

Bayesian Ridge Regression - Second Report

Dominik Strache, Nicolai Bäuerle & Joel Beck

Introduction

Our first report was focused on the underlying mathematical model of Bayesian Ridge Regression and provided a brief introduction of the `asp21bridge` package. Since then, the central `mcmc_ridge()` function, which implements the Markov Chain Monte Carlo sampler and whose name has been adapted to meet implicit naming conventions of other groups, has been slightly extended as well as fully unit tested and now shows a stable and reliable performance.

This report investigates the effect of manipulating some of the 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 of this report will be conducted on the following components:

- The **data** input (section 1 and section 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) and the **number of simulations** `num_sim` (section 4). These sections investigate, if a stabilization process happens 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 5) of the Inverse Gamma prior distribution of the variance parameters τ and ξ , 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, implementation as well as 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.

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 $y_i \sim \mathcal{N}(\mathbf{x}_i^T \boldsymbol{\beta}, \exp(\mathbf{z}_i^T \boldsymbol{\gamma})^2)$. All these conditions lead to an excellent performance of the MCMC sampler, but might arguably not represent the most challenging task.

The next two sections 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()` estimator and the Markov Chain Monte Carlo `mcmc()` sampler from the `lmls` package.

1.1 Simulation Setting

- The design matrix $\mathbf{X} = (\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3)$ is simulated from a three dimensional normal distribution $\mathcal{N}_3(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean vector $\boldsymbol{\mu} = (-5 \ 2 \ 0)^T$ 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 $\mathbf{Z} = (\mathbf{z}_1 \ \mathbf{z}_2)$ 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 $\mathbf{z}_2 = \mathbf{x}_2 - 0.5 \cdot \mathbf{x}_3$.
- In both design matrices intercept columns are added for estimation purposes. The true coefficient vectors are given by $\boldsymbol{\beta} = (\beta_0 \ \beta_1 \ \beta_2 \ \beta_4)^T = (0 \ 3 \ -1 \ 1)^T$ and $\boldsymbol{\gamma} = (\gamma_0 \ \gamma_1 \ \gamma_2)^T = (0 \ 2 \ 0)^T$.
- Three different values (0, -0.5 and 0.9) were chosen for ρ 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.

1.2 Simulation Results

Figure 1 displays the posterior mean estimates for the MCMC samplers and the Maximum Likelihood estimates for the `lmls()` function of one complete iteration of the simulation. 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 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 . 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 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 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 estimation procedures, the above procedure is repeated 50 times. The black points 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 2:

There are three conclusions from this first simulation study:

Model Performance for different Predictor Correlation Structures

True coefficient values are indicated by grey circles

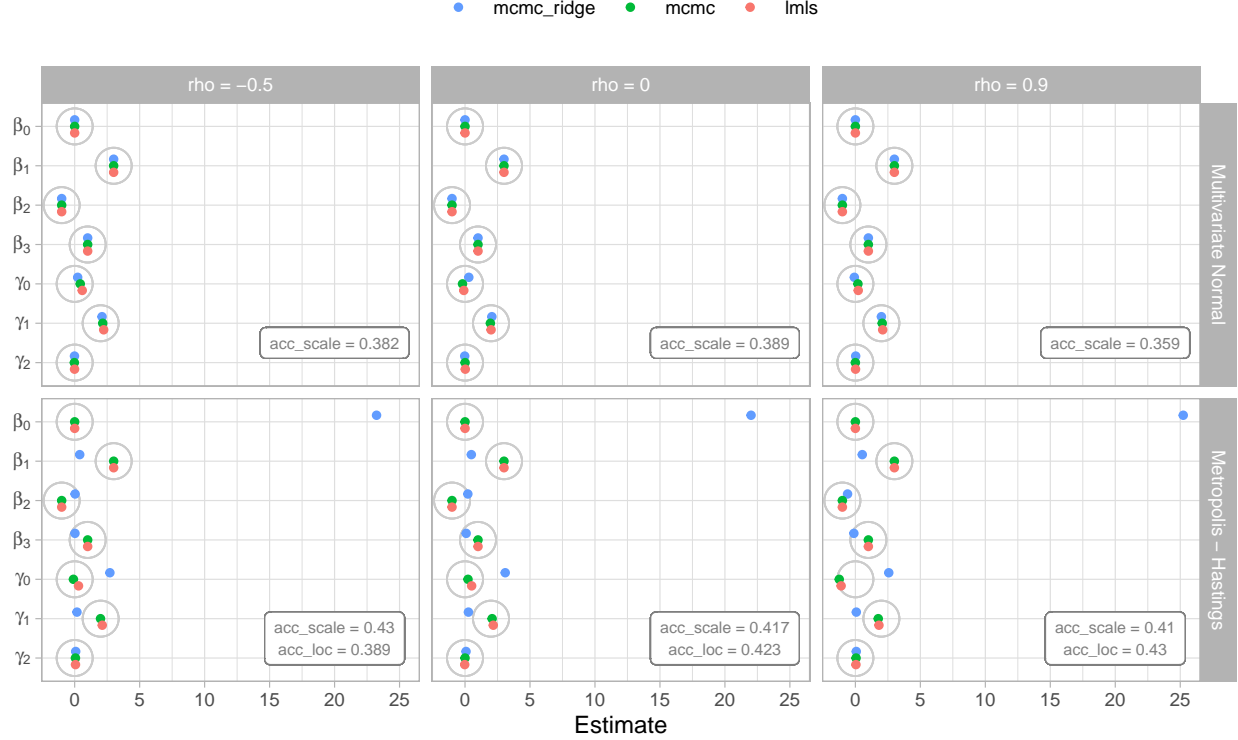


Figure 1: Comparison of Correlation Structures - one Simulation

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) provide close to unbiased estimates.
3. The variability among the β estimates is almost nonexistent, such that results from a single simulation are already reliable and representative. While the estimates for the γ 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.

2 Challenging the Model Assumptions

2.1 Simulation Setting

2.2 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 illustrated 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

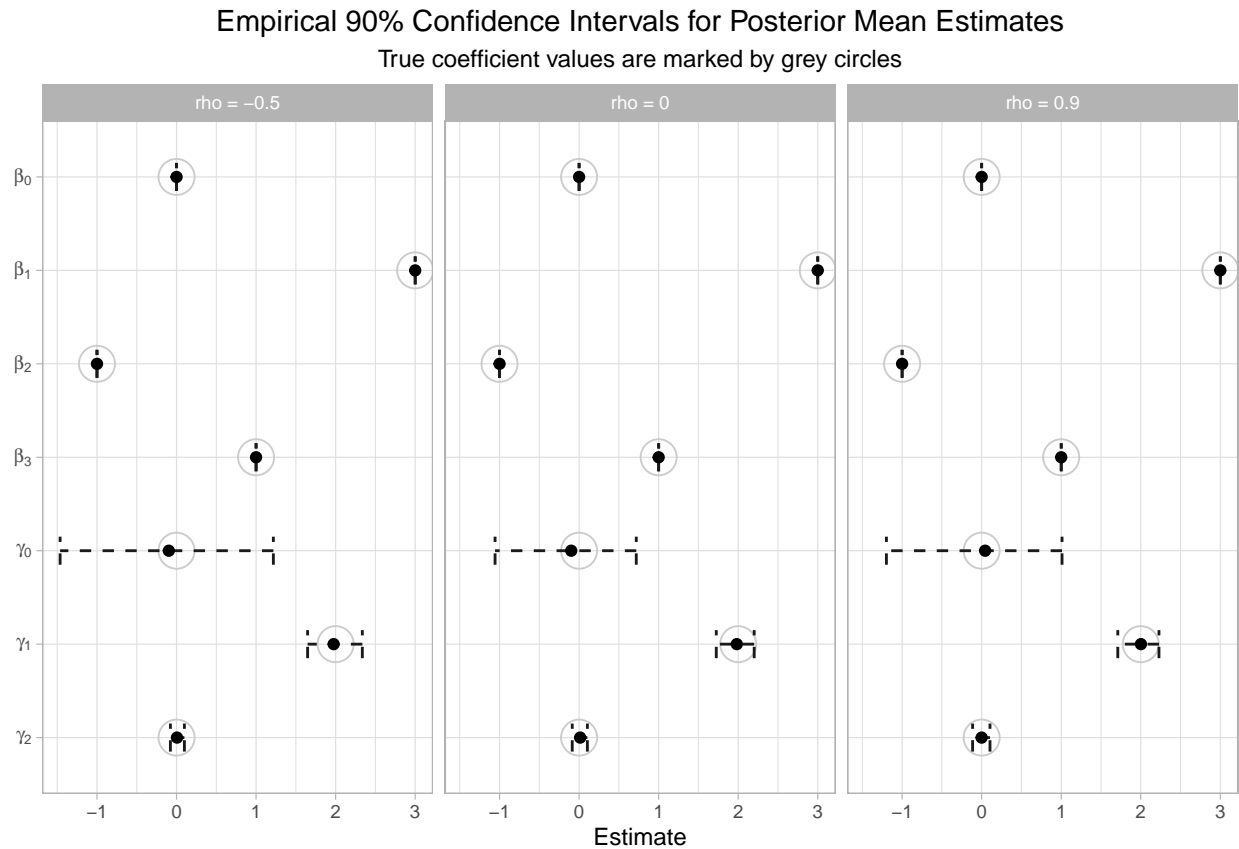


Figure 2: Comparison of Correlation Structures - 50 Simulations

well suited for simulation studies and provides some structural as well as minor performance advantages compared to the classical `for-loop` approach, which, however, are beyond the scope of this report.

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)
  ))
```

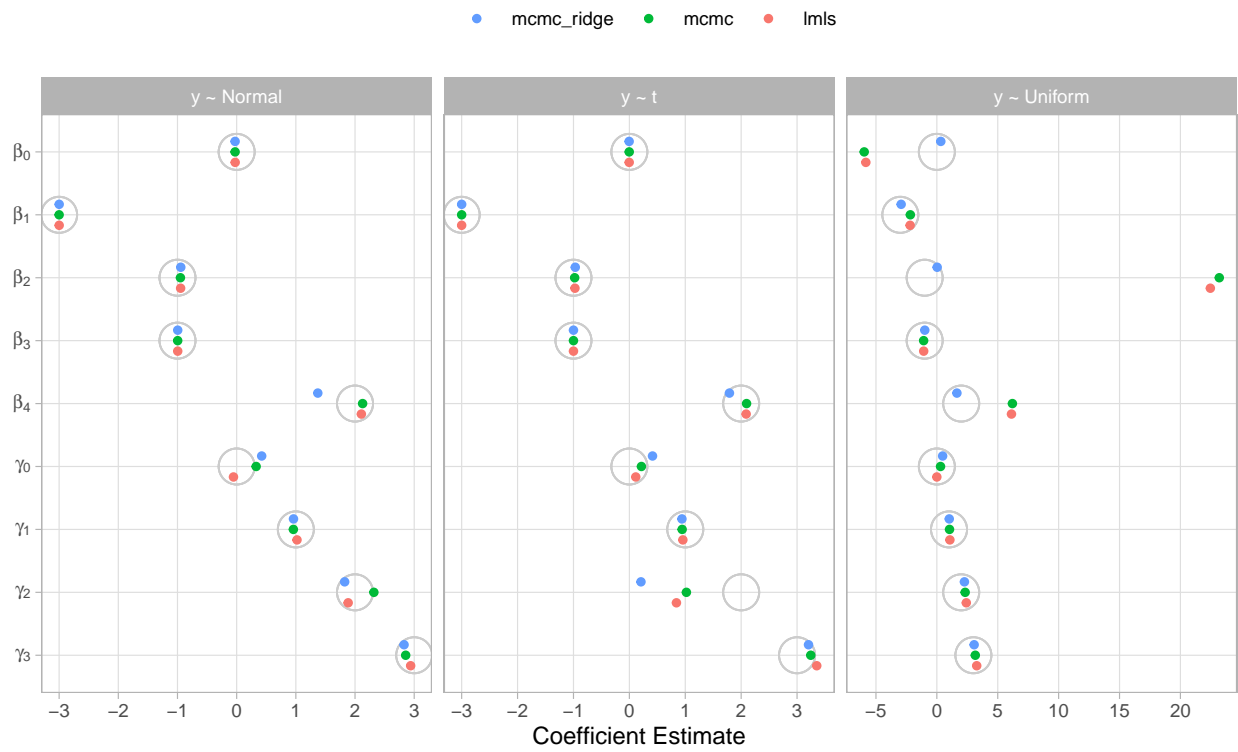
The `plan()` function borrowed from the `future` package initializes the parallel computing process and the number of cores available for computation. The `show_results()` helper function fits all three models `mcmc_ridge()`, `mcmc()` and `lmls()` for each outcome distribution in a single iteration.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

Finally, the `.options()` argument allows the specification of a random seed. Random number generation in the context of parallel computing is slightly more involved than in 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.

2.3 Simulation Results

Posterior Means / MLE for (misspecified) Regression Models

True coefficient values are marked by grey circles



Empirical 90% Confidence Intervals for Posterior Mean Estimates

True coefficient values are marked by grey circles

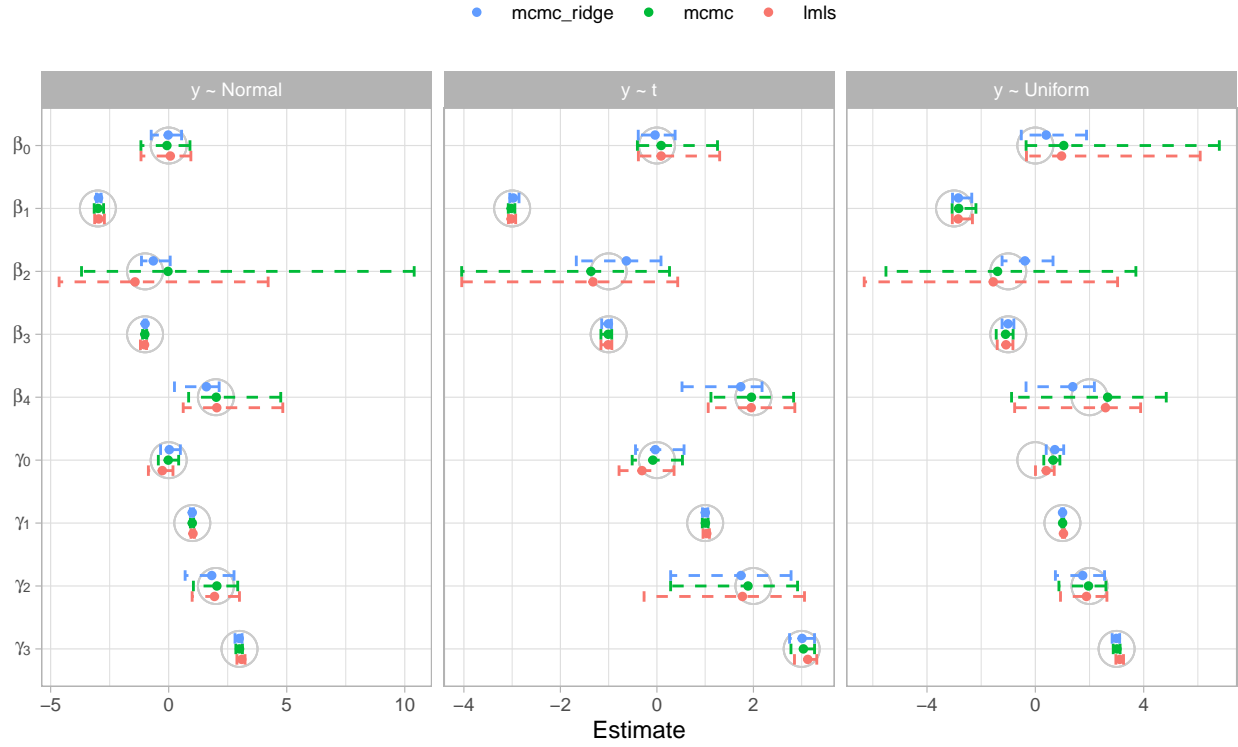


Table 1: Bias of Coefficient Estimates

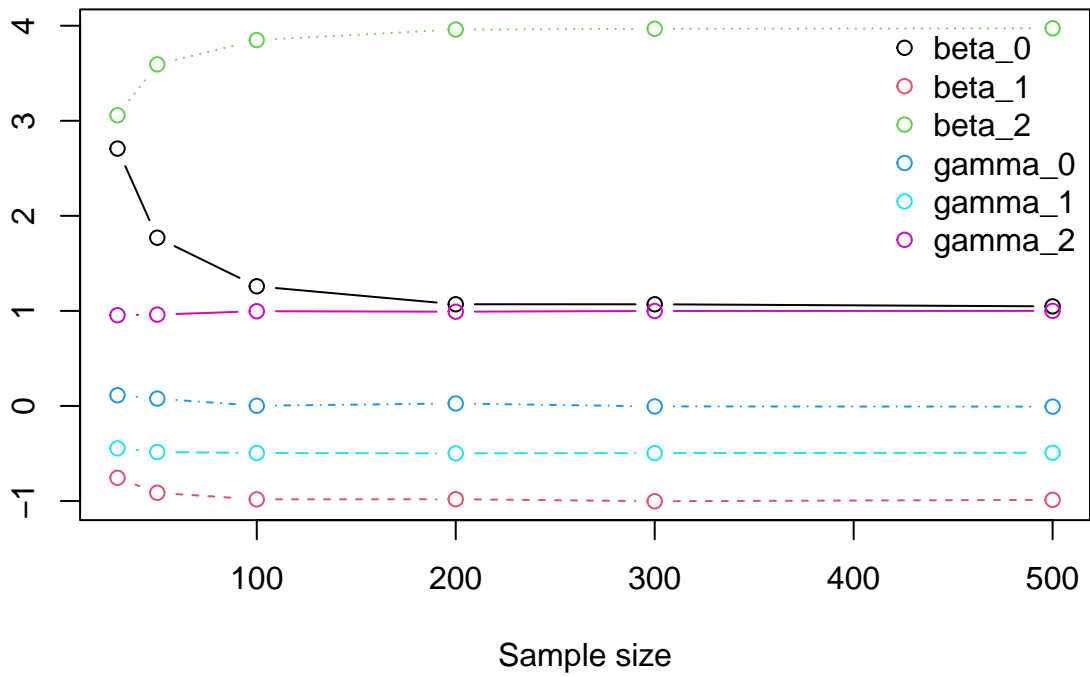
	Normal			t			Uniform		
	lmls	mcmc	mcmc_ridge	lmls	mcmc	mcmc_ridge	lmls	mcmc	mcmc_ridge
β_0	0.07	-0.07	-0.02	0.09	0.09	-0.04	0.97	1.05	0.40
β_2	-0.43	0.97	0.35	-0.32	-0.37	0.37	-0.55	-0.39	0.61
β_4	0.04	0.01	-0.40	-0.05	-0.04	-0.26	0.59	0.67	-0.62
γ_0	-0.27	-0.02	0.03	-0.31	-0.08	-0.03	0.40	0.66	0.72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

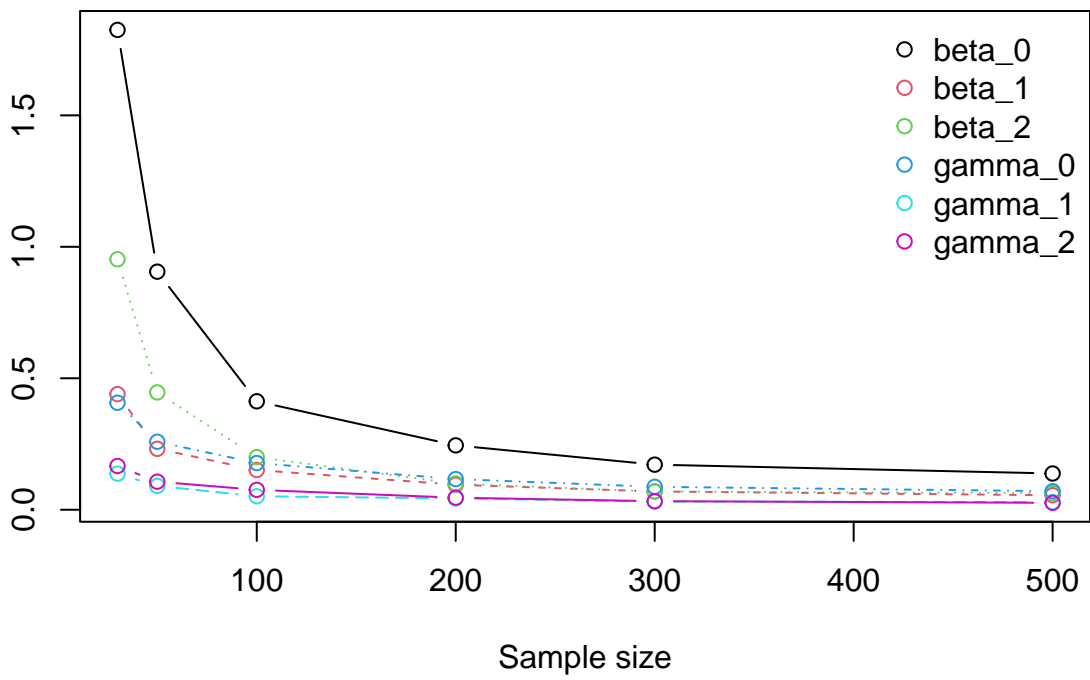
	Normal			t			Uniform		
	lmls	mcmc	mcmc_ridge	lmls	mcmc	mcmc_ridge	lmls	mcmc	mcmc_ridge
β_0	0.83	0.69	0.47	0.62	0.62	0.49	2.15	2.37	1.47
β_2	7.11	4.41	0.40	2.54	2.58	0.54	6.48	6.48	0.67
β_4	1.32	1.33	0.68	0.57	0.56	0.53	4.10	4.30	0.81
γ_0	0.33	0.28	0.27	0.35	0.33	0.34	0.22	0.19	0.21
γ_2	0.78	0.65	0.62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

3 Sample Size

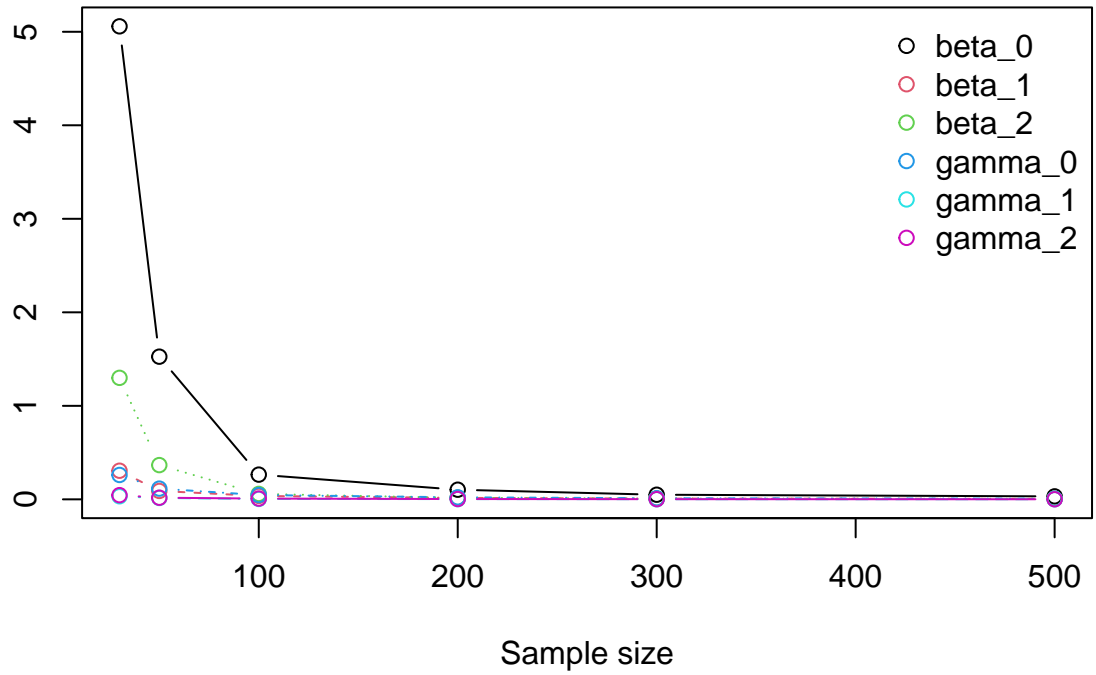
Mean of Posterior Means



MAE of Posterior Means

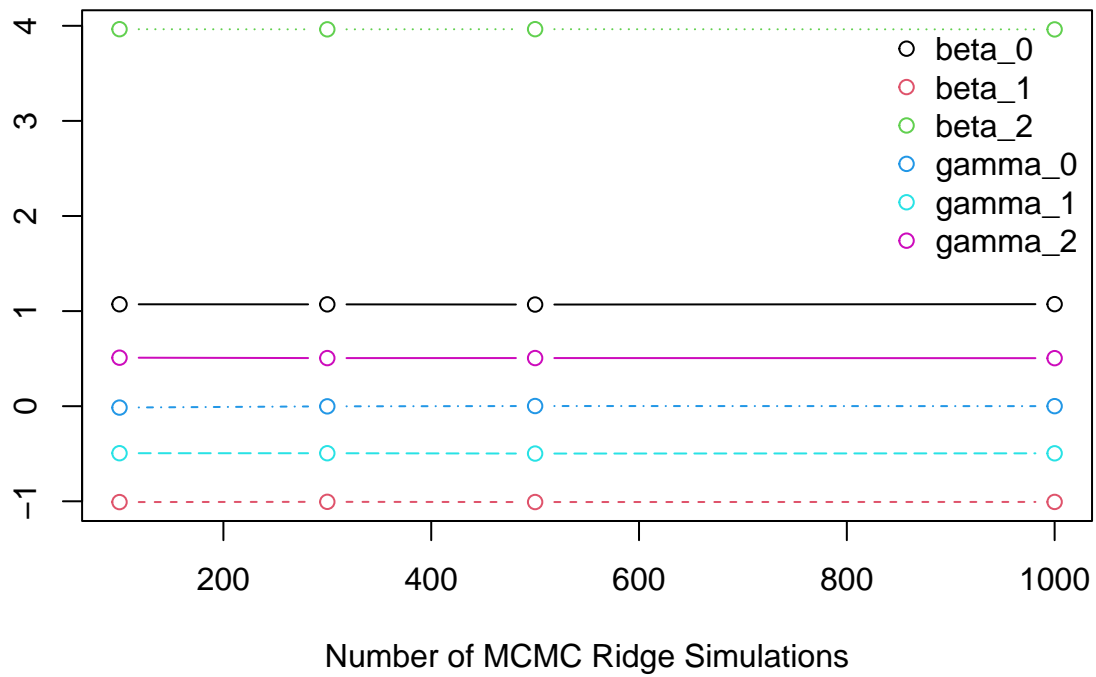


MSE of Posterior Means

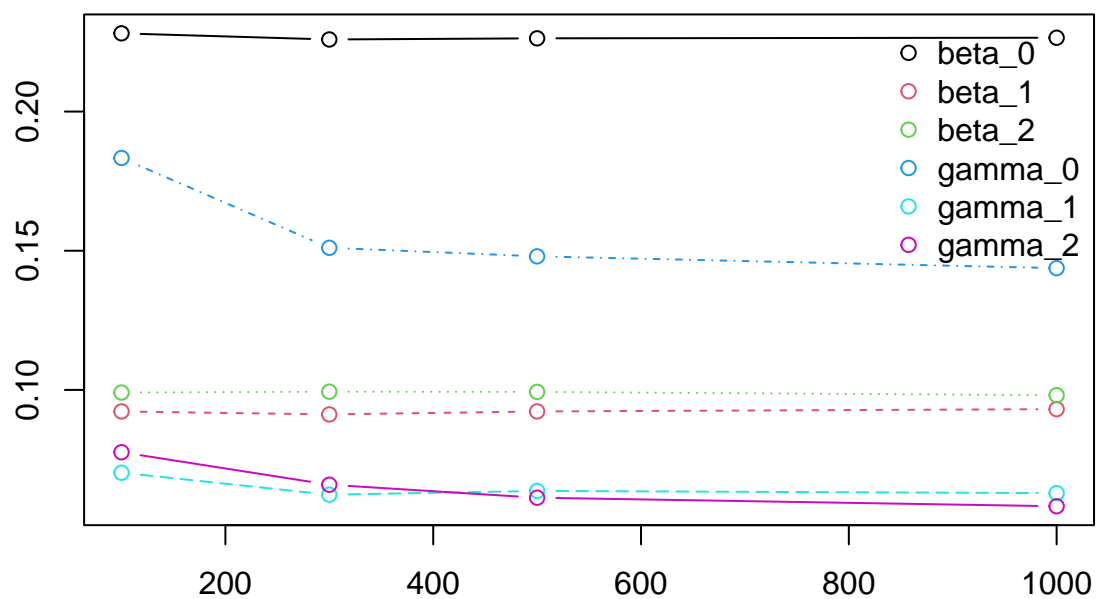


4 Number of Simulations

Mean of Posterior Means

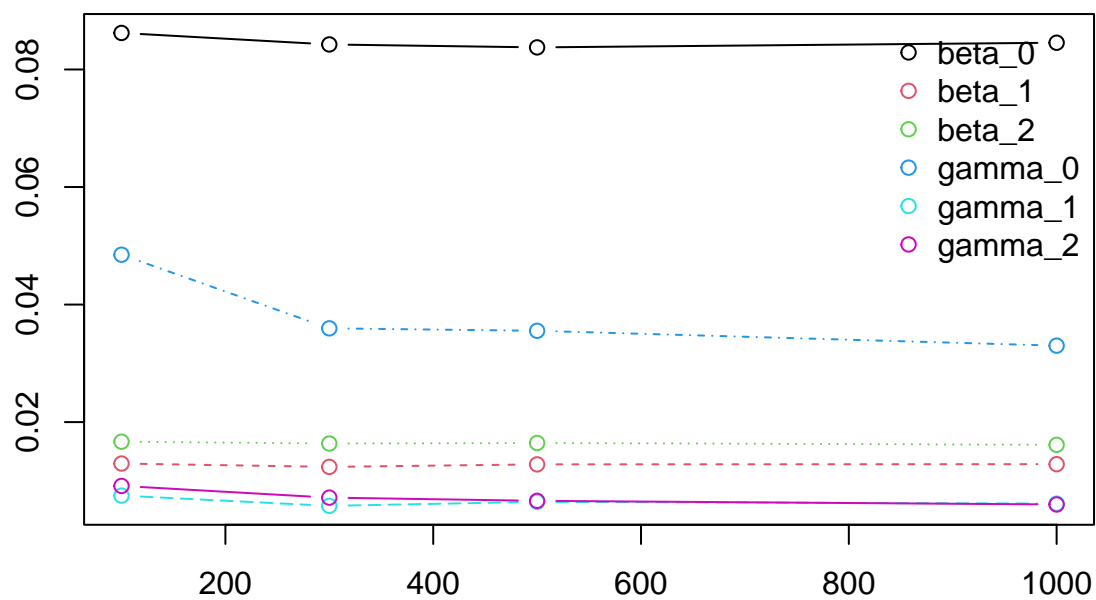


MAE of Posterior Means

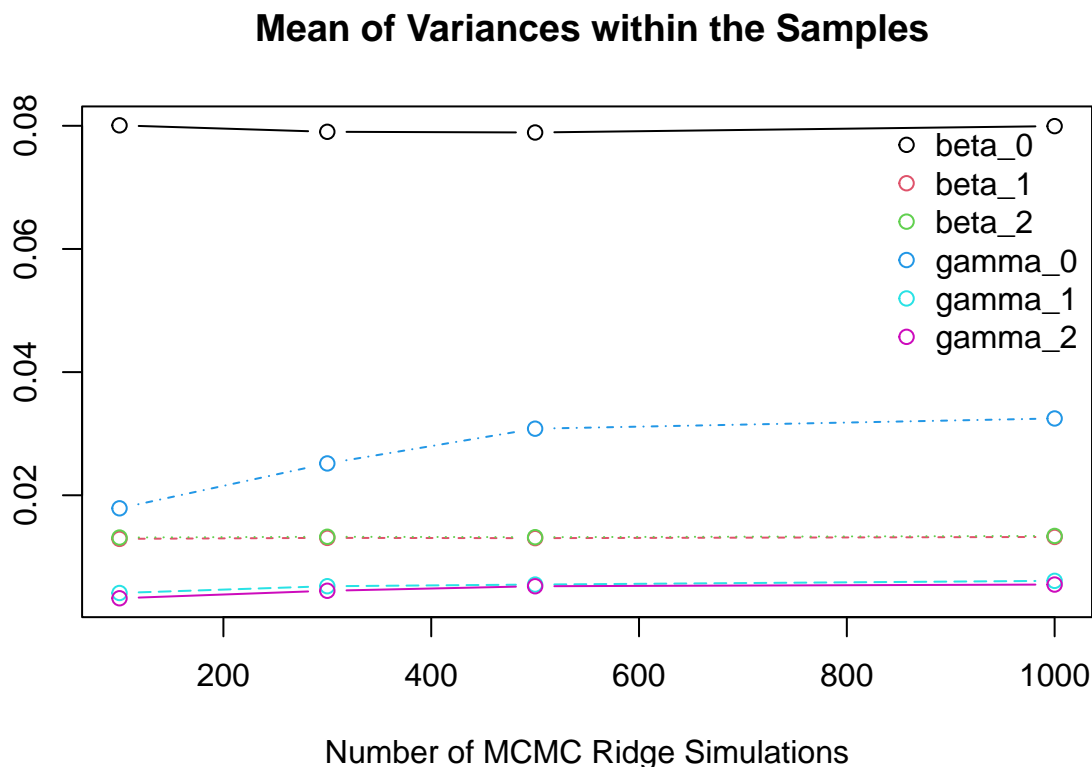


Number of MCMC Ridge Simulations

MSE of Posterior Means



Number of MCMC Ridge Simulations



5 Hyperparameters

Next Steps

In the process of writing the first and second report, the main work of developing both the mathematical foundation as well as the code base of the `asp21bridge` package has been done. Thus, the following weeks will focus on filling remaining gaps and working out details that were left open due to time constraints. This includes taking a closer look at code efficiency and possible code refactoring for the sake of modularity.

Further, we will reach out to other groups with similar topics (the Bayesian Lasso Regression group in particular) in order to initiate a collaboration. In addition to exploring similarities and differences between our own sampler and the procedures from the `lmls` package, this would allow for an interesting comparison between different penalization approaches.

The final step consists of combining all pieces of the project into a single structured and consistent final report. Aside of the work shown in the first two reports, some new elements like a brief discussion of design and implementation choices during the development stage will be added with the aim of providing the reader a solid understanding of the underlying ideas as well as the practical usage of the package.