The asp21bridge Package

1 The asp21bridge Package

This chapter introduces various facettes of the `asp21bridge` package. Section 1.1 explains how to use and combine the functions that are contained in the package most efficiently. Section 1.2 focuses on the implementation of the Markov Chain Monte Carlo Sampler with Ridge Penalty, links the source code to the mathematical model from chapter **??** 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 1.3.

1.1 User Guide

This section aims to provide help 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, that 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 1.2.1.
- *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.
- *Graphical Analysis:* The `trace_plot()`, `density_plot()` and `acf_plot()` functions provide the three most common visualization of a *single* chain's development over time, their 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 contained in the lower panel.

1.1.1 Sampling with the `mcmc_ridge()` function

As explained in section 1.2.1 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 simulation `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 **??**), must be manually specified.

We illustrate both use cases with the built-in data set `toy_data`, which is specifically designed for introductory tutorials, documentation and unit tests, and the more realistic `abdom` data set from the `lmls` package. In the former case, we use most of the default settings, except for number of simulations which we increase to 10000. In the latter 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*:

```
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)
```

```
# str(toy_fit)
# str(abdom_fit)
```

1.1.2 Statistical Analysis

1.1.3 Graphical Analysis

1.2 Implementation of the Markov Chain Monte Carlo Sampler

1.2.1 Iterative Parameter Sampling

1.2.2 Metropolis Hastings Step

1.3 Package Development

1.3.1 Code Coverage

- unit tests
- documentation

1.3.2 Design Choices

- pipe operator
- S3 Class