intercept estimates in each plot show the largest deviations from their true value. In Figure 14, however, the overall deviations of $\boldsymbol{\beta}$ estimates from their corresponding true value are small in absolute value. In contrast, deviations of the $\boldsymbol{\gamma}$ estimates in Figure 15 are fairly significant, especially for γ_0.

The functional chain that applies to the estimates of $\boldsymbol{\beta}$ can be described by the effect of the mean of the inverse gamma distribution on τ^2: A larger value for b_τ leads to larger values of τ^2, which are again affecting the full posterior parameters of $\boldsymbol{\beta}$ and, thus, potentially increase the absolute deviation of the corresponding estimates from their true values. a_τ causes the opposite effect. This numerically observable effect, however, is hided by the overall small deviation in the first two plots of Figure 14.

It is remarkable, that the deviation of $\boldsymbol{\beta}$ estimates is smallest when $a_\tau, b_\tau \in \{50, 100\}$. For values of $a_\tau \leq 1$, one obtains wider variances of absolute deviations, since the Posterior Mean requires values larger than one.

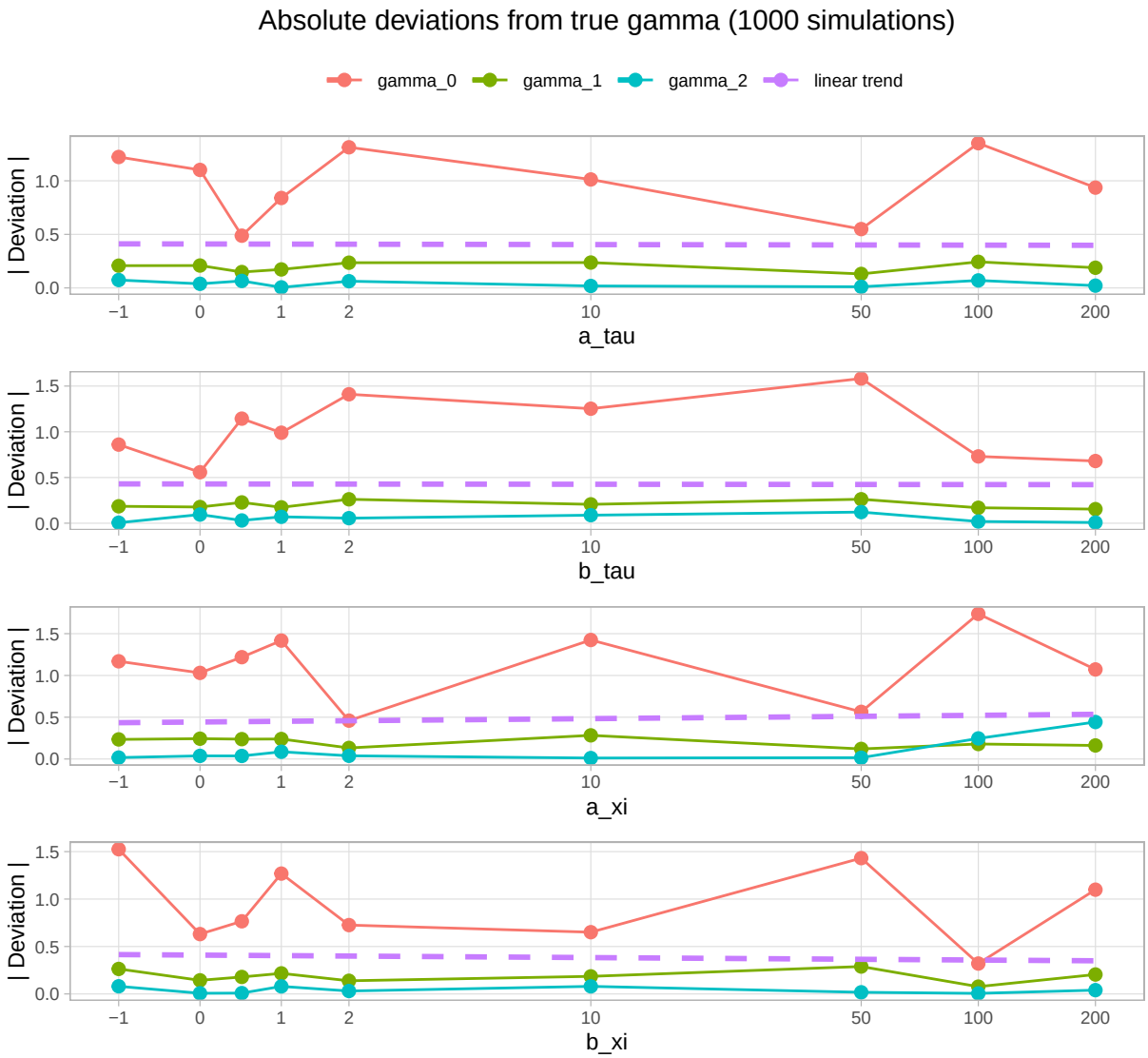

Figure 15: Comparison of the absolute deviations of gamma parameters

In the upper two plots of Figure 15, there is no clear impact of τ^2 and its parameters. Rooted in no direct effect of τ^2 on $\boldsymbol{\gamma}$ according to our underlying mathematical model, one observes cross-effects through the sampling procedure of the `mcmc_ridge()` sampler, where the full posterior $f(\boldsymbol{\gamma} \mid \cdot)$ depends on $\boldsymbol{\beta}$.

Anyway, our sampler produces the lowest deviation of $\boldsymbol{\gamma}$ estimates for $a_\tau, b_\tau \in \{0, 0.5, 200\}$, where 0.5 is chosen by coincidence for a_τ here, since wide variations for $a_\tau \leq 1$ of absolute deviations are observable again.

The lower two plots of Figures 14 and 15 are constructed analogously, but showing the impact of a_ξ and b_ξ on the location and scale parameters respectively. Again, the overall absolute deviations for the $\boldsymbol{\beta}$ estimates from their true values are small, whereas the deviations for the $\boldsymbol{\gamma}$ estimates are considerably larger. Once again, the intercept estimates display the largest deviations from their true value.

Arguing with the mean of the Inverse Gamma distribution of ξ^2 in a similar way, one obtains larger mean values for b_ξ, while a_ξ lowers them. The impact of ξ^2 on $\boldsymbol{\gamma}$ is assumed to decrease $f(\boldsymbol{\gamma} \mid \cdot)$ according to our underlying theoretical model. This effect is indicated by the linear trend lines in the second half of Figure 15.

In general, one obtains smaller deviations for larger values of b_ξ and lower ones of a_ξ, where especially lots of randomness occurs in the deviations of γ_0. Therefore, the impact of a_ξ and b_ξ on the scale intercepts is

overshadowed by the randomness induced by the Metropolis Hastings algorithm. The same wide variations exclusively for $a_\xi \leq 1$ cannot be obtained in the same manner as for a_τ.

The sampler exhibits the best results for the *scale* estimates for $a_\xi = 2$ and $b_\xi = 100$. However, due to the wide overall variation, these results must be taken with care.

The effect of a_ξ and b_ξ on $\boldsymbol{\beta}$ can be explained through the cross-effects of the matrix $\mathbf{W}$ and the vector $\mathbf{u}$ introduced at the beginning of this section, both containing $\boldsymbol{\gamma}$. These diminish with increasing values of the $\boldsymbol{\gamma}$ entries.

The matrix $\mathbf{W}$ affects the variance of the full conditional distribution of $\boldsymbol{\beta}$ negatively, while the mean is positively affected. Hence, larger values of a_ξ cause higher Posterior Means of the location parameters. The positive linear trend in the second half of Figure 14 for values of a_ξ is particularly interesting. For values of b_ξ, the trend comes off inferior. The wider variations of deviations for $a_\xi \leq 1$ is again not observable here. Nonetheless, the randomness observable for scale estimates does not show up for location estimates anymore.

The smallest deviations of the *location* estimates can be detected for $a_\xi = 1$ and $b_\xi = 200$.

Shortly noted, the acceptance rates of the Metropolis Hastings algorithm for sampling $\boldsymbol{\gamma}$ are always between 0.31 and 0.53. For the value range of a_τ, b_τ and a_ξ, no distinct pattern is observable in this regard. With growing values of b_ξ, however, acceptance rates are more likely to grow. Since the acceptance rates are in reasonable ranges enabling statistically valid estimation, these results are not further investigated here.

Conclusion