7-2021

Evaluating The Efficiency of Markov Chain Monte Carlo Algorithms

Thuy Scanlon
University of Arkansas, Fayetteville

Evaluating The Efficiency of Markov Chain Monte Carlo Algorithms

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Statistics and Analytics

by

Thuy Scanlon
University of Houston
Bachelor of Science in Life Science, 2009

July 2021
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

John Tipton Ph.D.
Thesis Director

Qingyang Zhang Ph.D.
Committee Member

Avishek Chakraborty Ph.D.
Committee Member

Abstract

Markov chain Monte Carlo (MCMC) is a simulation technique that produces a Markov chain designed to converge to a stationary distribution. In Bayesian statistics, MCMC is used to obtain samples from a posterior distribution for inference. To ensure the accuracy of estimates using MCMC samples, the convergence to the stationary distribution of an MCMC algorithm has to be checked. As computation time is a resource, optimizing the efficiency of an MCMC algorithm in terms of effective sample size (ESS) per time unit is an important goal for statisticians. In this paper, we use simulation studies to demonstrate how the Gibbs sampler and the Metropolis-Hasting algorithm works and how MCMC diagnostic tests are used to check for MCMC convergence. We investigated and compared the efficiency of different MCMC algorithms fit a linear and a spatial model. Our results showed that the Gibbs sampler and the Metropolis-Hasting algorithm give estimates similar to the maximum likelihood estimates, validating the accuracy of MCMC. The results also imply that the efficiency of an MCMC algorithm can be affected by different factors. In particular, a model with more parameters could still be more efficient in terms of ESS per time unit. For fitting large datasets, algorithms whose computation involves dividing a large matrix into smaller matrices can be more efficient than algorithms that use the entire large matrix.

Acknowledgements

I would like to express my sincere gratitude to my thesis advisor, Dr. John Tipton for the great help and support he provided me through out the process of writing this thesis. Dr. Tipton not only guides me with his expert knowledge of the subject, he also helps me with my writing skills by proofreading my thesis over and over again. I cannot thank enough for his patience and care through out this whole process.

I would like to extend my gratitude to Dr. Avishek Chakraborty and Dr. Qingyang Zhang, my committee members and also my professors, Dr. Giovanni Petris, my academic advisor and also my professor, and all faculty and staff in Department of Mathematic Science at the University of Arkansas. Thank you all for your help and support during my academic studies in the STAN program.

Finally, I could not have done this thesis without the encouragement and support from my family members. My parents are the inspiration for me to pursuit higher education, which was the dream that did not come true for them due to circumstances. A special thank to my husband, James Scanlon, who has been there every step of the journey encouraging and supporting me every possible way.

Table of Contents

1 Introduction

The goal of data modeling is to make inference about parameters of interest based on available sample data and use these estimated parameters to make predictions for out of sample data. One of the approaches of estimating parameters is Bayesian statistics. Bayesian statistics use probability distributions for both the model fitting and the resulting statements about the parameters. As a result, Bayesian inference naturally accounts for uncertainty – an important measurement in inference – and results in common-sense interpretation in terms of probabilities.

In general, Bayesian inference has the following steps. First, the full probability models for the observed quantities (the response and covariates) and the unobserved quantities (the parameters that govern the probability distribution of the observed quantities) are defined. Combined, these distributions define the posterior distribution (the probability distribution of the parameters after updating based on observation data) of the unobserved quantities. To obtain the posterior distribution, one must assign prior distributions (the probability distributions for the parameters before any data is collected or considered) to the unknown parameters. The choice of a prior is determined by domain knowledge such as expert knowledge of parameters or results from previous studies. In fact, a Bayesian model is often built from hierarchical framework which is comprised of models of individual parameters. In hierarchical models, one has many opportunities to use priors that incorporate domain knowledge into the final model. Finally, the fit of the model is checked to ensure the inference is appropriate and accurate. If there is a concern of bias introduced to the model through the use of priors, model checking step can ensure the priors do not mislead the inference [Gelman et al., 2013].

The Bayesian approach to inference estimates the posterior probability distribution of

unobserved quantities using Bayes' theorem

$$[\theta|\mathbf{y}] = \frac{[\mathbf{y}|\theta][\theta]}{[\mathbf{y}]}. \tag{1}$$

Here, θ is the parameter (unobserved quantity) we wish to learn about and $\mathbf{y} = (y_1, y_2, ..., y_n)'$ is the data. The notation $[\cdot]$ is a probability density/mass function and $[\cdot|\cdot]$ is a conditional probability. In the food industry, for example, estimating shelf-life is an essential and important step of launching a new product. In this example, the parameter θ is the shelf-life – "Best if used by date" – and $\mathbf{y}$ would be the observations of the product performance at different storing durations under different temperature and moisture conditions. $[\theta]$ is defined as the prior probability of θ, $[\theta|\mathbf{y}]$ is the posterior probability of θ given $\mathbf{y}$, and $[\mathbf{y}|\theta]$ is the likelihood of $\mathbf{y}$ given θ (the probability of the data $\mathbf{y}$ given the parameter θ). Because $\mathbf{y}$ is available data, $[\mathbf{y}]$ is an unknown constant called the marginal likelihood of the data. Therefore, the posterior distribution is proportional to the likelihood times the prior distribution

$$[\theta|\mathbf{y}] \propto [\mathbf{y}|\theta][\theta]. \tag{2}$$

This form of the posterior distribution in Equation 2 is called a non-normalized density because the integration over its domain is not 1, as required of a proper probability density.

If $\mathbf{y}$ is a vector of n independent and identically distributed (iid) samples, then the likelihood function is

$$[\mathbf{y}|\theta] = \prod_{i=1}^{n} [y_i|\theta] \tag{3}$$

where y_i is the independent ith observation with $i = 1, ..., n$. Under this assumption, the

posterior distribution can be expressed as

$$[\theta|\mathbf{y}] \propto [\mathbf{y}|\theta][\theta] \propto \prod_{i=1}^{n} [y_i|\theta][\theta]. \tag{4}$$

Once the posterior probability distribution of the parameter of interest is available, statistical summaries of the parameter can be calculated for inference. The statistical summaries can be point estimates such as means, medians, modes and standard deviations, or posterior intervals such as credible intervals or highest posterior densities for multimodal distributions. Calculating statistical summaries of the parameter θ analytically following Equation 1 involves computing $[\mathbf{y}] = \int [\mathbf{y}|\theta][\theta] d\theta$, which is the integral with respect to θ and can be difficult to calculate if the parameter θ is a vector or $[y|\theta][\theta]$ is not an analytic distribution.

A common method for estimating statistical summaries is Monte Carlo estimation. In Monte Carlo estimation, an adequate number of i.i.d samples drawn from the posterior distribution are used to estimate statistical summaries [Metropolis and Ulam, 1949]. However, obtaining iid samples from the posterior distribution is often difficult, especially when the posterior density function is in a non-normalized form as in Equation 2. An alternative sampling method called Markov chain Monte Carlo (MCMC) is often used instead.

MCMC is a sampling method that utilizes a Markov chain process where the stationary distribution (the limiting distribution) of the Markov process is the target distribution. A Markov chain is a stochastic process of k samples: $X_1, X_2, ..., X_k$, in which the conditional distribution of X_k given $X_1, X_2, ..., X_{k-1}$ only depends on X_{k-1} – meaning $[X_k|X_1, X_2, ..., X_{k-1}] = [X_k|X_{k-1}]$. The state space of a Markov chain is the set of possible values $\{X\}$ that can be reached using a transition kernel. An MCMC algorithm is composed of many repeating steps called iterations with a transition kernel that specifies how to transition between states. At each iteration k, X_k is sampled from the conditional distribution given X_{k-1} using an appropriate transition kernel. Samples generated in MCMC are not iid as in the traditional Monte Carlo method. However, it has been proved that the consistency

of the estimate is ensured as long as the Markov chain is stationary and converges to the stationary distribution [Brooks et al., 2011].

To ensure the accuracy of the estimates using MCMC samples, it is important to design a Markov chain that has a stationary distribution for the specific target distribution and run the chain long enough for the MCMC to reach convergence to the stationary distribution. In this thesis, we will discuss in detail some important MCMC methods and common practices for evaluating MCMC algorithms. Simulated data for multivariate linear regression and spatial models will be used to illustrate some MCMC algorithms. Finally, the efficiency of different MCMC algorithms for fitting large spatial data will be evaluated and compared using effective sample size per second (ESS/s).

1.1 History of MCMC

As soon as computers were invented, they were used for simulating of random processes using Monte Carlo simulation. Markov chain Monte Carlo was then invented not long after the Monte Carlo method at Los Alamos National Laboratory by Metropolis et al. [1953] using an algorithm that requires symmetric proposal distributions that was later called the Metropolis algorithm. Hastings [1970] generalized the method now called the Metropolis-Hasting algorithm by allowing the use of asymmetric proposal distributions. Without knowledge of Hasting's work, Geman and Geman [1984] published an algorithm called the Gibbs sampler to perform image resolution. Like many other scientific discoveries, MCMC was not recognized among statisticians until Gelfand et al. [1990] demonstrated MCMC applications in which the Gibbs sampler was found to be a special case of the Metropolis-Hasting algorithm [Brooks et al., 2011].

1.2 Metropolis-Hasting Algorithm

The Metropolis-Hasting (MH) algorithm is a simulation technique that can be used to sample from any target distribution, normalized or non-normalized. Samples are first generated from

a proposal distribution, then filtered by an accept-reject rule to better represent the target distribution [Gelman et al., 2013]. Suppose $\mathbf{y}$ are the observations, θ is the parameter of interest with a non-normalized target distribution $[\theta|\mathbf{y}]$. If θ at the MCMC iteration $k-1$ has the value $\theta^{(k-1)}$, the MH algorithm at iteration k will generate a candidate sample θ^* from $[\theta^*|\theta^{(k-1)}]$, the proposal distribution given the current state of θ. Next, the Metropolis-Hasting ratio is calculated as

$$\alpha_{MH} = \frac{[\theta^*|\mathbf{y}]}{[\theta^{(k-1)}|\mathbf{y}]} \frac{[\theta^{(k-1)}|\theta^*]}{[\theta^*|\theta^{(k-1)}]}$$

where the conditional distributions of θ^* given $\mathbf{y}$ is

$$[\theta^*|\mathbf{y}] = \frac{[\mathbf{y}|\theta^*][\theta^*]}{[\mathbf{y}]}$$

and the conditional distribution of $\theta^{(k-1)}$ given $\mathbf{y}$ is:

$$[\theta^{(k-1)}|\mathbf{y}] = \frac{[\mathbf{y}|\theta^{(k-1)}][\theta^{(k-1)}]}{[\mathbf{y}]}.$$

The advantage of using the ratio of these two conditional densities is that the marginal data distribution $[\mathbf{y}]$, which is difficult to evaluate, is canceled out

$$\frac{[\theta^*|\mathbf{y}]}{[\theta^{(k-1)}|\mathbf{y}]} = \frac{\frac{[\mathbf{y}|\theta^*][\theta^*]}{[\mathbf{y}]}}{\frac{[\mathbf{y}|\theta^{(k-1)}][\theta^{(k-1)}]}{[\mathbf{y}]}} = \frac{[\mathbf{y}|\theta^*][\theta^*]}{[\mathbf{y}|\theta^{(k-1)}][\theta^{(k-1)}]}.$$

Thus the MH ratio becomes

$$\alpha_{MH} = \frac{[\mathbf{y}|\theta^*][\theta^*]}{[\mathbf{y}|\theta^{(k-1)}][\theta^{(k-1)}]} \frac{[\theta^{(k-1)}|\theta^*]}{[\theta^*|\theta^{(k-1)}]}.$$

The new θ^* is accepted with probability $min(1, \alpha_{MH})$. If the new θ^* is rejected, then $\theta^{(k)} = \theta^{(k-1)}$.

The Metropolis-Hasting algorithm is different from the original Metropolis algorithm at the choice of the proposal distribution. The Metropolis algorithm is restricted to symmetric proposals – meaning $[\theta^*|\theta^{(k-1)}] = [\theta^{(k-1)}|\theta^*]$. The Metropolis ratio is

$$\alpha_M = \frac{[\theta^*|\mathbf{y}]}{[\theta^{(k-1)}|\mathbf{y}]},$$

whereas, MH algorithm is

$$\alpha_{MH} = \frac{[\theta^*|\mathbf{y}]}{[\theta^{(k-1)}|\mathbf{y}]} \frac{[\theta^{(k-1)}|\theta^*]}{[\theta^*|\theta^{(k-1)}]},$$

where $\frac{[\theta^{(k-1)}|\theta^*]}{[\theta^*|\theta^{(k-1)}]}$ is the correction factor for the use of an asymmetric proposal distribution. Notice that when the proposal is symmetric, the correction factor $\frac{[\theta^{(k-1)}|\theta^*]}{[\theta^*|\theta^{(k-1)}]} = 1$, and the MH ratio becomes the Metropolis ratio. Therefore, the Metropolis algorithm is a special case of the MH algorithm.

The choice of the proposal distribution is important to the success of a MH algorithm. For random walk proposals in particular, the variance of the proposal distribution determines the rate of acceptance and thereby the rate of convergence and mixing. A small proposal variance will give a high acceptance rate but the move may be too small and so the state space of the target distribution would not be explored efficiently. In contrast, a large variance will have many samples being rejected and the algorithm would not be efficient either. Both extremes should be avoided. Finding an optimal proposal variance requires much trial and error. However, an alternative method is adaptive MCMC, which is designed so that the algorithm learns and updates the proposal variance values to find an optimal value as the algorithm runs [Brooks et al., 2011]. Adaptive MCMC is an efficient method and is used widely, however, arbitrary adaptive MCMC algorithms may not preserve the stationary distribution. One can addresses this problem by designing an adaptive MCMC to satisfy certain conditions that are guaranteed to preserve the stationary distribution. In particular,

adaptive tuning algorithms that use vanishing adaptation preserve the stationary distribution [Roberts and Rosenthal, 2007].

1.3 Gibbs Sampler

The Gibbs sampler is an MCMC algorithm in which the Markov chain samples are generated directly from the conditional posterior distribution. Therefore, the Gibbs sampler is used only when the conditional posterior distribution of the parameter of interest has a known distribution that can be easily sampled. The priors used in Gibbs samplers give the conditional posterior a known distribution and are called conjugate priors.

If the vector of parameters of $\boldsymbol{\theta} = (\theta_1, ..., \theta_p)'$ at iteration $k-1$ has the value $\boldsymbol{\theta}^{(k-1)} = (\theta_1^{(k-1)}, \theta_2^{(k-1)}, ..., \theta_p^{(k-1)})$, then $\boldsymbol{\theta}^{(k)}$ at the iteration k can be obtained by sampling individual components $\theta_i^{(k)}$ with $i = 1, ..., p$ separately from the posterior distribution given all other components $[\theta_i^{(k)} | \theta_1^{(k)}, ..., \theta_{i-1}^{(k)}, \theta_{i+1}^{(k-1)}, ..., \theta_p^{(k-1)})]$.

Full conditional distributions as described above are often used, but conditioning on fewer parameters will work as well [Brooks et al., 2011]. Block Gibbs, for example, is the case where several components of $\boldsymbol{\theta}$ are sampled simultaneously. Because the samples are generated directly from the conditional posterior distributions, the Gibbs sampler is a special case of the MH algorithm with an acceptance rate of one.

1.4 MCMC Diagnostics

To ensure the inference made is accurate and appropriate, the MCMC samples used for estimation must well represent the posterior distribution of the parameter. When MCMC sampling is used, the sampling distribution has to converge to the stationary distribution, which is the posterior distribution by design. However, except for the Gibbs sampler, MCMC is a black box in that we do not know the transition kernel nor the stationary distribution [Brooks et al., 2011]. Therefore, it is important to use diagnostic tests to check for a lack of convergence. Specifically, MCMC diagnostics are useful in determining where to start and

when to stop the algorithm to reach a specific goal, in this case, convergence to the limiting distribution and obtaining a sufficient ESS for reliable inference.

The first step of an MCMC algorithm is choosing initial values. The goal is to have the sampling distribution evolve through the transition kernel so that the samples converge to the stationary distribution in a reasonable amount of time in a computationally efficient manner. To achieve this, the initial starting points can be in a high density region of the target distribution. However, determining the high density region (or regions) is difficult, so it is recommended that the starting points should be from an over-dispersed distribution to the target distribution [Gelman et al., 2013]. Starting points from an over-dispersed distribution will allow the state space of the target distribution to be explored efficiently, especially in the case of a multi-modal target distribution. An alternative approach is to initialize the algorithm with crude estimates obtained from a simple model or approximation and can be useful in diagnosing convergence issues.

Samples generated from the initial iterations of an MCMC algorithm typically reflect the initial condition more than the stationary distribution. This initial period is called a burn-in period, in which the Markov chain works toward convergence to the stationary distribution. The burn-in period is usually removed from an MCMC workflow and only post burn-in samples are retained. The length of a burn-in period has to be assessed on a case by case basis. Removing the burn-in period does not guarantee accuracy of the MCMC estimates nor is it always necessary, but it is a good practice for many situations to increase the accuracy of the estimation using MCMC samples.

There are many methods to check for non-convergence of an MCMC chain. The most basic and common tools for checking non-convergence are graphical diagnostics, which include trace plots, autocorrelation plots, and running mean plots. The trace plot is a time series plot of sample values against the iteration number. Trace plots can be used to determine initial effects and identify a lack of convergence and mixing. If the starting points are not in the high density region of the target distribution, the trace plot will show consecutive

iterations trending toward one direction indicating a slow convergence. One can control the speed of convergence and mixing by adjusting a tuning parameter – the variance of the proposal distribution. If the Markov chain is converging to stationarity quickly, the trace plot shows samples moving up and down across the sample median in a pattern commonly called a "fuzzy caterpillar". Any trends or changes in the trace plot can indicate that the Markov chain has not achieved stationarity.

Accompanying the trace plot is the autocorrelation plot, which shows the correlation of samples as a function of different iteration lags. If a MCMC algorithm has K iterations, the lag-k autocorrelation with $k = \{0, ..., K\}$ starts at one for $k = 0$, then drops down toward zero as k increases. The rate at which the lag-k autocorrelation goes to zero indicates how fast the chain is mixing within its state space. Theoretically, although the lag-k autocorrelation will never be zero regardless how big k is, the autocorrelation is practically insignificant when the lag reaches a certain k that we assume there is no meaningful correlation between the samples [Brooks et al., 2011].

Even though its use is controversial, the running mean plot has been used to show the convergence of the estimates to the population parameters. The running mean plot should stabilize to a constant value and visible changes in the running mean plot indicates that the simulation should not be stopped yet [Roy, 2020].

The most popular numeric MCMC diagnostic is the Gelman-Rubin diagnostic $\hat{R}$. The Gelman-Rubin diagnostic uses multiple chains with starting points initialized from an overdispersed distribution with respect to the target distribution. Similar to ANOVA, the ratio of pooled variance between-chains and within-chains is then evaluated to determine if there is a lack of convergence [Gelman et al., 2013].

Let θ be the parameter of interest, m be the number of chains, and K be the number of samples in each sequence, then $\theta_j^{(k)}(k = 1, .., K; j = 1, .., m)$ is the value of θ at iteration k

in sequence j. The sample within-chain variance, denoted W, is

$$W = \frac{1}{m}\sum_{j=1}^{m} s_j^2 \tag{5}$$

where s_j^2 is the sample variance of sequence j. The between-chain variance, denoted B, is

$$B = \frac{K}{m-1}\sum_{j=1}^{m}(\bar{\theta}_j - \bar{\theta})^2,$$

where $\bar{\theta}_j$ is the sample mean of sequence j and $\bar{\theta}$ is the overall sample mean over all MCMC samples. The pooled variance of the parameter of interest can be estimated by a weighted average of within-chain variance and between-chain variance

$$\widehat{var}(\theta|\mathbf{y}) = \frac{K-1}{K}W + \frac{1}{K}B.$$

If the starting points of sequences are initialized from an overdispersed distribution as recommended, $\widehat{var}(\theta|\mathbf{y})$ overestimates the marginal posterior variance. In contrast, the within-chain variance W underestimates the marginal posterior variance due to the finite numbers of samples generated from each sequence. As a result, the ratio of the pooled variance and within chain variance is almost always greater than one. The Gelman-Rubin method uses the ratio of pooled variance and within-chain variance, called the scale reduction

$$\hat{R} = \sqrt{\frac{\widehat{var}(\theta|\mathbf{y})}{W}},$$

to determine if there is evidence of a lack of convergence. As the number of sample increases, the scale reduction tends toward one. In practice, a scale reduction of 1.1 or less is a good stopping point for an MCMC algorithm.

Because MCMC samples are not independent, one would want to know the amount of information obtained from the MCMC samples. ESS is used for this purpose and can also be

used as a stopping rule for an MCMC algorithm. ESS is the number of independent samples required to obtain equivalent information that contained in the MCMC samples. Although there are different ways to calculate ESS, the estimation of ESS mainly depends on the estimation of sample variance and sample autocorrelation at different lags. For example, the ESS would be mK if the samples were independent. Then the sample variance is

$$var(\theta) = \frac{1}{mK} var(\theta|\mathbf{y}). \tag{6}$$

Because the samples of MCMC sequences are autocorrelated, the asymptotic variance is

$$lim_{K\to\infty} mK var(\theta) = (1 + 2\sum_{k=1}^{\infty} \rho_k) var(\theta|\mathbf{y}),$$

with ρ_k is the autocorrelation of the sequence at lag k. The theoretical ESS can be derived from this asymptotic variance

$$ESS = \frac{mK}{1 + 2\sum_{k=1}^{\infty} \rho_k}. \tag{7}$$

Due to the finite number of samples in MCMC sampling, ESS can be estimated using the partial sum: starting from lag 0, the sum of estimated lag-k autocorrelation terms $\hat{\rho}_k$ continues until the sum of autocorrelation of two successive lags becomes negative [Gelman et al., 2013].

An MCMC algorithm generates samples from the posterior distribution but the better the MCMC algorithm explores the posterior distribution the more desirable the algorithm is. Therefore, the efficiency of an MCMC algorithm is of great interest to statisticians when choosing an MCMC algorithm. Due to differences in run time, higher ESS per iteration does not guarantee a more efficient algorithm. The time it takes for an MCMC algorithm reaches a desired ESS depends on both the number of iterations and the speed of each iteration. Hence, the ultimate measure of efficiency of an MCMC algorithm is the ESS per time unit.

2 Fitting a Multivariate Linear Regression Model

We will demonstrate how MCMC algorithms work by fitting a simulated dataset using a multivariate linear regression model. For comparison, both a Gibbs sampler and a MH algorithm are used and the results will be compared. Estimated parameters from the MCMC algorithms will also be compared to the maximum likelihood estimates (MLEs) and the simulated values.

2.1 Multivariate Linear Regression Model

Consider a multivariate linear regression model for $i = 1, ..., n$ observations

$$y_i = \boldsymbol{x}_i'\boldsymbol{\beta} + \varepsilon_i, \tag{8}$$

where y_i is the ith observation, $\boldsymbol{x}_i = (x_{i1}, x_{i2}, ..., x_{ip})'$ is a known p-dimensional covariate vector, $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_p)'$ is an unknown p-dimensional vector called a coefficient vector, and $\varepsilon_i \sim N(0, \sigma^2)$ is called a residual, in which σ^2 is an unknown parameter.

We assume $\boldsymbol{\beta}$ follows a normal distribution $\boldsymbol{\beta} \sim N_p(\boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_\beta)$ with $\boldsymbol{\mu}_\beta$ and $\boldsymbol{\Sigma}_\beta$ fixed and known and σ^2 follows an inverse gamma distribution $\sigma^2 \sim IG(\alpha_0, \beta_0)$ with α_0 and β_0 fixed and known. The density of the $IG(\alpha_0, \beta_0)$ distribution is

$$[\sigma^2|\alpha_0, \beta_0] = \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}\sigma^{2(-\alpha_0-1)}exp\left\{-\frac{\beta_0}{\sigma^2}\right\}.$$

By Bayes' theorem, the posterior distribution of the parameters given $\mathbf{y} = (y_1, ..., y_n)'$ can be expressed in non-normalized form as

$$[\boldsymbol{\beta}, \sigma^2|\mathbf{y}] \propto \Pi_{i=1}^n[y_i|\boldsymbol{\beta}, \sigma^2][\boldsymbol{\beta}][\sigma^2].$$

Given the conditional posterior distribution, the full conditional distribution of each param-

eter is presented in the following sections.

2.2 Full Conditional Distribution of $\boldsymbol{\beta}$

The marginal posterior distribution of $\boldsymbol{\beta}$ written as $[\boldsymbol{\beta}|\cdot]$ is also called the full conditional distribution of $\boldsymbol{\beta}$ given all other parameters. The full conditional distribution of $\boldsymbol{\beta}$ is proportional to the likelihood of y given $\boldsymbol{\beta}$ and σ^2 times the prior of $\boldsymbol{\beta}$

$$\begin{aligned}[\boldsymbol{\beta}|\cdot] &\propto \Pi_{i=1}^{n}[y_i|\boldsymbol{\beta},\sigma^2][\boldsymbol{\beta}][\sigma^2],\\ &\propto \Pi_{i=1}^{n}[y_i|\boldsymbol{\beta},\sigma^2][\boldsymbol{\beta}].\end{aligned}$$

From Equation 8, the likelihood of $y_i|\beta,\sigma^2$ is

$$[y_i|\boldsymbol{\beta},\sigma^2] = (2\pi\sigma^2)^{-1/2}exp\{-\frac{1}{2}(\sigma^2)^{-1}(y_i-\boldsymbol{x}_i'\boldsymbol{\beta})^2\}.$$

With the prior $\boldsymbol{\beta}\sim N_p(\boldsymbol{\mu}_\beta,\boldsymbol{\Sigma}_\beta)$, the conditional posterior of $\boldsymbol{\beta}$ can be expanded as

$$\begin{aligned}[\boldsymbol{\beta}|\cdot] &\propto \Pi_{i=1}^{n}(2\pi\sigma^2)^{-1/2}exp\{-\frac{1}{2}(\sigma^2)^{-1}(y_i-\boldsymbol{x}_i'\boldsymbol{\beta})^2\}(2\pi)^{-p/2}|\boldsymbol{\Sigma}_\beta|^{-1/2}\\ &\quad\times exp\{-\frac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta)'\boldsymbol{\Sigma}_\beta^{-1}(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta)\}\\ &\propto \Pi_{i=1}^{n}exp\{-\frac{1}{2}(\sigma^2)^{-1/2}(\boldsymbol{\beta}'\boldsymbol{x}_i\boldsymbol{x}_i'\boldsymbol{\beta}-2y_i\boldsymbol{x}_i'\boldsymbol{\beta})\}exp\{-\frac{1}{2}(\boldsymbol{\beta}'\boldsymbol{\beta}-2\boldsymbol{\beta}'\boldsymbol{\mu}_\beta)\boldsymbol{\Sigma}_\beta^{-1}\}.\end{aligned}$$

In matrix form, the conditional posterior of $\boldsymbol{\beta}$ is proportional to

$$exp\{\boldsymbol{\beta}'[\mathbf{X}'(\sigma^2 I_n)^{-1}\mathbf{X}+\boldsymbol{\Sigma}_\beta^{-1}]\boldsymbol{\beta}-2\boldsymbol{\beta}'[\mathbf{X}'(\sigma^2 I_n)^{-1}\mathbf{y}+\boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta]\},$$

where $\mathbf{y} = (y_1, y_2, ..., y_n)'$ is a n-dimensional vector of observations and $\mathbf{X}$ is a $n \times p$ matrix of covariates. Thus, the marginal posterior distribution of $\boldsymbol{\beta}$ is a normal distribution with mean $\mathbf{A}_\beta^{-1}\mathbf{b}_\beta$ and variance $\mathbf{A}_\beta^{-1}$ where

$$\mathbf{A}_\beta = \mathbf{X}'(\sigma^2 I_n)^{-1}\mathbf{X} + \boldsymbol{\Sigma}_\beta^{-1},$$

and

$$\mathbf{b}_\beta = \mathbf{X}'(\sigma^2 I_n)^{-1}\mathbf{y} + \boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta.$$

2.3 Full Conditional Distribution for σ^2

Similarly, the full conditional distribution of σ^2 given all other parameters is written as $[\sigma^2|\cdot]$ and is proportional to the likelihood of $\mathbf{y}$ given $\boldsymbol{\beta}$ and σ^2 times the prior for σ^2

$$\begin{aligned}
[\sigma^2|\cdot] &\propto \Pi_{i=1}^n [y_i|\boldsymbol{\beta}, \sigma^2][\sigma^2] \\
&\propto \Pi_{i=1}^n (2\pi\sigma^2)^{-1/2} exp\{-\frac{1}{2}(\sigma^2)^{-1}(y_i - \boldsymbol{x}_i'\boldsymbol{\beta})^2\} \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)} \sigma^{2(-\alpha_0-1)} exp\{-\frac{\beta_0}{\sigma^2}\} \\
&\propto \Pi_{i=1}^n exp\{-(2\sigma^2)^{-1/2}(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2\} \sigma^{2(-\alpha_0-1)} exp\{-\frac{\boldsymbol{\beta}_0}{\sigma^2}\} \\
&\propto \sigma^{2(-\frac{N}{2}-\alpha_0-1)} exp\{-(\sigma^2)^{-1}(\sum_{i=1}^n \frac{1}{2}(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 + \boldsymbol{\beta}_0)\}.
\end{aligned}$$

So the marginal posterior distribution for σ^2 is an inverse gamma distribution with the shape parameter of $\alpha_0 + \frac{n}{2}$ and the scale parameter of $\boldsymbol{\beta}_0 + \frac{1}{2}\sum_{i=1}^n (y_i - \mathbf{x}_i'\boldsymbol{\beta})^2$.

2.4 Simulation Study

2.4.1 Multivariate Simulated Dataset

We create a set of simulated data to test different MCMC algorithms using the multivariate linear model with $n = 100$, $\boldsymbol{\beta} = (10, 2, 5)'$, $\sigma^2 = 4$, and $\mathbf{X}$ a 100×3 matrix where $\mathbf{X} =$

$(\mathbf{X}_0, \mathbf{X}_1, \mathbf{X}_2)$ where $\mathbf{x}_0$ is a 100x1 column vector of 1s, $\mathbf{x}_1$ is a 100x1 column vector of random samples from $N(2,2)$, and $\mathbf{x}_2$ is a 100x1 column vector of random samples from $N(3,3)$. The simulated data $\mathbf{y}$ is then a 100x1 vector.

The goal is to recover the simulated values of $\boldsymbol{\beta}$ and σ^2 using the simulated data $\mathbf{y}$.

2.4.2 Gibbs Sampler for Multivariate Linear Regression Model

As shown in linear regression model section, we assume $\boldsymbol{\beta} \sim N_3(\boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_\beta)$. To make the computation of the inverse matrix of $\boldsymbol{\Sigma}_\beta$ simple, we choose $\boldsymbol{\Sigma}_\beta$ to be a diagonal matrix. We generate the initial value of $\boldsymbol{\beta}$ by randomly draw a sample from the multivariate normal distribution with mean $\boldsymbol{\mu}_\beta = \mathbf{0}$ and variance $\boldsymbol{\Sigma}_\beta = 100\mathbf{I}_3$. The initial value of σ^2 is generated from an inverse gamma distribution with chosen shape α_0 and scale $\boldsymbol{\beta}_0$ to be 0.01.

We choose to run $K = 5000$ MCMC iterations. In each iteration k, we first calculate $\mathbf{A}_\beta$ and $\mathbf{b}_\beta$ using the current value $\sigma^{2(k-1)}$

$$\mathbf{A}_\beta = \mathbf{X}'\mathbf{X}\sigma^{-2(k-1)} + \boldsymbol{\Sigma}_\beta^{-1},$$
$$\mathbf{b}_\beta = \mathbf{X}'\mathbf{y}\sigma^{-2(k-1)} + \boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta$$

Then, $\boldsymbol{\beta}^{(k)}$ is generated from the multivariate normal using the current $\mathbf{A}_\beta$ and $\mathbf{b}_\beta$, where

$$\boldsymbol{\beta}^{(k)} \sim MVN(\boldsymbol{A}_\beta^{-1}\boldsymbol{b}_\beta, \boldsymbol{A}_\beta^{-1}).$$

The current $\boldsymbol{\beta}^{(k)}$ is then used to generate sample $\sigma^{2(k)}$ with

$$\sigma^{2(k)} \sim IG(\alpha_0 + \frac{n}{2}, \boldsymbol{\beta}_0 + \frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^{(k)})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^{(k)}))$$

2.4.3 Metropolis Hasting for Multivariate Model

For the MH algorithm, we choose a multivariate normal distribution and a truncated normal distribution with positive support as proposal distributions for $\boldsymbol{\beta}$ and σ^2, respectively. The acceptance rate of proposed values depends on the variance of the proposal distributions (i.e., the tuning values). We manually find the tuning values by trial and error, aiming for an acceptance rate between 0.234 and 0.44 following recommendations from Roberts and Rosenthal [2001]. For initial values, we choose $\boldsymbol{\beta}$ to be a 3-dimensional vector of 1s and σ^2 to be a random value drawn from $Unif(0, 10)$.

To sample $\boldsymbol{\beta}$, in each iteration k, a proposal value $\boldsymbol{\beta}^*$ is generated from the multivariate normal distribution using the current $\boldsymbol{\beta}^{(k-1)}$

$$\beta^* \sim N_3(\boldsymbol{\beta}^{(k-1)}, \sigma^2_{tune})$$

Because our proposal distribution, a multivariate normal, is a symmetric distribution (meaning $\frac{[\boldsymbol{\beta}^{(k-1)}|\boldsymbol{\beta}^*]}{[\boldsymbol{\beta}^*|\boldsymbol{\beta}^{(k-1)}]} = 1$), the MH ratio is calculated as

$$\alpha_{\boldsymbol{\beta}} = \frac{[\boldsymbol{y}|\boldsymbol{\beta}^*][\boldsymbol{\beta}^*]}{[\boldsymbol{y}|\boldsymbol{\beta}^{(k-1)}][\boldsymbol{\beta}^{(k-1)}]}.$$

Then $\boldsymbol{\beta}^*$ is accepted and set $\boldsymbol{\beta}^* = \boldsymbol{\beta}^{(k)}$ if $\alpha_{\boldsymbol{\beta}} > q$, where $q \sim Unif(0, 1)$. Otherwise, $\boldsymbol{\beta}^*$ is rejected and $\boldsymbol{\beta}^{(k)} = \boldsymbol{\beta}^{(k-1)}$ is the updated value.

Similarly, σ^{2*} is generated from a truncated normal $N^+(\sigma^{2(k-1)}, \sigma^2_{tune})$. Because a truncated normal is an asymmetric proposal, the MH ratio is calculated as follows

$$\alpha_{\sigma^2} = \frac{[\boldsymbol{y}|\sigma^{2*}][\sigma^{2*}][\sigma^{2(k-1)}|\sigma^{2*}]}{[\boldsymbol{y}|\sigma^{2(k-1)}][\sigma^{2(k-1)}][\sigma^{2*}|\sigma^{2(k-1)}]}.$$

2.4.4 Results and Discussion

After the MCMC algorithms are complete, we removed a burnin period of 2000 initial samples. The post-burin samples are then used for model checking and inference. We use visualization plots as diagnostic tools to check for convergence and compare the performance of the Gibbs sampler and the MH algorithm. Figure 1 shows the trace plots of post-burnin MCMC samples for both algorithms. In general, all the trace plots show patterns centered close to the MLEs without any evidence of a lack of convergence. As expected, the MCMC samples from the Gibbs sampler have better mixing with samples alternating on both sides of the medians, whereas the MCMC samples from the MH algorithm have higher autocorrelation and less mixing with more consecutive samples being on one side of the median. Autocorrelation among the MCMC samples can be examined further using the autocorrelation plots and ESS. Figure 2 shows almost no autocorrelation exists among consecutive samples of Gibbs sampler for each parameter. On the other hand, lag-k autocorrelation of MCMC samples from the MH algorithm in Figure 3 decreases much slower, especially for $\boldsymbol{\beta}$, indicating high levels of autocorrelation among the samples. The autocorrelation between iterations of each parameter of the MH algorithm is confirmed by the ESS in Table 1. The ESS for the Gibbs sampler is equal to the number of post-burnin samples for the $\boldsymbol{\beta}$ parameter meaning that the information contained in Gibbs samples is equivalent to an equal number of iid samples. In contrast, the ESS of MH algorithm is much smaller which is consistent with the high levels of autocorrelation among MH samples. Despite the differences in rate of convergence and mixing, the histograms in Figure 4 show that MCMC samples from Gibbs sampler and the MH algorithm have similar posterior distributions which is expected because the two algorithms both target the same posteriors.

Finally, Table 2 lists point estimates, the MLEs, and the simulated parameters for comparison. The estimated posterior medians of all parameters from both the Gibbs sampler and MH algorithm are almost the same as the MLEs. Both of the MCMC algorithms were

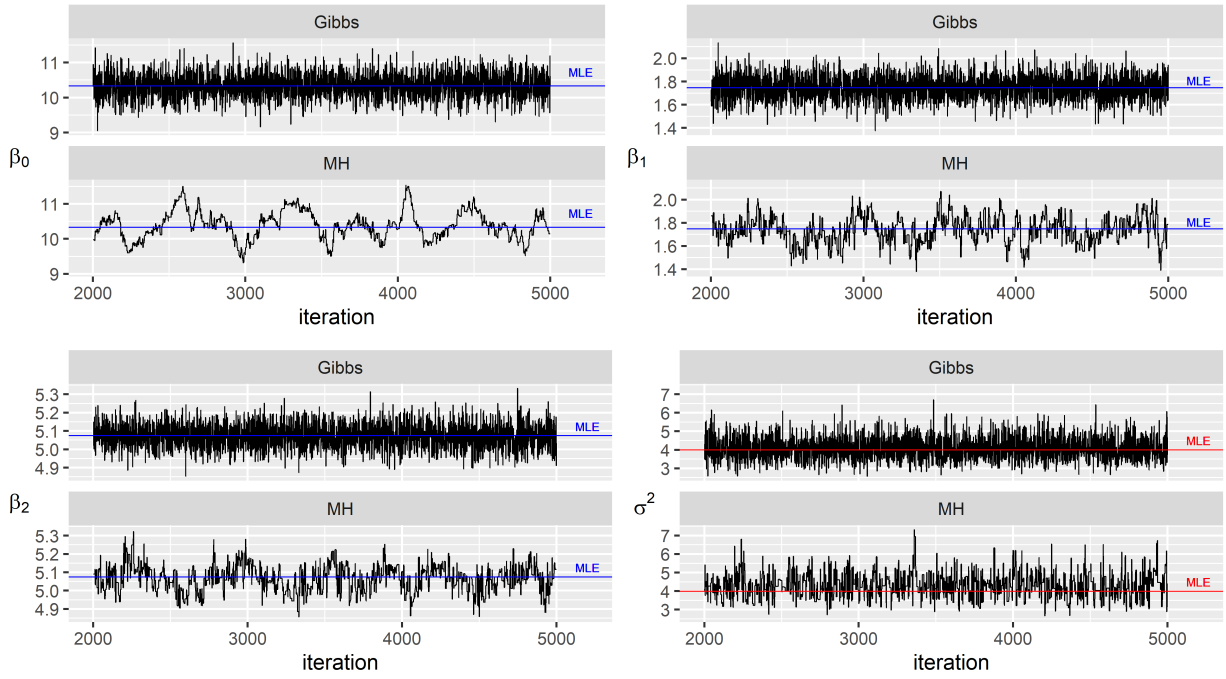

Figure 1: Trace Plots for Gibbs Sampler vs. Metropolis-Hasting Algorithm

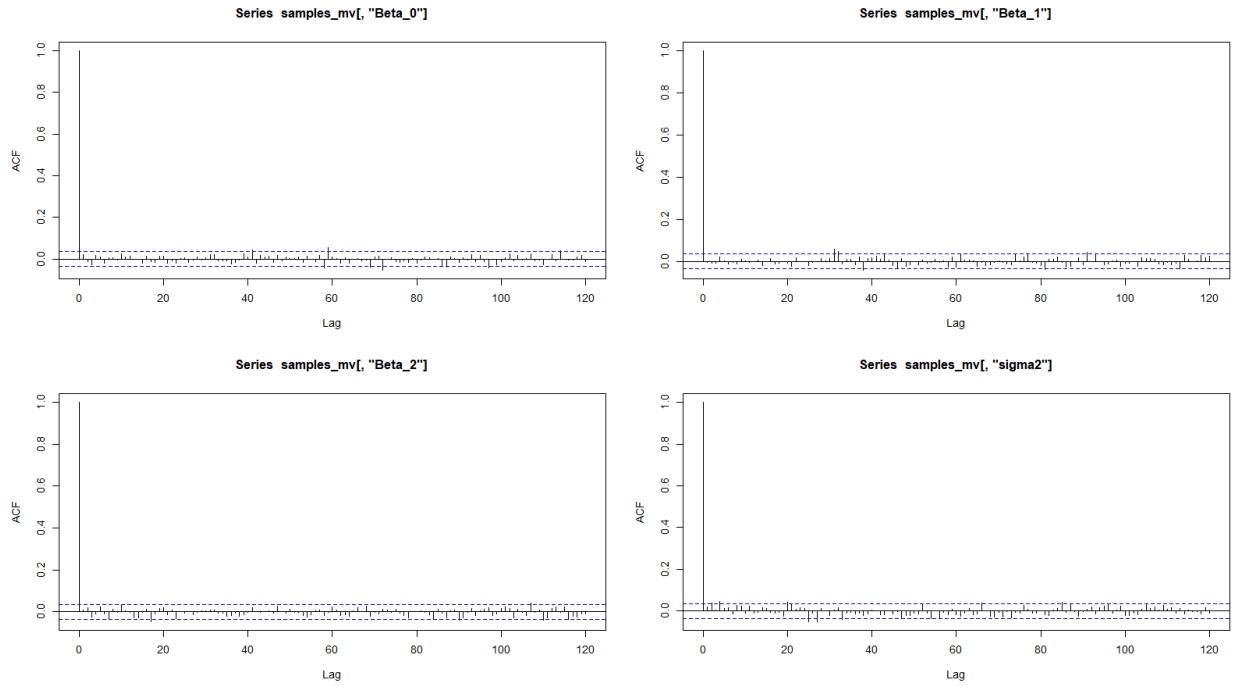

Figure 2: Autocorrelation Plots for Gibbs Sampler

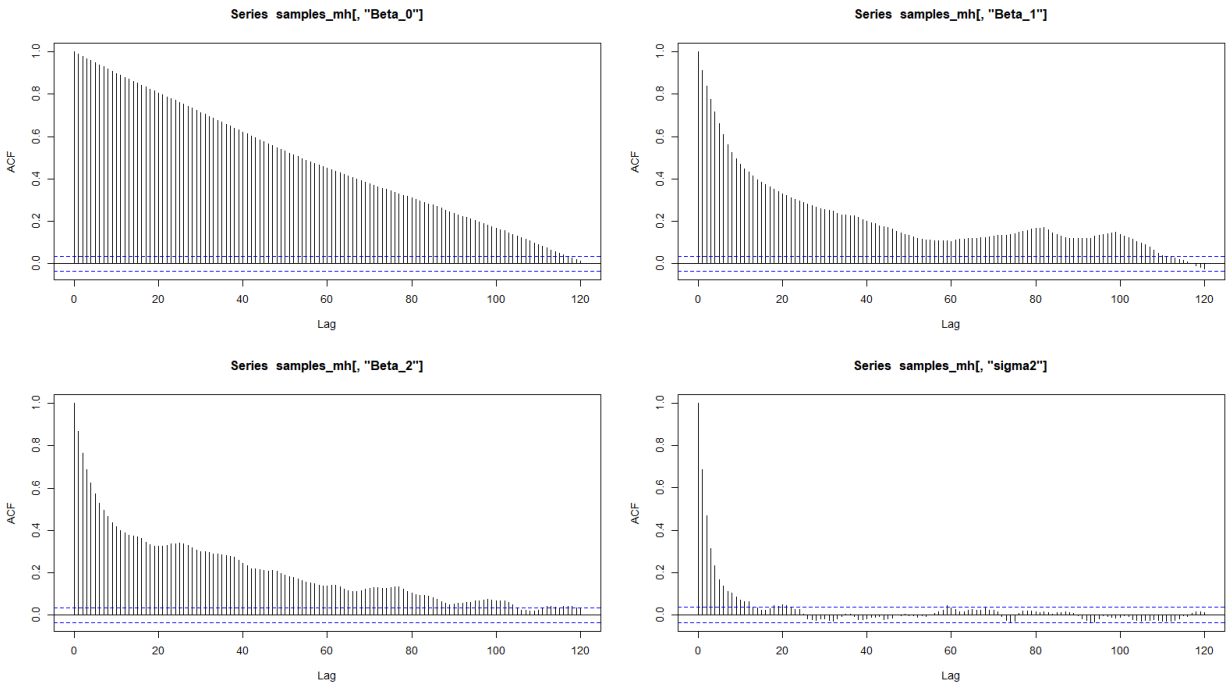

Figure 3: Autocorrelation Plots for Metropolis-Hasting Algorithm

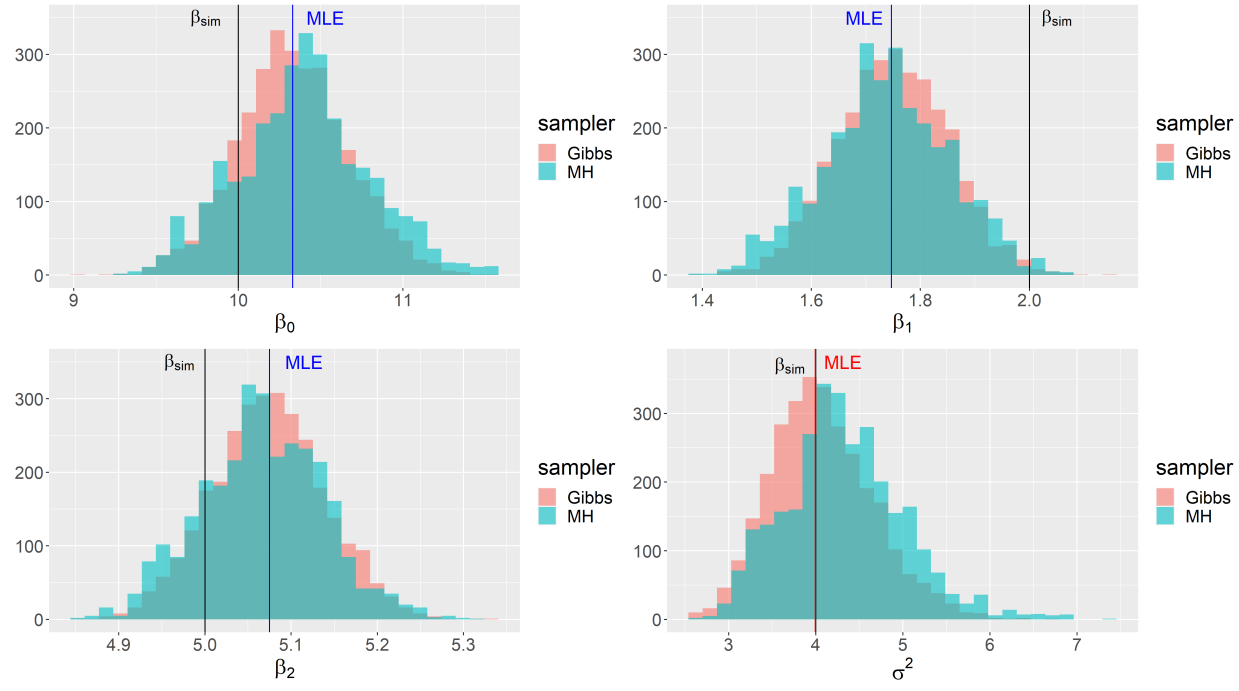

Figure 4: Histograms for Gibbs Sampler vs. Metropolis-Hasting Algorithm

Table 1: Effective sample size for multivariate linear regression model

	Gibbs ESS	MH ESS
β_0	3000	16
β_1	3000	128
β_2	3000	68
σ^2	2412	561

able to recover the simulated parameters with high accuracy. Due to sampling error, the first coefficient β_1 medians are not as close to the simulated value as other parameters, but they are comparable to the MLE of β_1.

Table 2: Point estimates for multivariate linear regression model

	Gibbs sampler		MH algorithm		MLE	Sim_value
	Median	95 percent CI	Median	95 percent CI		
β_0	10.32	(10.11,10.55)	10.39	(10.13,10.63)	10.33	10
β_1	1.75	(1.68,1.82)	1.74	(1.66,1.82)	1.75	2
β_2	5.07	(5.03,5.12)	5.07	(5.02,5.12)	5.07	5
σ^2	4.02	(3.66,4.44)	4.27	(3.90,4.74)	3.99	4

In summary, our simulation study shows that both the Gibbs sampler and the MH algorithm give similar estimates, which are comparable to the MLEs. However, the Gibbs sampler has higher rate of convergence and mixing than the MH algorithm. Therefore, the Gibbs sampler is an efficient sampling method being used widely in practice. Despite having slower convergence rate, the MH algorithm remains an important sampling method because it can handle non-normalized posterior distributions, which the Gibbs sampler cannot. In fact, both Gibbs sampling and MH can be used together in an MCMC algorithm in a combined form called Metropolis-within-Gibbs update (Brooks et al. [2011]).

3 Fitting a Linear Spatial Model

In this section, we will use different MCMC algorithms to fit a simulated spatial dataset using the `BayesMRA` R package (Tipton [2021]). These MCMC algorithms (functions `mcmc_mra()`

and `mcmc_mra_integrated()`) are applications of the spatial model presented in Nychka et al. [2015] but fit within a Bayesian framework using a first- and second-order representations, respectively. With different possible algorithms to fit one model, we are interested in finding if one algorithm is more efficient than the others. In particular, we want to know how much i) the structures of the model (first-order model vs. second-order model) influence computation and ii) the sum-to-zero constraints on random effects change the efficiency of the MCMC algorithms, and iii) whether it is more efficient to compute fewer iterations with large matrices or more iterations with smaller matrices (full-resolution vs. conditional-resolution updates). The efficiency that we refer to is the ESS/s which is calculated using the R package `coda`.

3.1 Simulated Spatial Dataset

The spatial dataset used in this section is from Heaton et al. [2019] and can be found in `BayesMRA` package as the data `code_test`. The dataset is in a `data.frame` format with 10,000 observations and 4 variables: `Lon` is the observation longitude, `Lat` is the observation latitude, `MaskTemp` is the simulated temperature observations in which some of the values have been masked, and `TrueTemp` is the full simulated temperature observations used to evaluate out of sample prediction.

The linear spatial model that is used to fit the simulated dataset is summarized in the following section.

3.2 Linear Spatial Model

Consider an observation $y(\mathbf{s})$ at location $\mathbf{s}$ in a spatial domain $\mathcal{D}$ modeled by

$$y(\mathbf{s}) = \mathbf{x}(\mathbf{s})'\boldsymbol{\beta} + \eta(\mathbf{s}) + \varepsilon(\mathbf{s}),$$

where $\mathbf{x}(\mathbf{s})$ is a covariate vector at site $\mathbf{s}$, $\boldsymbol{\beta}$ is a vector of regression coefficients, $\eta(\mathbf{s})$ is called the spatial random effect and $\varepsilon(\mathbf{s})$ is the realization of an uncorrelated Gaussian error

process. The spatial random effect $\eta(\mathbf{s})$ is a realization of a Gaussian process (a infinite-dimensional distribution where any finite-dimensional marginal distribution is Gaussian) at location $\mathbf{s}$.

In matrix form, this is written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\eta} + \boldsymbol{\varepsilon},$$

where the residual $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2 I)$ accounts for the measurement error. Let $\mathbf{C}$ is a covariance matrix where C_{ij} is determined by a covariance function $cov(\mathbf{s}_i, \mathbf{s}_j)$ between site $\mathbf{s}_i$ and site $\mathbf{s}_j$. Using this notation, $\boldsymbol{\eta} \sim N(\mathbf{0}, \mathbf{C})$ accounts for the spatial effect among locations.

How one models $\boldsymbol{\eta}$ determines the properties of a spatial model. Instead of assuming a parametric form of the covariance function $cov(\mathbf{s}_i, \mathbf{s}_j)$ as in traditional geostatistics, the covariance function can be approximated using multi-resolution spatial basis functions. The spatial effect is then approximated as

$$\boldsymbol{\eta} \approx \sum_{m=1}^{M} \mathbf{W}_m \boldsymbol{\alpha}_m = \mathbf{W}\boldsymbol{\alpha},$$

where $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1', ..., \boldsymbol{\alpha}_M']'$ is a vector of all random effects where $\boldsymbol{\alpha}_m$ is the $\kappa_m \times 1$ vector of random effects at resolution m with $m = 1, ..., M$ and κ_m is the number of basis functions chosen at resolution m. Similarly, $\mathbf{W} = (\mathbf{W}_1 \cdots \mathbf{W}_M)$ is a matrix of all spatial basis functions where each component $\mathbf{W}_m$ is a matrix of compactly supported spatial basis functions at resolution m. For this work, we use Wendland basis functions where the radius of basis functions reduces by half at each level of resolutions. Using Wendland basis functions results in most of the elements of $\mathbf{W}_m$ being zero. As a result, the computational demand is alleviated by leveraging sparse matrix routines.

A conditionally independent normal distribution is chosen as the prior for the random

effects $\boldsymbol{\alpha}_m$ at each resolution

$$\boldsymbol{\alpha}_m \sim N(\mathbf{0}, (\tau_m^2 \mathbf{Q}_m)^{-1}),$$

where τ_m^2 is a precision (inverse variance) parameter that accounts for the global variation of the random effect at the mth resolution and $\mathbf{Q}_m$ is a precision matrix that arises from a graphical structure over a regular lattice. As there are many basis functions (perhaps more than data points), a generic choice of prior for the random effect $\boldsymbol{\alpha}_m$ may result in overfitting. To reduce overfitting, $\mathbf{Q}_m$ can be chosen to be a precision matrix of an intrinsic conditional autoregression (ICAR) or a simultaneous autoregression (SAR) process, which limits the variability in $\boldsymbol{\alpha}_m$ (Lang and Brezger [2004]).

3.2.1 First-order and Second-order Models

In matrix form, the observation model using the multiresolution approximation can be written as

$$\mathbf{y} \approx \mathbf{X}\boldsymbol{\beta} + \mathbf{W}\boldsymbol{\alpha} + \boldsymbol{\varepsilon}. \tag{9}$$

Using this approximation, the model can be fitted in multiple ways. If we approximate the spatial effect by $\mathbf{W}\boldsymbol{\alpha}$ and consider the linear spatial model as a linear regression with the mean being shifted by $\mathbf{W}\boldsymbol{\alpha}$, then we have the first-order model defined as

$$\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\beta} + \mathbf{W}\alpha, \sigma^2\mathbf{I}). \tag{10}$$

On the other hand, because both $\boldsymbol{\alpha}$ and $\boldsymbol{\varepsilon}$ are assumed to have normal distributions, $\boldsymbol{\alpha}$ can be integrated out of the likelihood and the second-order model specifies the random process

as part of the covariance giving

$$\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\beta}, \boldsymbol{W}(\mathbf{Q}\boldsymbol{\tau}^2)^{-1}\mathbf{W}' + \sigma^2 I). \tag{11}$$

where $\mathbf{Q}\boldsymbol{\tau}^2 = blockdiag(\tau_1^2\mathbf{Q}_1, ..., \tau_M^2\mathbf{Q}_M)$ and there is no longer the need to sample the latent parameters $\boldsymbol{\alpha}$.

In summary, the first-order model treats $\boldsymbol{\alpha}$ as one of the parameters of interest, which need to be sampled along with the parameters $\boldsymbol{\beta}$ and σ^2. In contrast, the second-order model has integrated out $\boldsymbol{\alpha}$ and it does not need to sample $\boldsymbol{\alpha}$ in the algorithm. As a result, the second-order model is more efficient in term of ESS per iteration. However, due to the different ways these two models define the observation models, the computation costs are different. Thus, the time it takes for each MCMC algorithm to generate an effective sample may be different. We explore how the differences in computational algorithms affect the MCMC algorithm's efficiency and whether the second-order model is also more efficient in term of ESS per time unit.

3.2.2 Full-resolution and Conditional-resolution Algorithms

The function `mcmc_mra()` in `BayesMRA` package uses the first-order model, which has the observation distribution described in Equation 10. In these MCMC samplers, if $\mathbf{W}\boldsymbol{\alpha}$ is treated as a large matrix of all resolutions (`joint = T` in `mcmc_mra()` function), corresponding to full-resolution MCMC algorithms, the observation model can be approximated as

$$\mathbf{y} \approx \mathbf{X}\boldsymbol{\beta} + \mathbf{W}\boldsymbol{\alpha} + \boldsymbol{\varepsilon}.$$

and $\boldsymbol{\alpha}$ is sampled as a single Gibbs block.

On the other hand, if $\boldsymbol{\alpha}_m$ is sampled for each mth resolution, corresponding to

conditional-resolution MCMC algorithms, the observation model is

$$\mathbf{y} \approx \mathbf{X}\boldsymbol{\beta} + \sum_{m=1}^{M} \mathbf{W}_m \boldsymbol{\alpha}_m + \varepsilon. \tag{12}$$

and each $\boldsymbol{\alpha}_m$ is sampled using its own Gibbs block update conditional on all the other resolution $\boldsymbol{\alpha}_{-\boldsymbol{m}}$.

Choosing to estimate the spatial effect $\boldsymbol{W}_m \boldsymbol{\alpha}_m$ at each resolution separately or not has its advantages and disadvantages. The full-resolution algorithm samples $\boldsymbol{\alpha}$ parameter for all resolution simultaneously requiring the computation of the Cholesky factor of a large but sparse matrix, which can be computational demanding for big data but results in better mixing of the $\boldsymbol{\alpha}$ parameter. On the other hand, the conditional-resolution algorithm samples $\boldsymbol{\alpha}_m$ for each resolution separately reducing the computational demand because the computation involves in smaller matrices but results in slower mixing of $\boldsymbol{\alpha}$. We would like to know if decreasing the computational challenging with smaller matrices offsets the disadvantage of slower mixing. In other words, we would want to know if conditional-resolution algorithm yields higher ESS per time unit.

3.2.3 Sum-to-zero Constraint Algorithms

To reduce non-identifiability between the intercept β_0 and the approximated random effect $\mathbf{W}\boldsymbol{\alpha}$, a sum-to-zero constraint can be applied on the spatial random process $\mathbf{W}\boldsymbol{\alpha}$. For the conditional-resolution MCMC algorithm described in Equation 12, the constraint is at each resolution level so that each row of $\boldsymbol{W}_m \boldsymbol{\alpha}_m$ matrix has a sum of zero $\sum_{i=1}^{n} W_{mi} \boldsymbol{\alpha}_m = 0$. For the full-resolution MCMC, where $\mathbf{W}$ is treated as a large matrix and $\boldsymbol{\alpha}$ as a large vector, the constraint can be applied at each row of the basis expansion $\mathbf{W}\boldsymbol{\alpha}$ so that $\sum_{i=1}^{n} \boldsymbol{W}_i \boldsymbol{\alpha} = 0$. However, due to the non-identifiability among resolutions, the constraint is better to be applied at each level for the full-resolution model as well. In other words, sum-to-zero constraint $\sum_{i=1}^{n} W_{mi} \boldsymbol{\alpha}_m = 0$ can be applied if desired for both samplers of the first-order

model.

The choice of applying sum-to-zero constraint is mostly for interpretation of the intercept β_0 and to improve mixing. Therefore, we would like to compare the efficiency of the algorithms with and without sum-to-zero constraint to find out whether applying the constraint affects the algorithm in term of ESS per unit time.

3.3 Results and Discussion of the Spatial Model

We ran 2000 iterations for each algorithm described above and removed the first 1000 iterations as burn-in period. All of the inference and ESS were calculated using 1000 post burn-in samples.

The trace plots of the conditional-resolution first-order model with constraint are in Figure 5 and these look similar to other first-order trace plots except for the intercept term which varies its behavior according to the constraint. With the constraint on $\mathbf{W}\boldsymbol{\alpha}$ being applied as in Figure 5, the intercept is steady at 0, whereas without constraint, the intercept has higher variance and shows a lack of identifiability. While the trace plots of $\boldsymbol{\tau^2} = (\tau_1^2, \tau_2^2, \tau_3^2)'$ and σ^2 show pattern without evidence of a lack of convergence, the trace plots of β_1 and β_2 show some trends indicating a potential issue. Because the fixed covariates $\mathbf{X}$ in this dataset are latitude and longitude, which are spatially explicit, we believe that columns of $\mathbf{X}$ matrix are collinear with columns of $\mathbf{W}$ matrix. In the first-order model, because the algorithms sample both $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$, the non-identifiability of $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ could explain the behaviors of the trace plots of $\boldsymbol{\beta}$. Examining the posterior distributions of the first-order model in Figure 6, we could see that β_1 distribution seems to have dual modes, whereas the posterior distributions of σ^2 and τ^2 are uni-modal as expected.

Compared to the first-order model, trace plots of the second-order model in Figure 7 show more horizontal straight lines indicating a slower convergence and less mixing for all three parameters $\boldsymbol{\tau^2}$, σ^2, and $\boldsymbol{\beta}$. However, the trace plots of $\boldsymbol{\beta}$ do not show trends like the ones in the first-order model. The second-order model does not sample $\boldsymbol{\alpha}$ and so the $\boldsymbol{\beta}$

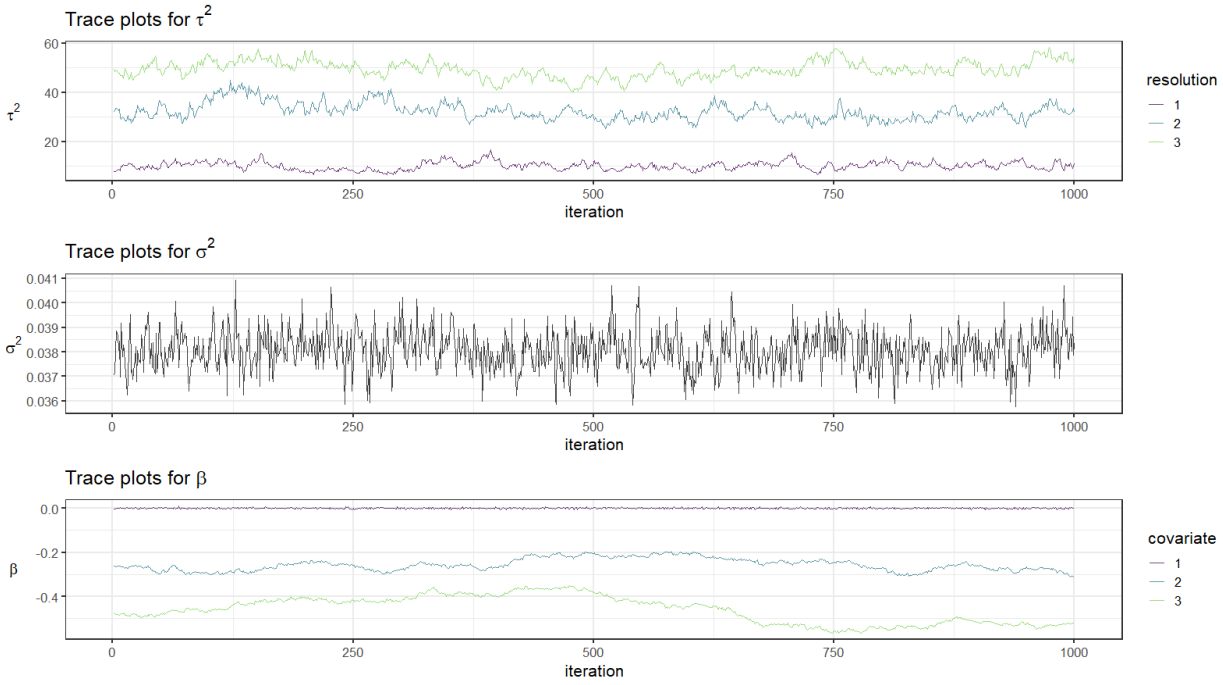

Figure 5: Trace plots of conditional and constrained algorithm first-order model

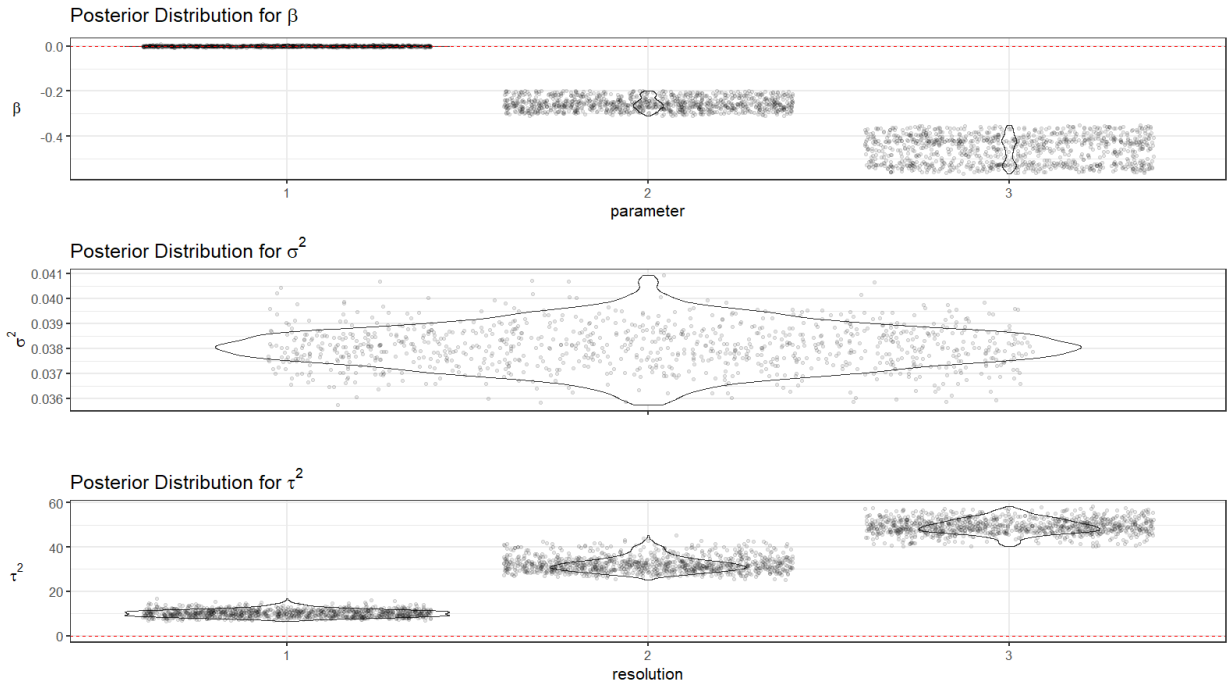

Figure 6: Posterior distributions of conditional and constrained algorithm first-order model

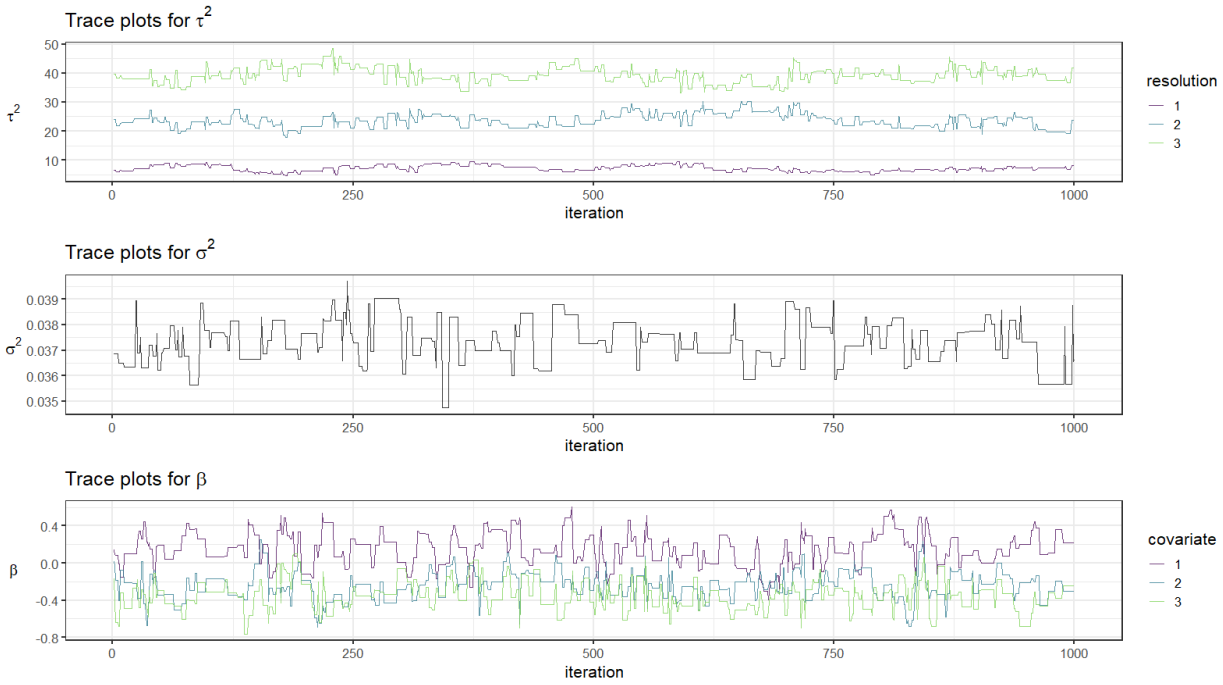

Figure 7: Trace plots of second-order model

samples are less influenced by the colinearity of the covariates with columns of $\mathbf{W}$ matrix. Figure 8 shows the posterior distributions of all parameters in the second-order model having single modes as expected.

In term of efficiency, Table 3 shows the relative ESS/s of the first-order model with the second-order model chosen as the baseline. Values greater than 1 means that the ESS/s of the first-order model is greater than ESS/s of the second-order model. We can examine the algorithm efficiency for the parameters in two groups: one for $\boldsymbol{\tau^2}$ and σ^2 and one for $\boldsymbol{\beta}$. For $\boldsymbol{\tau^2}$ and σ^2, the first-order model is more efficient than the second-order model in term of ESS/s. Within the first-order model, the conditional-resolution algorithms are more efficient than the full-resolution. Between the algorithms with and without sum-to-zero constraint, there is not evidence that one algorithm is doing better than another other than the ESS/s for the intercept term where the models with constraints show higher ESS/s.

In contrast, due to the collinearity between $\mathbf{X}$ and $\mathbf{W}$, $\boldsymbol{\beta}$ behaves differently than $\boldsymbol{\tau^2}$ and σ^2. The first-order model is less efficient than the second-order model for $\boldsymbol{\beta}$. Within the

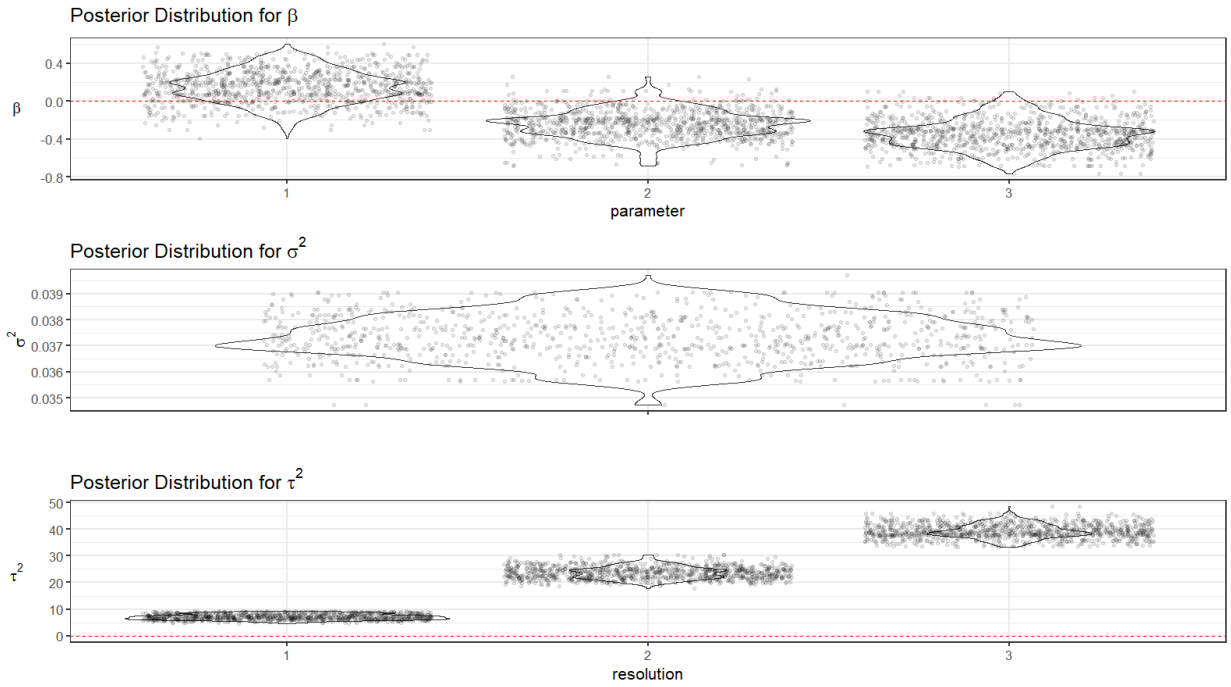

Figure 8: Posterior distributions of second-order model

first-order model, there is not consistent evidence that the full-resolution algorithm is more efficient than the conditional-resolution algorithm.

Table 3: ESS/s of the first-order model relative to the second-order model

	Full-resolution		Conditional-resolution	
	Unconstrained	Constrained	Unconstrained	Constrained
β_0	0.14	55.10	0.30	170.87
β_1	0.11	0.03	0.42	0.72
β_2	0.05	0.19	0.60	0.16
τ_1^2	5.21	5.48	24.33	39.22
τ_2^2	3.29	4.01	9.27	10.66
τ_3^2	2.92	3.38	19.21	10.04
σ^2	12.88	13.12	72.33	75.23

MCMC samples of all algorithms for all three parameters $\boldsymbol{\beta}$, $\boldsymbol{\tau^2}$ and σ^2 are summarized in the boxplots of Figure 9. In general, we expect the two order representations give similar inferences because they target the same posteriors. However, we observe differences between

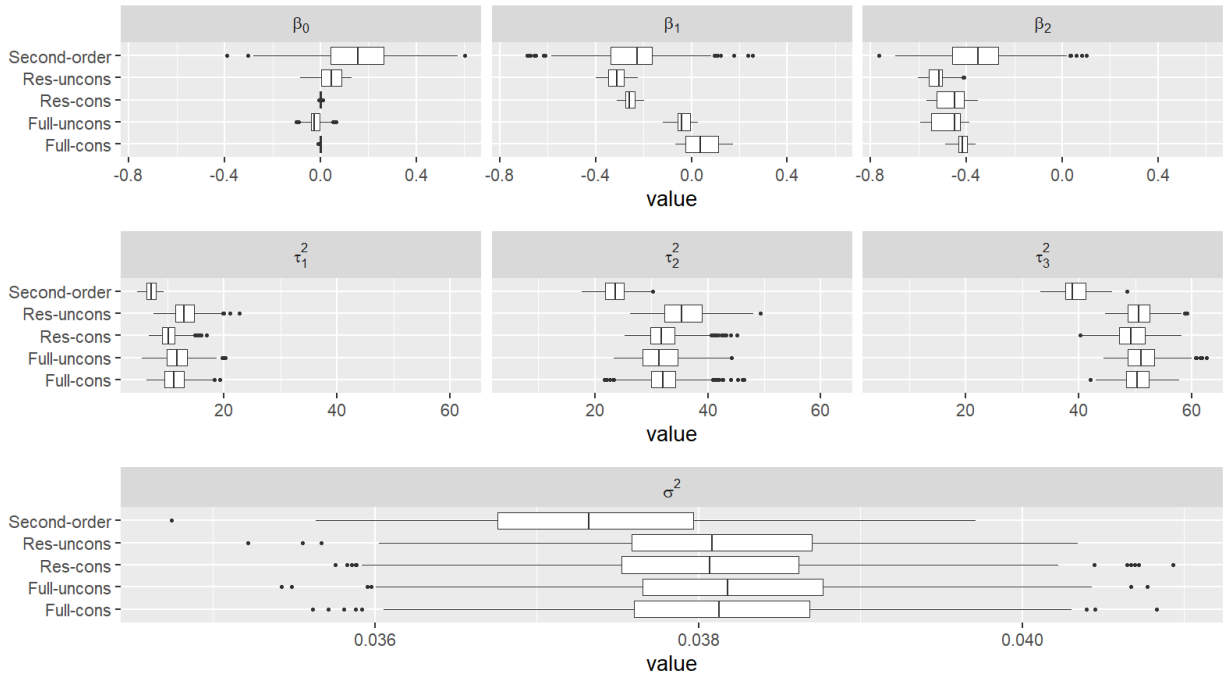

Figure 9: Posterior boxplots of all algorithms

the first- and second-order models in terms of MCMC samples values and inferences. The second-order model tends to have lower MCMC sample values for both $\boldsymbol{\tau^2}$ and σ^2. The differences is harder to discern for $\boldsymbol{\beta}$ because of its inconsistent behaviors. The exact values of median estimates can be found in Table 4. As lower convergence and less mixing observed in the trace plots of the second-order model in Figure 7, it is possible that MCMC samples of both order models, especially the second-order model, do not represent well the posterior distributions. We only ran 1000 post-burnin iterations with the goal of comparing the efficiency of the algorithms. If more iterations had run with better mixing rates, we believe that the two order models would give similar inferences.

In summary, the first-order model is more efficient in term of ESS per time unit than the second-order model for most of the parameters except for $\boldsymbol{\beta}$ even though the second-order has a higher ESS per iteration. This result indicates that model structure could greatly affect the efficiency of an algorithm. In the first-order model, the condition-resolution algorithm is more efficient in term of ESS/s than the full-resolution. This means that designing an

Table 4: Median point estimates

	First-order				Second-order
	Full-resolution		Conditional-resolution		
	Unconstrained	Constrained	Unconstrained	Constrained	
β_0	-0.03	0.00	0.05	0.00	0.16
β_1	-0.04	0.04	-0.31	-0.26	-0.23
β_2	-0.45	-0.42	-0.51	-0.45	-0.35
τ_1^2	11.71	11.08	12.98	10.16	7.10
τ_2^2	31.29	31.99	35.43	31.78	23.67
τ_3^2	51.06	50.29	50.56	49.24	38.93
σ^2	0.04	0.04	0.04	0.04	0.04

MCMC algorithm so that the computation involves in smaller matrices can significantly improve the efficiency when fitting dataset as big as or greater than ours. Finally, because there is no evidence that applying the sum-to-zero constraint affects the efficiency of an algorithm except for the intercept term, the sum-to-zero constraint is a great option when fitting this spatial model to increase mixing and for the interpretation of $\boldsymbol{\beta}$.

4 Conclusion

In this thesis, we have demonstrated how MCMC can be used to obtain posterior samples for inference in a Bayesian framework. Through our simulation study of multivariate linear regression, we have shown that both the Gibbs sampler and the Metropolis-Hasting algorithm give similar results to the MLEs. Therefore, MCMC is a reliable sampling technique despite its working mechanism being not as straightforward as Monte Carlo estimation or analytic approximation. Theoretically, MCMC will converge to the stationary distribution given infinite time, but we want to know if it has not converged in finite time. Because we may not know the transition kernel nor the stationary distribution in an MCMC algorithm, we can not prove a convergence but only check for the evidence of a lack of convergence. However, there are many MCMC diagnostic tests that can be used to ensure the convergence and

check the fit of the model. In addition, for many models, it is feasible and practical to adjust the algorithm until it reaches one's goals. We also showed that the efficiency of an MCMC algorithm could depend on many factors such as the structure of the model and the size of matrices used in computation. Specifically, having fewer parameters does not guarantee higher efficiency in term of ESS per time unit. Algorithms that have computation using smaller matrices can be preferred over algorithms using large matrices, especially when fitting a large dataset. Finally, constraints can be important for model interpretation and can be applied if desired without significantly affecting the efficiency of the algorithm.

In conclusion, with the great advances in computer power, MCMC is a powerful tool for model fitting in a Bayesian framework. However, the use of MCMC has its challenges and one must take great care in checking convergence to ensure the accuracy of the estimates. In addition, the efficiency of an MCMC algorithm can be affected by many factors. As a result, users have different options to improve the efficiency of an MCMC algorithm. We only discussed and demonstrated the two most basic MCMC algorithms: the Gibbs sampler and the Metropolis-Hasting update. MCMC techniques continue to be improved and developed to solve more specific and more sophisticated problems. For example, reversible jump MCMC was introduced by Green [1995] to allow the target distribution to be a mix of distributions with different dimensions. For our illustration, we only used simulated data, which is convenient for model checking and comparing. However, fitting real data could further demonstrate how Bayesian statistics in general and MCMC in particular can be used to solve real life problems.

References

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

Alan E Gelfand, Susan E Hills, Amy Racine-Poon, and Adrian FM Smith. Illustration of bayesian inference in normal data models using gibbs sampling. *Journal of the American Statistical Association*, 85(412):972–985, 1990.

Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

PJ Green. Reversible jump mcmc computation and bayesian model determination. biomeirika, 82: 711-732 hastings, wk 1970. monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1995.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

Matthew J Heaton, Abhirup Datta, Andrew O Finley, Reinhard Furrer, Joseph Guinness, Rajarshi Guhaniyogi, Florian Gerber, Robert B Gramacy, Dorit Hammerling, Matthias Katzfuss, et al. A case study competition among methods for analyzing large spatial data. *Journal of Agricultural, Biological and Environmental Statistics*, 24(3):398–425, 2019.

Stefan Lang and Andreas Brezger. Bayesian p-splines. *Journal of computational and graphical statistics*, 13(1):183–212, 2004.

Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Douglas Nychka, Soutir Bandyopadhyay, Dorit Hammerling, Finn Lindgren, and Stephan Sain. A multiresolution gaussian process model for the analysis of large spatial datasets. *Journal of Computational and Graphical Statistics*, 24(2):579–599, 2015.

Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling for various metropolis-hastings algorithms. *Statistical science*, 16(4):351–367, 2001.

Gareth O Roberts and Jeffrey S Rosenthal. Coupling and ergodicity of adaptive markov chain monte carlo algorithms. *Journal of applied probability*, 44(2):458–475, 2007.

Vivekananda Roy. Convergence diagnostics for markov chain monte carlo. *Annual Review of Statistics and Its Application*, 7:387–412, 2020.

John Tipton. *BayesMRA: Bayesian Multi-Resolution Gaussian Process Approximations*, 2021. URL https://github.com/jtipton25/BayesMRA. R package version 1.0.0.