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 to 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 existing 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 (sections 1 and 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, 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 simulation are included in the `simulation-studies` folder inside the `asp21bridge` package. Thus, each figure as well as all numerical results are fully reproducible and can be repeated and extended by the reader.

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 sampler, but might arguably not represent the most challenging task.

Sections 1 and 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()` estimator and the Markov Chain Monte Carlo `mcmc()` sampler from the `lmls` package.

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 $\mathbf{z}_2 = \mathbf{x}_2 - 0.5 \cdot \mathbf{x}_3$.
- 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 (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.

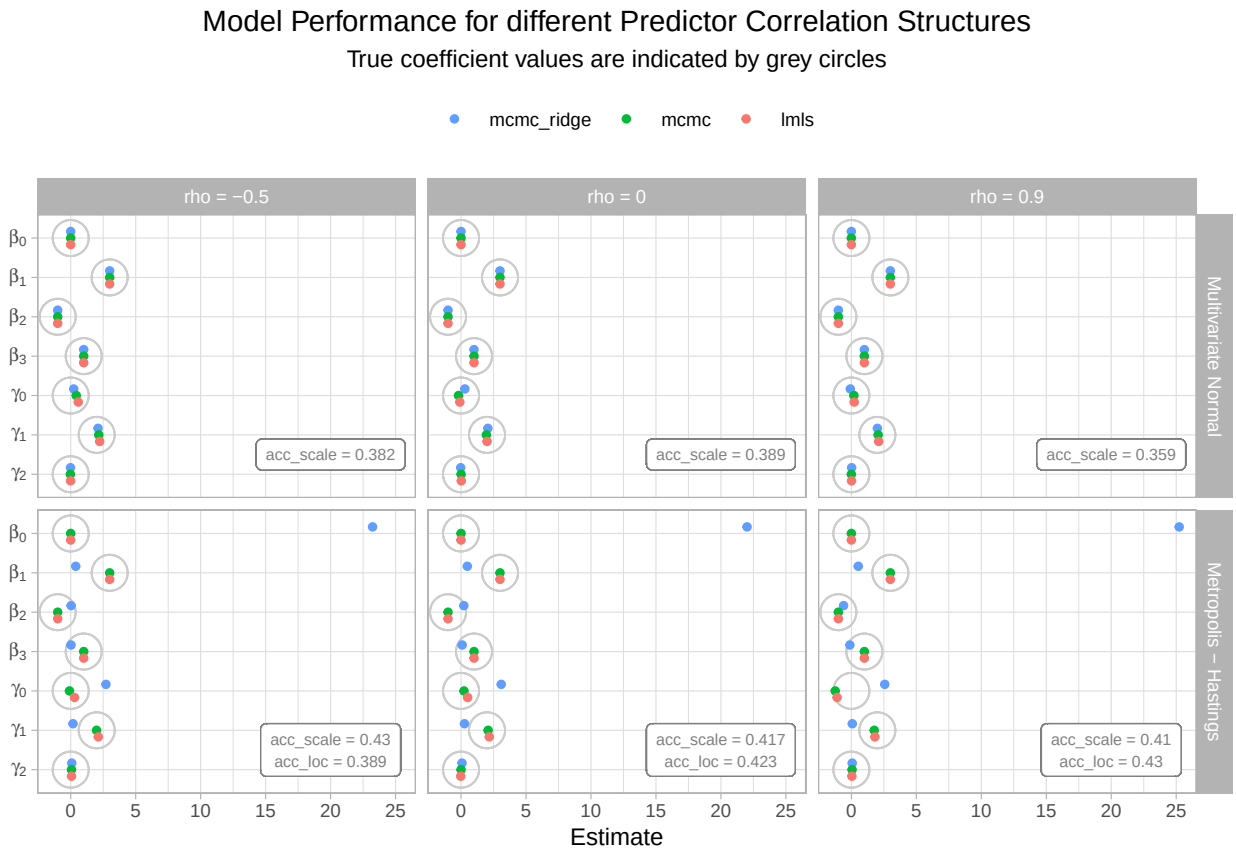


Figure 1: Comparison of Correlation Structures - One Simulation Cycle

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:

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 $\boldsymbol{\beta}$ estimates is almost nonexistent, such that results from a single simulation 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.

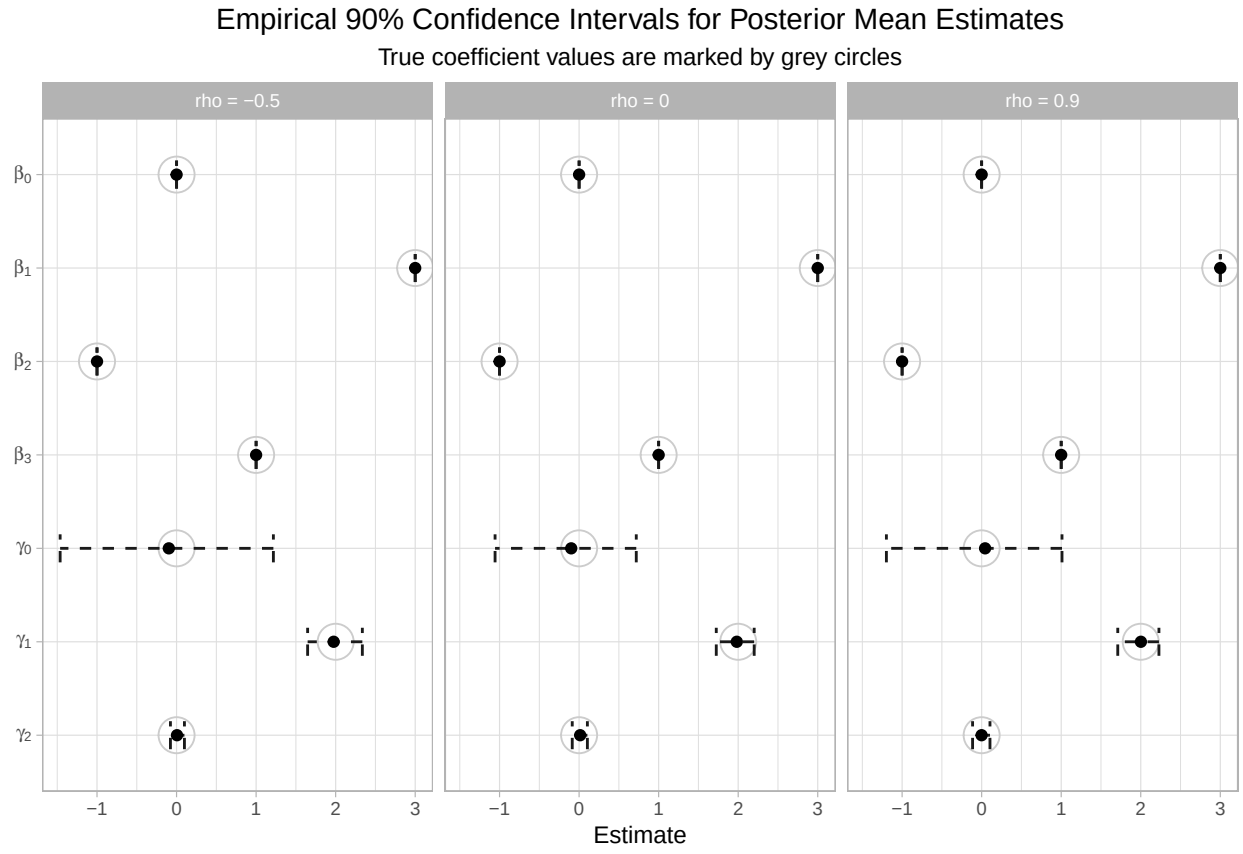


Figure 2: Comparison of Correlation Structures - 50 Simulation Cycles

2 Challenging the Model Assumptions

This simulation study is structured in a very similar way to the study considered in section 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.

2.1 Simulation Setting

- The design matrix $\mathbf{X} = \begin{pmatrix} \mathbf{1}_n & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \end{pmatrix}^T$ 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 $\mathbf{Z} = \begin{pmatrix} \mathbf{1}_n & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{z}_3 \end{pmatrix}^T$ contains the additional regressor variable $\mathbf{x}_3 \overset{iid}{\sim} t_{10}$, which is independently sampled from all other columns.
- 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 $\boldsymbol{\gamma} = \begin{pmatrix} \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 \end{pmatrix}^T = \begin{pmatrix} 0 & 1 & 2 & 3 \end{pmatrix}^T$.

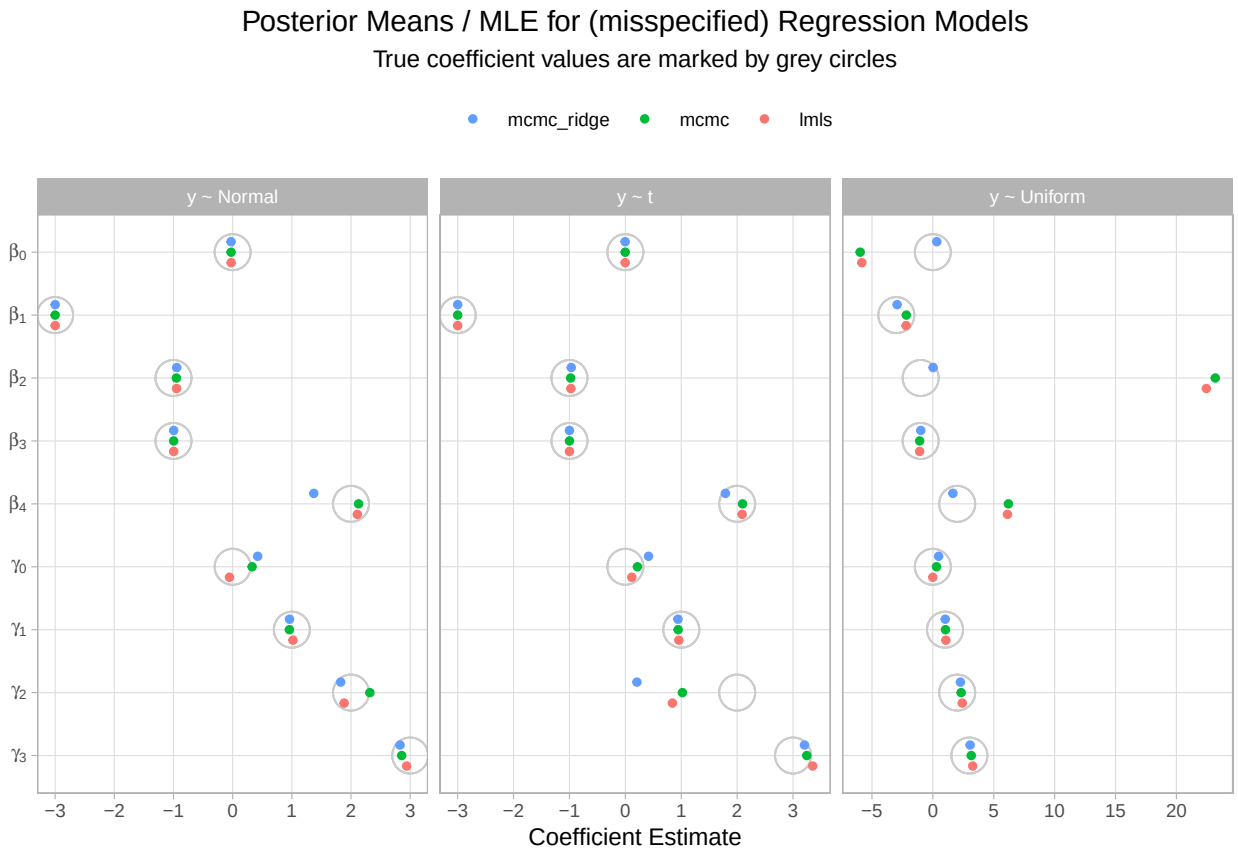


Figure 3: Comparison of Outcome Distributions - One Simulation Cycle

- Three different specifications for the outcome distribution were chosen:
  - $y_i \sim \mathcal{N}(\mu, \sigma^2)$
  - $y_i \sim \mu + \left(\sigma \cdot \sqrt{\frac{3}{5}}\right) T$, where $T \sim t_5$
  - $y_i \sim \mu + \sigma \cdot U$, where $U \sim \mathcal{U}([0,\ 1])$

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 $y_i \sim \mathcal{N}(\mu, \sigma^2)$. 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 the model assumptions.

2.2 Simulation Results

Just as in section 1 the results of one complete iteration (each of the $3 \cdot 3 = 9$ models was fitted once / each data point represents one estimate) are displayed in Figure 3. 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

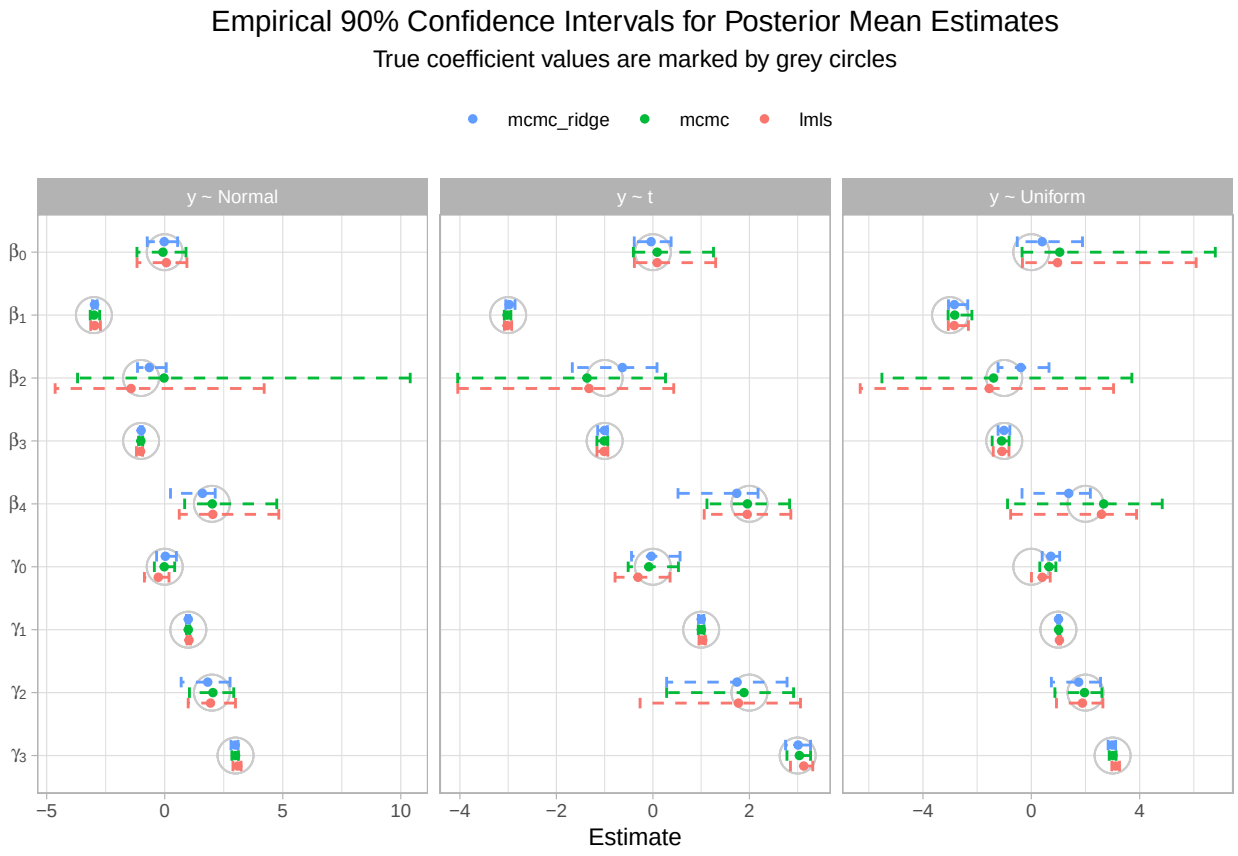


Figure 4: Comparison of Outcome Distributions - 50 Simulation Cycles

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 1) are plotted in Figure 4.

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.

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

Considering the bias estimates 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.

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

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 4) 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 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 interpretation.

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:

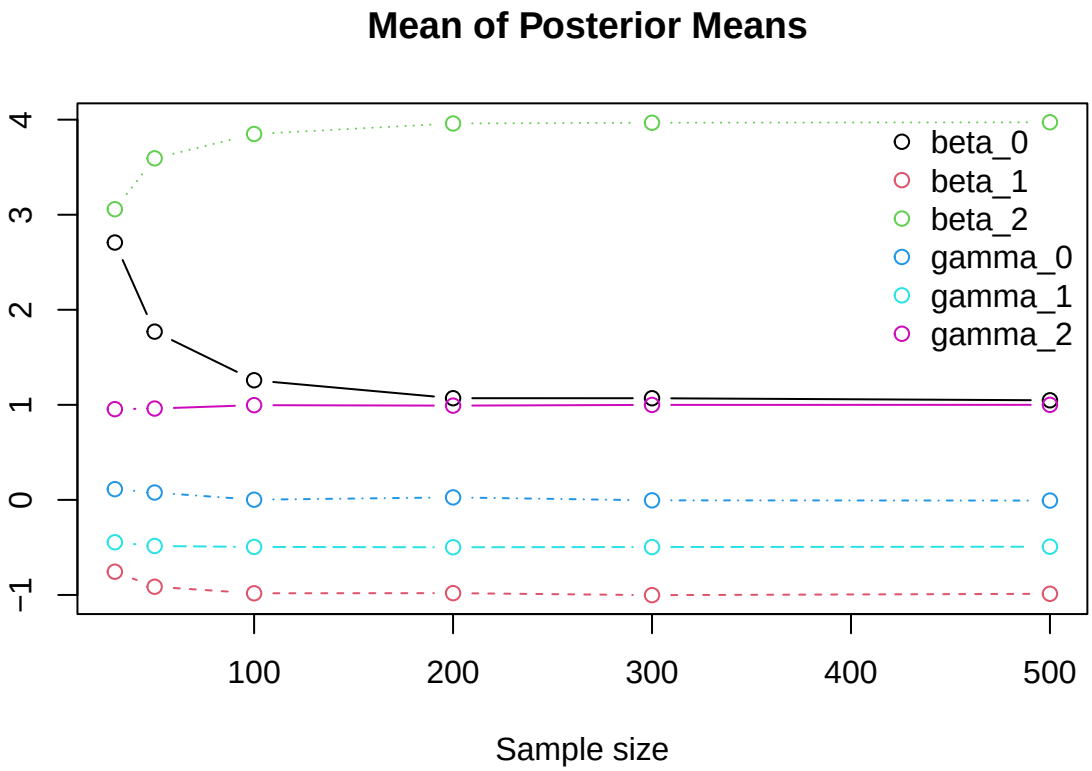
```
plan(multisession, workers = 8)

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)
  ))
```

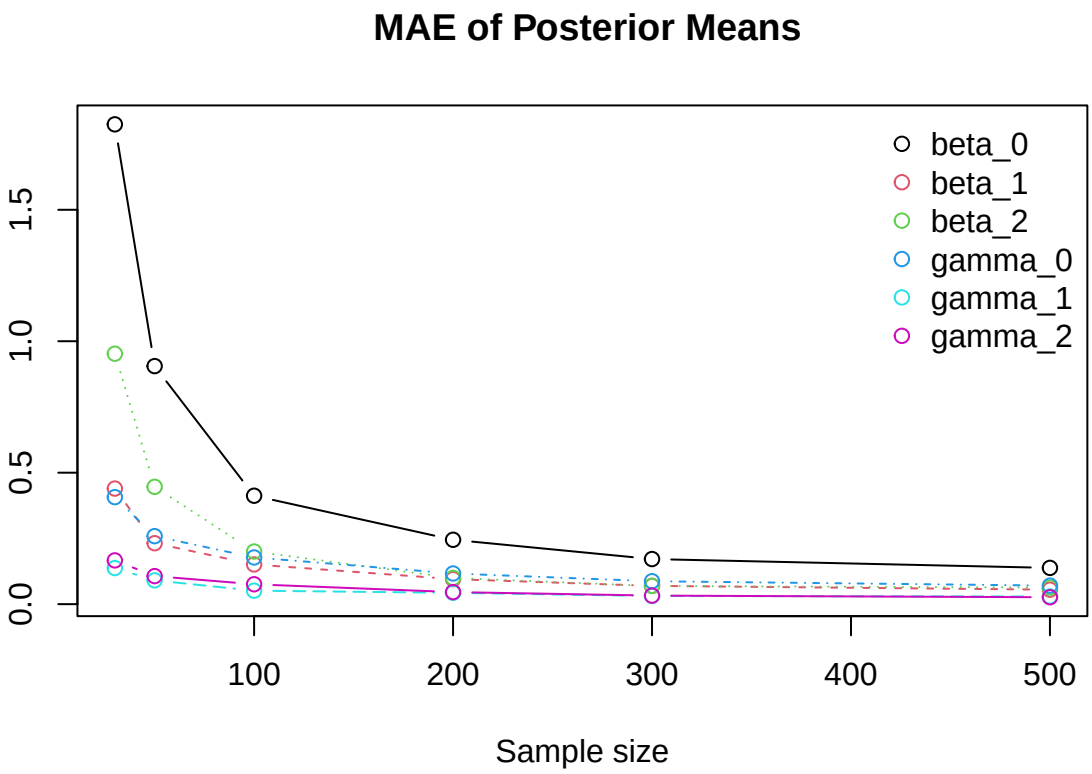
The `plan()` function borrowed from the `future` package initializes the parallel computing process and the number of cores/workers available for computation. The `show_results()` helper function fits all three models `mcmc_ridge()`, `mcmc()` and `lmls()` for each outcome distribution in a single simulation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

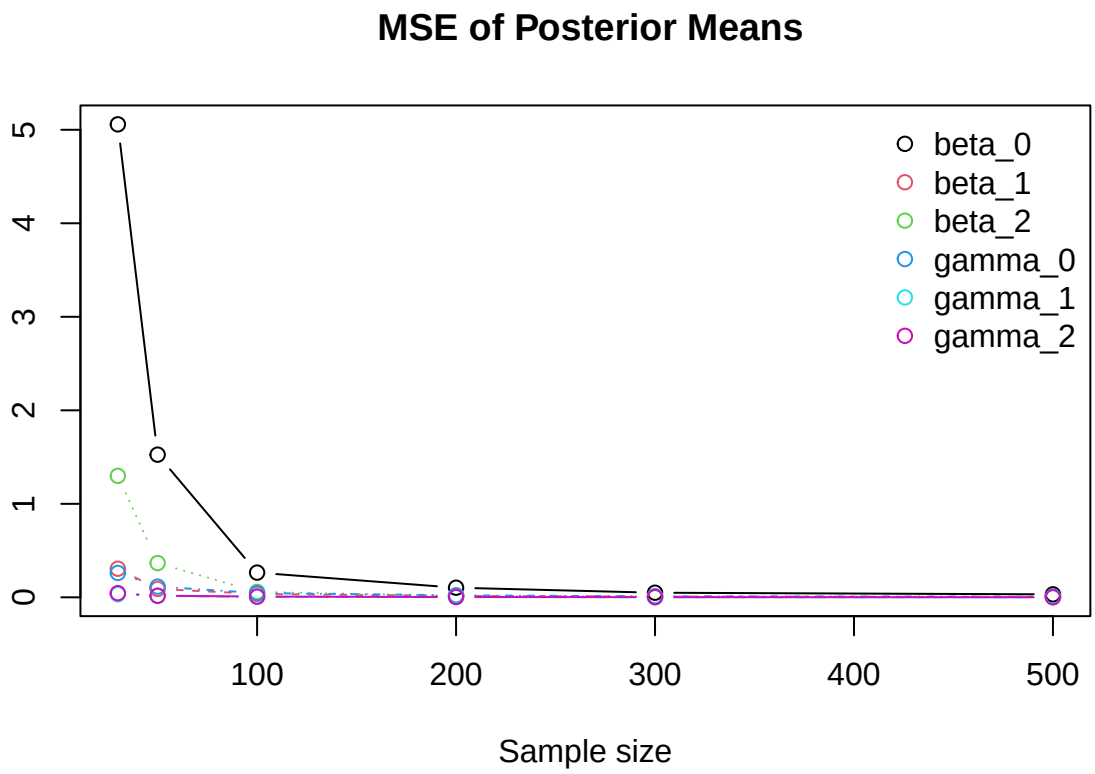
This entire procedure is repeated 50 times in parallel using the `future_map()` function from the `furrr` package, where the results of all 450 models are saved in a well organized structure inside of a list column. This new column of the data frame contains complete information about all simulations,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

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.

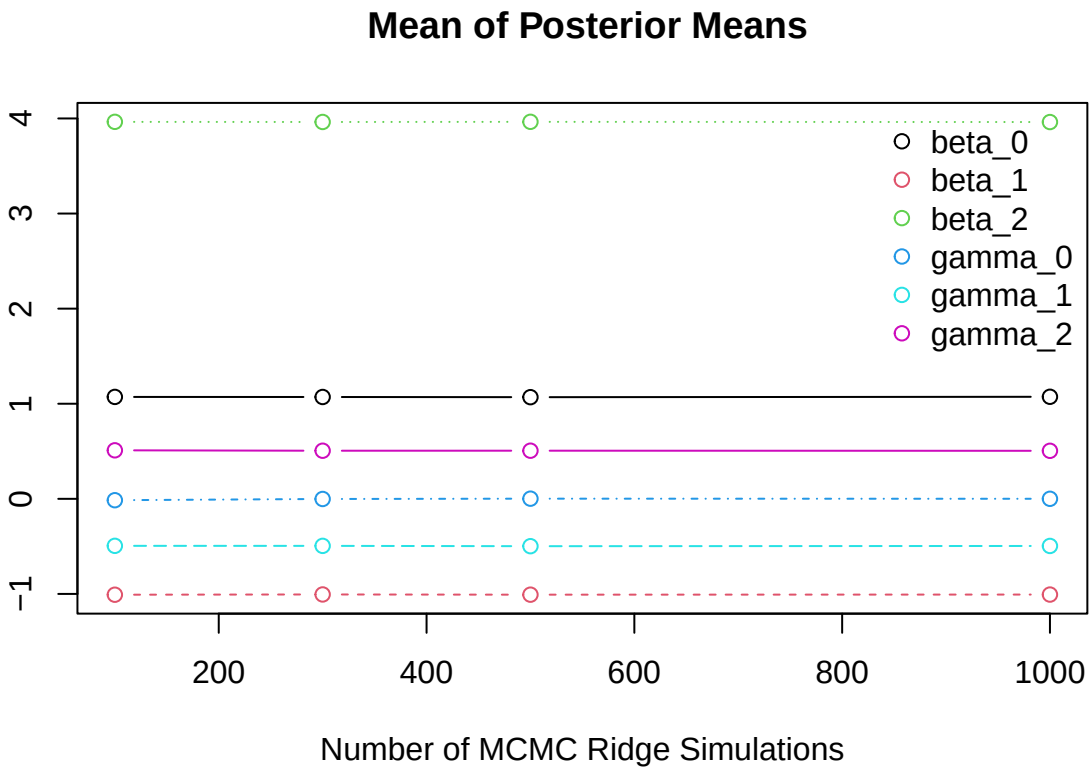
3 Sample Size

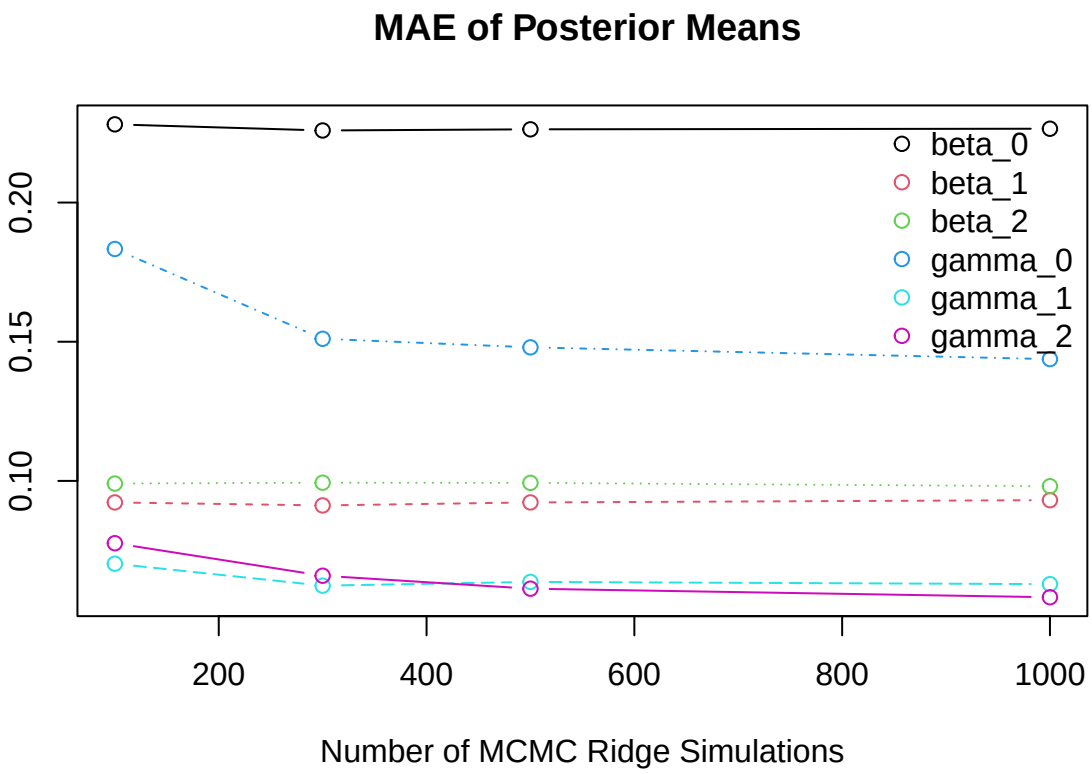


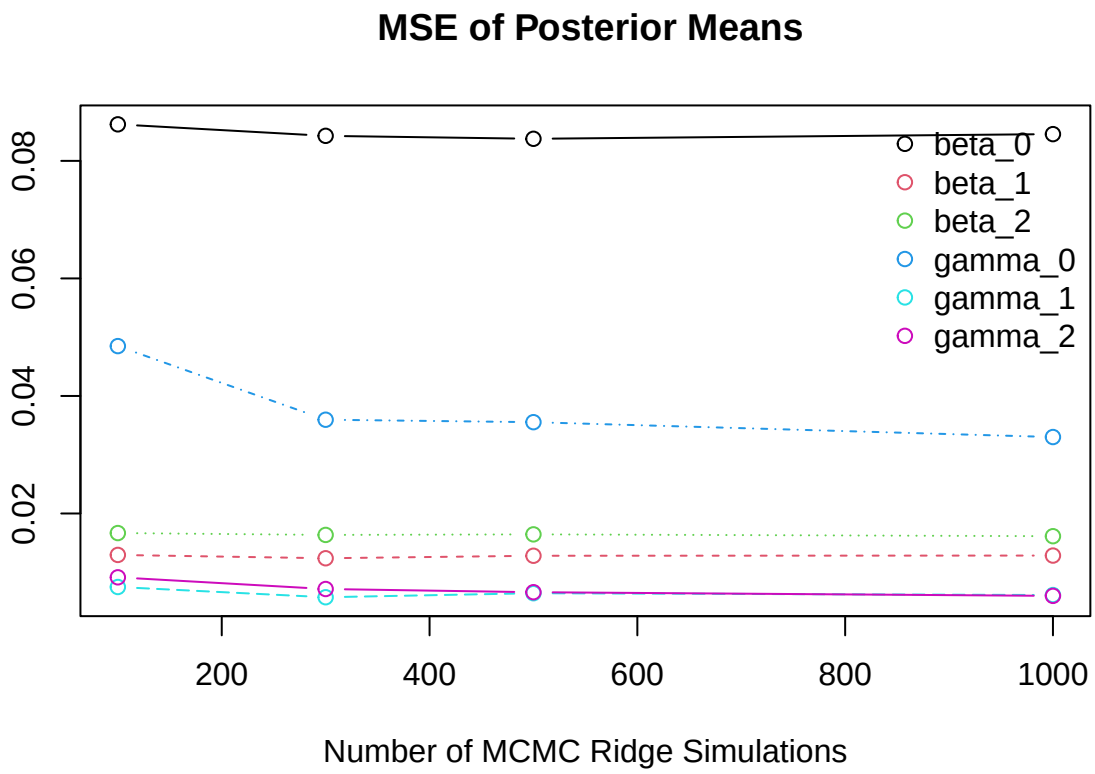

MAE of Posterior Means
beta_0
beta_1
beta_2
gamma_0
gamma_1
gamma_2
0.0
0.5
1.0
1.5
100
200
300
400
500
Sample size

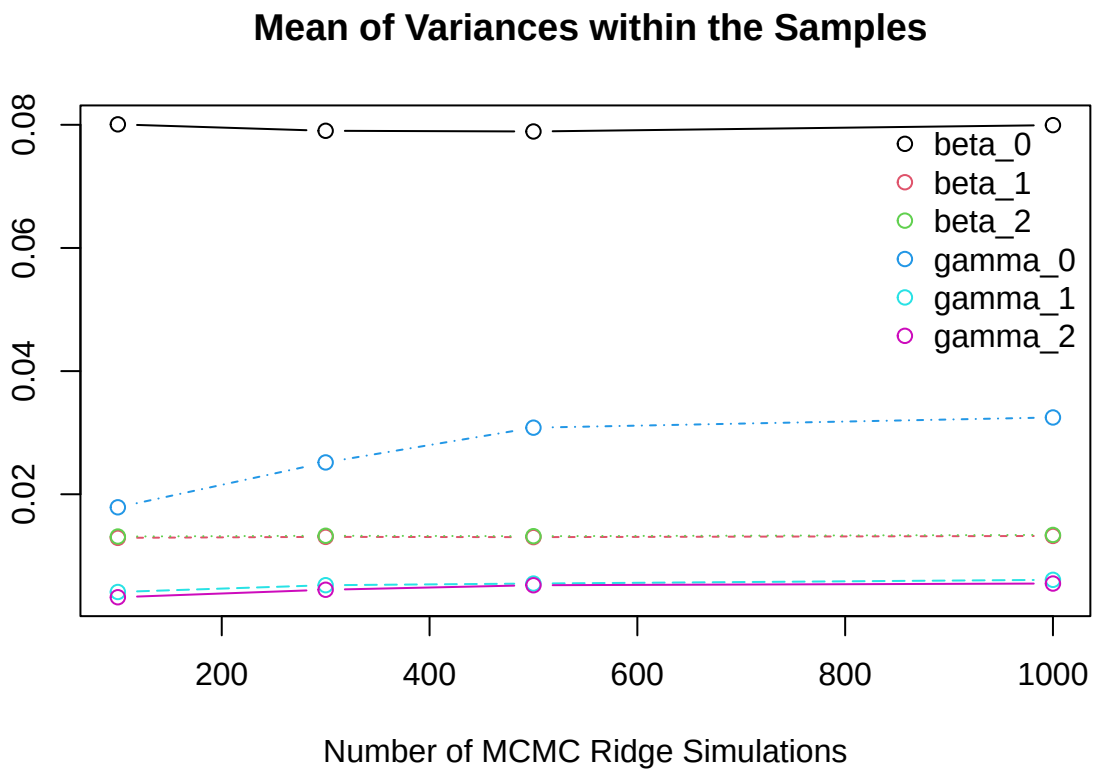

MSE of Posterior Means
beta_0
beta_1
beta_2
gamma_0
gamma_1
gamma_2
0
1
2
3
4
5
100
200
300
400
500
Sample size

4 Number of Simulations




MAE of Posterior Means
0.10
0.15
0.20
200
400
600
800
1000
Number of MCMC Ridge Simulations
beta_0
beta_1
beta_2
gamma_0
gamma_1
gamma_2


MSE of Posterior Means
0.02
0.04
0.06
0.08
200
400
600
800
1000
Number of MCMC Ridge Simulations
beta_0
beta_1
beta_2
gamma_0
gamma_1
gamma_2



5 Hyperparameters

In the past, we have been sampling data by our `mcmc_ridge` function without having a closer look on the effect of the hyperparameters `a_tau`, `b_tau`, `a_xi` and `b_xi`. However, they affect the full conditionals of τ^2 and ξ^2, which are Inverse Gamma distributed, directly as stated here again for convenience:

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

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

Simulating data samples for different hyperparameters values is therefore inevitable.

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 $\mathcal{N}_2\left(\boldsymbol{\mu}, \sigma^2\mathbf{I}\right)$ with mean vector $\boldsymbol{\mu} = \begin{pmatrix}1 & 2\end{pmatrix}^T$ and unit variance $\boldsymbol{\sigma_1^2 = \sigma_2^2 = 1}$. Same as the design matrix $\mathbf{Z} = \begin{pmatrix}\mathbf{z}_1 & \mathbf{z}_2\end{pmatrix}$ with mean vector $\boldsymbol{\mu} = \begin{pmatrix}5 & 3\end{pmatrix}^T$ also having a unit variance.
- 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 simulating the influence of the hyperparameters, nine different values are chosen for $\boldsymbol{a_\tau = b_\tau = a_\xi = b_{xi}} = \begin{pmatrix}\mathbf{-1} & \mathbf{0} & \mathbf{0.5} & \mathbf{1} & \mathbf{2} & \mathbf{10} & \mathbf{50} & \mathbf{100} & \mathbf{200}\end{pmatrix}$. Since for properties like

the mean of an Inverse Gamma distribution, values for $\boldsymbol{a_\tau = a_\xi > 0}$ are required, particular attention is focused on larger values. Anyway, it is an aim to inspect the performance of the sampler for values $\leq \mathbf{1}$ as well.

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.