Bayesian Computations via MCMC, with applications to Big Data and Spatial Data

by

Reihaneh Entezari

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Statistical Sciences
University of Toronto

Abstract

Bayesian Computations via MCMC,
with applications to Big Data and Spatial Data

Reihaneh Entezari

Doctor of Philosophy

Graduate Department of Statistical Sciences

University of Toronto

2018

Markov Chain Monte Carlo (MCMC) methods are fundamental tools for sampling highly complex distributions. They are crucial to Bayesian inference as posterior distributions are generally analytically intractable. In this thesis, we tackle two Bayesian inference problems via MCMC methods, that will lie on both methodology and application aspects.

The first part of this thesis tackles the computational challenges of Bayesian inference from big data. We develop a new communication-free parallel method, the *Likelihood Inflating Sampling Algorithm (LISA)*, that significantly reduces computational costs by randomly splitting the dataset into smaller subsets and running MCMC methods *independently* in parallel on each subset using different processors. Each processor will be used to run an MCMC chain that samples sub-posterior distributions which are defined using an "inflated" likelihood function. We then discuss on the approaches to combine all sub-samples from all processors to build a highly accurate posterior distribution that is consistent with the full posterior distribution. More importantly, we learn a strategy in combining LISA's draws to study the full posterior of the more complex Bayesian Additive Regression Trees (BART) model,

which is highly important in non-parametric regression. In addition, we also successfully examine the consistency in performance of LISA on BART with new efficient Metropolis-Hastings (MH) proposals introduced by Pratola (2016).

The second part of this thesis is more focused on the applied aspect of performing Bayesian inference via MCMC methods. We study a Bayesian Geostatistical model to analyze spatial data from the Timiskaming & Abitibi River forests in Ontario Canada, provided by the First Resource Management Group Inc.. We implement an MCMC algorithm to perform Bayesian inference on predicting the proportion of hardwood trees from elevation and vegetation index. Spatial predictions are made for new sites in the forests and results are compared with a Logistic Regression model without a spatial effect. We study the trend of accuracy in predictions when fitting fewer data to the model, and present useful insights on performance related to the number of ground truth data collected, that can be costly. We further discuss a stratified sampling approach in choosing the subsets of data that allows for potential better predictions.

Acknowledgements

I would like to first thank my supervisor, Professor Jeffrey S. Rosenthal, for all his support, encouragement, and guidance throughout my studies. His thoughtful insights, enthusiasm, and motivation was the main reason I was able to complete my PhD thesis. I was very lucky to be his student where I had the chance to learn a lot about MCMC.

I also would like to thank my co-supervisor, Professor Radu V. Craiu, for all his intelligent ideas, advice, and support. I truly appreciate the time he has spent in helping me complete this thesis.

I would like to express my gratitude to Professor Patrick E. Brown for all his help, support, and guidance throughout my PhD work. I was very lucky to learn a lot about Geostatistics from him.

I also like to thank my thesis committee members, Professor Ruslan Salakhutdinov and Professor David Duvenaud for all their helpful comments. And special thanks to Dermot Whelan for his unlimited help in my computing questions.

Finally, I sincerely thank my family, specially my parents for always supporting me and loving me. Without them, I would have never had the opportunities to become the person I am today.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Nowadays we face many complicated real-world problems that require scientific understanding of data and design of its underlying model that will guide through predictions for the future. For this purpose, statistical inference has become essential in order to satisfy our needs in better understanding the world around us. However in such inferences, there still exist many uncertainties about the true underlying process as there may be insufficient information available, and hence require the powerful fundamentals of probability distributions. Bayesian inference (Bernardo and Smith (2001); Gelman et al. (2014)) is the key subject that resolves these issues in that matter, and is widely used in various areas of research such as artificial intelligence, bio-informatics, finance, etc. In a Bayesian framework, we are able to apply our own beliefs and experiences along with the use of data to describe all uncertainties using probability distributions.

Bayesian approaches attempt to quantify uncertainties about the true underlying model parameters θ, as a probability distribution called the *posterior distribution*.

They require prior distributions $\pi(\theta)$, that are formed from our initial beliefs about the parameter of interest, along with the distribution of the observed data to form the overall posterior distribution. Bayesian methods are built upon the foundations of Bayes' Theorem which we will describe later on in Chapter 2.

Our particular focus in this thesis will lie on the Bayesian computations aspect, where numerical integration regarding the posterior distribution becomes intractable for complicated models with higher parameter dimensions. There are various approaches proposed in the literature that overcome this challenge by approximating the posterior distribution either with family of known distributions (MacKay, 2003; Bishop, 2006; Rue et al., 2009), or by simulating samples from it. However, some simulating methods are to some level indirect (Beaumont et al., 2002; Blum, 2010; Drovandi et al., 2015), that is, they approximate the posterior distribution by simulating data from a given prior parameter and decide through a discrepancy dependent of summary statistics. They also require the ability to sample from the prior distribution. Although all these methods may be useful and efficient, but they are somehow model-specific and can be less reliable or convenient given their limitations.

In this thesis, we will tackle Bayesian computations with Markov Chain Monte Carlo (MCMC) simulation methods (Neal, 1993; Craiu and Rosenthal, 2014; Brooks et al., 2011; Brooks, 1998). MCMC methods are fundamental tools for sampling highly complex distributions, and are more powerful and reliable compared to their alternatives, as they deal directly with the posterior density regardless of its complexity. With the widespread adoption of Bayesian inference in science and engineering problems, MCMC tools are vital to many applications such as statistical learning, image processing, medical imaging and natural language processing.

Our main contribution in this thesis is to tackle two Bayesian inference problems under the umbrella of MCMC methods. The type of problems solved in this thesis lie

in both methodology terms and applications on real-world problems. We will describe in detail the context and the overall order of the thesis chapters in the next section.

1.2 Overview of Thesis Chapters

We begin with reviewing some fundamentals of Bayesian modelling and MCMC methods in Chapter 2. We will explain the differences between Bayesian and Frequentist approaches with an emphasis on Bayesian methods. We later on discuss Monte Carlo methods and Markov chain convergence theorem that will build the foundations of MCMC methods. At last, we will explain in detail the various MCMC methods that will be used for the rest of this thesis.

The first half of the thesis deals with the computational challenges of Bayesian inference via MCMC methods from big data, i.e. huge sets of data that are too big to fit even on one computer. We develop a new method for posterior sampling, in big data applications, that lends itself to highly parallel computation. In Chapter 3, we introduce the *Likelihood Inflating Sampling Algorithm (LISA)* (Entezari et al., 2018b), that significantly reduces computational costs by randomly partitioning the dataset into smaller subsets and running MCMC methods *independently* in parallel on each subset using different processors. Each processor will be used to run an MCMC chain that samples from sub-posterior distributions which are defined using an *"inflated"* likelihood function. The sub-samples drawn from all processors are then aggregated in a way to build a highly accurate posterior distribution that is consistent with the full posterior distribution. More importantly, we develop a strategy for combining the draws from different sub-posteriors to study the full posterior of the complex non-parametric regression model, the *Bayesian Additive Regression Trees (BART)*. We prove in theory and a simple application that LISA can outperform its competing

method, the popular Consensus Monte Carlo (CMC) (Scott et al., 2013). LISA also shows superior performance (in terms of accuracy and efficiency) compared to CMC in BART, when tested on simulated data and a large socio-economic study.

Chapter 4 consists of the performance of LISA with a more efficient MCMC method for BART proposed by Pratola (2016). We present results that show with a simulated dataset, LISA's performance is consistent with the new MCMC method used for BART, and continues to successfully generate approximate full posterior samples (Chkrebtii et al., 2016).

In Chapter 5, we tackle a different problem that also involves Bayesian inference via MCMC methods. Our challenge in this chapter is to present an analysis for a real-world problem related to spatial data from forests in Canada. As it is known, the monetary value of forests depends on their timber values and timber values differ depending on the species of the trees. Hardwood trees are worth more than softwoods, as they provide longer lasting wood and hence can be of great asset for owners. Thus, the problem of interest will be in predicting the proportion of hardwood trees in different areas around forests which is highly important in determining the timber value. The *First Resource Management Group Inc.* has provided us data from the Timiskaming and Abitibi River forests in Ontario, that consists of ground truth data, as well as remotely sensed data that was collected using their new remote sensing technology called the "SkyForest". However, since the collection of ground truth data is costly and time consuming, our aim is to study this data and build up a statistical model where we can examine and analyze predictions with fewer ground truth data collected. We build a Bayesian Geostatistical model to predict the proportion of hardwood trees from elevation and vegetation index image data, and implement an MCMC algorithm for posterior simulation of the model parameters. Our analysis (Entezari et al., 2018a) shows that with fewer data fitted to the model, unbiased esti-

mates of hardwood counts along with reasonable uncertainty quantities are achieved. Finally, we discuss a stratified sampling approach for choosing subsets of spatial data that will show potential improvements.

We make our overall conclusions in Chapter 6 and show results from different simulations related to the analysis of Chapter 5, in the Appendix.

Chapter 2

Bayesian Modelling & Computations

Statistical inference has become vital to scientists, as there are essentials and desire in learning the underlying mechanisms of observed data. However taking into account all possible uncertainties in predictions will be an important and challenging step in this analysis. Hence, the common solution is constructed in a Bayesian setting, where expert knowledge can play a significant role in exploring the distribution of parameters. The broad and complete inference that can be derived from Bayesian methods is the main reason that brings huge attention to this area and hence our focus in this thesis.

In this chapter we will review all the essential fundamentals and notations of the Bayesian modelling used in this thesis. More details on Bayesian Statistics can be found in Bernardo and Smith (2001); Gelman et al. (2014). We will also describe various Markov Chain Monte Carlo (MCMC) algorithms that are essential for the remaining part of this thesis. We begin with a comparison between Bayesian and Frequentist approaches by emphasizing the importance of Bayesian modelling in sta-

tistical inference.

2.1 Bayesian vs Frequentist Approaches

Bayesian methods play important roles in statistical inference, as they quantify uncertainties with probability distributions, using our prior knowledge on unknown model parameters. More specifically, in a Bayesian framework, data are observed and assumed fixed, while unknown model parameters are described by a probability distribution. In contrast, in a frequentist approach, parameters are considered to be fixed while data are generated from a repeatable random sample process. In frequentist methods, point estimations are derived along with confidence intervals, while in Bayesian methods, a probability distribution is describing the behaviour of parameters, with means or quantiles as possible candidates for estimation along with credible intervals for uncertainties.

To illustrate this difference, let $y_1, ..., y_n \overset{i.i.d.}{\sim} f(.|\theta)$ be a set of observed data from a specific distribution with parameter (vector) θ. The main aim is to make inference about θ. As a frequentist, one would consider θ to be fixed and concentrate on its estimation, for example using the Maximum Likelihood Estimate (MLE) $\hat{\theta}$, which is calculated by maximizing the Likelihood function:

$$\hat{\theta} = \arg\max_{\theta} L(\theta|y_1, ..., y_n) = \arg\max_{\theta} \prod_{i=1}^{n} f(y_i|\theta)$$

Hence, in a frequentist approach, all efforts are made towards point estimations of the model parameter and confidence intervals are also constructed to represent its variability. On the other hand, in a Bayesian approach, we consider the parameter to be random and find a probability distribution that can best describe it. Therefore,

in a Bayesian framework we define initial prior distributions from our intuitive beliefs or experiences about a particular model parameter, and update the probability distribution of the parameter as more and more data are available. Thus using the notation $\pi(\theta)$ for the prior distribution, the Bayesian probability distribution of the model parameters, called the *posterior distribution*, can be calculated through Bayes' Theorem:

$$\pi(\theta|y) = \frac{f(y|\theta)\pi(\theta)}{\int_\theta f(y|\theta)\pi(\theta)} \propto f(y|\theta)\pi(\theta) \tag{2.1}$$

where $f(y|\theta)$ is called the likelihood function and θ is the vector of model parameters. The normalizing constant $Z = \int_\theta f(y|\theta)\pi(\theta)$ is usually analytically tractable if conjugate priors are considered, i.e. priors that will generate posterior distributions with the same form of distributions. However, if the main goal is to sample θ, then Z can be discarded from the formulations as it is independent of θ.

2.2 Posterior Simulation

In practice, the posterior distribution becomes intractable for complex problems, especially as the integration becomes complicated with higher dimensional parameters. There are various methods proposed in the literature to challenge this problem. Strictly speaking, there is great area of research done on Variational Bayesian methods (MacKay, 2003; Bishop, 2006) that mainly find simplified analytical approximates of the posterior distribution using an optimization problem involving the Kullback–Leibler divergence (KL-divergence) distance. Approximate Bayesian Computation (ABC) methods (Beaumont et al., 2002; Blum, 2010; Drovandi et al., 2015) tackle intractable likelihood approximations in a Bayesian framework using summary statistics and simulation. Other approaches such as the integrated nested Laplace approxima-

tions (INLA) (Rue et al., 2009) are also proposed to make approximate Bayesian inference for only subset of structured additive regression models, the *latent gaussian models*. However, this approach only approximates marginal posterior distributions and lacks in approximating the joint posterior distribution. Although all these methods can be fast and computationally efficient, they are still problem-specific or are limited to a family of known analytical distributions.

Alternatively, Markov Chain Monte Carlo (MCMC) methods attempt to approximate the (joint) posterior distribution by sampling from its *exact* formulation, and hence produce more accurate results. Thus in this thesis, our focus will be on using and implementing various MCMC methods for different problems as we move further on. However, to understand the framework of MCMC methods and how they work, we will need to first describe the idea behind Monte Carlo Methods in the following subsection. We will later describe the different types of MCMC methods that are used in the remaining part of this thesis.

2.2.1 Monte Carlo Methods

Monte Carlo methods are computational algorithms that can estimate numerical values for various problems such as integrals, using repeated random sampling. Suppose we are interested in estimating an expected value defined as:

$$E(f(X)) = \int_{-\infty}^{+\infty} f(x) \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$$

where $X \sim N(0, 1)$. Since this integral may be hard to compute, we can propose a Monte Carlo method that can approximate $E(f(X))$. Thus, by generating S random samples $x_1, ..., x_S$ from the the standard normal distribution, we can calculate $f(x_s)$

for all $s \in \{1, ..., S\}$ and approximate $E(f(X))$ with the following unbiased estimator:

$$E(f(X)) \approx \bar{\mu}_S = \frac{1}{S} \sum_{s=1}^{S} f(x_s),$$

where:

$$\bar{\mu}_S \xrightarrow[S \to \infty]{L.L.N.} E(f(X))$$

by Law of Large Numbers (L.L.N.) (assuming Lebesgue integrability of $f(X)$).

However, in this case it is simple to draw samples from the standard normal distribution, but what happens if the samples you need are from a complicated distribution π, say a posterior distribution? The answer to this question is solved by Markov Chain Monte Carlo (MCMC) methods which are useful tools for sampling highly complex distributions. The basic idea is to generate a Markov Chain with stationary distribution π (which we will further explain). The theory behind the convergence of Markov Chains, plays an important role in explaining how MCMC algorithms work. Thus we will briefly explain this within the next section.

2.2.2 Markov Chain Convergence Theorem

A discrete-time Markov Chain is defined by a sequence of random variables $X_0, X_1, X_2, ...$ that can take possible values in the *state space* $\mathcal{X}$ with an *initial distribution* defined for X_0 and *transition probabilities* defined as:

$$p(x, A) = P(X_{n+1} \in A | X_n = x), \qquad \forall\ A \subseteq \mathcal{X}$$

One of the fundamental properties of Markov Chains is the *Markov Property*, that is:

$$P(X_{n+1} \in A | X_0, X_1, ..., X_n) = P(X_{n+1} \in A | X_n) \qquad \forall\ A \subseteq \mathcal{X}$$

In other words, the probability that the chain moves to the next state depends only on the *current* state and not the previous states.

Although this thesis is not focused on the theory of MCMC algorithms and their convergence, but we will review some definitions and theory that justifies how MCMC methods work. A complete background on Markov Chains can be found in the book by Rosenthal (2006).

Definition 1. *Consider a Markov chain* $\{X_i\}$ *on state space* $\mathcal{X}$ *with transition probability* $P(x, .)$. *Let* $\pi(.)$ *be a probability distribution defined on* $\mathcal{X}$. *Then* π *is a stationary distribution for the Markov chain if for* $x, y \in \mathcal{X}$*:*

$$\int_{x \in \mathcal{X}} \pi(dx) P(x, dy) = \pi(dy)$$

Definition 2. *A Markov chain is* ϕ*-irreducible, if there exists a non-zero* σ*-finite measure* ϕ *on* $\mathcal{X}$ *such that:*

$$\forall A: \ A \subseteq \mathcal{X} \text{ with } \phi(A) > 0 \quad \& \quad \forall x \in \mathcal{X}$$

$$\implies \exists \ n \in \mathbb{N}: \quad P^n(x, A) > 0$$

Definition 3. *A Markov chain is aperiodic, if there are no disjoint non-empty subsets* $\mathcal{X}_1, ..., \mathcal{X}_d \subseteq \mathcal{X}$ *for* $d \geq 2$, *such that* $P(x, \mathcal{X}_{i+1}) = 1$ *for all* $x \in \mathcal{X}_i \ \ (1 \leq i \leq d-1)$ *and* $P(x, \mathcal{X}_1) = 1$ *for all* $x \in \mathcal{X}_d$.

Now we will state the Markov chain convergence theorem which specifies how MCMC algorithms work.

Theorem 1. *Consider an aperiodic and* ϕ*-irreducible Markov chain defined on a state*

space $\mathcal{X}$ with stationary distribution π. Then for π.a.e. $x \in \mathcal{X}$:

$$\lim_{n\to\infty} ||P^n(x,.) - \pi(.)|| = 0$$

That is $\lim_{n\to\infty} P^n(x, A) = \pi(A)$ *for all measurable* $A \subseteq \mathcal{X}$.

Proof. See Meyn and Tweedie (2012); Rosenthal (2006). □

Assuming the goal is to sample from a complex π, MCMC methods are designed such that a Markov chain is generated with π stationary distribution. The *aperiodicity* of such Markov chains almost always hold (as there is usually positive probability of rejection in most MCMC methods), and *ϕ-irreducibility* is also straightforward to check and usually holds for MCMC algorithms. Hence, MCMC methods become reliable as they are built under the foundation of Markov chains. In the following section, we will describe most common MCMC algorithms that are also used in this thesis.

2.3 Most common MCMC algorithms

2.3.1 Metropolis-Hastings algorithm

One of the most popular MCMC methods is the Metropolis algorithm that was first introduced by Metropolis et al. (1953) and then extended as the Metropolis-Hastings algorithm by Hastings (1970). Let $\pi(x) = c\ g(x)$ be the complex distribution where we are interested to sample from (also called the target distribution) and assume it is only known up to a normalizing constant, i.e. $g(x)$ is known. The steps of the more generalized method, the Metropolis-Hastings algorithm, is described in Algorithm 1

below. The algorithm states that by starting at x, a proposal y is *accepted* with probability:

$$\alpha(x, y) = \min\left(1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right)$$

where $q(x, y)$ is the conditional probability of proposing state y given state x.

Algorithm 1: Metropolis-Hastings Algorithm

Input: Initial value X_0, burn-in iterations B, number of samples M
Output: $S_1, ..., S_M$ samples
1 **for** n **in** $\{1, ..., (B + M)\}$ **do**
2 Draw $Y_n \sim Q(X_{n-1}, .)$, where Q is the proposal distribution with probability density function q.
3 Calculate $A_n = \frac{\pi(Y_n)q(Y_n, X_{n-1})}{\pi(X_{n-1})q(X_{n-1}, Y_n)} = \frac{g(Y_n)q(Y_n, X_{n-1})}{g(X_{n-1})q(X_{n-1}, Y_n)}$.
4 Draw $U_n \sim$ Uniform[0,1]
5 **if** $U_n < A_n$ **then** $X_n = Y_n$ ("accept"), **else** $X_n = X_{n-1}$ ("reject")
6 **end for**
7 **return** $S_1 = X_{B+1}, ..., S_M = X_{B+M}$.

The Metropolis algorithm is the simplified version of Algorithm 1, with an exception in Step 2, where the proposal distribution Q is symmetric, i.e. $q(X_{n-1}, Y_n) = q(Y_n, X_{n-1})$, hence Step 3 also simplifies to $A_n = \frac{\pi(Y_n)}{\pi(X_{n-1})} = \frac{g(Y_n)}{g(X_{n-1})}$.

2.3.2 Random Walk Metropolis-Hastings (RWMH)

The Random Walk Metropolis-Hastings (RWMH) algorithm is a special case of the Metropolis-Hastings algorithm where $q(x, y) = q(y - x)$. For example $Q(x, .) \sim$ Uniform$[x - \epsilon, x + \epsilon]$ or $Q(x, .) \sim N(x, \sigma^2)$ can be considered as possible proposal distributions.

2.3.3 Metropolis-Hastings-within-Gibbs

The idea behind the Metropolis-Hastings-within-Gibbs algorithm is to update each coordinate at a time when the Markov chain is high-dimensional, and hence resulting

to be computationally more efficient. Assume the chain has d dimensions and denote $X_{-i} = (X_1, ..., X_{i-1}, X_{i+1}, ..., X_d)$ as the vector of all components except i. Then Step 2 in the Metropolis-Hastings algorithm will instead update each component at a time. In other words, at iteration n, the proposal Y_n will be constructed such that $Y_{n,i} \sim Q_i(X_{n-1,i}, .)$ with $Y_{n,-i} = X_{n-1,-i}$, where $Y_{n,i}$ is the i-th coordinate of Y_n.

2.3.4 Gibbs Sampler

The Gibbs Sampler is a special case of the Metropolis-Hastings-within-Gibbs with proposal distribution of the i-th component being the conditional distribution of that component with respect to the target distribution π, given the current values of all other components. Thus, the proposal density can be written as $q_i(x, y) = C(x_{-i})\pi(y)$ where $C(x_{-i})$ is the appropriate normalizing constant and $x_{-i} = y_{-i}$. Then proposals will always be *accepted* since:

$$\begin{aligned}
\alpha(x, y) &= \min\left(1, \frac{\pi(y)q_i(y, x)}{\pi(x)q_i(x, y)}\right) \\
&= \min\left(1, \frac{\pi(y)C(y_{-i})\pi(x)}{\pi(x)C(x_{-i})\pi(y)}\right) \\
&= \min(1, 1) = 1
\end{aligned}$$

The last equation holds since $C(x_{-i}) = C(y_{-i})$.

2.3.5 Langevin-Hastings algorihtm

The Langevin-Hastings algorithm is a special case of the Metropolis-Hastings algorithm where the proposal is defined as $Y_n \sim MVN(X_{n-1} + \frac{1}{2}\sigma^2 \nabla \log \pi(X_{n-1}), \sigma^2 I)$. This method tries to move towards a direction where π is increasing and hence can be computationally efficient. The theoretical background of this method can be found

in Roberts and Tweedie (1996); Roberts and Rosenthal (1998).

2.4 Optimal Acceptance Rates

To ensure that the MCMC method is mixing well and hence converging, we need to examine our choice of proposal distributions, as they can control the optimality in performance. For this purpose, *acceptance rates*, which are the proportion of *accepted* proposals to the total proposals, are useful tools to help determine such issues. The acceptance rates can specify how much the target distribution is being explored, that said, they should not be too low (close to 0), as this shows more rejection of proposals and hence a chain that is stuck and is not moving much. On the other hand, acceptance rates that are too high (close to 1) also show small movements of the chain, as only close-by states can be accepted highly. Hence, there needs to be an acceptance rate in between that can help the chain move more and explore the target distribution. It is proven that under mild conditions, the *optimal acceptance rates* of a d-dimensional Metropolis algorithm with a Gaussian proposal distribution is 0.234 as $d \to \infty$ (e.g. see Roberts et al. (1997); Roberts and Rosenthal (2001)). It is also shown that acceptance rates between 15% and 50% can still perform well (Roberts and Rosenthal (2001)).

Chapter 3

Likelihood Inflating Sampling Algorithm (LISA)

3.1 Introduction

Markov Chain Monte Carlo (MCMC) methods are essential for sampling highly complex distributions. They are of paramount importance in Bayesian inference as posterior distributions are generally difficult to characterize analytically (e.g., Brooks et al., 2011; Craiu and Rosenthal, 2014). When the posterior distribution is based on a massive sample of size N, posterior sampling can be computationally prohibitive since for some widely-used samplers at least $O(N)$ operations are needed to draw one MCMC sample. Additional issues include memory and storage bottlenecks where datasets are too large to be stored on one computer.

A common solution relies on parallelizing the computation task, i.e. dividing the load among a number of parallel *workers*, where a worker can be a processing unit, a computer, etc. Given the abundant availability of processing units, such strategies can be extremely efficient as long as there is no need for frequent communication

between workers. Some have discussed parallel MCMC methods (Wilkinson, 2006; Rosenthal, 2000; Laskey and Myers, 2003) such that each worker runs on the full dataset. However, these methods do not resolve memory overload, and can face difficulties in assessing the number of burn-in iterations for each processor.

Alternative subsampling MCMC approaches (Maclaurin and Adams, 2014; Bardenet et al., 2017) propose to reduce computational costs by only evaluating the likelihood of a subset of data at each iteration. However these methods are serial and can still suffer from memory bottlenecks.

A truly parallel approach is to divide the dataset into smaller groups and run parallel MCMC methods on each subset using different workers. Such techniques benefit from not demanding space on each computer to store the full dataset. Generally, one needs to avoid frequent communication between workers, as it is time consuming. In a typical divide and conquer strategy the data is partitioned into non-overlapping sub-sets, called *shards*, and each shard is analyzed by a different worker. For such strategies some essential MCMC-related questions are: 1) how to define the sub-posterior distributions for each shard, and 2) how to combine the MCMC samples obtained from each sub-posterior so that we can recover the same information that would have been obtained by sampling the full posterior distribution. Existing communication-free parallel methods proposed by Scott et al. (2013), Neiswanger et al. (2013) and Wang and Dunson (2013) have in common the fact that the product of the unnormalized sub-posteriors is equal to the unnormalized full posterior distribution, but differ in the strategies used to combine the samples. Specifically, Neiswanger et al. (2013) approximate each sub-posterior using kernel density estimators, while Wang and Dunson (2013) use the Weierstrass transformation. The popular Consensus Monte Carlo (CMC) method (Scott et al., 2013) relies on a weighted averaging approach to combine sub-posterior samples. The CMC relies on theoretical deriva-

tions that guarantee its validity when the full-data posterior and all sub-posteriors are Gaussian or mixtures of Gaussian.

We introduce a new communication-free parallel method, the *Likelihood Inflating Sampling Algorithm (LISA)*, that also relies on independent and parallel processing of the shards by different workers to sample the sub-posterior distributions. The latter are defined differently than in the competing approaches described above. In this chapter, we develop techniques to combine the sub-posterior draws obtained for LISA in the case of Bayesian Additive Regression Trees (BART) (Chipman et al., 1998, 2010; Kapelner and Bleich, 2013) and compare the performance of our method with CMC.

Sections 3.2 and 3.3 contain a brief review of the CMC algorithm and the detailed description of LISA, respectively. Section 3.4 illustrates the potential difference brought by LISA over CMC in a simple Bernoulli example, and includes a simple application of LISA to linear regression models. Section 3.5 contains the justification for a modified and improved version of LISA for BART. Numerical experiments and the analysis of socio-economic data presented in Section 3.6 examine the computational performance of the algorithms proposed here and compare it with CMC. We end the chapter with some ideas for future work. The Appendix contains theoretical derivations and descriptions of the steps used when running BART.

3.2 Review of Consensus Monte Carlo

In this chapter we assume that of interest is to generate samples from $\pi(\theta|\vec{Y}_N)$, the posterior distribution θ given the sample $\vec{Y}_N = \{Y_1, \ldots, Y_N\}$ of size N. The assumption is that N is large enough to prohibit running a standard MCMC algorithm in which draws from π are obtained on a single computer. We use the notation

$\pi(\theta|\vec{Y}_N) \propto f(\vec{Y}_N|\theta)p(\theta)$, where $f(\vec{Y}_N|\theta)$ is the likelihood function corresponding to the observed data $\vec{Y}_N$ and $p(\theta)$ is the prior. Major issues with MCMC posterior sampling for big data can be triggered because a) the data sample is too large to be stored on a single computer, or b) each chain update is too costly, e.g. if π is sampled via a Metropolis-Hastings type of algorithm each update requires N likelihood calculations.

In order to reduce the computational costs, the CMC method of Scott et al. (2013) partitions the sample into K independent batches (i.e. $\vec{Y}_N = \cup_{j=1}^{K} Y^{(j)}$) and uses the workers independently and in parallel to sample each sub-posterior. More precisely, the j-th worker ($j = 1, ..., K$) will generate samples from the j-th sub-posterior distribution defined as:

$$\pi_{j,CMC}(\theta|Y^{(j)}) \propto f(Y^{(j)}|\theta)p(\theta)^{1/K}.$$

Note that the prior for each batch is considered to be $p_j(\theta) = [p(\theta)]^{1/K}$ such that $p(\theta) = \prod_{j=1}^{K} p_j(\theta)$ and thus the overall full-data unnormalized posterior distribution which we denote as $\pi_{Full}(\theta|\vec{Y}_N)$ is equal to the product of unnormalized sub-posterior distributions, i.e.

$$\pi_{Full}(\theta|\vec{Y}_N) \propto \prod_{j=1}^{K} \pi_{j,CMC}(\theta|Y^{(j)}).$$

When the full posterior is Gaussian, the weighted averages of the sub-samples from all batches can be used as full-data posterior draws. That is, assuming $\theta_1^{(k)}, ..., \theta_S^{(k)}$ are S sub-samples from the kth worker then the s-th approximate full posterior draw will be:

$$\theta_s = (\sum_k w_k)^{-1} \sum_k w_k \theta_s^{(k)}$$

where the weights $w_k = \Sigma_k^{-1}$ are optimal for Gaussian models with $\Sigma_k = \mathrm{Var}(\theta|y^{(k)})$.

In the next section we introduce an alternative method to define the sub-posteriors

in each batch.

3.3 Likelihood Inflating Sampling Algorithm (LISA)

LISA is an alternative to CMC that also benefits from independently processing each batch on a different worker. Assuming that the data are i.i.d., the dataset is randomly divided with equal probability into K batches of approximately equal size n. We then define the sub-posterior distributions for each machine by adjusting the likelihood function without making changes to the prior. Thus the j-th sub-posterior distribution will be:

$$\pi_{j,LISA}(\theta|Y^{(j)}) \propto [f(Y^{(j)}|\theta)]^K p(\theta).$$

Assuming the data is iid, inflating the likelihood function K-times is intuitive because the sub-posterior from each batch of data will be a closer representation of the whole data posterior. We expect that sub-posteriors sampled by each worker will be closer to the full posterior thus improving the computational efficiency.

We indeed prove in a theorem below that under mild conditions, LISA's sub-posterior distributions are asymptotically closer to the full posterior than those produced by the CMC-type approach.

The Taylor's series expansion for a log-posterior density $\log \pi(\theta|\vec{Y}_N)$ around its posterior mode $\hat{\theta}_N$ yields the approximation

$$\log \pi(\theta|\vec{Y}_N) \approx \log \pi(\hat{\theta}_N|\vec{Y}_N) - \frac{1}{2}(\theta - \hat{\theta}_N)^T \hat{I}_N (\theta - \hat{\theta}_N)$$

where $\hat{I}_N = -\frac{\partial^2 \log(\pi(\theta|\vec{Y}_N))}{\partial\theta\partial\theta^T}|_{\theta=\hat{\theta}_N}$. Exponentiating both sides will result in

$$\pi(\theta|\vec{Y}_N) \approx \pi(\hat{\theta}_N|\vec{Y}_N) \exp\left[-\frac{1}{2}(\theta - \hat{\theta}_N)^T \hat{I}_N (\theta - \hat{\theta}_N)\right]$$

which shows asymptotic normality, i.e. $\hat{I}_N^{1/2}(\Theta - \hat{\theta}_N) \xrightarrow{D} N(0, I)$ as $N \to \infty$ where $\Theta \sim \pi(.|\vec{Y}_N)$. Let $\hat{\theta}_{n,L}^{(j)}$ and $\hat{\theta}_{n,C}^{(j)}$ denote the *j-th* sub-posterior modes in LISA and CMC, respectively. Similarly, $\hat{I}_{n,L}^{(j)}$ and $\hat{I}_{n,C}^{(j)}$ denote the negative second derivative of the *j-th* log sub-posterior for LISA and CMC, respectively, when calculated at the mode. Assuming K is fixed, consider the assumptions:

A1: There exist θ_L, θ_C such that if we define $\epsilon_{n,L}^{(j)} = |\hat{\theta}_{n,L}^{(j)} - \theta_L|$ and $\epsilon_{n,C}^{(j)} = |\hat{\theta}_{n,C}^{(j)} - \theta_C|$, then $\max_{1 \le j \le K} \epsilon_{n,L}^{(j)} \to 0$ and $\max_{1 \le j \le K} \epsilon_{n,C}^{(j)} \to 0$ w.p. 1 as $n \to \infty$.

A2: $|\hat{I}_{n,L}^{(i)} - \hat{I}_{n,L}^{(j)}| \longrightarrow 0$ and $|\hat{I}_{n,C}^{(i)} - \hat{I}_{n,C}^{(j)}| \to 0$ w.p. 1 $\forall\ i \neq j$ as $n \to \infty$.

A3: π_{Full}, $\pi_{j,LISA}$, and $\pi_{j,CMC}$ are unimodal distributions that have continuous derivatives of order 2.

Theorem 2. *Assume that assumptions* **A1** *through* **A3** *hold and if* $\Theta_{Full} \sim \pi_{Full}(.|\vec{Y}_N)$ *we also assume* $\hat{I}_N^{1/2}(\Theta_{Full} - \hat{\theta}_N) \xrightarrow{D} N(0, I)$ *as* $N \to \infty$. *If* $\Theta_{j,LISA} \sim \pi_{j,LISA}(.|Y^{(j)})$ *and* $\Theta_{j,CMC} \sim \pi_{j,CMC}(.|Y^{(j)})$ *then as* $N \to \infty$

$$\hat{I}_N^{1/2}(\Theta_{j,LISA} - \hat{\theta}_N) \xrightarrow{D} N(0, I) \quad \textit{and} \quad \hat{I}_N^{1/2}(\Theta_{j,CMC} - \hat{\theta}_N) \xrightarrow{D} N(0, KI) \quad \forall\ j \in \{1, \ldots, K\}.$$

Proof. For simplicity, assume $n = N/K$ is the number of observations in each batch and consider θ to be a one-dimensional parameter. We will show Theorem 1's statements separately for LISA and CMC.

LISA: Given assumption **A1**, $\forall\ j$ *w.p.*1:

$$\forall\ \epsilon_1^{(j)} > 0\ \ \exists\ \ M_1 > 0\ \ s.t.\ \ \forall\ n > M_1 \quad |\hat{\theta}_{n,L}^{(j)} - \theta_L| < \epsilon_1^{(j)} \tag{3.1}$$

hence with the continuous assumption in **A3**, we have $\forall\ j\ \ w.p.1$:

$$\forall\, \gamma_1^{(j)} > 0\ \exists\ M_1 > 0\ \ s.t.\ \forall\, n > M_1\ \ \Big|\log\left(\pi_{j,LISA}(\hat{\theta}_{n,L}^{(j)}|Y^{(j)})\right) - \log\left(\pi_{j,LISA}(\theta_L|Y^{(j)})\right)\Big| < \gamma_1^{(j)} \tag{3.2}$$

We know that $\left(\pi_{Full}(\theta|Y_N)\right)^K \propto \prod_{j=1}^K \pi_{j,LISA}(\theta|Y^{(j)})$, hence:

$$\log\left(\pi_{Full}(\theta|Y_N)\right) = \frac{1}{K}\sum_{j=1}^{K}\log\left(\pi_{j,LISA}(\theta|Y^{(j)})\right) + c \tag{3.3}$$

where c is a constant. This implies that

$$\log\left(\pi_{Full}(\theta|Y_N)\right)\Big|_{\theta=\hat{\theta}_N} = \frac{1}{K}\sum_{j=1}^{K}\log\left(\pi_{j,LISA}(\hat{\theta}_N|Y^{(j)})\right) + c \tag{3.4}$$

Since $\hat{\theta}_N$ is the full posterior mode:

$$\Big[\frac{1}{K}\sum_{j=1}^{K}\log\left(\pi_{j,LISA}(\hat{\theta}_N|Y^{(j)})\right)\Big] - \Big[\frac{1}{K}\sum_{j=1}^{K}\log\left(\pi_{j,LISA}(\theta_L|Y^{(j)})\right)\Big] \geq 0 \tag{3.5}$$

and because $\hat{\theta}_{n,L}^{(j)}$ is the mode of $\pi_{j,LISA}$:

$$\frac{1}{K}\sum_{j=1}^{K}\log\left(\pi_{j,LISA}(\hat{\theta}_N|Y^{(j)})\right) \leq \frac{1}{K}\sum_{j=1}^{K}\log\left(\pi_{j,LISA}(\hat{\theta}_{n,L}^{(j)}|Y^{(j)})\right) \tag{3.6}$$

and thus from (3.5) and (3.6), we will have:

$$\begin{aligned} 0 &\leq \Big[\frac{1}{K}\sum_{j=1}^{K}\log\left(\pi_{j,LISA}(\hat{\theta}_N|Y^{(j)})\right)\Big] - \Big[\frac{1}{K}\sum_{j=1}^{K}\log\left(\pi_{j,LISA}(\theta_L|Y^{(j)})\right)\Big] \\ &\leq \Big[\frac{1}{K}\sum_{j=1}^{K}\log\left(\pi_{j,LISA}(\hat{\theta}_{n,L}^{(j)}|Y^{(j)})\right)\Big] - \Big[\frac{1}{K}\sum_{j=1}^{K}\log\left(\pi_{j,LISA}(\theta_L|Y^{(j)})\right)\Big] \end{aligned} \tag{3.7}$$

Taking absolute values from last inequality in (3.7) and using the triangle inequality,

we have $w.p.1$:

$$\frac{1}{K}\Big|\sum_{j=1}^{K}\Big[\log\left(\pi_{j,LISA}(\hat{\theta}_N|Y^{(j)})\right)-\log\left(\pi_{j,LISA}(\theta_L|Y^{(j)})\right)\Big]\Big|\leq$$
$$\frac{1}{K}\Big|\sum_{j=1}^{K}\Big[\log\left(\pi_{j,LISA}(\hat{\theta}^{(j)}_{n,L}|Y^{(j)})\right)-\log\left(\pi_{j,LISA}(\theta_L|Y^{(j)})\right)\Big]\Big|\leq$$
$$\frac{1}{K}\sum_{j=1}^{K}\Big|\log\left(\pi_{j,LISA}(\hat{\theta}^{(j)}_{n,L}|Y^{(j)})\right)-\log\left(\pi_{j,LISA}(\theta_L|Y^{(j)})\right)\Big|\leq\frac{1}{K}\sum_{j=1}^{K}\gamma_1^{(j)}=\gamma_1 \tag{3.8}$$

The last inequality in (3.8) is followed by (3.2). From inequality (3.8) and the fact that the posteriors are unimodal as stated in assumption **A3**, we can conclude $w.p.1$:

$$|\hat{\theta}_N-\theta_L|\longrightarrow 0\quad\text{as}\quad N\to\infty \tag{3.9}$$

From (3.9) and assumption **A1**, we can conclude $\forall\ j\ \ w.p.1$:

$$|\hat{\theta}_N-\hat{\theta}^{(j)}_{n,L}|\longrightarrow 0\quad\text{as}\quad n\to\infty \tag{3.10}$$

And from (3.10) and assumption **A3**, $wp.1$, $\forall\ j$:

$$\Big|\frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,LISA}(\theta|Y^{(j)})\right)\Big|_{\theta=\hat{\theta}_N}-\frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,LISA}(\theta|Y^{(j)})\right)\Big|_{\theta=\hat{\theta}^{(j)}_{n,L}}\Big|\longrightarrow 0\quad\text{as}\quad n\to\infty \tag{3.11}$$

In addition from (3.10), we can also conclude that for any i and j such that $i\neq j$:

$$|\hat{\theta}^{(i)}_{n,L}-\hat{\theta}^{(j)}_{n,L}|\longrightarrow 0\quad\text{as}\quad n\to\infty \tag{3.12}$$

And thus benefitting from (3.11), (3.12), and the structural form of sub-posterior

distributions in LISA (or assumption **A2**) for $i \neq j$, we have $w.p.1$:

$$\Big|\frac{\partial^2}{\partial\theta^2}\log\left(\pi_{i,LISA}(\theta|Y^{(i)})\right)\Big|_{\theta=\hat{\theta}^{(i)}_{n,L}} - \frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,LISA}(\theta|Y^{(j)})\right)\Big|_{\theta=\hat{\theta}^{(j)}_{n,L}}\Big| \longrightarrow 0 \quad \text{as} \quad n\to\infty \tag{3.13}$$

Now take the second derivative with respect to θ from both sides of (3.3) evaluated at $\theta = \hat{\theta}_N$:

$$-\hat{I}_N := \frac{\partial^2}{\partial\theta^2}\log\left(\pi_{Full}(\theta|Y_N)\right)\Big|_{\theta=\hat{\theta}_N} = \frac{1}{K}\sum_{j=1}^{K}\frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,LISA}(\theta|Y^{(j)})\right)\Big|_{\theta=\hat{\theta}_N} \tag{3.14}$$

Denoting:

$$-\hat{I}^{(j)}_{n,L} := \frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,LISA}(\theta|Y^{(j)})\right)\Big|_{\theta=\hat{\theta}^{(j)}_{n,L}} \tag{3.15}$$

Using (3.11), (3.13), and (3.14), will result in:

$$\begin{aligned}
|\hat{I}_N - \hat{I}^{(j)}_{n,L}| &= \Big|\frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,LISA}(\theta|Y^{(j)})\right)|_{\theta=\hat{\theta}^{(j)}_{n,L}} - \frac{1}{K}\sum_{i=1}^{K}\frac{\partial^2}{\partial\theta^2}\log\left(\pi_{i,LISA}(\theta|Y^{(i)})\right)|_{\theta=\hat{\theta}_N}\Big| \\
&\leq \frac{1}{K}\Big|\frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,LISA}(\theta|Y^{(j)})\right)|_{\theta=\hat{\theta}^{(j)}_{n,L}} - \frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,LISA}(\theta|Y^{(j)})\right)|_{\theta=\hat{\theta}_N}\Big| + \\
&\frac{1}{K}\Big|\sum_{i\neq j}\Big[\frac{\partial^2}{\partial\theta^2}\log\left(\pi_{i,LISA}(\theta|Y^{(i)})\right)|_{\theta=\hat{\theta}_N} - \frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,LISA}(\theta|Y^{(j)})\right)|_{\theta=\hat{\theta}^{(j)}_{n,L}}\Big]\Big| \longrightarrow 0
\end{aligned} \tag{3.16}$$

$w.p.1\ \forall\ j$.

CMC: In CMC, since $\pi_{Full}(\theta|Y_N) \propto \prod_{j=1}^{K} \pi_{j,CMC}(\theta|Y^{(j)})$, we will have

$$\log\left(\pi_{Full}(\theta|Y_N)\right) = \sum_{j=1}^{K} \log\left(\pi_{j,CMC}(\theta|Y^{(j)})\right) + c \tag{3.17}$$

where c is a constant. Thus, using **A1** through **A3** with a similar proof as in LISA, we can show that $w.p.1$:

$$|\hat{\theta}_N - \theta_C| \longrightarrow 0 \quad \text{as} \quad N \to \infty \tag{3.18}$$

and hence $\forall\ j \quad w.p.1$:

$$|\hat{\theta}_N - \hat{\theta}^{(j)}_{n,C}| \longrightarrow 0 \quad \text{as} \quad n \to \infty \tag{3.19}$$

$$|\hat{\theta}^{(i)}_{n,C} - \hat{\theta}^{(j)}_{n,C}| \longrightarrow 0 \quad \text{as} \quad n \to \infty \quad \text{for} \ \ i \neq j \tag{3.20}$$

Similarly, from (3.19) and assumption **A3**, $wp.1$, $\forall\ j$:

$$\Big|\frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,CMC}(\theta|Y^{(j)})\right)\Big|_{\theta=\hat{\theta}_N} - \frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,CMC}(\theta|Y^{(j)})\right)\Big|_{\theta=\hat{\theta}^{(j)}_{n,C}}\Big| \longrightarrow 0 \quad \text{as} \quad n \to \infty \tag{3.21}$$

And again benefitting from (3.20), (3.21), and the structural form of sub-posterior distributions in CMC (or assumption **A2**), for $i \neq j$, we have $w.p.1$:

$$\Big|\frac{\partial^2}{\partial\theta^2}\log\left(\pi_{i,CMC}(\theta|Y^{(i)})\right)\Big|_{\theta=\hat{\theta}^{(i)}_{n,C}} - \frac{\partial^2}{\partial\theta^2}\log\left(\pi_{j,CMC}(\theta|Y^{(j)})\right)\Big|_{\theta=\hat{\theta}^{(j)}_{n,C}}\Big| \longrightarrow 0 \quad \text{as} \quad n \to \infty \tag{3.22}$$

Now taking the second derivative with respect to θ from both sides of (3.17)

evaluated at $\theta = \hat{\theta}_N$:

$$-\hat{I}_N := \frac{\partial^2}{\partial\theta^2} \log\left(\pi_{Full}(\theta|Y_N)\right)\Big|_{\theta=\hat{\theta}_N} = \sum_{j=1}^{K} \frac{\partial^2}{\partial\theta^2} \log\left(\pi_{j,CMC}(\theta|Y^{(j)})\right)\Big|_{\theta=\hat{\theta}_N} \tag{3.23}$$

Denoting:

$$-\hat{I}^{(j)}_{n,C} := \frac{\partial^2}{\partial\theta^2} \log\left(\pi_{j,CMC}(\theta|Y^{(j)})\right)\Big|_{\theta=\hat{\theta}^{(j)}_{n,C}} \tag{3.24}$$

Using (3.21), (3.22), and (3.23), will similarly result in:

$$\begin{aligned}
|\frac{\hat{I}_N}{K} - \hat{I}^{(j)}_{n,C}| &= \Big|\frac{\partial^2}{\partial\theta^2} \log\left(\pi_{j,CMC}(\theta|Y^{(j)})\right)|_{\theta=\hat{\theta}^{(j)}_{n,C}} - \frac{1}{K}\sum_{i=1}^{K} \frac{\partial^2}{\partial\theta^2} \log\left(\pi_{i,CMC}(\theta|Y^{(i)})\right)|_{\theta=\hat{\theta}_N}\Big| \\
&\leq \frac{1}{K}\Big|\frac{\partial^2}{\partial\theta^2} \log\left(\pi_{j,CMC}(\theta|Y^{(j)})\right)|_{\theta=\hat{\theta}^{(j)}_{n,C}} - \frac{\partial^2}{\partial\theta^2} \log\left(\pi_{j,CMC}(\theta|Y^{(j)})\right)|_{\theta=\hat{\theta}_N}\Big| + \\
&\frac{1}{K}\Big|\sum_{i\neq j}\Big[\frac{\partial^2}{\partial\theta^2} \log\left(\pi_{i,CMC}(\theta|Y^{(i)})\right)|_{\theta=\hat{\theta}_N} - \frac{\partial^2}{\partial\theta^2} \log\left(\pi_{j,CMC}(\theta|Y^{(j)})\right)|_{\theta=\hat{\theta}^{(j)}_{n,C}}\Big]\Big| \longrightarrow 0
\end{aligned} \tag{3.25}$$

$w.p.1\ \forall\ j.$

□

Theorem 1 shows the difference between sub-posterior distributions for CMC and LISA, with LISA's sub-posterior distributions being asymptotically similar to the full posterior distribution. This suggests that draws from LISA sub-posteriors can be combined using uniform weights.

Remarks:

1. When data are iid we expect the shards to become more and more similar as N (and thus $n = N/K$) increases and assumption **A1** is expected to hold for

general models.

2. Assumption **A2** in Theorem 1 holds due to the structural form of sub-posteriors in LISA and CMC.

3. The validity of using uniform weights with LISA's sub-posterior draws is justified asymptotically, but we will see that this approximation can be exact in some examples, e.g. for a Bernoulli model with balanced batch samples, while in others modified weights can improve the performance of the sampler. In this respect LISA is similar to other embarrassingly parallel strategies where one must carefully consider the model of interest in order to find the best way to combine the sub-posterior samples.

In the next section we will illustrate LISA in some simple examples and compare its performance to the full-data posterior sampling as well as CMC.

3.4 Motivating Examples

In this section we examine some simple examples where theoretical derivations can be carried out in detail. We emphasize the difference between LISA and CMC.

3.4.1 Bernoulli Random Variables

Consider $y_1, ..., y_N$ to be N i.i.d. Bernoulli random variables with parameter θ. Hence, we consider a prior $p(\theta) = \text{Beta}(a, b)$. Assuming that we know little about the size of θ we set $a = b = 1$ which corresponds to a $U(0, 1)$ prior. The resulting full-data posterior $\pi_{Full}(\theta|\vec{Y}_N)$ is $\text{Beta}(S+a, N-S+b)$ where $S = \sum_{i=1}^{N} y_i$ is the total number of ones. Suppose we divide the data into K batches with S_j number of ones in batch j, such that $S_j = \frac{S}{K} \quad \forall\ j \in \{1, .., K\}$, i.e. the number of 1's are divided equally

between batches. Then the j^{th} sub-posterior based on batch-data of size $n = \frac{N}{K}$ for each method will be:

CMC:

$$\begin{aligned}\pi_{j,CMC}(\theta|Y^{(j)}) &= \text{Beta}\left(S_j + \frac{a-1}{K} + 1, n - S_j + \frac{b-1}{K} + 1\right) \\ &= \text{Beta}\left(\frac{S}{K} + \frac{a-1}{K} + 1, \frac{N-S}{K} + \frac{b-1}{K} + 1\right)\end{aligned}$$

LISA:

$$\begin{aligned}\pi_{j,LISA}(\theta|Y^{(j)}) &= \text{Beta}(S_jK + a, (n - S_j)K + b) \\ &= \text{Beta}(S + a, N - S + b)\end{aligned}$$

which implies

$$\pi_{j,LISA}(\theta|Y^{(j)}) = \pi_{Full}(\theta|\vec{Y}_N) \quad \forall\ j \in \{1, ..., K\}.$$

In this simple case any one of LISA 's sub-posterior distributions is equal to the full posterior distribution if the batches are balanced, i.e. the number of 1's are equally split across all batches. Thus, LISA's sub-samples from any batch will represent correctly the full posterior. On the other hand, the draws from the CMC sub-posterior distributions will need to be recombined to obtain a representative sample from the true full posterior $\pi_{Full}(\theta|\vec{Y}_N)$.

However, when the number of ones is unequally distributed among the batches it is not easy to pick the winner between CMC and LISA as both require a careful weighting of each batch sub-posterior samples.

In the remaining part of this chapter, we will mainly focus on the performance of

LISA when it is applied to the Bayesian Additive Regression Trees (BART) model. Interestingly, we discover that using a minor modification inspired by running LISA on the simpler Bayesian Linear Regression model we can approximate the full posterior. The idea behind the modification is described in the next section.

3.4.2 Bayesian Linear Regression

Consider a standard linear regression model

$$Y = X\beta + \epsilon \tag{3.26}$$

where $\beta \in \mathbf{R}^p$, $X \in \mathbf{R}^{N\times p}$ and $Y, \epsilon \in \mathbf{R}^N$ with $\epsilon \sim \mathcal{N}_N(0, \sigma^2\mathbf{I}_N)$. To simplify the presentation we consider the improper prior

$$p(\beta, \sigma^2) \propto \sigma^{-2}. \tag{3.27}$$

Straightforward calculations show that the conditional posterior distributions for the full data are

$$\pi_{Full}(\sigma^2|Y, X) = \text{Inv-Gamma}\left(\frac{N-p}{2}, \frac{s^2(N-p)}{2}\right) \tag{3.28}$$

$$\pi_{Full}(\beta|\sigma^2, Y, X) = N(\hat{\beta}, \sigma^2(X^TX)^{-1}) \tag{3.29}$$

where $\hat{\beta} = (X^TX)^{-1}X^TY$ and $s^2 = \frac{(Y-X\hat{\beta})^T(Y-X\hat{\beta})}{N-p}$.

A Monte Carlo sampler designed to sample from $\pi_{Full}(\beta, \sigma^2|Y, X)$ will iteratively sample σ^2 using (3.28) and then β via (3.29). If we denote β_{Full} the r.v. with density $\pi_{Full}(\beta|Y, X)$ then, using the iterative formulas for conditional mean and variance we

obtain

$$E[\beta_{Full}|Y,X] = (X^TX)^{-1}X^TY$$

and

$$\text{Var}(\beta_{Full}|Y,X) = (X^TX)^{-1}\frac{(N-p)/2}{(N-p)/2-1}s^2 = (X^TX)^{-1}s^2 + O(N^{-1}). \tag{3.30}$$

We examine below the statistical properties of the samples produced by LISA. If the data are divided into K equal batches of size $n = N/K$, let us denote $Y^{(j)}$ and $X^{(j)}$ the response vector and model matrix from the jth batch, respectively.

With the prior given in (3.27), the sub-posteriors produced by LISA have the following conditional densities

$$\pi_j(\sigma^2|Y^{(j)},X^{(j)}) = \text{Inv-Gamma}\left(\frac{N-p}{2}, \frac{Ks_j^2(n-p)}{2}\right) \tag{3.31}$$

$$\pi_j(\beta|\sigma^2,Y^{(j)},X^{(j)}) = N(\hat{\beta}_j, \frac{\sigma^2}{K}(X^{(j)\,T}X^{(j)})^{-1}), \tag{3.32}$$

where $\hat{\beta}_j = (X^{(j)\,T}X^{(j)})^{-1}X^{(j)\,T}Y^{(j)}$ and $s_j^2 = \frac{(Y^{(j)}-X^{(j)}\hat{\beta}_j)^T(Y^{(j)}-X^{(j)}\hat{\beta}_j)}{n-p}$ for all $1 \le j \le K$.

Similarly, a Monte Carlo sampler designed to sample from $\pi_j(\beta,\sigma^2|Y^{(j)},X^{(j)})$ will iteratively sample σ^2 from (3.31) and then β from (3.32). It can be shown using the iterative formulas for conditional means and variances that

$$E[\beta|Y^{(j)},X^{(j)}] = \hat{\beta}_j$$

and

$$\text{Var}(\beta|Y^{(j)},X^{(j)}) = (X^{(j)\,T}X^{(j)})^{-1}\frac{s_j^2(n-p)/2}{(N-p)/2-1} = (X^{(j)\,T}X^{(j)})^{-1}\frac{s_j^2(n-p)}{(N-p)} + O(N^{-1}).$$

In order to combine the sub-posterior samples we propose using the weighted average

$$\beta_{LISA} = (\sum_{j=1}^{K} W_j)^{-1} \sum_{j=1}^{K} W_j \beta_j, \tag{3.33}$$

where $\beta_j \sim \pi_j(\beta|Y^{(j)}, X^{(j)})$ and $W_j = \frac{X^{(j)\,T} X^{(j)}}{\sigma^2}$. Since $\sum_{j=1}^{K} X^{(j)\,T} X^{(j)} = X^T X$ we get

$$E[\beta_{LISA}|Y, X] = \hat{\beta} = (X^T X)^{-1} X^T Y \tag{3.34}$$

and

$$\mathrm{Var}(\beta_{LISA}|Y, X) = (X^T X)^{-1} \frac{n-p}{N-p} \left[\sum_{j=1}^{K} s_j^2 (X^{(j)\,T} X^{(j)}) \right] (X^T X)^{-1} \approx (X^T X)^{-1} \frac{n-p}{N-p} s^2, \tag{3.35}$$

where the last approximation in (3.35) is based on the assumption that $s_j^2 \approx s^2$ as both are unbiased estimators for σ^2 based on n and, respectively, N observations. It is apparent that the variance computed in (3.35) is roughly K times smaller than the target given in (3.30). In order to avoid underestimating the variance of the posterior distribution we propose a modified LISA sampling algorithm which consists of the following steps:

$$\begin{aligned} \sigma^2 &\sim \text{Inv-Gamma}\left(\frac{N-p}{2}, \frac{K s_j^2 (n-p)}{2}\right) \\ \tilde{\sigma} &= \sqrt{K}\sigma \\ \tilde{\beta} &\sim N(\hat{\beta}_j, \frac{\tilde{\sigma}^2}{K}(X^{(j)\,T} X^{(j)})^{-1}) = N(\hat{\beta}_j, \sigma^2 (X^{(j)\,T} X^{(j)})^{-1}). \end{aligned}$$

The intermediate step simply adjusts the variance samples so that

$$\mathrm{Var}(\tilde{\beta}|Y^{(j)}, X^{(j)}) = (X^{(j)\,T} X^{(j)})^{-1} \frac{s_j^2 K(n-p)/2}{(N-p)/2 - 1} = (X^{(j)\,T} X^{(j)})^{-1} \frac{s_j^2 K(n-p)}{(N-p)} + O(N^{-1}).$$

In turn, if we define

$$\beta_{modLISA} = (\sum_{j=1}^{K} W_j)^{-1} \sum_{j=1}^{K} W_j \tilde{\beta}_j, \tag{3.36}$$

then $E[\beta_{modLISA}|Y, X] = (X^T X)^{-1} X^T Y$ and

$$\text{Var}(\beta_{modLISA}|Y, X) = (X^T X)^{-1} \frac{K(n-p)}{N-p} \left[\sum_{j=1}^{K} s_j^2 (X^{(j)\,T} X^{(j)}) \right] (X^T X)^{-1} \approx (X^T X)^{-1} \frac{K(n-p)}{N-p} s^2. \tag{3.37}$$

While both (3.36) and (3.33) produce samples that have the correct mean, from equations (3.30), (3.35) and (3.37) we can see that the weighted average of the modified LISA samples have the variance closer to the desired target.

In the next section, we will examine LISA's performance on a more complex model, the Bayesian Additive Regression Trees (BART). The discussion above will guide our construction of a modified version of LISA for BART.

3.5 Bayesian Additive Regression Trees (BART)

Consider the nonparametric regression model:

$$y_i = f(x_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \sigma^2) \quad i.i.d.$$

where $x_i = (x_{i1}, ..., x_{ip})$ is a p-dimensional vector of inputs and f is approximated by a sum of m regression trees:

$$f(x) \approx \sum_{j=1}^{m} g(x; T_j, M_j)$$

where T_j denotes a binary tree consisting of a set of interior node decision rules and a set of terminal nodes. $M_j = \{\mu_{1j}, ..., \mu_{bj}\}$ is the set of parameter values associated

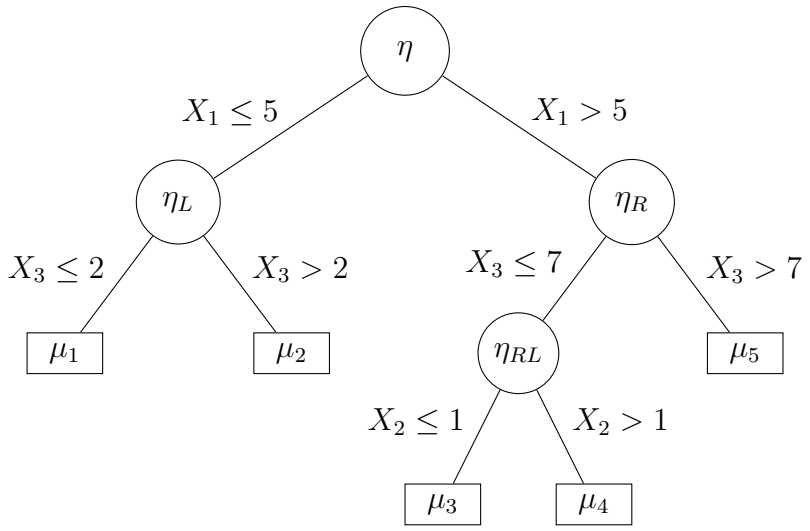


Figure 3.1: A binary tree with internal nodes η, η_L, η_R, and η_{RL} that maps each $x = (x_1, x_2, x_3)$ to one of its five terminal nodes according to the rules, and lastly assigns parameter μ_i.

with the b terminal nodes of T_j. In addition, $g(x; T_j, M_j)$ is the function that maps each x to a $\mu_{ij} \in M_j$. Figure 3.1 illustrates such binary trees. The regression model is then approximated by a sum-of-trees model

$$y_i = \sum_{j=1}^{m} g(x_i; T_j, M_j) + \epsilon_i \ , \qquad \epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$$

Let $\theta := ((T_1, M_1), ..., (T_m, M_m), \sigma^2)$ denote the vector of model parameters. Below, we briefly describe the prior specifications stated in Chipman et al. (2010) and Chipman et al. (1998).

Prior Specifications:

1. Prior Independence and Symmetry:

$$p((T_1, M_1), ..., (T_m, M_m), \sigma) = [\prod_j p(M_j|T_j)p(T_j)]p(\sigma)$$

where $p(M_j|T_j) = \prod_i p(\mu_{ij}|T_j)$.

2. Recommended number of trees: m=200 (Chipman et al., 2010) and m=50 (Kapelner and Bleich, 2013)

3. Tree prior $p(T_j)$, is characterised by three aspects:

   a. The probability that a node at depth $d = 0, 1, ...$ is non-terminal, which is assumed to have the form $\alpha(1+d)^{-\beta}$, where $\alpha \in (0, 1)$ and $\beta \geq 0$. (recommended values are $\alpha = 0.95$ and $\beta = 2$)

   b. The distribution on the splitting variable assignments at each interior node which is recommended to have a uniform distribution.

   c. The distribution on the splitting rule assignment in each interior node, conditional on the splitting variable which is also recommended to have a uniform distribution.

4. The conditional prior for μ_{ij} is $\mathcal{N}(\mu_\mu, \sigma_\mu^2)$ such that:

$$\begin{cases} m\mu_\mu - k\sqrt{m}\sigma_\mu = y_{min} \\ m\mu_\mu + k\sqrt{m}\sigma_\mu = y_{max} \end{cases}$$

   with $k = 2$ recommended.

5. The prior for σ^2 is Inv-Gamma$(\frac{\nu}{2}, \frac{\nu\lambda}{2})$ where $\nu = 3$ is recommended and λ is chosen such that $p(\sigma < \hat{\sigma}) = q$ with recommended $q = 0.9$ and sample variance $\hat{\sigma}$.

Hence the posterior distribution will have the form:

$$\pi(\theta) = \pi(\theta|Y,X) \propto \underbrace{\left\{(\sigma^2)^{-\frac{n}{2}} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{m} g(x_i;M_j,T_j))^2}\right\}}_{\text{Likelihood}} \times$$
$$\underbrace{\left\{\underbrace{(\sigma^2)^{-\frac{\nu}{2}-1} e^{-\frac{\nu\lambda}{2\sigma^2}}}_{\text{Prior of } \sigma^2} \left[\prod_{j=1}^{m} \sigma_\mu^{-b_j} (2\pi)^{-\frac{b_j}{2}} e^{-\frac{1}{2\sigma_\mu^2}\sum_{k=1}^{b_j}(\mu_{kj}-\mu_\mu)^2} p(T_j)\right]\right\}}_{\text{Prior}}. \quad (3.38)$$

Gibbs Sampling is used to sample from this posterior distribution. The algorithm iterates between the following steps:

1. $\sigma^2 \mid (T_1,M_1),...,(T_m,M_m),Y,X \ \propto$ Inv-Gamma(ρ,γ)
   where $\rho = \frac{\nu+n}{2}$ and $\gamma = \frac{1}{2}\ [\ \sum_{i=1}^{n}(y_i - \sum_{j=1}^{m} g(x_i;M_j,T_j))^2 + \lambda\nu\ ]$.

2. $(T_j,M_j) \mid T_{(j)},M_{(j)},\sigma,Y,X$ which is the same as drawing from the conditional $(T_j,M_j) \mid R_j,\sigma$ where $T_{(j)}$ denotes all trees except the j-th tree, and residual R_j is defined as:

$$R_j = \ g(x;,M_j,T_j) + \epsilon = \ y - \sum_{k\neq j} g(x;M_k,T_k).$$

   The sampling of (T_j,M_j) is performed in two steps:

   a. $T_j \mid R_j,\sigma$ and

   b. $M_j \mid T_j,R_j,\sigma$.

   Step b involves sampling from each component of M_j using

$$\mu_{ij} \mid T_j,R_j,\sigma \ \sim \ \mathcal{N}\left(\frac{\frac{\sigma^2}{\sigma_\mu^2}\ \mu_\mu \ + \ n_i\bar{R}_{j(i)}}{\frac{\sigma^2}{\sigma_\mu^2} \ + \ n_i}, \frac{\sigma^2}{\frac{\sigma^2}{\sigma_\mu^2} \ + \ n_i}\right)$$

where $\bar{R}_{j(i)}$ denotes the average residual (computed without tree j) at terminal node i with total number of observations n_i. The conditional density of T_j in step a can be expressed as:

$$p(T_j \mid R_j, \sigma) \quad \propto \quad p(T_j) \int p(R_j \mid M_j, T_j, \sigma)\; p(M_j \mid T_j, \sigma)\; dM_j. \tag{3.39}$$

The Metropolis-Hastings (MH) algorithm is then applied to draw T_j from (4.1) with four different proposal moves on trees:

- **GROW:** growing a terminal node (with probability 0.25);
- **PRUNE:** pruning a pair of terminal nodes (with probability 0.25);
- **CHANGE:** changing a non-terminal rule (with probability 0.4) (Kapelner and Bleich, 2013, change rules only for parent nodes with terminal children);
- **SWAP:** swapping a rule between parent and child (with probability 0.1) (This proposal move was removed by Kapelner and Bleich, 2013).

Detailed derivations involving the Metropolis-Hastings acceptance ratios are described in section 3.8.

Two existing packages in R, "BayesTree" and "bartMachine", can be used to run BART on any dataset, but as the sample size increases, these packages tend to run slower. In these situations we expect methods such as LISA or CMC to become useful, and for a fair illustration of the advantages gained we have used our own R implementation of BART and applied the same structure to implement LISA and CMC algorithm for BART. The Metropolis-Hastings acceptance ratios for LISA and CMC are also reported in the Appendix.

As discussed by Scott et al. (2013), the approximation to the posterior produced by the CMC algorithm can be poor. Thus, for comparison reasons, we applied both LISA and CMC to BART using a simulated dataset (described further) with $K = 30$ batches. Given Theorem 1, since LISA's sub-posterior distributions are asymptotically equivalent to the full posterior distribution, we examined its performance by uniformly taking sub-samples from all its batches as an approximation to full posterior samples. We will see further that LISA with uniform weights produces higher prediction accuracy compared to CMC. However, they both perform poorly in approximating the posterior samples as they generate larger trees and under-estimate σ^2, which results in over-dispersed posterior distributions.

The following sub-section discusses a modified version of LISA for BART which will have significant improvement in performance.

3.5.1 Modified LISA for BART

The under estimation of σ^2 when applying LISA to BART is similar to the problem encountered when using LISA for the linear regression model discussed in Section 3.4.2. This is not a coincidence since BART is also a linear regression model, albeit one where the set of independent variables is determined through a highly sophisticated process. We will show below that when applying a similar variance adjustment to the one discussed in Section 3.4.2, the Modified LISA (modLISA) for BART will exhibit superior computational and statistical efficiency compared to either LISA or CMC.

Just like in the regression model we "correct" the sampling algorithm by adjusting the residual variance. We start with the conditional distribution of tree j from

expression (4.1) which takes the form

$$p(T_j \mid R_j, \sigma) \quad \propto \quad p(T_j) \int p(R_j \mid M_j, T_j, \sigma)\; p(M_j \mid T_j, \sigma)\; dM_j.$$

Note that only the conditional distribution of the residuals, $R_j \mid M_j, T_j, \sigma$ is affected by the modifications brought by LISA. The Metropolis-Hastings acceptance ratios for tree proposals contain three parts: the transition ratio, the likelihood ratio and the tree structure ratio. The modifications brought by LISA will influence only the likelihood ratio which is constructed from the conditional distributions of residuals. Consider the likelihood ratio for GROW proposal in LISA (full details are presented in the Appendix)

$$\frac{P(R \mid T_*, \sigma^2)}{P(R \mid T, \sigma^2)} = \sqrt{\frac{\sigma^2(\sigma^2 + Kn_l\sigma_\mu^2)}{(\sigma^2 + Kn_{l_L}\sigma_\mu^2)(\sigma^2 + Kn_{l_R}\sigma_\mu^2)}} \;\times$$

$$\exp\left\{\frac{K^2\sigma_\mu^2}{2\sigma^2}\left[\frac{(\sum_{i=1}^{n_{l_L}} R_{l_L,i})^2}{\sigma^2 + Kn_{l_L}\sigma_\mu^2} + \frac{(\sum_{i=1}^{n_{l_R}} R_{l_R,i})^2}{\sigma^2 + Kn_{l_R}\sigma_\mu^2} - \frac{(\sum_{i=1}^{n_l} R_{l,i})^2}{\sigma^2 + Kn_l\sigma_\mu^2}\right]\right\} \quad (3.40)$$

where n_l is the total number of observations from batch-data that end up in terminal node l. The newly grown tree, T_*, splits terminal node l into two terminal nodes (children) l_L and l_R, which will also divide n_l to n_{l_L} and n_{l_R} which are the corresponding number of observations in each new terminal node. By factoring out K in (3.40), we

can rewrite it as

$$\frac{P(R \mid T_*, \sigma^2)}{P(R \mid T, \sigma^2)} = \sqrt{\frac{\frac{\sigma^2}{K}(\frac{\sigma^2}{K} + n_l\sigma_\mu^2)}{(\frac{\sigma^2}{K} + n_{l_L}\sigma_\mu^2)(\frac{\sigma^2}{K} + n_{l_R}\sigma_\mu^2)}} \times$$

$$\exp\left\{\frac{\sigma_\mu^2}{2\frac{\sigma^2}{K}}\left[\frac{(\sum_{i=1}^{n_{l_L}} R_{l_L,i})^2}{\frac{\sigma^2}{K} + n_{l_L}\sigma_\mu^2} + \frac{(\sum_{i=1}^{n_{l_R}} R_{l_R,i})^2}{\frac{\sigma^2}{K} + n_{l_R}\sigma_\mu^2} - \frac{(\sum_{i=1}^{n_l} R_{l,i})^2}{\frac{\sigma^2}{K} + n_l\sigma_\mu^2}\right]\right\}. \quad (3.41)$$

Expression (3.41) shows a similar residual variance that is K times smaller in each batch, and hence following the discussion in Section 3.4.2, to achieve similar variance, we need to modify LISA for BART by adding the intermediate step $\tilde{\sigma}^2 = K\sigma^2$ when updating *trees* in each batch, and then taking a weighted average combination of sub-samples (similar to Bayesian linear regression). As in Section 3.4.2, we don't apply any changes when updating σ^2. All our numerical experiments show that modLISA also generates accurate predictions in BART, since the modification corrects the bias in the posterior draws of σ^2 and properly calibrates the size of the trees.

The BART algorithm will split the covariate space into disjoint subsets and on each subset a regression with only an intercept is fitted. Therefore, as suggested by the discussion in 3.4.2 the weight assigned to each batch will be proportional to the inverse sample variance in that batch. In the following sections we examine the improvement brought by modLISA when compared to LISA and CMC.

3.6 Numerical Experiments

3.6.1 The Friedman's function

We have simulated data of size $N = 20,000$ from Friedman's test function (Friedman, 1991)

$$f(x) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5,$$

where the covariates $x = (x_1, \ldots, x_{10})$ are simulated independently from a $U(0,1)$ and $y \sim \mathcal{N}(f(x), \sigma^2)$ with $\sigma^2 = 9$. Note that five of the ten covariates are unrelated to the response variable. We have also generated test data containing 5000 cases. We apply BART to this simulated dataset using the default hyperparameters stated in Section 3.5 with $m = 50$ to generate posterior draws of (T, M, σ^2) that, in turn, yield posterior draws for $f(x)$ using the approximation $\hat{f}(x) \approx \sum_{j=1}^{m} g(x; \hat{T}_j, \hat{M}_j)$ for each $x = (x_1, \ldots, x_{10})$. Since in this case the true f is known, one can compute the root mean squared error (RMSE) using average posterior draws of $\hat{f}(x)$ for each x (i.e. $\overline{\hat{f}(x)}$), as an estimate to measure its performance, i.e. RMSE $= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(x_i) - \overline{\hat{f}(x_i)})^2}$. It is known that SingleMachine BART may mix poorly when it is run on an extremely large dataset with small residual variance. However since the data simulated is of reasonable size and σ is not very small the SingleMachine BART is expected to be a good benchmark for comparison (see discussion in Pratola, 2016).

Comparison of modLISA with Competing Methods

We have implemented modLISA, LISA, and CMC for BART with $K = 30$ batches on the simulated data for 5000 iterations (and 4000 burn-in) with a total of 1000 posterior draws. Table 3.1 shows results from all methods including the SingleMachine

which runs BART on the full dataset using only one machine. Results are averaged over three different realizations of train and test data, and include the Train and Test RMSE for each method, along with tree sizes (average maximum tree size from all machines), σ^2 estimates and their 95% Credible Intervals (CI). The summaries presented in Table 3.1 show that although LISA has better prediction performance than CMC, it does a terrible job at estimating σ^2, its estimate being orders of magnitude smaller than the one produced by CMC. CMC and LISA both generate larger trees compared to SingleMachine, with CMC generating trees that are ten times larger than LISA's. One can see that modLISA with weighted averages dominates both CMC and LISA across all performance indicators since it yields the smallest RMSE, the smallest tree size, and less biased σ^2 estimates. Generally, modLISA generates results that are by far the closest to the ones produced by SingleMachine.

Table 3.1: Comparing Train & Test RMSE, tree sizes, and average post burn-in $\hat{\sigma}^2$ with 95% CI in each method for $K = 30$ to SingleMachine BART (all results are averaged over three different realizations of data).

Method	TrainRMSE	TestRMSE	Tree Nodes	Avg $\hat{\sigma}^2$	95% CI for σ^2
CMC	2.73	2.94	602	1.91	[1.45 , 2.88]
$LISA\ (unif\ wgh)$	1.18	1.19	55	0.001	[0.0009 , 0.0011]
$modLISA\ (wgh\ avg)$	0.57	0.59	7	7.97	[7.87 , 8.08]
$SingleMachine$	0.55	0.56	7	9.04	[8.85 , 9.21]

Table 3.2: Average acceptance rates of tree proposal moves.

Method	GROW	PRUNE	CHANGE
CMC	21%	0.03%	34%
$LISA$	1.8%	0.5%	1.6%
$modLISA$	20%	26%	19%
$SingleMachine$	9%	10%	6%

The size of trees produced by each method is in sync with the average acceptance

rates of each tree proposal move shown in Table 3.2. It is noticeable the difference between CMC and LISA 's average acceptance rates between growing a tree and pruning one. However, the prior plays an important role in determining the tree sizes and since in CMC the prior is weakened by a power of $1/K$, this difference may become more apparent as observed in Table 3.2. On the other hand, modLISA has overall larger acceptance rates with the smallest relative absolute difference between growing and pruning probabilities compared to LISA and CMC ($6/26 = 23.1\%$ for modLISA, 99.9% for CMC, and 72.2% for LISA) and is closest to SingleMachine (10%). Overall, modLISA induced a significant reduction in tree sizes by preserving a balance between growing and pruning trees which also improves exploring the posterior distribution.

Table 3.3: Average coverage for 95% credible intervals constructed for training (TrainCredCov) and test (TestCredCov) data and 95% prediction intervals constructed for training (TrainPredCov) and test (TestPredCov) data. The prediction interval coverage is estimated based on 1000 iid samples, $N = 20,000$ and $K = 30$. All results are averaged over three different realizations of data.

Method	TrainPredCov	TestPredCov	TrainCredCov	TestCredCov
CMC	45.71 %	47.83 %	81.95 %	99.99 %
$LISA\ (unif\ wgh)$	1.54 %	1.54 %	100 %	100 %
$modLISA\ (wgh\ avg)$	92.93 %	92.91 %	60.88 %	58.45 %
$SingleMachine$	94.67 %	94.65 %	71.58 %	71.54 %

For a more clear comparison of the methods, Table 3.3 shows the average coverage of 95% credible intervals (CI) for predictors $f(x)$ and 95% prediction intervals (PI) for future responses y. The calculations are made for the values of y and $f(x)$ in the training and test data sets.

The coverage for CI is given by the averaging for all training or test data of

$$\frac{\#\{f(x_i) \in \hat{I}_{f(x_i)} : \ 1 \leq i \leq N\}}{N}$$

where $\hat{I}_f(x_i)$ is the CI for $f(x_i)$ estimated based on the MCMC draws from π.

The coverage of the PI corresponding to a pair $(y_i, f(x_i))$ is given by the proportion of 1000 iid samples generated from the true generative model $N(f(x_i), \sigma^2)$ that fall between its limits, i.e. the average over training or test data of

$$\frac{\#\{\tilde{y}_j \in \hat{J}_{y_i} : \ \tilde{y}_j \overset{iid}{\sim} N(f(x_i), \sigma^2) 1 \leq j \leq 1000\}}{1000},$$

where $\hat{J}_{y_i}$ is the PI for y_i. The PI coverage in modLISA and SingleMachine are very close to nominal and vastly outperform the PI's produced using LISA or CMC.

One can see that coverages of the CI built via CMC and LISA are high, which is not surprising since both algorithms produce over-dispersed approximations to the conditional distributions of $f(x)$. Our observation is that the CI for LISA and CMC are too wide to be practically useful. Also, modLISA and SingleMachine have much lower CI coverage than nominal which is also expected due to the systematic bias induced by the discrepancy between the functional forms of the true predictor (continuous) and of the one fitted by BART (piecewise constant). Thus, the CI for $f(x)$ will exhibit poor coverage as they are centered around a biased estimate of $f(x)$.

In order to verify that this is indeed the case we have generated a dataset of size 20,000 from the piecewise constant function:

$$f(x) = \mathbf{1}_{[0,0.2)}(x_1) + 2 \cdot \mathbf{1}_{[0.2,0.4)}(x_1) + 3 \cdot \mathbf{1}_{[0.4,0.6)}(x_1) + 4 \cdot \mathbf{1}_{[0.6,0.8)}(x_1) + 5 \cdot \mathbf{1}_{[0.8,1)}(x_1)$$

where $\mathbf{1}_{[a,b)}(x) = 1$ if $x \in [a, b)$ and 0 otherwise, $x = (x_1, \ldots, x_{10}) \in (0, 1)^{10}$ is a ten-dimensional input vector, with $x_i \sim$ Uniform$(0, 1)$, and $y \sim \mathcal{N}(f(x), 9)$. Additional 5000 data have also been simulated as test cases. Table 3.4 summarizes the analysis with $K = 30$ and confirms a sharp decrease in RMSEs even though the noise has the

same variance $\sigma^2 = 9$. We note that the coverages of CI build under modLISA and SingleMachine are much higher than with Friedman's function.

Table 3.4: Comparing test data RMSE and coverage of 95% credible intervals for piecewise $f(x)$ with $N = 20,000$ and $K = 30$.

Method	TestRMSE	TestCredCov
CMC	1.35	100 %
$LISA\ (unif\ wgh)$	0.94	100 %
$modLISA\ (wgh\ avg)$	0.24	90.16 %
$SingleMachine$	0.15	98.76 %

Comparison with SingleMachine BART

In order to investigate the closeness of posterior samples in each method to the SingleMachine BART, we have plotted in Figure 3.2 the empirical distribution functions of $\hat{f}(x)$ generated from each algorithm for two pairs of observations in the training and test dataset. One can see that the empirical distribution functions in LISA and CMC don't match the ones from SingleMachine, and look over-dispersed. However, the empirical distribution functions in modLISA weighted average look much closer to SingleMachine with a slight shift in location.

In order to assess the performance of the sampling procedures considered, we use the Cramér-von Mises distance to assess the difference between empirical distribution functions. This distance is defined to be $\omega^2 = \int_{-\infty}^{\infty}(F_n(x) - F(x))^2 dF(x)$ where in our case we assume $F(x) = F_{BART}(x)$ to be the empirical distribution function generated from posterior samples in SingleMachine BART and $F_n(x)$ is similarly computed for the alternative method that is considered for comparison.

Using a set of $T = 1000$ equispaced points, we compute the average squared difference between the single machine and all other alternative methods for each

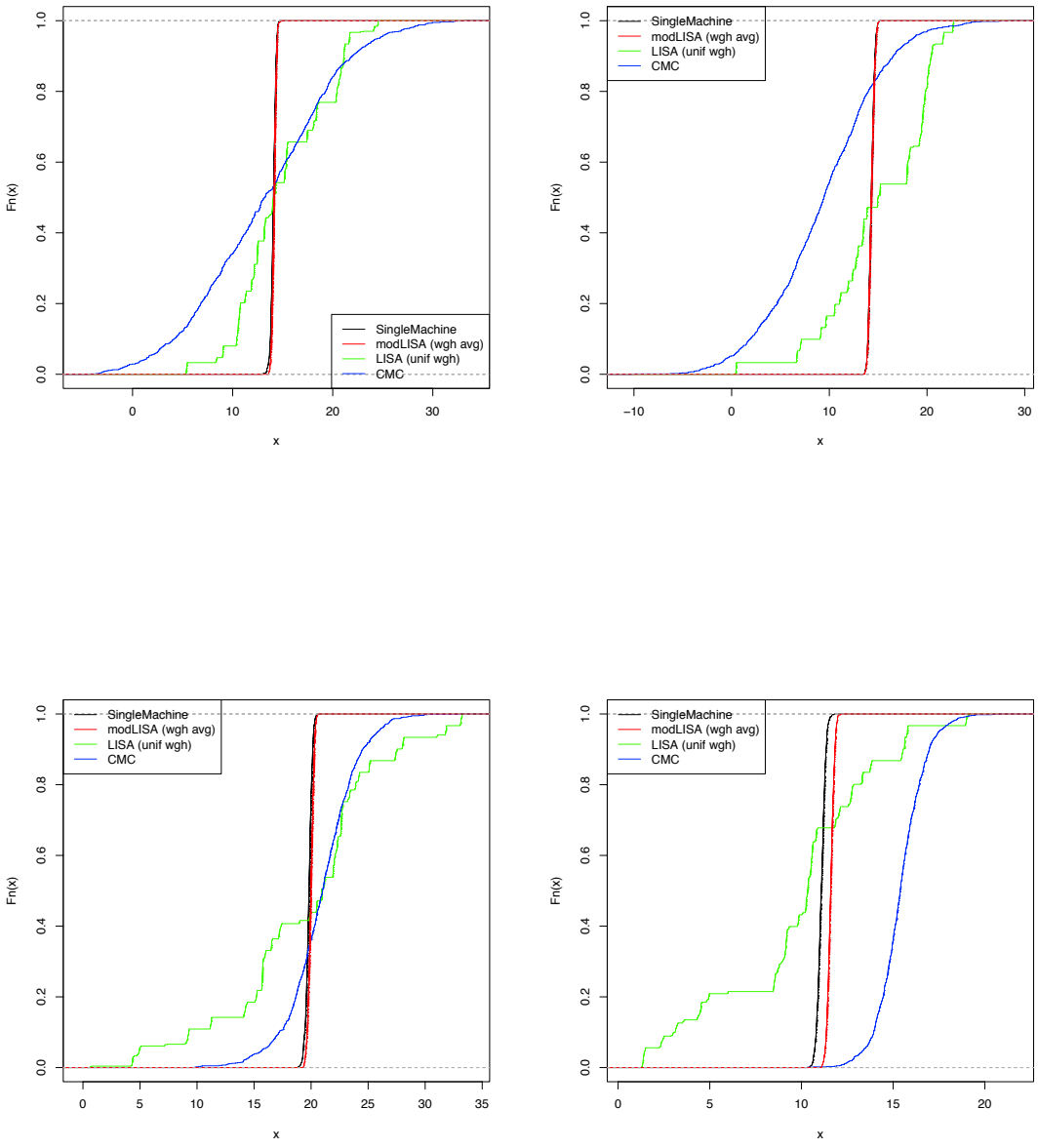


Figure 3.2: Empirical distribution functions of $\hat{f}(x)$ obtained from MCMC samples produced by modLISA (red line), LISA (green line), CMC (blue line), and SingleMachine BART (black line) for two different pairs of training and test data. In this example $K = 30$. Top left: Test $x^* = 2000$, $f(x^*) = 14.4$. Top right: Test $x^* = 2000$, $f(x^*) = 14.4$. Bottom left: Training $x = 999$, $f(x) = 19.8$. Bottom right: Training $x = 2001$, $f(x) = 11.2$.

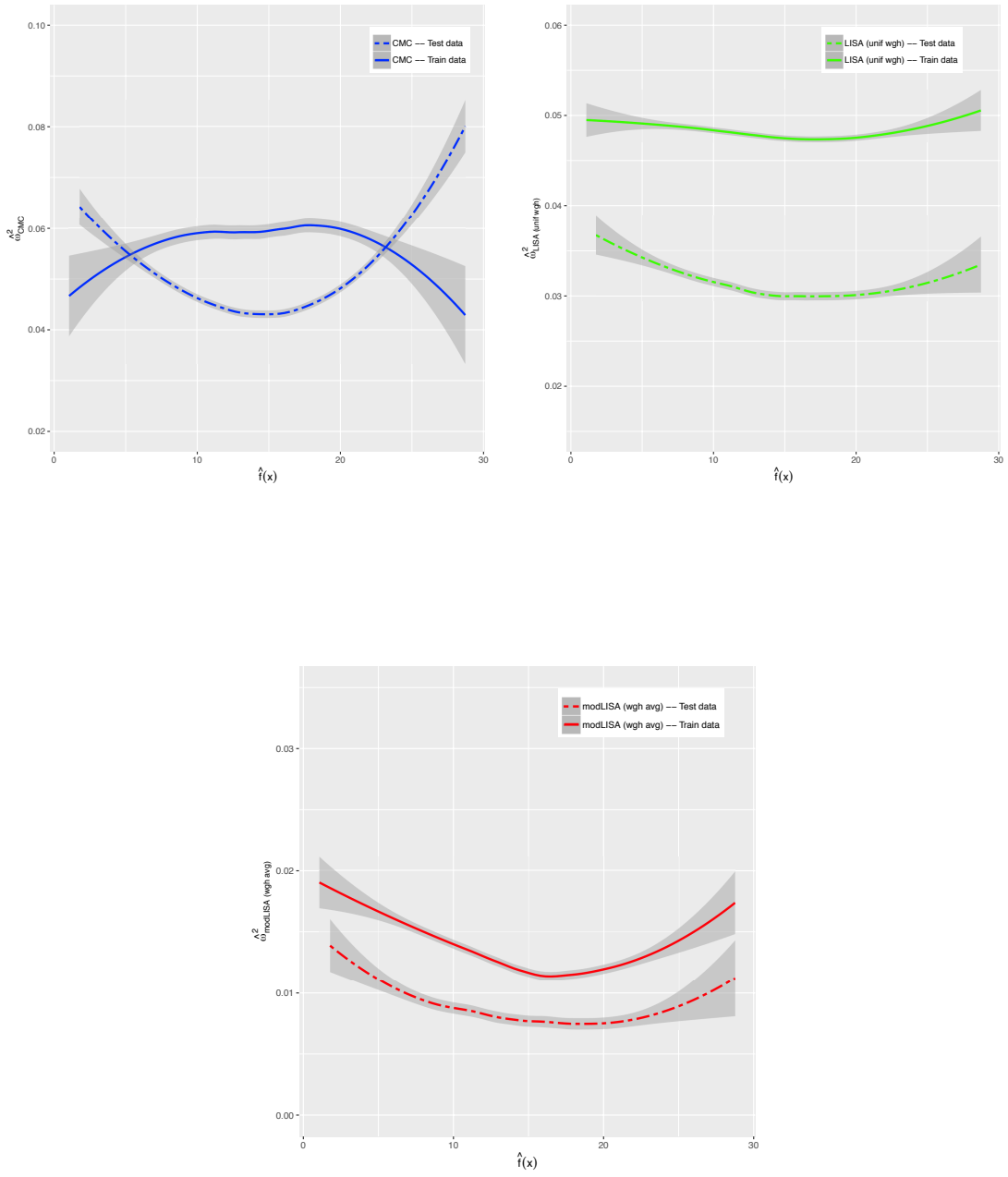


Figure 3.3: Fitted polynomial trends (for both train and test data) of average squared difference between empirical distribution functions of SingleMachine and the following: (a) CMC for training (blue solid line) and test (blue dot dashed line) data (top left panel), (b) LISA with uniform weights for training (green solid line) and test (green dot dashed line) data (top right panel) and (c) modLISA with weighted average for training (red solid line) and test (red dot dashed line) data (bottom panel). The difference is plotted against the mean prediction $\hat{f}(x)$ produced by SingleMachine. Grey areas represent the 95% credible intervals constructed from 100 independent replicates.

observation in the dataset. To illustrate, for LISA we estimate ω using $\hat{\omega}^2_{LISA} = \frac{1}{T}\sum_{j=1}^{T}(F_{LISA}(t_j) - F_{BART}(t_j))^2$.

Figure 3.3 is comparing the fitted polynomial trends of $\hat{\omega}^2$ (in each method) versus mean predicted $\hat{f}(x)$ in SingleMachine with their corresponding 95% credible regions (for both train and test data). Clearly in LISA and modLISA, there are small variations around the trends with not much changes seen in values of $\hat{\omega}^2$ among different mean predicted $\hat{f}(x)$, which specifies consistency within different train or test observations. In addition, the gap between trends from train and test data indicate that the average distance between LISA/modLISA and SingleMachine's distributions are smaller for test data compared to train data. Furthermore, there are still small variations seen around CMC's trends, but with slight changes in values of $\hat{\omega}^2$ among different mean predicted $\hat{f}(x)$, especially for the test dataset which indicates inconsistency within different observations.

To emphasize the difference in performance between modLISA and its competitors, Figure 3.4 shows all the fitted polynomial trends without their credible regions for the train and test data. One can see that there is a large gap between $\hat{\omega}^2$ values in modLISA weighted average and other alternative methods (for both train and test data), with modLISA having the lowest value. Thus the weighted average of samples produced by modLISA yields the closest results to SingleMachine. This can also be justified by comparing average $\hat{\omega}^2$ over all train observations for each trend which is calculated to be 0.013 for modLISA that is significantly smaller than 0.059, 0.048 for CMC, and LISA, respectively. Similarly, the average $\hat{\omega}^2$ over test data are 0.008, 0.047, and 0.031 for modLISA, CMC, and LISA respectively, which again the smallest value is seen in modLISA. We conclude that modLISA weighted average sample yields the closest representation of the BART posterior and exhibits the best performance compared to alternative methods.

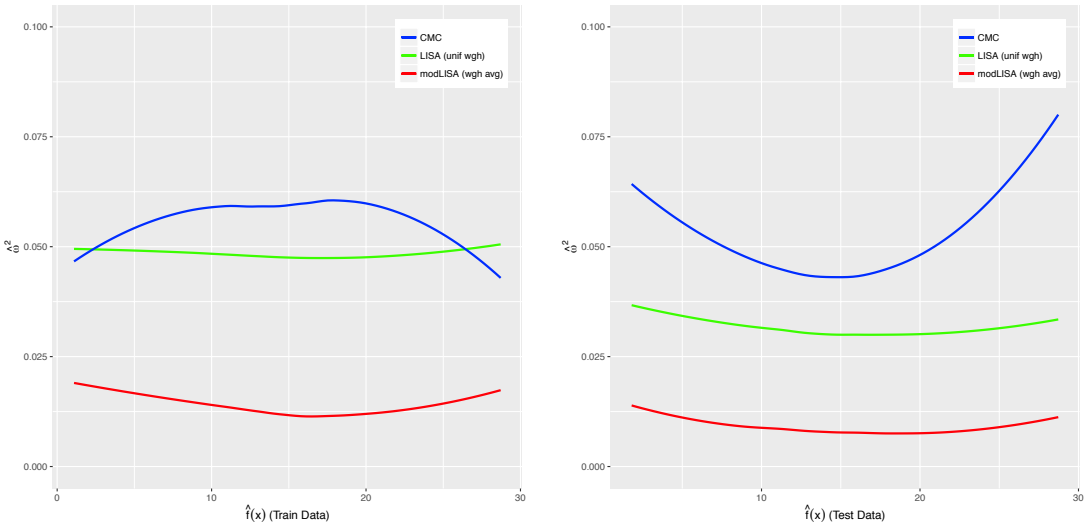


Figure 3.4: Comparing fitted polynomial trends of average squared difference in empirical distribution functions of each method and SingleMachine, as functions of mean predicted $\hat{f}(x)$ in SingleMachine for train (left panel) and test data (right panel).

At last we compare run time per iteration for each method so we can draw some conclusions regarding the overall efficiency.

Run Time Comparisons

The main goal of methods such as LISA and CMC was to reduce run times regarding big data applications. Here we have compared average run times per iteration (from one processor) for each method using our implementation of BART. All computations were conducted on a cluster of Intel® Xeon® CPUs (@ 3.20GHz).

Table 3.5: Running times for CMC, LISA, modLISA and SingleMachine when $K = 30$.

Method	Avg Time per iteration (Secs)	Speed-up
CMC	11.99	31%
$LISA$	5.04	71%
$modLISA$	1.81	90%
$SingleMachine$	17.28	—

As it is seen in Table 3.5, modLISA, LISA and CMC with $K = 30$ are all faster compared to SingleMachine since they are influenced by the smaller subsets of data used. However, since LISA and CMC generate much larger trees, they become slower compared to modLISA which is the fastest method. We have also reported the speed-up percentages with respect to SingleMachine, which is defined to be $(1 - t/17.28) \times 100\%$ where t is the average time per iteration in each method. Clearly, CMC shows the smallest speed-up (31%) while modLISA has the highest (90%).

3.6.2 Additional Considerations

Effect of N (number of training data) on Posterior Accuracy

To see how the number of training data (N) can effect the posterior accuracy, we have examined the performance of all methods when N is increased to 60,000 while we keep the same number of batches $K = 30$. Tables 3.6 shows the results of 1000 posterior samples generated from fitting the BART model to the training set with additional 5000 data considered as test cases.

Table 3.6: Tree sizes, estimates and 95% credible intervals for σ^2, RMSE for training data (TrainRMSE) of size $N = 60,000$ and for test data (TestRMSE) of size $5,000$ for each method run with $K = 30$.

Method	TrainRMSE	TestRMSE	Tree Nodes	Avg $\hat{\sigma}^2$	95% CI for σ^2
CMC	2.85	5.56	983	0.48	[0.30 , 0.66]
LISA (*unif wgh*)	1.17	1.19	125	0.0003	[0.00031 , 0.00035]
modLISA (*wgh avg*)	0.41	0.42	7	8.82	[8.79 , 8.86]
SingleMachine	0.41	0.41	11	9.04	[8.94 , 9.16]

Table 3.7: Average coverage for 95% credible intervals constructed for training (TrainCredCov) and test (TestCredCov) data and 95% prediction intervals constructed for training (TrainPredCov) and test (TestPredCov) data. The prediction interval coverage is estimated based on 1000 iid samples, $N = 60,000$ and $K = 30$.

Method	TrainPredCov	TestPredCov	TrainCredCov	TestCredCov
CMC	25.74 %	17.28 %	51.37 %	85.92 %
LISA (*unif wgh*)	0.84 %	0.84 %	100 %	100 %
modLISA (*wgh avg*)	94.54 %	94.53 %	53.68 %	52.68 %
SingleMachine	94.83 %	94.84 %	57.79 %	58.90 %

Unsurprisingly, Tables 3.1 and 3.6 show that the RMSE for training and test data in LISA, modLISA, and SingleMachine decrease as N increases. More importantly, while LISA and CMC estimates for σ^2 get worse, modLISA generates more accurate estimates of σ^2 with a larger N.

Trees have a stable size in modLISA, but tend to grow larger in CMC and LISA, as N increases. Note that tree size has remained unchanged for modLISA while for SingleMachine it has slightly increased; this is reasonable for two possible reasons, one that the tree sizes are averaged over all machines and hence error can be present and other is that modLISA is running on a much smaller dataset compared to SingleMachine which indicates better mixing of the chain and hence more balanced trees. Table 3.7 shows that coverage of PI decreases in CMC and LISA, but increases in modLISA and SingleMachine for larger training data. We find it particularly promising that modLISA competes with SingleMachine for larger N. Note that, coverage of CI in LISA and CMC are still unreliable because of their over-dispersion, while in modLISA and SingleMachine they decrease as N increases, which is reasonable since larger sample size creates narrow CI that are around a biased $f(x)$ estimate, as discussed in the previous section. Overall, as N increases, modLISA seems to be a more reliable method as it shows a better performance compared to all other alternatives.

Effect of K (number of batches) on Posterior Accuracy

To examine the effect of K on posterior accuracy, we have generated 1000 posterior draws for training data of size $N = 20,000$ and $K = 10$. The test data sample is of size 5,000. The results are shown in Table 3.8.

Table 3.8: Tree sizes, estimates and 95% credible intervals for σ^2, RMSE for training data (TrainRMSE) of size $N = 20,000$ and for test data (TestRMSE) of size $5,000$ for each method run with $K = 10$.

Method	TrainRMSE	TestRMSE	Tree Nodes	Avg $\hat{\sigma}^2$	95% CI for σ^2
CMC	2.92	3.18	951	0.73	[0.57 , 0.90]
LISA (unif wgh)	1.70	1.78	131	0.001	[0.0010 , 0.0012]
modLISA (wgh avg)	0.46	0.47	7	8.69	[8.61 , 8.77]
SingleMachine	0.55	0.56	7	9.04	[8.85 , 9.21]

Table 3.9: Average coverage for 95% credible intervals constructed for training (TrainCredCov) and test (TestCredCov) data and 95% prediction intervals constructed for training (TrainPredCov) and test (TestPredCov) data. The prediction interval coverage is estimated based on 1000 iid samples, $N = 20,000$ and $K = 10$.

Method	TrainPredCov	TestPredCov	TrainCredCov	TestCredCov
CMC	31.08 %	29.83 %	48.18 %	99.80 %
$LISA\ (unif\ wgh)$	1.44 %	1.43 %	99.98 %	99.96 %
$modLISA\ (wgh\ avg)$	94.30 %	94.29 %	71.08 %	70.32 %
$SingleMachine$	94.67 %	94.65 %	71.58 %	71.54 %

As K decreases, the performance of LISA and CMC drops while modLISA generates stronger results, which is intuitively expected as each batch is larger and closer to the full sample when K is smaller. We also note the improvement of modLISA over SingleMachine in terms of RMSE. In addition, Table 3.9 shows that the PI and CI coverages for modLISA and SingleMachine are very close.

3.6.3 Varying the Underlying Model – Different $f(x)$

Consistency in performance of modLISA can also be seen when the underlying model is changed. For instance, we also considered a sample of size 20,000 using

$$f(x) = 3\sqrt{x_1} - 2{x_2}^2 + 5x_3x_4, \tag{3.42}$$

where $x = (x_1, \ldots, x_4)$ is a four-dimensional input vector that is simulated independently from a $U(0, 1)$ and $y \sim \mathcal{N}(f(x), \sigma^2)$ with $\sigma^2 = 1$. Additional 5000 data have also been simulated as test cases. Similarly, by fitting this newly simulated dataset to each method with $K = 30$, we have generated 1000 posterior samples with results averaged across three different realizations of data shown in Tables 3.10 and 3.11.

Again modLISA outperforms all alternative methods, and its performance is closest to SingleMachine. This confirms the previous simulation results and allows us

Table 3.10: Tree sizes, estimates and 95% credible intervals for σ^2, RMSE for training data (TrainRMSE) of size $N = 20,000$ generated from (3.42) and for test data (TestRMSE) of size $5,000$ for each method run with $K = 30$. Results are averaged over three different data replications.

Method	TrainRMSE	TestRMSE	Tree Nodes	Avg $\hat{\sigma}^2$	95% CI for σ^2
CMC	0.89	0.76	614	0.21	[0.18 , 0.34]
LISA (*unif wgh*)	0.32	0.33	57	0.0001	[0.000083 , 0.000103]
modLISA (*wgh avg*)	0.11	0.11	7	0.88	[0.87 , 0.89]
SingleMachine	0.14	0.14	7	1.00	[0.99 , 1.03]

Table 3.11: Average coverage for 95% credible intervals constructed for training (TrainCredCov) and test (TestCredCov) data and 95% prediction intervals constructed for training (TrainPredCov) and test (TestPredCov) data generated from (3.42). The prediction interval coverage is estimated based on 1000 iid samples, $N = 20,000$ and $K = 30$. Results are averaged over three different data replications.

Method	TrainPredCov	TestPredCov	TrainCredCov	TestCredCov
CMC	49.74 %	52.97 %	84.16 %	100 %
LISA (*unif wgh*)	1.50 %	1.49 %	100 %	100 %
modLISA (*wgh avg*)	93.07 %	93.18 %	82.88 %	83.50 %
SingleMachine	94.82 %	94.81 %	79.13 %	78.47 %

to conclude that modLISA is a more reliable method for BART models with large datasets.

In the next section we will apply modLISA weighted average BART to a large socio-economic study.

3.6.4 Real Data Analysis

The American Community Survey (ACS) is a growing survey from the US Census Bureau and the Public Use Microdata Sample (PUMS) is a sample of responses to ACS which consists of various variables related to people and housing units (see US Bureau of Census, 2013). Considering the person-level data from PUMS 2013, we would like to predict a person's total income based on variables such as sex, age, education, class

of worker, living state, and citizenship status. We have collected information related to people who are employed and have total income of at least $5000 with education level of either Bachelor's degree, Master's degree, or a PhD which resulted in $437,297$ observations. We randomly divided the dataset into approximately 80% training and 20% testing sets, with $K = 100$ batches considered for splitting the training data to apply modLISA. Computations were performed on the GPC supercomputer at the SciNet HPC Consortium (Loken et al., 2010) using 100 cores, each running on $3,500$ observations. Considering the logarithm of total income for each person as the response variable, we ran modLISA with weighted average and SingleMachine BART on this dataset for 1500 iterations (since SingleMachine is very slow) and discarded the first 1000 draws which resulted in 500 posterior samples. Table 3.12 contains the results of Test RMSE as well as average post burn-in σ^2 estimates and tree sizes.

Table 3.12: Perfomance summaries computed from 1000 posterior samples generated from modLISA with $K = 100$ and SingleMachine BART on PUMS 2013 test data.

Method	TestRMSE	Avg $\hat{\sigma}^2$	Tree Nodes	Speed-up
modLISA (wgh avg)	0.71	0.488	7	90%
SingleMachine	0.70	0.485	23	–

Table 3.13: Average acceptance rates of tree proposal moves.

Method	GROW	PRUNE	CHANGE
modLISA	10 %	11 %	14 %
SingleMachine	8%	7%	7%

One can see that Test RMSE in modLISA is similar to the one from SingleMachine, but with a 90% speed-up of modLISA over SingleMachine. The speed-up can be explained by the larger acceptance probabilities and by the smaller tree sizes

reported in Tables 3.13 and 3.12, respectively. The 90% speedup is important for applications like the one considered here, as it takes more than a day to simulate $1,500$ samples from the posterior using SingleMachine. The result indicate the potential of the proposed method for reducing computational costs while producing accurate predictions.

3.7 Discussion

The challenge of using MCMC algorithms to sample posterior distributions obtained from a massive sample of observations is a serious one.

In this chapter, we introduced a new method based on the idea of randomly dividing the data into batches and drawing samples from each of the resulting sub-posteriors independently and in parallel on different machines. We proposed a novel way to define the sub-posteriors and theoretically justified our reasoning behind the chosen sub-posteriors. We showed for a Bernoulli distributed data, LISA with uniform weights outperforms its competing method, CMC. However, we developed a different strategy to combine the samples produced by each batch analysis for the important class of Bayesian Additive Regression Trees (BART) models. We examined LISA on BART with both simulated and real datasets. Our proposed methodology for BART generates approximate exact samples as the full posterior samples and outperforms CMC with reduction in computation time that are as high as 90%.

In future work one can find a procedure for combining the sub-posterior samples that will make LISA easy to adapt to a wide variety of models.

3.8 Derivations of Metropolis-Hastings (MH) Acceptance Ratios

3.8.1 MH Acceptance ratios for BART

In this section we will use a similar explanation and notation given by Kapelner and Bleich (2013) to derive the acceptance ratios of the Metropolis-Hastings step in updating trees of BART. We will further extend these calculations for LISA and CMC.

The Metropolis-Hastings algorithm is used to draw samples from conditional distribution given in equation (14)

$$p(T \mid R, \sigma) \quad \propto \quad p(T) \int p(R \mid M, T, \sigma)\ p(M \mid T, \sigma)\ dM$$

Assume we propose T_*, then the acceptance ratio will be:

$$r \ = \ \underbrace{\frac{P(T_* \to T)}{P(T \to T_*)}}_{\text{transition ratio}} \ \times \ \underbrace{\frac{P(R \mid T_*, \sigma^2)}{P(R \mid T, \sigma^2)}}_{\text{likelihood ratio}} \ \times \ \underbrace{\frac{P(T_*)}{P(T)}}_{\text{tree structure ratio}}$$

We will calculate r for each possible proposal:

GROW Proposal:

- **Transition ratio:** Consider growing one of the b terminal nodes of tree T, say

node η, to two children nodes. Then we will have:

$$\begin{aligned} P(T \to T_*) &= P(GROW)\ P(\text{choosing } \eta)\ P(\text{choosing a predictor to split on}) \times \\ &\quad P(\text{choosing a splitting value}) \\ &= P(GROW)\frac{1}{b}\frac{1}{p(\eta)}\frac{1}{n_p(\eta)} \end{aligned}$$

where $p(\eta)$ denotes the number of predictors left available to split on at node η (there must be at least two unique values in each predictor to consider), and $n_p(\eta)$ denotes the number of unique splitting values left in the chosen pth attribute.

In addition, we have:

$$P(T_* \to T) = P(PRUNE)\ P(\text{choosing } \eta \text{ to prune}) = P(PRUNE)\frac{1}{w_*}$$

where w_* is the number of nodes with two terminal nodes in the new tree T_*. Hence the transition ratio will be:

$$\frac{P(T_* \to T)}{P(T \to T_*)} = \frac{P(PRUNE)}{P(GROW)}\frac{b\ p(\eta)\ n_p(\eta)}{w_*}$$

- **Likelihood ratio:** For computing the likelihood ratio, we have:

$$P(R_1, ..., R_n \mid T, \sigma^2) = \prod_{l=1}^{b} P(R_{l_1}, ..., R_{l_{n_l}} \mid \sigma^2)$$

since the data are partitioned across all b terminal nodes of tree T. R_{l_j} denotes the j-th data (residual) in the l-th terminal node and n_l is the number of observations in the l-th terminal node. From BART we know that $\mu_l \sim N(0, \sigma_\mu^2)$,

hence we will have:

$$P(R_{l_1}, ..., R_{l_{n_l}} \mid \sigma^2) = \int_{\mathbb{R}} P(R_{l_1}, ..., R_{l_{n_l}} \mid \mu_l, \sigma^2)\ P(\mu_l; \sigma_\mu^2)\ d\mu_l.$$

By completion of the square this will equal to:

$$P(R_{l_1}, ..., R_{l_{n_l}} \mid \sigma^2) =$$

$$\frac{1}{(2\pi\sigma^2)^{n_l/2}} \sqrt{\frac{\sigma^2}{\sigma^2 + n_l\sigma_\mu^2}}\ \exp\Big(-\frac{1}{2\sigma^2}\Big[\sum_{i=1}^{n_l} (R_{l_i} - \bar{R}_l)^2\ -\ \frac{{\bar{R}_l}^2 {n_l}^2}{n_l + \frac{\sigma^2}{\sigma_\mu^2}}\ +\ n_l {\bar{R}_l}^2 \Big]\Big), \tag{3.43}$$

where $\bar{R}_l$ is the average residual at terminal node l. Note that the likelihood is specified by all terminal nodes, and since T differs from T_* only at its l-th terminal node which splits into two terminal children l_L and l_R, the probability terms from other terminal nodes will be canceled in the likelihood ratio which results in (using (3.43)):

$$\frac{P(R \mid T_*, \sigma^2)}{P(R \mid T, \sigma^2)} = \sqrt{\frac{\sigma^2(\sigma^2 + n_l\sigma_\mu^2)}{(\sigma^2 + n_{l_L}\sigma_\mu^2)(\sigma^2 + n_{l_R}\sigma_\mu^2)}}\ \times$$

$$\exp\Big(\frac{\sigma_\mu^2}{2\sigma^2}\Big[\frac{(\sum_{i=1}^{n_{l_L}} R_{l_L,i})^2}{\sigma^2 + n_{l_L}\sigma_\mu^2} + \frac{(\sum_{i=1}^{n_{l_R}} R_{l_R,i})^2}{\sigma^2 + n_{l_R}\sigma_\mu^2} - \frac{(\sum_{i=1}^{n_l} R_{l,i})^2}{\sigma^2 + n_l\sigma_\mu^2}\Big]\Big), \tag{3.44}$$

where R_{l_L} and R_{l_R} are residuals in the left and right child (respectively) with corresponding number of observations n_{l_L} and n_{l_R}.

- **Tree Structure ratio:** Recall the descriptions given in BART related to the

probability that node η at depth d_η is non-terminal:

$$P_{\text{Split}}(\eta) = \frac{\alpha}{(1+d_\eta)^\beta}$$

with probability of assigning a rule given as:

$$P_{\text{Rule}}(\eta) = \frac{1}{p(\eta)}\frac{1}{n_p(\eta)}$$

Hence, the prior on each tree will be:

$$P(T) = \prod_{\eta \in \text{ non-terminal nodes}} P_{\text{Split}}(\eta)\ P_{\text{Rule}}(\eta) \quad \times \prod_{\eta \in \text{ terminal nodes}} (1 - P_{\text{Split}}(\eta))$$

which will result in the following tree structure ratio:

$$\frac{P(T_*)}{P(T)} = \alpha \frac{(1-\frac{\alpha}{(2+d_\eta)^\beta})^2}{((1+d_\eta)^\beta - \alpha)\ p(\eta)\ n_p(\eta)}. \tag{3.45}$$

PRUNE Proposal:

- **Transition ratio:** A similar description as in the GROW step will lead to:

$$\frac{P(T_* \to T)}{P(T \to T_*)} = \frac{P(GROW)}{P(PRUNE)}\frac{w}{(b-1)\ p(\eta^*)\ n_p(\eta^*)}$$

  where w is the number of nodes with two terminal nodes in tree T. Note that tree T_* has one less terminal nodes $(b-1)$.

- **Likelihood ratio:** This is the inverse of the likelihood ratio in the GROW proposal.

- **Tree Structure ratio:** This is also the inverse of the tree structure in the GROW proposal.

CHANGE Proposal:

- **Transition ratio:** As described by Kapelner and Bleich (2013), for simplicity, we will only change the rule assignments for nodes with two terminal children. Hence:

$$P(T \to T_*) = P(CHANGE)\ P(\text{choosing } \eta)\ P(\text{choosing a predictor to split on}) \times P(\text{choosing a splitting value})$$

with the first three terms canceling in the transition ratio given as:

$$\frac{P(T_* \to T)}{P(T \to T_*)} = \frac{n_{p^*}(\eta^*)}{n_p(\eta)}.$$

- **Likelihood ratio:** T_* differs from T only from the two terminal children effected by the changed rules from their parents. Hence, by canceling the probabilities from other terminal nodes, we will achieve the likelihood ratio:

$$\frac{P(R \mid T_*, \sigma^2)}{P(R \mid T, \sigma^2)} = \sqrt{\frac{(\frac{\sigma^2}{\sigma_\mu^2} + n_1)(\frac{\sigma^2}{\sigma_\mu^2} + n_2)}{(\frac{\sigma^2}{\sigma_\mu^2} + n_1^*)(\frac{\sigma^2}{\sigma_\mu^2} + n_2^*)}} \times$$

$$\exp\Big(\frac{1}{2\sigma^2}\Big[\frac{(\sum_{i=1}^{n_{1^*}} R_{1^*,i})^2}{\frac{\sigma^2}{\sigma_\mu^2} + n_1^*} + \frac{(\sum_{i=1}^{n_{2^*}} R_{2^*,i})^2}{\frac{\sigma^2}{\sigma_\mu^2} + n_2^*} - \frac{(\sum_{i=1}^{n_1} R_{1,i})^2}{\frac{\sigma^2}{\sigma_\mu^2} + n_1} - \frac{(\sum_{i=1}^{n_2} R_{2,i})^2}{\frac{\sigma^2}{\sigma_\mu^2} + n_2}\Big]\Big), \tag{3.46}$$

where subscripts 1 and 2 denote the two terminal children, while the asterisk

refers to the proposed tree T_*.

- **Tree Structure ratio:** Following the definition of $P(T)$, we will have:

$$\frac{P(T_*)}{P(T)} = \frac{n_p(\eta)}{n_{p^*}(\eta^*)}.$$

Note that:

$$\frac{P(T_* \to T)}{P(T \to T_*)} \times \frac{P(T_*)}{P(T)} = 1.$$

3.8.2 MH Acceptance ratios of LISA for BART

GROW Proposal:

- **Transition ratio:** No change.

- **Likelihood ratio:** Equation (3.43) changes to:

$$P(R_{l_1}, ..., R_{l_{n_l}} \mid \sigma^2) =$$

$$\frac{1}{(2\pi\sigma^2)^{n_l/2}} \sqrt{\frac{\sigma^2}{\sigma^2 + K n_l \sigma_\mu^2}} \exp\Big(-\frac{K}{2\sigma^2}\Big[\sum_{i=1}^{n_l}(R_{l_i} - \bar{R}_l)^2 - \frac{K \bar{R_l}^2 {n_l}^2}{K n_l + \frac{\sigma^2}{\sigma_\mu^2}} + n_l \bar{R_l}^2\Big]\Big). \tag{3.47}$$

Thus the likelihood ratio will change to:

$$\frac{P(R \mid T_*, \sigma^2)}{P(R \mid T, \sigma^2)} = \sqrt{\frac{\sigma^2(\sigma^2 + Kn_l\sigma_\mu^2)}{(\sigma^2 + Kn_{l_L}\sigma_\mu^2)(\sigma^2 + Kn_{l_R}\sigma_\mu^2)}} \times$$

$$\exp\Big(\frac{K^2\sigma_\mu^2}{2\sigma^2}\Big[\frac{(\sum_{i=1}^{n_{l_L}} R_{l_L,i})^2}{\sigma^2 + Kn_{l_L}\sigma_\mu^2} + \frac{(\sum_{i=1}^{n_{l_R}} R_{l_R,i})^2}{\sigma^2 + Kn_{l_R}\sigma_\mu^2} - \frac{(\sum_{i=1}^{n_l} R_{l,i})^2}{\sigma^2 + Kn_l\sigma_\mu^2}\Big]\Big). \quad (3.48)$$

- **Tree Structure ratio:** No change.

PRUNE Proposal:

- **Transition ratio:** No change.
- **Likelihood ratio:** This is the inverse of the likelihood ratio in the GROW proposal.
- **Tree Structure ratio:** No change.

CHANGE Proposal:

- **Transition ratio:** No change.
- **Likelihood ratio:**

$$\frac{P(R \mid T_*, \sigma^2)}{P(R \mid T, \sigma^2)} = \sqrt{\frac{(\frac{\sigma^2}{\sigma_\mu^2} + Kn_1)(\frac{\sigma^2}{\sigma_\mu^2} + Kn_2)}{(\frac{\sigma^2}{\sigma_\mu^2} + Kn_1^*)(\frac{\sigma^2}{\sigma_\mu^2} + Kn_2^*)}} \times$$

$$\exp\Big(\frac{K^2}{2\sigma^2}\Big[\frac{(\sum_{i=1}^{n_{1^*}} R_{1^*,i})^2}{\frac{\sigma^2}{\sigma_\mu^2} + Kn_1^*} + \frac{(\sum_{i=1}^{n_{2^*}} R_{2^*,i})^2}{\frac{\sigma^2}{\sigma_\mu^2} + Kn_2^*} - \frac{(\sum_{i=1}^{n_1} R_{1,i})^2}{\frac{\sigma^2}{\sigma_\mu^2} + Kn_1} - \frac{(\sum_{i=1}^{n_2} R_{2,i})^2}{\frac{\sigma^2}{\sigma_\mu^2} + Kn_2}\Big]\Big). \quad (3.49)$$

- **Tree Structure ratio:** No change.

The conditional posterior of σ^2 and M_j changes to:

- $\sigma^2 \mid (T_1, M_1), ..., (T_m, M_m), Y, X \quad \propto Inv - Gamma(\rho, \gamma)$

  where $\rho = \frac{\nu + Kn}{2}$ and $\gamma = \frac{1}{2}\ [\ K \sum_{i=1}^{n} (y_i - \sum_{j=1}^{m} g(x_i; M_j, T_j))^2 + \lambda\nu\ ]$.

- For the conditional posterior $M_j \mid T_j, R_j, \sigma$, we have:

$$\mu_{ij} \mid T_j, R_j, \sigma \ \sim \ \mathcal{N}\left(\frac{\frac{\sigma^2}{\sigma_\mu^2}\ \mu_\mu \ + \ K n_i \bar{R}_{j(i)}}{\frac{\sigma^2}{\sigma_\mu^2} \ + \ K n_i}, \frac{\sigma^2}{\frac{\sigma^2}{\sigma_\mu^2} \ + \ K n_i} \right),$$

  where $\bar{R}_{j(i)}$ denotes the average residual (computed without tree j) at terminal node i with total number of data n_i. Note that we can consider $\mu_\mu = 0$.

3.8.3 MH Acceptance ratios of CMC for BART

GROW Proposal:

- **Transition ratio:** No change.

- **Likelihood ratio:** Equation (3.43) changes to:

$$P(R_{l_1}, ..., R_{l_{n_l}} \mid \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n_l/2}} \ \left(\sqrt{2\pi\sigma_\mu^2}\right)^{1-\frac{1}{K}} \ \sqrt{\frac{\sigma^2}{\frac{\sigma^2}{K} + n_l\sigma_\mu^2}} \ \times$$

$$\exp\Big(- \frac{1}{2\sigma^2}\Big[\sum_{i=1}^{n_l} (R_{l_i} - \bar{R}_l)^2 - \frac{{\bar{R}_l}^2 {n_l}^2}{n_l + \frac{\sigma^2}{K\sigma_\mu^2}} + n_l {\bar{R}_l}^2 \Big]\Big) \quad (3.50)$$

Thus the likelihood ratio will change to:

$$\frac{P(R \mid T_*, \sigma^2)}{P(R \mid T, \sigma^2)} = (\sqrt{2\pi\sigma_\mu^2})^{1-\frac{1}{K}} \sqrt{\frac{\sigma^2(\frac{\sigma^2}{K} + n_l\sigma_\mu^2)}{(\frac{\sigma^2}{K} + n_{l_L}\sigma_\mu^2)(\frac{\sigma^2}{K} + n_{l_R}\sigma_\mu^2)}} \times$$

$$\exp\Big(\frac{\sigma_\mu^2}{2\sigma^2}\Big[\frac{(\sum_{i=1}^{n_{l_L}} R_{l_L,i})^2}{\frac{\sigma^2}{K} + n_{l_L}\sigma_\mu^2} + \frac{(\sum_{i=1}^{n_{l_R}} R_{l_R,i})^2}{\frac{\sigma^2}{K} + n_{l_R}\sigma_\mu^2} - \frac{(\sum_{i=1}^{n_l} R_{l,i})^2}{\frac{\sigma^2}{K} + n_l\sigma_\mu^2}\Big]\Big) \quad (3.51)$$

- **Tree Structure ratio:** The tree structure ratio will be raised to the power $1/K$:

$$\Big[\frac{P(T_*)}{P(T)}\Big]^{\frac{1}{K}}.$$

PRUNE Proposal:

- **Transition ratio:** No change.

- **Likelihood ratio:** This is the inverse of the likelihood ratio in the GROW proposal.

- **Tree Structure ratio:** This is also the inverse of the tree structure ratio in the GROW proposal.

CHANGE Proposal:

- **Transition ratio:** No change.

- **Likelihood ratio:**

$$\frac{P(R \mid T_*, \sigma^2)}{P(R \mid T, \sigma^2)} = \sqrt{\frac{(\frac{\sigma^2}{K\sigma_\mu^2} + n_1)(\frac{\sigma^2}{K\sigma_\mu^2} + n_2)}{(\frac{\sigma^2}{K\sigma_\mu^2} + n_1^*)(\frac{\sigma^2}{K\sigma_\mu^2} + n_2^*)}} \times$$

$$\exp\Big(\frac{1}{2\sigma^2}\Big[\frac{(\sum_{i=1}^{n_{1^*}} R_{1^*,i})^2}{\frac{\sigma^2}{K\sigma_\mu^2} + n_1^*} + \frac{(\sum_{i=1}^{n_{2^*}} R_{2^*,i})^2}{\frac{\sigma^2}{K\sigma_\mu^2} + n_2^*} - \frac{(\sum_{i=1}^{n_1} R_{1,i})^2}{\frac{\sigma^2}{K\sigma_\mu^2} + n_1} - \frac{(\sum_{i=1}^{n_2} R_{2,i})^2}{\frac{\sigma^2}{K\sigma_\mu^2} + n_2}\Big]\Big). \tag{3.52}$$

- **Tree Structure ratio:** The tree structure ratio will be raised to the power $1/K$.

Now the product of transition ratio and tree structure ratio is not 1 anymore:

$$\frac{P(T_* \to T)}{P(T \to T_*)} \times \frac{P(T_*)}{P(T)} = n_p(\eta)^{\frac{1}{K}-1} \; n_{p^*}(\eta^*)^{1-\frac{1}{K}}.$$

The conditional posterior of σ^2 and M_j changes to:

- $\sigma^2 \mid (T_1, M_1), ..., (T_m, M_m), Y, X \quad \propto Inv - Gamma(\rho, \gamma)$

  where $\rho = \frac{\nu+2+K(n-2)}{2K}$ and $\gamma = \frac{1}{2}\ [\ \sum_{i=1}^{n}(y_i - \sum_{j=1}^{m} g(x_i; M_j, T_j))^2 + \frac{\lambda\nu}{K}\ ]$.

- For the conditional posterior $M_j \mid T_j, R_j, \sigma$, we have:

$$\mu_{ij} \mid T_j, R_j, \sigma \sim \mathcal{N}(\frac{\frac{\sigma^2}{K\sigma_\mu^2}\ \mu_\mu + n_i \bar{R}_{j(i)}}{\frac{\sigma^2}{K\sigma_\mu^2} + n_i}, \frac{\sigma^2}{\frac{\sigma^2}{K\sigma_\mu^2} + n_i})$$

  where we can consider $\mu_\mu = 0$.

Chapter 4

Application of LISA to BART with efficient MH proposals

4.1 Introduction

Regression tree models (Chipman et al., 1998) have become popular in many applications as they are powerful tools for describing complex nonlinear relationships. They are efficient, simple to interpret and flexible in managing many nonlinear problems. Moreover, a Bayesian framework for the regression tree models (Chipman et al., 2002) is also of high importance as the complex nonlinear relationships are explained through all possible uncertainties. More specifically, the Bayesian Additive Regression Tree (BART) model (Chipman et al., 2010; Kapelner and Bleich, 2013) has become one of the most popular ensemble methods that perform significantly better compared to the single-tree models.

The most common approach for inference on BART models use Markov Chain Monte Carlo (MCMC) methods. However, existing Metropolis-Hastings (MH) proposals for trees in this model can suffer from poor mixing of the MCMC (Wu et al.,

2007), and hence result in overfitting issues. To overcome this problem, Pratola (2016) has introduced two new proposal moves for trees that can efficiently discover the tree space and hence improve the mixing of the MCMC.

On the other hand, BART models are often used to analyze large datasets and this can pose serious challenges as the run time can be prohibitively slow. In the previous chapter, we introduced the *Likelihood Inflating Sampling Algorithm (LISA)* (Entezari et al., 2018b) which is a new communication-free parallel method for posterior sampling of big datasets, with specific application on the BART model. In this chapter, we will examine the performance of LISA for BART with new tree proposals introduced by Matthew Pratola (henceforth, MP) (Pratola, 2016) and hence present consistency in LISA's performance. We discuss the use of MP's novel algorithm together with LISA to sample from posterior distributions arising from datasets which are too large to fit into a single machine's memory.

This chapter is organized as following. Section 4.2 describes the previous and new tree proposals proposed by MP for sampling trees from the BART model. Section 4.3 presents the results achieved from applying LISA to the BART model with the new MH proposals. Finally, section 4.4 closes the chapter with a brief summary.

4.2 Tree Proposals

Previously in chapter 3, we discussed the Bayesian framework performed for the BART model via MCMC. One of the main steps of the MCMC was to draw tree T_j using a Metropolis-Hastings (MH) algorithm from the conditional distribution:

$$p(T_j \mid R_j, \sigma) \quad \propto \quad p(T_j) \int p(R_j \mid M_j, T_j, \sigma)\, p(M_j \mid T_j, \sigma)\, dM_j. \tag{4.1}$$

Recall the four proposal moves for trees introduced by Chipman et al. (1998, 2010, 2002):

- **GROW:** Randomly choose a terminal node and split into two new nodes with splitting rules defined from the prior.
- **PRUNE:** Randomly choose a parent node with two terminal children nodes, and turn it into a terminal node by removing its children.
- **CHANGE:** Randomly select an internal node and reassign a splitting rule used in the prior.
- **SWAP:** Randomly select parent-child pair internal nodes and swap their splitting rules unless the other child has the identical rule, in which case the splitting rule of the parent is swapped with both children.

In chapter 3, we implemented LISA on BART using only the first three proposal moves for trees (ignoring SWAP) similar to the suggestions made by Kapelner and Bleich (2013). However, Pratola (2016) studies the issue of poor mixing of the MCMC with these tree proposals and hence introduces two new proposal moves that he suggests to be considered along with the GROW and PRUNE proposals. MP discusses that the only previous proposals that changed the dimensionality of the trees were the GROW and PRUNE moves, where the alterations only happened at the bottom of trees and thus preventing the MCMC to search in a broader space of trees. For this purpose, he has developed a novel ROTATION proposal that will change the dimension of the tree and explore the tree space more efficiently. On the other hand, the original CHANGE proposal chooses the variable and cutpoints from a uniform distribution such that no empty nodes will be created, which requires propagating the data through the tree to check this restriction, hence resulting to be computationally

expensive. To overcome this issue, MP has introduced an enhanced version to select the cutpoints and variables to split on in two separate moves called PERTURB and PERTURB within CHANGE-of-VARIABLE, which we will describe briefly below.

- **ROTATION:** Randomly select an internal node η_i. If it is the left child of its parent $p(\eta_i)$, a right-rotation will be applied (similarly for right child, a left-rotation). For a right-rotation, rules of node η_i will be swapped with its parent $p(\eta_i)$, and a new node will be added as the right child for the parent, $r(p(\eta_i))$, with the same rule previously in $p(\eta_i)$. At the same time, the sub-tree in the right child of η_i, $r(\eta_i)$, is moved to be a sub-tree at $l(r(p(\eta_i)))$. In addition, in the new tree, sub-trees in $r(r(p(\eta_i))$ and $r(l(p(\eta_i))$ will be duplicates of T_s, where T_s is the sub-tree in $r(p(\eta_i))$ from the original tree. The splitting rules are then updated accordingly.

- **PERTURB:** For node i, let $C^{v_i}_{p(i)}$ be the collection of cutpoints for all nodes ancestral of node i that split on variable v_i. The ancestral nodes can be partitioned into left and right ancestors of node i, i.e. $C^{v_i}_{p(i)} = \{C^{v_i}_{p_l(i)}, C^{v_i}_{p_r(i)}\}$, where a left ancestor is an ancestor of node i such that node i is on the right sub-branch of the ancestor node (similarly for right ancestor). Also let $C^{v_i}_{l(i)}$ (similarly $C^{v_i}_{r(i)}$) be the collection of cutpoints for all nodes in the left (similarly right) sub-tree of node i that split on variable v_i. Then a cutpoint for variable v_i is uniformly drawn from the interval $(a_i^{v_i}, b_i^{v_i})$ where:

$$a_i^{v_i} = max(0, max(C^{v_i}_{p_l(i)}), max(C^{v_i}_{l(i)})); \quad b_i^{v_i} = min(1, min(C^{v_i}_{p_r(i)}), min(C^{v_i}_{r(i)}))$$

  Note that the cutpoint proposal does not directly depend on data and hence it is efficient to calculate.

- **PERTURB within CHANGE-of-VARIABLE:** This change-of-variable proposal takes into account the fact that high correlation between two variables will result in the same partition of the observations throughout the tree structure. Thus MP proposes a transition from variable v_k to v_j with probability proportional to:

$$\frac{|Cor(X_k, X_j)| \times \mathcal{I}_{(a_i^{v_j}, b_i^{v_j}) \neq \{\}}}{\sum_l |Cor(X_k, X_l)| \times \mathcal{I}_{(a_i^{v_l}, b_i^{v_l}) \neq \{\}}}$$

  where $Cor(.,.)$ is a function that measures the relatedness level between variables.

In the next section, we will discuss the performance of LISA on BART using the new tree proposals in the MCMC.

4.3 Divide and Conquer Analysis via BART and LISA

In order to apply LISA, the data is divided into K batches and for each batch j we compute the partial posterior $\pi_j(\theta|\vec{x}^{(j)}) \propto p(\theta)[L(\theta|\vec{x}^{(j)})]^K$ where $p(\theta)$ is the model's prior and $L(\theta|\vec{x}^{(j)})$ is the likelihood for the data in the jth batch. Samples obtained from each partial posterior are combined to perform inference about $\pi(\theta)$, the full data posterior.

Previously, Entezari et al. (2018b) applied LISA to BART using the methods proposed in Chipman et al. (2010, 1998), and Kapelner and Bleich (2013), and concluded that by taking a weighted average of batch-draws that were generated with a minor modification to LISA (modLISA), one can produce indistinguishable posterior distributions from the full posterior distribution of BART.

In this chapter, we will apply modLISA to BART using the tree proposals presented by Pratola (2016) to examine consistency in results and time savings.

We consider the Friedman's test function (Friedman, 1991):

$$f(x) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5,$$

and simulate 20,000 observations $y \sim N(f(x), \sigma^2)$ where $\sigma = 0.1$, and $x = (x_1, \ldots, x_{10})$ are uniformly drawn from $(0, 1)$. The sample size is chosen so that we can still run MP's algorithm to sample the full-data posterior in reasonable time. We have used the implementation of BART by Pratola (2016) to apply modLISA to this dataset with $K = 30$ batches.

Table 4.1 is comparing the results of 1000 posterior samples generated from modLISA after 1000 burn-in iterations, to the SingleMachine which ran MP's algorithm on the full dataset on one single machine. Note that we also simulated an additional 5000 observations as test data to fully compare the methods. Table 4.1 contains root mean squared error (RMSE) of $f(x)$ for both train and test data as well as the mean σ estimate. Both methods were performed with 30% rotate proposals without any adaptation. As seen in Table 4.1 the parallel algorithm produces results that are very similar to the ones produced by SingleMachine. This is in line with the findings in Entezari et al. (2018b). Table 4.2 shows, for each algorithm, the empirical test data coverage of the 90% credible interval for $f(x)$, average tree depth, total run time (real world time in seconds) and the inverse product of Test RMSE and running time which can be thought of as a measure of computational efficiency. Interestingly, modLISA has higher coverage and lower average tree depth than SingleMachine. Total run time is more than 10 times faster for modLISA.

Figure 4.1 compares the empirical distribution functions of $\hat{f}(x)$ in modLISA to

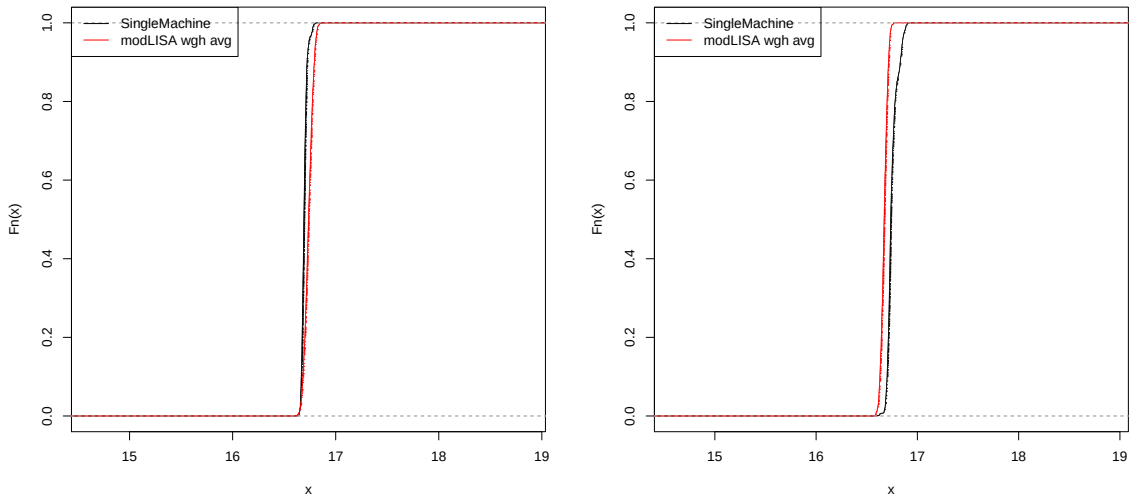
Table 4.1: Results of training data RMSE, test data RMSE and mean post burn-in $\hat{\sigma}$ from each method with 30% rotate proposals. There are $K = 30$ batches in total.

Method	Train RMSE	Test RMSE	Mean $\hat{\sigma}$
modLISA	0.137	0.147	0.176
SingleMachine	0.075	0.087	0.123

Table 4.2: Computational efficiency comparison between modLISA and SingleMachine

Method	Test Coverage	Avg tree depth	Total Run Time (secs)	1/(Test RMSE × Time)
modLISA	70.8 %	1.01	121.6	0.056
SingleMachine	63.7 %	2.07	1585.5	0.007

SingleMachine for two different observations in the test data. As it is seen, the two empirical distribution functions are indistinguishable.



(a) observation 3300 (b) observation 4600

Figure 4.1: Comparing empirical distribution functions of $\hat{f}(x)$ in modLISA weighted average with $K = 30$ to SingleMachine BART for two different test observations.

4.4 Discussion

In this chapter we examined the performance of LISA on BART with efficient Metropolis-Hastings proposals for trees introduced by MP. We applied LISA (modLISA) on BART using a simulated dataset from the Friedman's test function with $\sigma = 0.1$ where poor mixing was observed with the original MH proposals as discussed by Pratola (2016). By applying the new ROTATION and PERTURB tree proposals introduced by MP along with the original GROW and PRUNE proposals, we were able to show that modLISA still generates accurate prediction results with much faster run time compared to the SingleMachine for large datasets. More importantly, modLISA generated higher test coverage with approximately similar empirical distribution function as the SingleMachine.

Overall, modLISA for BART showed consistent results with the ones found in Entezari et al. (2018b), which illustrates the ability of modLISA to effectively sample from posterior distributions when the datasets are too large and need to be divided into K batches before proceeding. It is also important to note that poor mixing of the MCMC on BART can still take place when large datasets are fitted to the model, whereas with modLISA this issue does not occur as the dataset is partitioned into smaller sets. Thus modLISA on BART along with efficient MH proposals can create a powerful combination that will prevent poor mixing of MCMC with BART.

Chapter 5

Bayesian Spatial Analysis of Hardwood Tree Counts

5.1 Introduction

5.1.1 The forest inventory problem

The forest industry has a significant impact on the economy of countries such as Canada, making forests an important financial asset. The monetary value of forest assets is mainly determined by their timber, the value of which depends on different features of trees such as size, species, age, defects, etc. Tree species have different types of wood with different qualities, and hence influence the timber value.

Tree species have two main categories, hardwood (deciduous) trees and softwood (coniferous) trees, with hardwood trees generally having wider leaves that are lost annually, while softwood trees have smaller leaves and retain their leaves throughout the year. Hardwood trees provide much longer lasting wood compared to softwood trees, with slower growth rates which makes them more expensive compared to softwood.

Hence, knowing the number of hardwood trees in a forest is valuable information. Collecting data on forests requires hiring workers to travel to different sites around the forests and measure the quantities needed, which can be costly and time consuming.

Remote sensing technologies can overcome this issue. Although they are cheap and efficient and can cover a wide range of geographical areas, they can suffer from lack of accuracy. Geostatistical models are powerful tools for analyzing and predicting such spatial data, and can be used to calibrate remotely sensed data (see Curran and Atkinson, 1998). Existing literatures by Giorgi et al. (2017); Shaby and Reich (2012); Abellan et al. (2007) are examples of the importance of statistical models for spatial analysis. The focus of this chapter will also be to take advantage of statistical tools to predict the number of hardwood trees using geostatistical models that take into account the spatial factor.

5.1.2 Model-based geostatistics

In the past few decades, spatial statistics has become an established field of statistics with well developed models applied to many real-world problems. Conventional geostatistical models for Gaussian spatial data were first popularized by Matheron (1962) and later on built upon by Cressie (1993). The generalization of these models for non-Gaussian data were introduced by Diggle et al. (1998).

Let Y_i be the observed spatial data at location s_i, with arbitrary distribution f that has mean λ and possible additional parameters γ. Consider $X(s_i)$ as the covariates at location s_i. Modelling this data with the Generalized Linear Geostatistical Model (GLGM) described in Diggle et al. (1998) and Diggle and Ribeiro (2007), will be as following:

$$\begin{aligned} Y_i|U(s_i) &\sim f[\lambda(s_i), \gamma] \\ g[\lambda(s_i)] &= \mu + \beta X(s_i) + U(s_i) \end{aligned} \tag{5.1}$$

where $g(.)$ is the link function (i.e. logit or log). Here $U(s)$ is a Gaussian random field U evaluated at location s, which is characterized by the joint multivariate normal distribution:

$$[U(s_1), ..., U(s_N)]^T \sim MVN(0, \Sigma)$$

where the elements of Σ are defined by a spatial correlation function ρ as

$$\Sigma_{ij} = \text{cov}[U(s_i), U(s_j)] = \sigma^2 \rho(||s_i - s_j||/\phi, \nu)$$

where ϕ is a range parameter and ν is a vector of other possible parameters. The range parameter ϕ controls the rate at which the correlation decreases with distance. There are many possible parametric functions for ρ, with Matérn correlation function (see Stein, 1999) being the most commonly used. The Matérn correlation is defined as:

$$\rho(h; \phi, \kappa) = \frac{1}{2^{\kappa-1}\Gamma(\kappa)} \Big(\frac{||h||}{\phi}\Big)^{\kappa} K_{\kappa}\Big(\frac{||h||}{\phi}\Big), \tag{5.2}$$

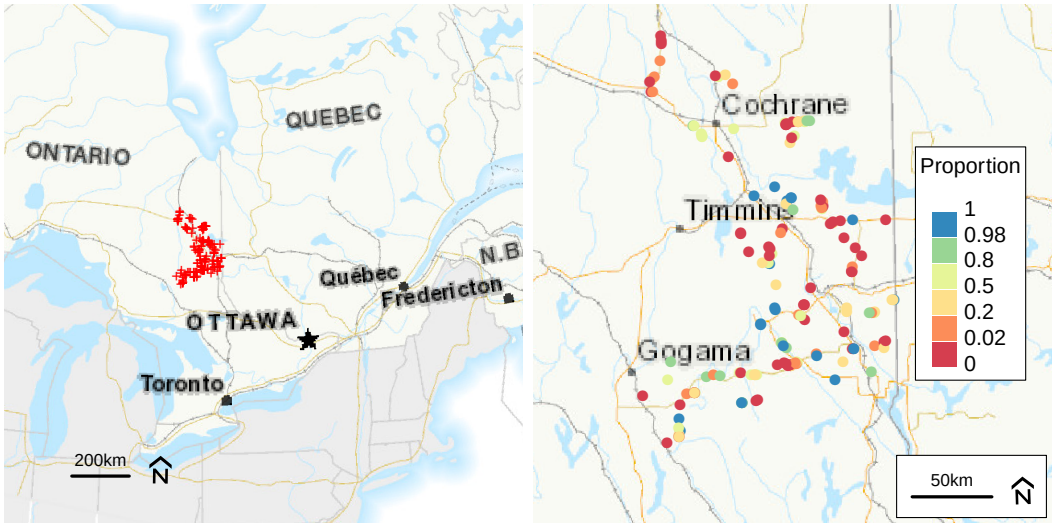
where $\Gamma(.)$ is the gamma function and $K_{\kappa}(.)$ is the modified Bessel function of the second kind of order $\kappa > 0$ (κ being a shape parameter). This function is particularly interesting, as it is flexible in the differentiability of the Gaussian process $U(s)$ by adjusting κ (Stein, 1999).

In the case where the data is Gaussian, Maximum Likelihood Estimates (MLEs) can be used as point estimates for the model parameters. However, when the data is non-Gaussian, because of the unobserved latent variables U present in the model, the likelihood function becomes intractable and it is difficult to calculate the MLEs. Performing Bayesian inference on these models via Markov Chain Monte Carlo (MCMC) methods (Brooks et al., 2011; Craiu and Rosenthal, 2014) has many advantages (as discussed in Diggle et al. (1998)). The Integrated Nested Laplace Approximation (INLA) algorithm introduced by Rue et al. (2009), is an alternative to MCMC for Bayesian Inference on latent Gaussian models. However this method has some drawbacks as it approximates marginal posterior distributions rather than joint posterior distributions. There are facilities in the R-INLA software for producing approximate joint posterior samples, but the properties of these samples have yet to be explored.

In this chapter, we will analyze the spatial hardwood tree count data collected from the Timiskaming & Abitibi River forests in Ontario, Canada. Our analysis is constructed in a Bayesian framework for a binomial geostatistical model to predict the proportion of hardwood trees from remotely sensed elevation and vegetation data. For posterior simulations, we implement an MCMC method using the Langevin-Hastings (see Roberts and Rosenthal, 1998) and the Random-Walk Metropolis Hastings (see Roberts et al., 1997; Roberts and Rosenthal, 2001) algorithms. By reducing the amount of training data fitted to the model, and predicting for the same validation set, we are able to answer questions related to the accuracy of predictions given small amounts of ground truth data collected. We will show that with training data size as small as 10 spatial locations, despite the increase in uncertainty, the true number of hardwood trees lies within a 95% prediction interval. This conclusion is very valuable as it will significantly reduce costs of collecting ground truth data. We will also compare our results with the logistic regression model where there is no spatial effect.

Furthermore, we explore a stratified sampling approach in choosing the training data that will show a potential improvement in predictions.

The chapter is organized as follows. The spatial data from the Timiskaming & Abitibi River Forests are described in section 5.1.3. Section 5.2 describes the geostatistical model used for our data and the MCMC algorithm applied to perform Bayesian Inference. In addition, we explain our stratified approach and describe the measurements we will use to compare and assess predictions. Section 5.3 discusses the numerical results from fitting the data, where comparisons are also made with the Logistic Regression. At last, we summarize our results in Section 5.4. The Appendix includes results from different simulations.



(a) Sample locations (b) Proportions of hardwood trees

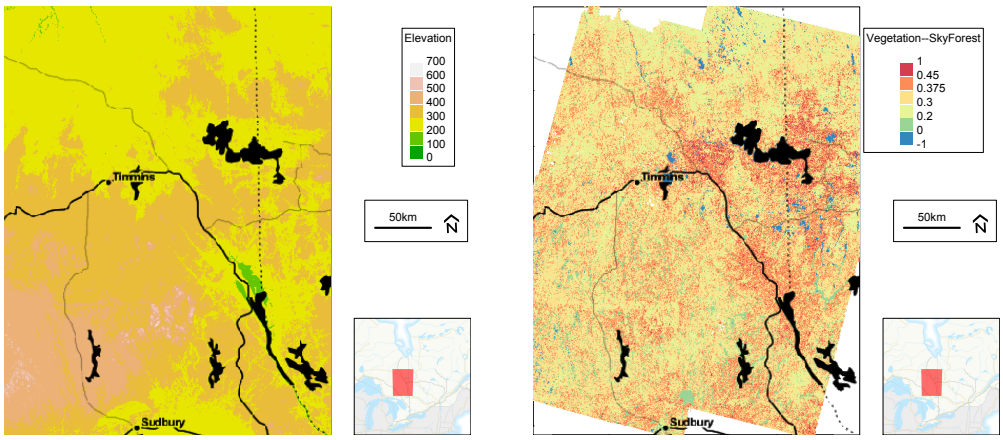
Figure 5.1: Locations of 162 forest plots in the Timiskaming and Abitibi River Forests.

5.1.3 Description of Data

The Timiskaming and Abitibi River forests are geographically located next to one-another in northern Ontario, Canada. The First Resource Management Group Inc. has provided detailed data from 162 individual forest plots inside these adjacent

forests. Each forest plot is 11.28m in radius to provide a $400m^2$ surface. The geographical locations of these 162 sites are shown in Figure 5.1.

The data from each site consists of information on the total number of trees, whether each tree is living or dead, and the species of each tree. Figure 5.1b shows the proportion of live trees which are hardwood from the 162 sites. As can be seen, many sites have no hardwood trees and such sites are scattered throughout the forests.



(a) Elevation (b) SkyForest^{TM} vegetation index

Figure 5.2: Elevation & Vegetation index around the Timiskaming and Abitibi River Forests (Background ©Stamen Design).

The remotely sensed data considered includes elevation values from satellite data provided by the SRTM program (Figure 5.2a). A measure of forest vegetation was provided by the First Resource Management Group Inc. using the proprietary remote sensing technology "SkyForest^{TM}", which is shown in Figure 5.2b. This vegetation measure is predicted by SkyForest^{TM} across the forest landscape by selecting an arithmetic transformation of spectral bands (ATSB) from a candidate list of ATSBs. The ATSBs are constructed similarly to well known vegetation indices such as the Normalized Difference Vegetation Index (NDVI), with some of them being multi-temporal. It is thus expected that hardwood trees are located where this measure is high.

In the next section, we will describe the geostatistical model for our dataset, along with the steps taken to perform a Bayesian analysis.

5.2 Methods

5.2.1 Logistic Regression

Before describing the full geostatistical model for our data, a simple Logistic Regression model with binomial response will be outlined. Consider Y_i to be the count of hardwood trees in forest plot i, and write $Y_i \sim \text{Binom}(n_i, p_i)$, where n_i is the total number of live trees at site i (s_i) and p_i is the probability of a tree in plot i being hardwood. Elevation and the SkyForest$^{\text{TM}}$ index are covariates in the model. The SkyForest$^{\text{TM}}$ covariate is treated as a linear effect with change point at 0.3 (approximately its average value), giving some additional flexibility to this covariate in the regression model. The elevation values are also centered at the average value of about 320. For computational reasons, we normalize the covariates by dividing by the standard deviation. The model is:

$$\begin{aligned} Y_i &\sim \text{Binom}(n_i, p_i) \quad i = 1, ..., 162 \\ \log\left(\frac{p_i}{1-p_i}\right) &= X(s_i)\beta \end{aligned} \tag{5.3}$$

Writing $A(s)$ as the SRTM-measured altitude at location s and $V(s)$ as the SkyForest^{TM} vegetation index, the normalized vector of covariates $X(s)$ is constructed by:

$$\begin{aligned} X_1(s) &= 1 \\ X_2(s) &= \frac{A(s) - 320}{50} \\ X_3(s) &= \frac{\min(V(s) - 0.3, 0)}{0.05} \\ X_4(s) &= \frac{\max(V(s) - 0.3, 0)}{0.05} \end{aligned}$$

5.2.2 The geostatistical model

Spatial dependence in the prevalence of hardwood trees should be expected as sites in the forests close to one another may benefit from the same soil, weather, etc, and hence may have similar tree types. Thus we expect a geographical effect to play an important role in explaining such data with a more sophisticated model such as the Generalized Linear Geostatistical Model (GLGM). A geostatistical model for our spatial data will have an extra spatial term $U(s)$ and an independent term Z compared to the model in (5.3), resulting:

$$\begin{aligned} Y_i &\sim \text{Binom}(n_i, p_i) \quad i = 1, ..., 162 \\ \log\left(\frac{p_i}{1-p_i}\right) &= t_i = X(s_i)\beta + U(s_i) + Z_i \end{aligned} \tag{5.4}$$

where

$$Z_i \overset{i.i.d.}{\sim} N(0, \tau^2),$$

$$U(s) \sim N(0, \sigma^2),$$

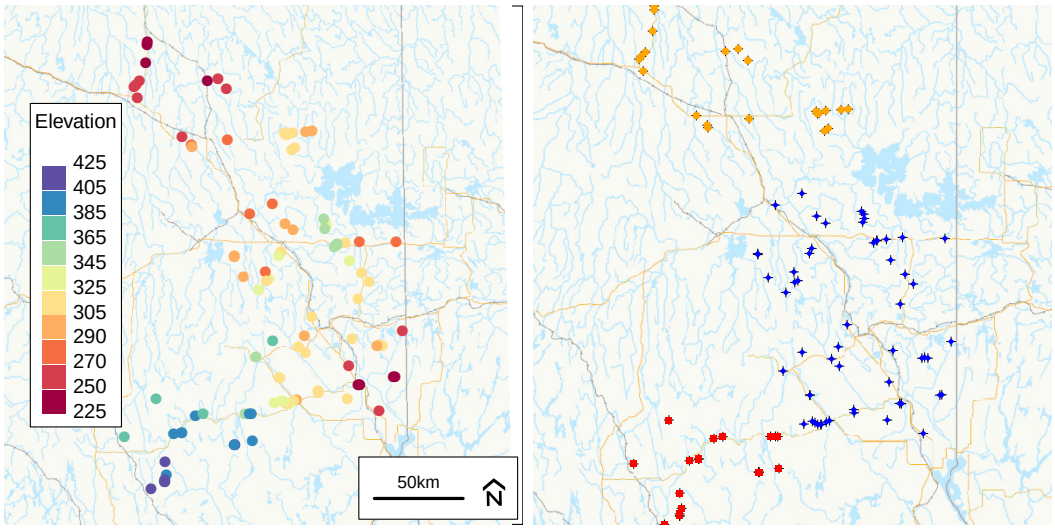
$$\text{cov}(U(s+h), U(s)) = \sigma^2 \rho(||h||; \phi, \kappa)$$

This model is equivalent to (5.1) where f is Binomial and g is a logit link function.

5.2.3 Random Sampling vs Stratified Sampling

For our analysis, we explore reducing the size of the training data fitted to the model, to observe and examine the trade-off between prediction accuracy and costs of collecting ground truth data. More specifically, if we were only given data from 25 or 10 plots on the ground, could useful predictions still be made? To answer this question, the 162 plots in the dataset were divided into 100 training and 62 validations sets. Keeping the 62 validation set fixed, we examine the performance of results generated by fitting 100, 25, and 10 training data to the model. For this purpose, we can do this by two different approaches, 1) choosing random subsets of data and 2) choosing stratified subsets of data. Since the spatial data is correlated, choosing the subset of data with a stratified approach should be expected to improve the results, as it can force the training plots to be as scattered as possible. Both elevation and vegetation covariates are taken into account for choosing the 25 and 10 dataset from the 100. Hence, we begin by looking at the elevation from all the 100 training data (first simulation) as shown in Figure 5.3a.

The 100 plots are stratified into three groups as shown in Figure 5.3b, and both location of the points as well as elevation values are taken into account equally. Keeping the proportion of the data from each strata constant, we systematically sample 25 plots from the 100 by sorting the vegetation index in each strata and taking every $j-th$ element depending on the number of data needed (similarly for the 10 data points from the 25). The Results section will explore how stratified sampling can (potentially) improve prediction accuracy with smaller training data



(a) Elevation for 100 training data (b) Stratified regions

Figure 5.3: Plots of elevation from 100 training data, along with the plot of stratified regions.

fitted to the model, compared to random sampling.

5.2.4 Inference

We will apply a Bayesian approach to the model in (5.4), and this methodology will be referred to as the Bayesian Generalized Linear Geostatistical Model (BGLGM). Let $\beta^T = (\beta_0, \beta_1, \beta_2, \beta_3)$, $\theta^T = (\sigma^2, \phi, \tau)$, and $t^T = (t_1, ..., t_n)$ with $t_i = X(s_i)\beta + U(s_i) + Z_i$, be the three sets of parameters. We treat κ as fixed at 1.5, since it is not of direct interest and according to Zhang (2004), not all the parameters (σ^2, ϕ, and κ) are consistently estimable. We define priors for each parameter as

$$\theta \sim \pi_1(.) \quad \& \quad \beta|\theta \sim \pi_2(.) = N(\mu, \sigma^2\Omega) \quad \& \quad t|\beta, \theta \sim \pi_3(.) = MVN(X\beta, \Sigma(\theta))$$

with joint posterior distribution given as:

$$\pi(\beta, \theta, t|y) \propto \pi_1(\theta)\pi_2(\beta|\theta)\pi_3(t|\beta, \theta)f(y|t) \tag{5.5}$$

where $f(y|t) = \prod_{i=1}^{n} f(y_i|t_i)$ is the likelihood function. Here $\Sigma(\theta)$ is the covariance matrix with diagonal elements equal to $\sigma^2 + \tau^2$ and off-diagonal elements of $\sigma^2\rho(||s_i - s_j||; \phi, \kappa)$ where ρ is the Matérn correlation function. We consider *Exponential(0.5)* priors for σ and τ, and a *Gamma(3,35)* prior for ϕ.

There are a number of R packages available for posterior estimation of the BGLGM. The **geostatsp** package by Brown (2015) uses INLA to approximate the marginal posterior distributions, while the recent **PrevMap** package (Giorgi and Diggle, 2017) uses an MCMC method to generate joint posterior draws. In this chapter, our focus will be on using MCMC methods to generate joint posterior samples of BGLGM. However, **PrevMap** performed poorly with this dataset when the number of data points was very small, and a bespoke MCMC algorithm was developed as a result.

As reparamterization and standardization help reduce correlation between variables, they will be play an important role in improving the mixing and convergence of MCMC algorithms. The transformations applied to all the model parameters in (5.5) follow the recommendations of Christensen et al. (2006).

Let $\Lambda(t)$ be a diagonal matrix with elements $-\partial^2/\partial t_i^2 \ \log f(y_i|t_i)$ for $i = 1, ..., n$, and denote $\hat{t}_i = \arg\max f(y_i|t_i)$. Assuming a prior $N(\mu, \Omega)$ for β, let $\tilde{\Sigma} = (\Sigma^{-1} + \Lambda(\hat{t}))^{-1}$ and $\tilde{\Omega} = (\Omega^{-1} + X^T(\Sigma^{-1} - \Sigma^{-1}\tilde{\Sigma}\Sigma^{-1})X)^{-1}$. Then by factorizing the posterior distribution in (5.5) into two parts: $\pi(\beta, \theta, t|y) \propto \pi_1(\theta)f(t, \beta|\theta, y)$, we will be able to

simplify the second factor $f(t, \beta|\theta, y)$ as following:

$$\begin{aligned}\log f(t,\beta|\theta,y) &\approx -0.5(t-\hat{t})^T\Lambda(\hat{t})(t-\hat{t}) - 0.5(t-X\beta)^T\Sigma^{-1}(t-X\beta) - 0.5(\beta-\mu)^T\Omega^{-1}(\beta-\mu) \\ &= -0.5(t-\tilde{\Sigma}(\Lambda(\hat{t})\hat{t}+\Sigma^{-1}X\beta))^T\tilde{\Sigma}^{-1}(t-\tilde{\Sigma}(\Lambda(\hat{t})\hat{t}+\Sigma^{-1}X\beta)) \\ &\quad - 0.5(\beta-\tilde{\Omega}(X^T\Sigma^{-1}\tilde{\Sigma}\Lambda(\hat{t})\hat{t}+\Omega^{-1}\mu))^T\tilde{\Omega}^{-1}(\beta-\tilde{\Omega}(X^T\Sigma^{-1}\tilde{\Sigma}\Lambda(\hat{t})\hat{t}+\Omega^{-1}\mu))\end{aligned} \tag{5.6}$$

where the first expression $-0.5(t-\hat{t})^T\Lambda(\hat{t})(t-\hat{t})$ is derived from the Taylor expansion of $\log f(y|t)$ around $\hat{t}$. From equation (5.6), we can simply use the transformations:

$$\tilde{\boldsymbol{t}} = (\tilde{\Sigma}^{1/2})^{-1}(\boldsymbol{t} - \tilde{\Sigma}(\Lambda(\hat{t})\hat{t} + \Sigma^{-1}X\beta)) \tag{5.7}$$

$$\tilde{\boldsymbol{\beta}} = (\tilde{\Omega}^{1/2})^{-1}(\boldsymbol{\beta} - \tilde{\Omega}(X^T\Sigma^{-1}\tilde{\Sigma}\Lambda(\hat{t})\hat{t} + \Omega^{-1}\mu)) \tag{5.8}$$

where $\tilde{t}_1, ..., \tilde{t}_n$ and $\tilde{\beta}_1, ..., \tilde{\beta}_p$ are now approximately uncorrelated with mean zero and variance one. These parameters are also uncorrelated with θ and hence there will be no posterior dependence between $\tilde{t}, \tilde{\beta}$, and θ. However, according to Christensen et al. (2006) and Giorgi and Diggle (2017), there is posterior dependence within the parameters of $\theta^T = (\theta_1, \theta_2, \theta_3) = (\sigma^2, \phi, \tau)$, and hence a reparameterization is proposed as following:

$$\tilde{\boldsymbol{\theta}} = (\tilde{\theta}_1, \tilde{\theta}_2, \tilde{\theta}_3) = (\log\sigma, \log\sigma^2/\phi^{2\kappa}, \log\tau^2)$$

In general, using these transformations will help facilitate the choice of proposal densities as well as reducing the correlation between variables that will significantly improve mixing and convergence of the MCMC algorithm.

Using these reparameterizations, we have implemented a Metropolis-Hastings-within-Gibbs sampling method that updates each blocks of $\tilde{\theta}, \tilde{\beta}$, and $\tilde{t}$ at a time.

However, for high-dimensional parameters, it is more suitable to use the Langevin-Hastings algorithm as they will have much faster convergence rates (Roberts and Rosenthal, 1998; Roberts and Tweedie, 1996; Møller et al., 1998). For our model and data, $\tilde{t}$ has the highest dimension, hence we will use Langevin-Hastings algorithm to update $\tilde{t}$. For the remaining blocks we will use the Random-Walk Metropolis Hastings (RWMH) algorithm. The summary of the steps used to run the MCMC algorithm is shown in the diagram below.

Algorithm 2: MCMC algorithm

1. Initialize θ, β, and t
2. Transform to $\tilde{\theta}, \tilde{\beta}$, and $\tilde{t}$
3. Update $\tilde{\theta}_1, \tilde{\theta}_2$ and $\tilde{\theta}_3$ using a RWMH, each with standard deviation s_i calculated iteratively as:

$$s_i = s_{i-1} + c_1 i^{-c_2}(\alpha_i - 0.45)$$

   where $c_1 > 0$ and $c_2 \in (0, 1]$ are constants, and α_i is the acceptance probability up to $i - th$ iteration with optimal acceptance probability of 0.45.
4. Update $\tilde{\beta}$ using a RWMH
5. Update $\tilde{t}$ with a Langevin-Hastings algorithm, i.e. $\tilde{t}' \sim MVN(\tilde{t} + 0.5h\nabla \log \pi(\tilde{t}), hI)$ where h is recommended to be $1.65^2/n^{1/3}$.
6. Repeat steps 3-5 until the desired number of samples are collected.
7. Transform samples of $\tilde{\theta}, \tilde{\beta}$, and $\tilde{t}$ back to θ, β, and t.

5.2.5 Prediction & Assessment

After running our MCMC algorithm on the BGLGM, we will combine the posterior samples for each parameter to generate posterior distributions for hardwood probabilities at each of the 62 validation locations. We will then emphasize on assessing the predictions from the number of hardwood counts rather than proportions, since the observed proportions are often 0 or 1, while predictions are $0 < p < 1$. Below we describe the various assessments we have considered:

1. *Coverage Probability*: For each of the 62 validation points, we generate posterior samples of hardwood counts from the corresponding posterior probability samples, then examine whether the true hardwood count is inside the (say) 95% posterior interval. The coverage probability will be the proportion of 62 points that are inside their posterior intervals, i.e.:

$$\#(true\ hardwood\ count\ \in\ posterior\ interval\ of\ hardwood\ counts)/62$$

2. *RMSE (root mean squared error)*: We will also compare RMSE of hardwood probabilities from both BGLGM and GLM (Logistic Regression), calculated as:

$$RMSE = \sqrt{\frac{1}{62}\sum_{j=1}^{62}(\hat{p}_j - p_j)^2}$$

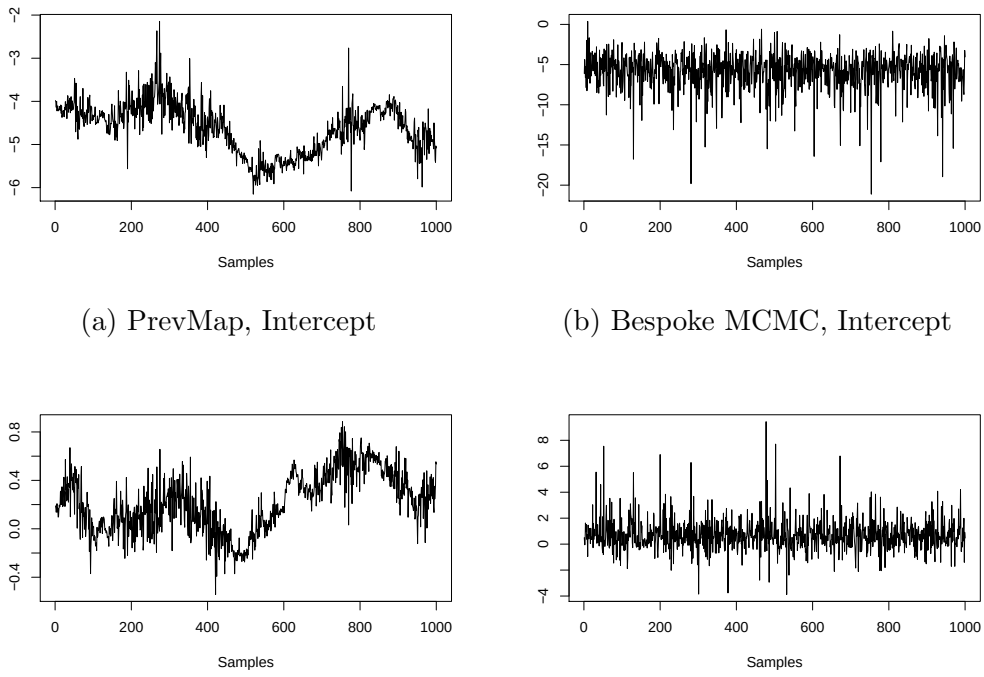
where p_j is the true proportion of hardwoods in plot i (often 0 or 1) and $\hat{p}_j$ is the predicted hardwood probability in GLM and posterior mean in BGLGM.

3. *Total hardwood count distribution*: We also consider the distribution of the total number of hardwoods in *all* 62 validation sites and examine whether the true total hardwood counts is covered within the 95% posterior interval. Unlike the posterior distributions of hardwood counts in each of the 62 plot, the total count has a reasonably symmetric distribution. In addition, we have compared this to the corresponding distribution generated from GLM via bootstrapping.

5.3 Results

For the main analysis we have run the MCMC algorithm for 2,000,000 iterations with 1,000,000 burnin and 100 thinning. Runs consist of fitting 100, 25, and 10 sites as

training data, both via random and stratified sampling, with predictions made for the 62 validation data. We have repeated this procedure for five different simulations by randomly choosing five different validation sets of size 62.



(a) PrevMap, Intercept

(b) Bespoke MCMC, Intercept

(c) PrevMap, elevation

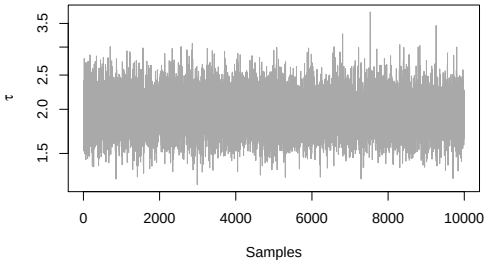
(d) Bespoke MCMC, elevation

Figure 5.4: Comparing trace plots of β_0 and β_1 from the bespoke MCMC implementation and the PrevMap package.

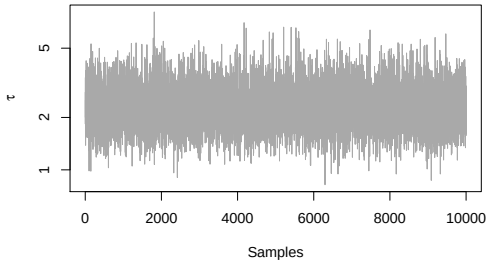
5.3.1 MCMC Convergence and Mixing

Figure 5.4 is showing the trace plots of posterior samples for β generated from the bespoke MCMC implementation from section 5.2.4 versus PrevMap package, using 10 training sites. The bespoke MCMC mixes well and has converged, while PrevMap trace plots have not converged although they have been ran for the same number of iterations. Different tuning parameters for running PrevMap were considered without

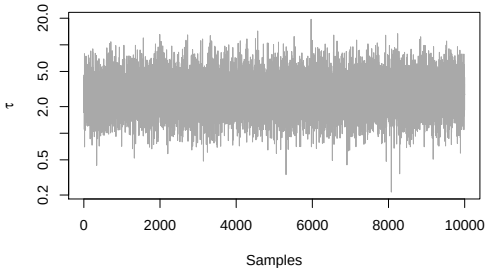
the chains being improved. Thus we will be using the bespoke MCMC implementation for the rest of the analysis in this chapter.



(a) 100 training sites — τ



(b) 25 training sites — τ



(c) 10 training sites — τ

Figure 5.5: Trace plots of 10,000 MCMC posterior samples for τ (simulation 1).

Figures 5.5a, 5.5b, and 5.5c show the MCMC trace plots for only the τ parameter with 100, 25, and 10 data fitted to the model respectively. All trace plots show that the MCMC is mixing well and thus, the chains have converged. In addition, the trace plots show larger variability with less training data fitted. The remaining trace plots for other parameters as well as other simulations are included in the appendix.

For quantitively verifying this variability between different training data size, we have compared the numerical values of posterior mean, 2.5 %, and 97.5 % quantiles of all model parameters in Table 5.1. While the posterior means remain almost unchanged, the 95% posterior intervals for each model parameter (except ϕ), become

Table 5.1: Comparison of posterior mean, 2.5 %, and 97.5 % quantiles of model parameters, for different sizes of training data. These results are from only the first of five training samples.

Parameters	# of training	Mean	2.5% quantile	97.5% quantile
Intercept - β_0	100	-3.47	-4.33	-2.65
	25	-3.54	-5.67	-1.66
	10	-2.37	-6.38	1.31
Elevation - β_1	100	0.53	0.07	0.99
	25	0.12	-1.03	1.13
	10	2.16	-0.96	6.38
SkyF<0.3 - β_2	100	2.89	1.19	4.87
	25	2.09	-0.50	5.26
	10	3.38	-1.39	10.42
SkyF>0.3 - β_3	100	2.61	2.10	3.17
	25	3.02	1.86	4.44
	10	4.20	1.85	7.70
Spatial sd - σ	100	0.04	0.02	0.11
	25	0.04	0.02	0.12
	10	0.06	0.02	0.17
Indep. sd - τ	100	1.98	1.52	2.55
	25	2.38	1.36	4.05
	10	3.09	1.04	7.23
Range(km) - ϕ	100	104.94	21.89	255.14
	25	105.42	22.00	252.93
	10	105.06	21.30	255.27

wider with less training data fitted to the model, indicating more uncertainty in parameter estimation.

5.3.2 Parameter posteriors & spatial surfaces

The prior and posterior densities of model parameters from the first simulation are shown in Figures 5.6 and 5.7. From these figures we can ascertain that with fewer training data, posterior densities become wider and hence result in more uncertainty

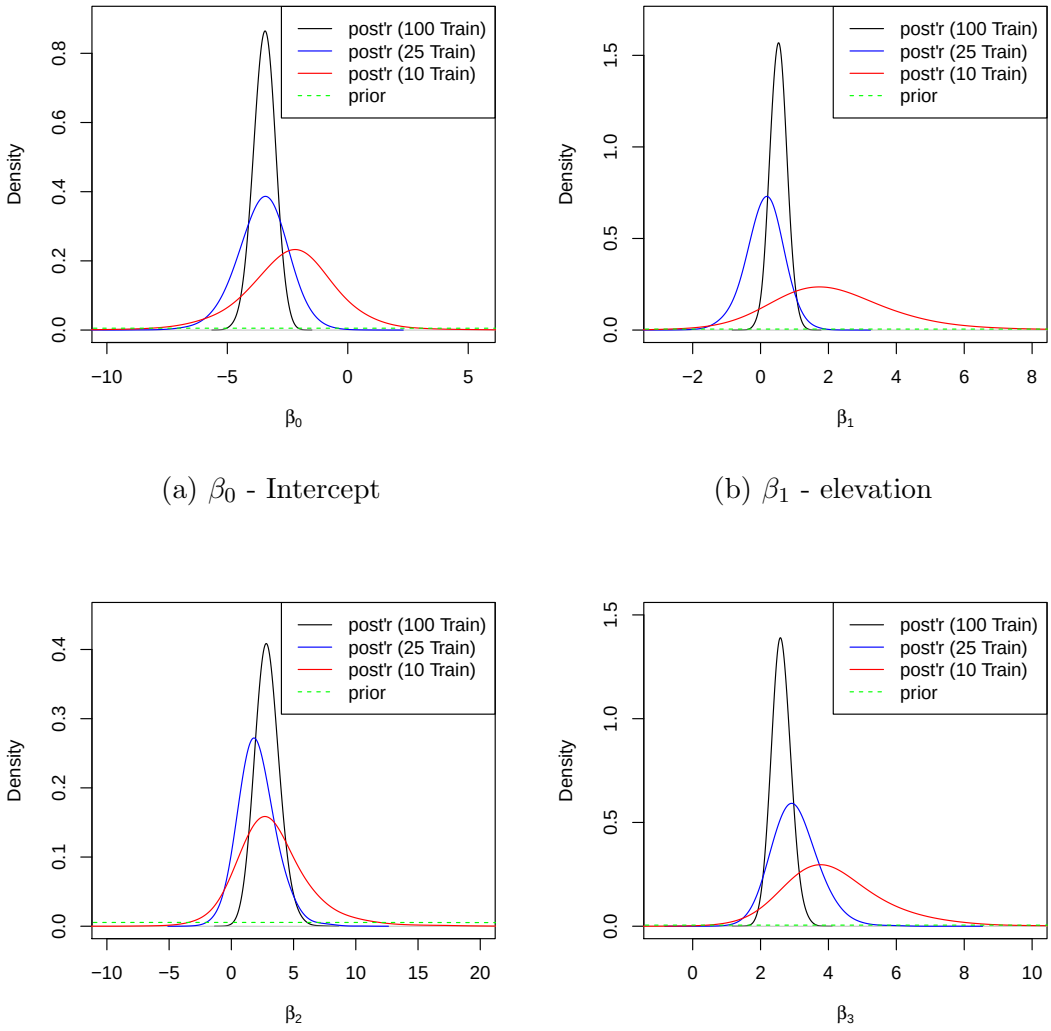
of predictions. The posterior distributions of σ suggest small spatial random effects for this dataset, as they have modes concentrated at smaller values. Posterior densities of ϕ are all similar and remain unchanged for different training data, as small σ causes weak spatial signal which can't identify ϕ.

One surprising feature of Figure 5.7b, is the posterior density with 10 training data points does not resemble the prior. Even the smallest training dataset considered provides clear evidence that there is more variation in the observed counts than the covariates predict, which is manifest in the results as τ has a posterior distribution concentrated away from zero. There is also evidence that this extra variation is not spatially structured, since σ is clearly much smaller than τ.

The main goal is to predict the composition of trees at unmeasured sites in the forests via simulating posterior samples of $U(g_l)$ for new locations $g_l : l = 1, ..., L$, conditional on MCMC posteriors $\{U(s_i) + Z(s_i) : i = 1, ..., n\}$. Considering a 100×100 grid with $L = 10,000$ cells inside the forests as our new locations, we can simulate $U(g_l)$ using the `RFsimulate` function in the "RandomFields" package and make predictions for hardwood probabilities $p(g_l)$ for each cell. The RandomFields package has very efficient algorithms for simulating from conditional distributions of spatial processes without using the full variance matrix. Thus assuming we have grid cells $g_1, ..., g_L$, we simulate $[U(g_1), ..., U(g_L)|Y]$ and independent $Z_1, ..., Z_L$ along with the use of other posterior samples to generate $[p(g_1), ..., p(g_L)|Y]$.

Figure 5.8 shows images of three different posterior samples along with posterior means (in each row) generated from fitting different training data sizes. With fewer training data the posterior rasters appear to become smoother, possibly indicating less precise predictions.

The 62 validation sites with their ground truth number of hardwood trees are used to evaluate predictions by summarizing results over all corresponding sites. The



(a) β_0 - Intercept

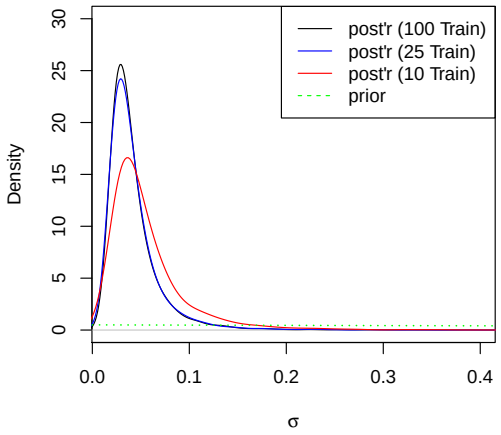
(b) β_1 - elevation

(c) β_2 - SkyForest^{TM} vegetation below 0.3

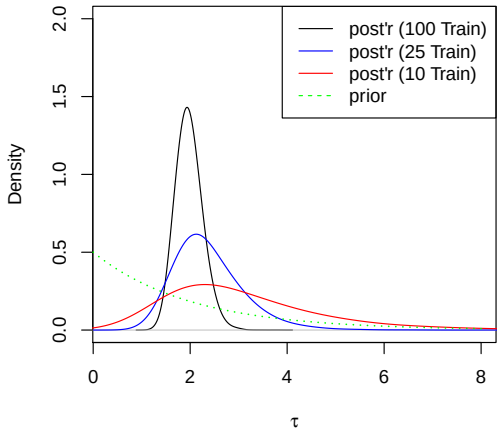
(d) β_3 - SkyForest^{TM} vegetation above 0.3

Figure 5.6: Prior and posterior distributions of parameters from the first simulation.

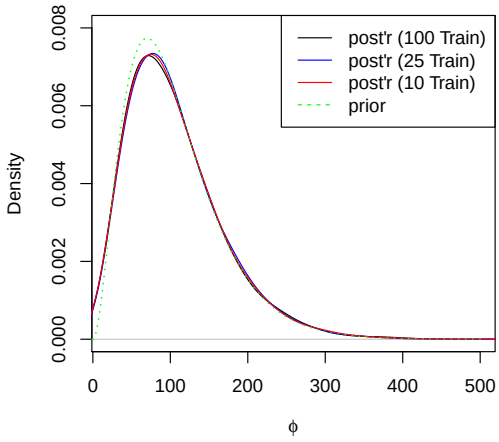
number of hardwood trees in each validation site is predicted and their coverage probabilities calculated from posterior intervals of hardwood counts. Table 5.2 shows the corresponding coverage probabilities of 95%, 80%, and 50% Posterior Credible Intervals (CI) for different training data size, averaged over five different simulations. Note that many observed proportions are 0 or 1, and the hardwood count posteriors



(a) σ - spatial sd



(b) τ - indep sd

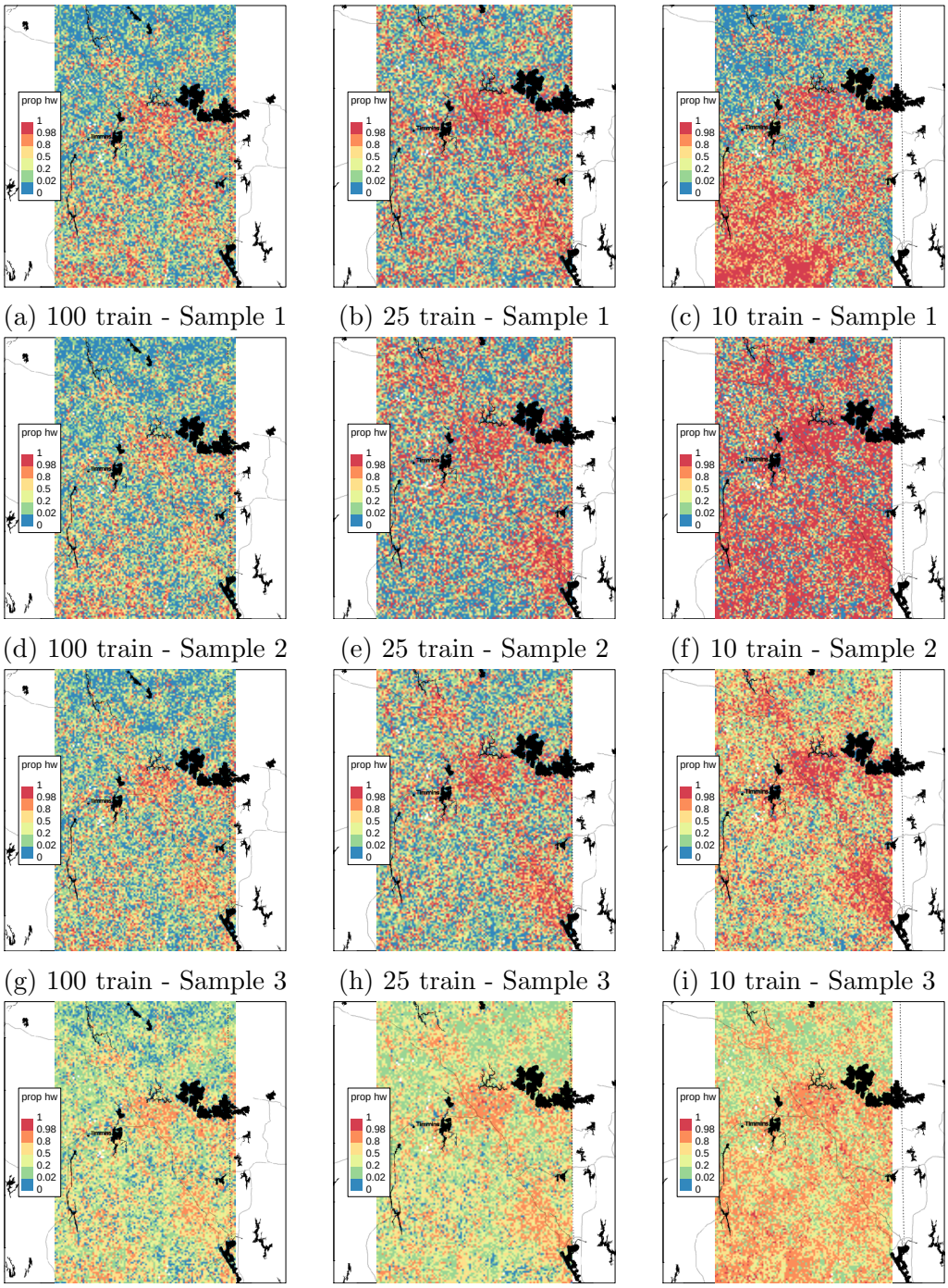


(c) ϕ (in km) - range

Figure 5.7: Priors and posteriors from the first simulation.

will not be symmetric. To illustrate this, Figure 5.9 shows the histograms of hardwood count posteriors for two validation plots where in one all are hardwoods and in the other none. We calculate the narrowest credible intervals for each validation plot, and compute their average coverages and widths as shown in Table 5.2.

The empirical coverage probabilities tend to exceed their theoretical values, mean-



(a) 100 train - Sample 1 (b) 25 train - Sample 1 (c) 10 train - Sample 1

(d) 100 train - Sample 2 (e) 25 train - Sample 2 (f) 10 train - Sample 2

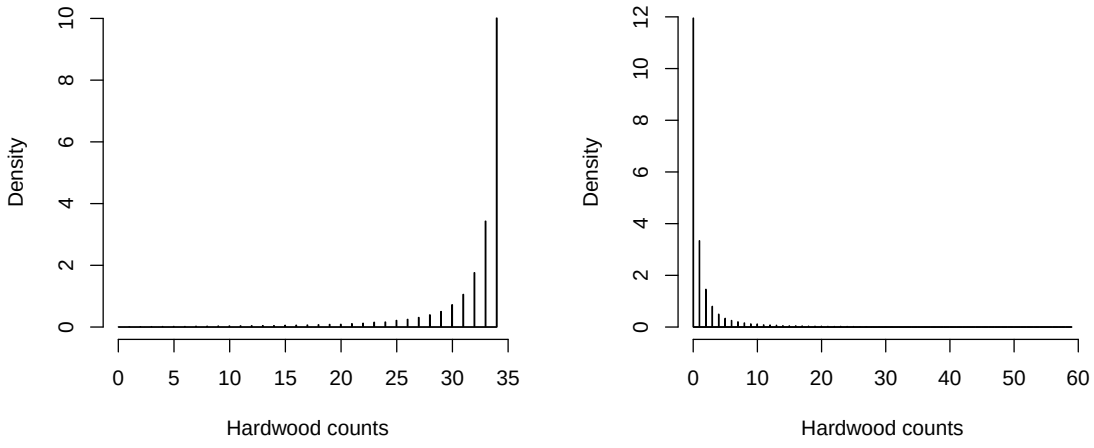
(g) 100 train - Sample 3 (h) 25 train - Sample 3 (i) 10 train - Sample 3

(j) 100 train - Sample mean (k) 25 train - Sample mean (l) 10 train - Sample mean

Figure 5.8: Three posterior samples of the hardwood proportion surface $p(s^*)$ along with their posterior means from different training data sizes (Background ©Stamen Design).

Table 5.2: Empirical Coverage of Posterior Credible Intervals and their Average Width. All results are averaged over 5 different simulations.

	Empirical Coverage of CI			Average CI Width		
#ofTrain	**95 %**	**80 %**	**50 %**	**95 %**	**80 %**	**50 %**
100 Sites	97 %	87 %	59 %	19.98	11.42	4.41
25 Sites	96 %	86 %	55 %	21.91	12.12	4.49
10 Sites	95 %	78 %	55 %	26.64	13.71	4.35



(a) All hardwoods, plot 37

(b) No hardwoods, plot 15

Figure 5.9: Posterior distributions of hardwood counts from two validation plots.

ing the intervals provided are on the conservative side. Overall, coverage probabilities are all at a desirable value. Table 5.2 also includes the average width of the posterior intervals, which shows *on average* wider intervals with fewer training data, as expected. Note that the slight decrease in the average width of the 50% posterior credible intervals with 10 training data may be due to monte carlo error.

Table 5.3: RMSE of predicted hardwood probabilities

	RMSE					Avg RMSE
#ofTrain	Sim 1	Sim 2	Sim 3	Sim 4	Sim 5	
100(BGLGM)	0.2276	0.2072	0.2096	0.1983	0.1569	0.1999
100(GLM)	0.2333	0.2060	0.2137	0.1969	0.1575	0.2015
25(BGLGM)	0.2284	0.2106	0.2335	0.1985	0.2268	0.2196
25(GLM)	0.2493	0.2256	0.2455	0.2005	0.2569	0.2356
10(BGLGM)	0.2912	0.2435	0.3639	0.2080	0.2472	0.2708
10(GLM)	0.3509	0.2796	0.4805	0.2357	0.3168	0.3327

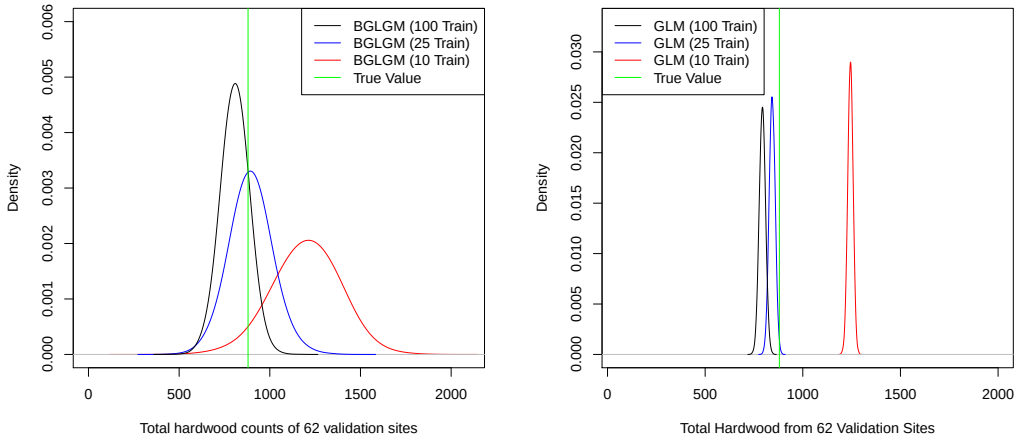
5.3.3 Comparison of BGLGM with Logistic Regression

In this section we will show the difference in performances between the BGLGM and a simple Logistic Regression. Fitting a Logistic Regression model to this dataset using the function `glm` in R is a frequentist way of analyzing this dataset, while BGLGM is a Bayesian approach. We will compare them both through their performance in point estimations via RMSEs, as well as their performance of distributions.

Table 5.3 is reporting the RMSEs of hardwood probabilities for the 62 validation sites, computed from runs with 100, 25, and 10 training data, for five different simulations. The RMSEs of BGLGM are calculated using posterior means. As it is observed, on average, RMSEs of BGLGM are smaller compared to Logistic Regression (GLM), indicating more accurate predictions. RMSEs increase with less ground truth data fitted to the model; verifying the results shown in the previous section.

To compare the predictive distributions of the GLM and BGLGM, we simulated 10,000 hardwood counts for each validation site using the estimated probabilities from GLM. These are compared to 10,000 MCMC posterior samples from the BGLGM using the distributions of the *total* hardwood counts from all 62 validation sites. Figure 5.10 is showing the corresponding distributions from BGLGM and GLM for only the

first simulation. The distributions from GLM are significantly narrower compared to the ones from BGLGM, as should be expected, since the GLM is ignoring errors in the parameter estimates. The BGLGM posterior distributions with all training data sizes capture the true value shown in green within their 95% intervals, while GLM with 10 and even 100 training data points fails to do so. In addition, from Figure 5.10a, we also observe that the posterior distributions become wider with less training data as expected. In conclusion, the BGLGM is a more reliable method compared to the GLM, in terms of both prediction accuracy and the ability of explaining uncertainties. Note that this process has been repeated for four other simulations with figures shown in the Appendix.

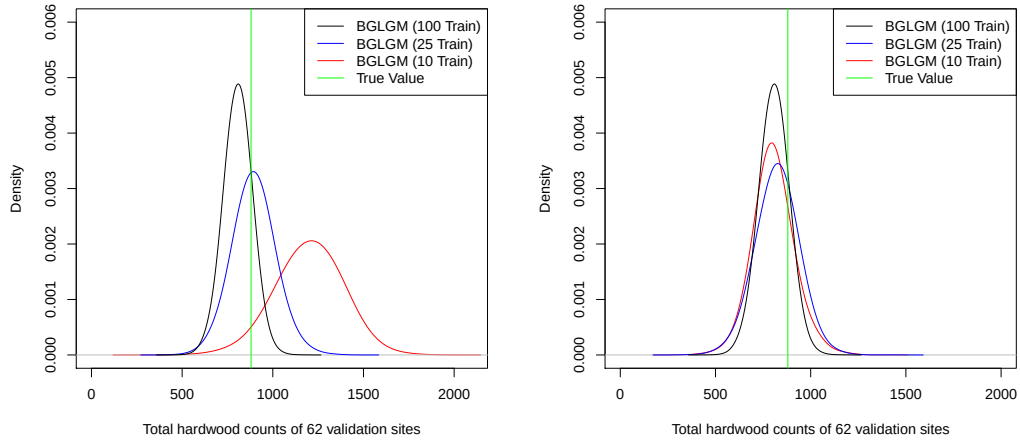


(a) BGLGM (b) GLM

Figure 5.10: Comparing BGLGM posterior distributions of total number of hardwood trees to the frequentist distributions from GLM.

Overall, a Bayesian approach is more reliable compared to a frequentist approach, since more types of uncertainty are taken into account. The simple Logistic Regression has artificially narrow prediction intervals, while BGLGM includes the true value within its 95% intervals for this dataset.

Stratified Sampling of training data



(a) Random (b) Stratified

Figure 5.11: Comparing random vs stratified sampling for total hardwood posterior distributions.

Figure 5.11 compares the posterior distributions of the total hardwood trees from both random sampling and stratified sampling on the first of five simulations. Prediction intervals from all five simulations are shown in Figure 5.12. The posterior distributions all contain the true value within their 95% posterior interval, however the uncertainty is generally less under stratified sampling in most cases. In Figure 5.11 it is notable that the stratified posterior with 10 training data contains the true value near its mode, while with the random posterior it is covered around the tail area.

In simulations 2 and 4 results are roughly comparable, while in simulation 3 the stratified posterior with 10 data points captures the true value around its mode. On the other hand, in simulation 5, the stratified posterior with 10 data points becomes more dispersed while with 25 more narrow. Overall, the stratified sampling approach shows only *potential* in improving results and thus may not be of significant improve-

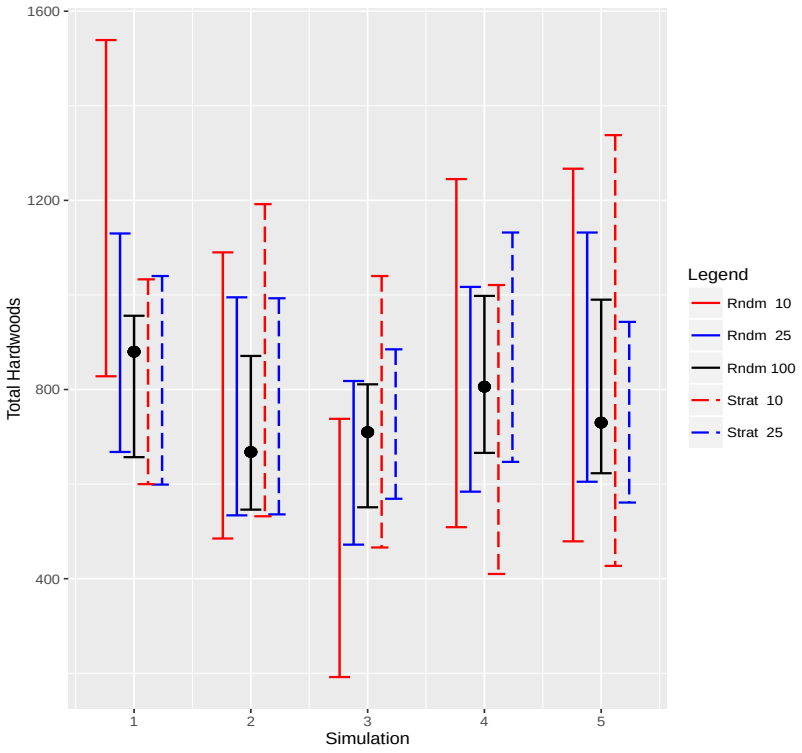


Figure 5.12: 95% posterior intervals of Random sampling vs Stratified sampling from all five simulations.

ments.

More details on the MCMC trace plots and the posterior distributions of each model parameter for the stratified sampling is included in the Appendix (for all simulations).

5.4 Discussion

In this chapter, we have analyzed the spatial data from the Timiskaming and Abitibi River forests in Ontario, Canada. We have studied the prediction of hardwood tree counts from elevation and vegetation index. We implemented a bespoke MCMC algorithm for posterior simulation of a Bayesian Generalized Linear Geostatistical Model (BGLGM), in order to make spatial predictions for new sites in the forests. The

bespoke MCMC performed well with this dataset while the general purpose "PrevMap" package struggled. We compared the Bayesian model with the frequentist Logistic Regression model. Although the dataset is imbalanced and contains many zero hardwood counts, the Bayesian approach provided unbiased estimates with reasonable uncertainties, while the overly simplistic Logistic Regression underestimated the uncertainty associated with the predictions. More importantly, with ground truth data as small as 10 points, BGLGM captured the true value of hardwood tree counts within its 95% posterior intervals, while the Logistic Regression failed even with 100 training points. This suggests with fewer ground truth data collected and hence reduction in expenses, good estimates of hardwood counts are still present and can capture the true value but perhaps with more uncertainties involved. This result is fairly important in terms of saving time and money for companies to gather such data.

Furthermore, a stratified sampling strategy of choosing the subset training data showed potential improvements in terms of predictions and uncertainties. However, these improvements are not always guaranteed.

As future work, one can further extend this model for multiple forests, where forests with similar features are considered to have high correlation indicated within priors and hence facilitate future spatial predictions for similar forests. This will significantly help reduce the redundant collection of data from similar forests.

Chapter 6

Conclusion

6.1 Summary

Bayesian methods are essential for exploring the underlying models through probability distributions and uncertainties. However computing the corresponding probability distributions (posterior distributions) are infeasible in most applications and hence require computational methods such as Markov Chain Monte Carlo (MCMC) methods to draw samples from such distributions. In this thesis, we have tackled two different Bayesian inference problems via MCMC methods. Our contributions involved both methodology and application aspects.

In Chapter 2, we briefly reviewed the fundamentals of Bayesian modelling and discussed the foundations of MCMC methods along with the descriptions of the most common ones used in this thesis.

In Chapter 3, we addressed the computational challenges of Bayesian inference caused by large-scale data, and introduced a divide and conquer MCMC method that splits tasks among different machines. The *Likelihood Inflating Sampling Algorithm (LISA)*, significantly reduces computational costs by randomly partitioning

the dataset into smaller subsets and running MCMC methods *independently* in parallel on each subset using different processors. Each processor runs an MCMC chain that samples from sub-posterior distributions which are defined using an "inflated" likelihood function. The draws from all processors are then combined in a way to provide approximate full posterior draws. We derived theoretical results showing that LISA's sub-posterior distributions are asymptotically equivalent to the full posterior distribution while the sub-posteriors of its competing method, the popular Consensus Monte Carlo (CMC), are asymptotically over-dispersed. Keeping this into account, we showed with a Bernoulli distributed data, LISA with uniform weights (uniformly choosing draws from all processors) outperforms CMC. However, our main contribution was to examine the performance of LISA on the complex nonparametric regression model, the Bayesian Additive Regression Trees (BART) model, where CMC fails. We developed a different strategy for combining the draws of LISA from all processors to study the full posterior of BART. We proved theoretically that with a minor modification on LISA's draws, we are able to generate full posterior draws of BART by simply taking a weighted average. Using simulated and real datasets, LISA showed superior performance in terms of accuracy and efficiency compared to CMC in the widely used nonparametric regression model BART, where computations were performed on the GPC supercomputer at the SciNet HPC Consortium.

In Chapter 4, we successfully examined LISA on BART with efficient Metropolis-Hastings (MH) proposals introduced by Pratola (2016). Pratola (2016) discussed that the existing MCMC methods for BART suffer from poor mixing and hence proposed new MH proposals that efficiently searches the tree space in BART. The performance of LISA on BART using the new MH proposals were consistent with the results previously presented in Chapter 3 and in fact shows that both LISA and the new efficient MCMC for BART can create a powerful combination that will significantly

help reduce poor mixing and hence prevent overfitting issues.

Finally in Chapter 5, we analyzed a spatial forestry dataset provided by the First Resource Management Group Inc. from the Timiskaming & Abitibi River forests in Ontario, Canada. We built a Bayesian Geostatistical model to predict the proportion of hardwood trees from elevation and vegetation index, and implemented an MCMC method for posterior simulations. We compared our results with the simple Logistic Regression model without spatial effect. As collecting ground truth data is very costly, we have analyzed the trend of predictions with fewer data fitted to the model. Our analysis suggest that with data as small as 10 points, the Bayesian Geostatistical model generates unbiased estimates of hardwood counts with reasonable estimates of uncertainty, while the Logistic Regression underestimates these uncertainties. Furthermore, we examined that by stratifying the subsets of spatial points, the posterior distributions show potential improvements, however do not always guarantee more accurate predictions. Our analysis help reduce expenses of collecting ground truth data, as the statistical model presented in this thesis shows the ability to generate reasonable predictions with fewer data.

6.2 Future Work

In this thesis we studied the theoretical concepts behind performing Bayesian inference on large-scale data, and constructed the formulation of batch-posteriors that guided us to develop LISA. With the asymptotic theoretical justifications, we proposed to use uniform weights to combine LISA's draws from all processors and successfully examined this on a Bernoulli distributed data. The intuition of using uniform weights for LISA can also be justified as future work by showing if the geometric and arithmetic means of LISA's unnormalized sub-posterior densities are approxi-

mately equal, i.e. the full posterior can be approximately seen as a mixture of LISA's sub-posteriors. However, we discovered that this weighting scheme cannot be used in general for all models, specially complex models like BART, and thus is highly model-specific. An avenue for future work can be to theoretically explore a general weighting scheme to combine LISA's draws, and examine LISA on a wider range of models. One can also study and examine the behaviour of multimodal sub-posterior distributions in LISA. Additional future work can be pursued to adaptively find the best K (total number of batches) that will have the best performance or specify upper bounds on K that will ensure enough data is in each batch for the overall algorithm to perform well.

Appendix A

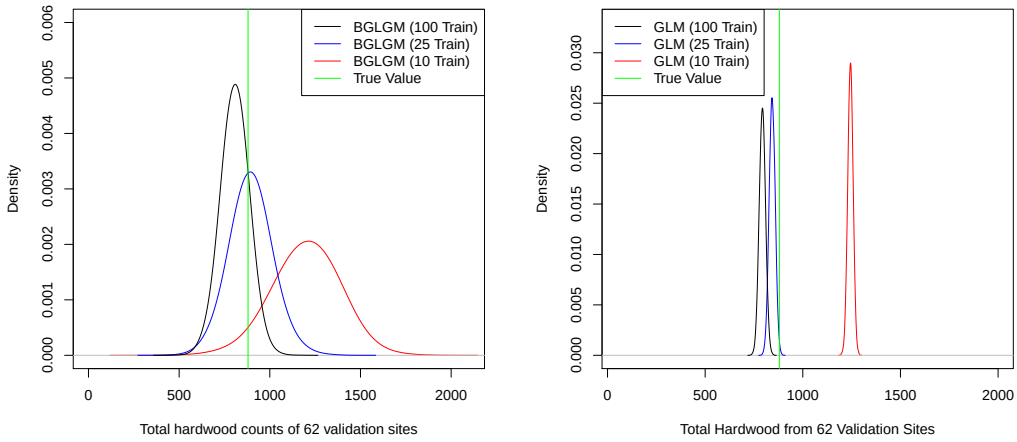
Appendix for Chapter 5

This Appendix includes the results from runs on five different simulations related to Chapter 5. Mainly the 162 dataset is randomly divided into 100 (training) and 62 (validation) sets, five times, resulting in five different simulations. For each simulation, we will then choose 25 and 10 data points from the 100, using random sampling and stratified sampling.

Section A.0.1 compares the total hardwood count posteriors (in validation set) from BGLGM and GLM for both random sampling and stratified sampling. Section A.0.2 presents the prior and posteriors of all model parameters from all runs. At last, Section A.0.3 contains the MCMC traceplots.

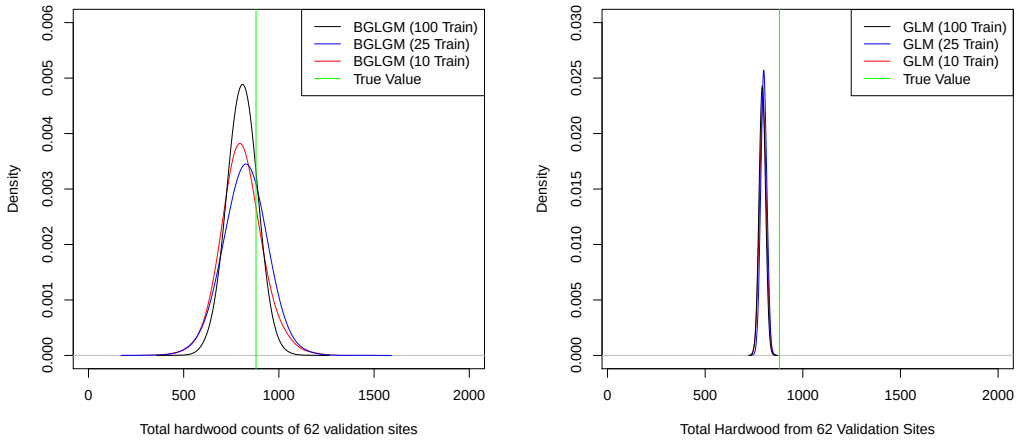
A.0.1 Total Hardwood Count Posteriors from BGLGM & GLM

Simulation 1 - Random & Stratified Comparisons



(a) BGLGM / Random

(b) GLM / Random

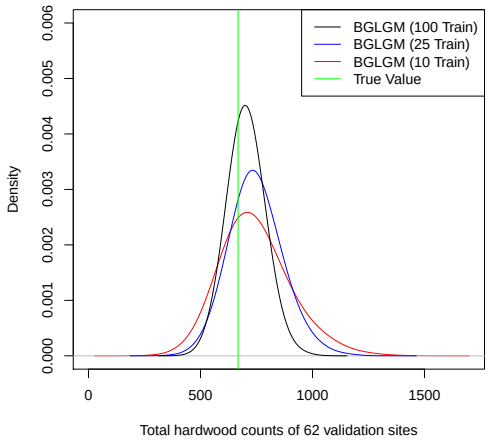


(c) BGLGM / Stratified

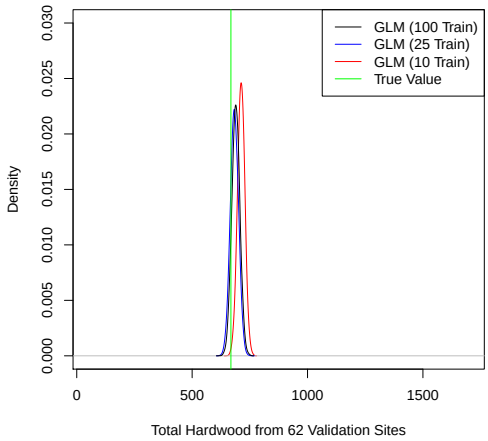
(d) GLM / Stratified

Figure A.1: Comparing BGLGM posterior distributions of total number of hardwood trees to the frequentist distributions from GLM - simulation 1.

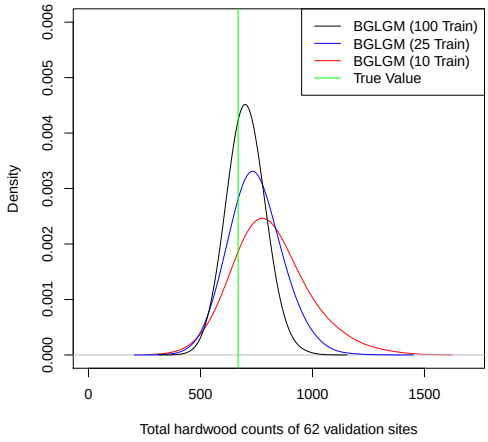
Simulation 2 - Random & Stratified Comparisons



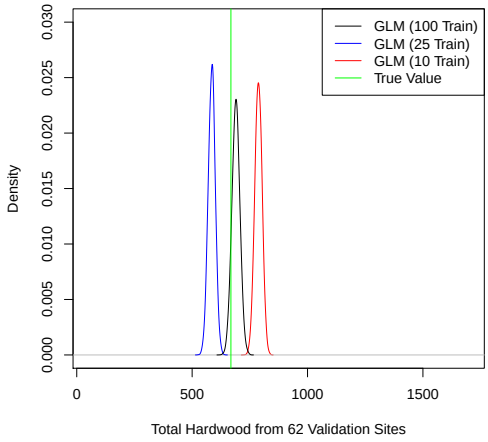
(a) BGLGM / Random



(b) GLM / Random



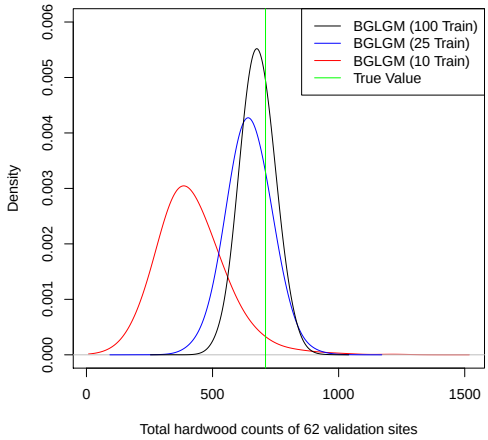
(c) BGLGM / Stratified



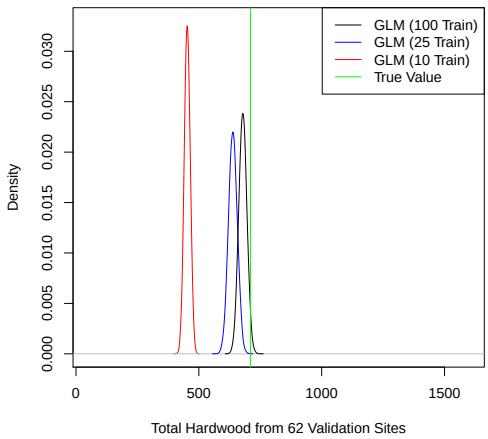
(d) GLM / Stratified

Figure A.2: Comparing BGLGM posterior distributions of total number of hardwood trees to the frequentist distributions from GLM - simulation 2.

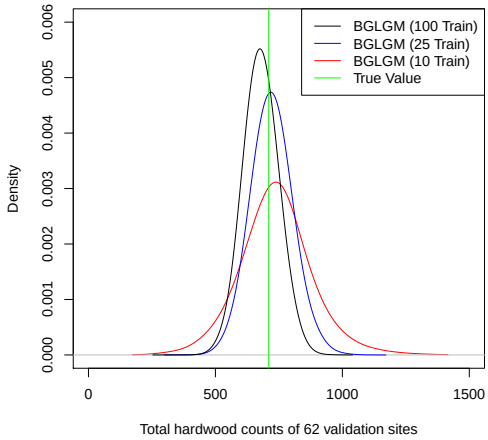
Simulation 3 - Random & Stratified Comparisons



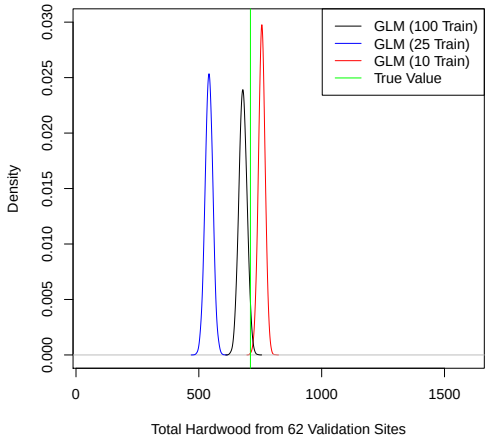
(a) BGLGM / Random



(b) GLM / Random



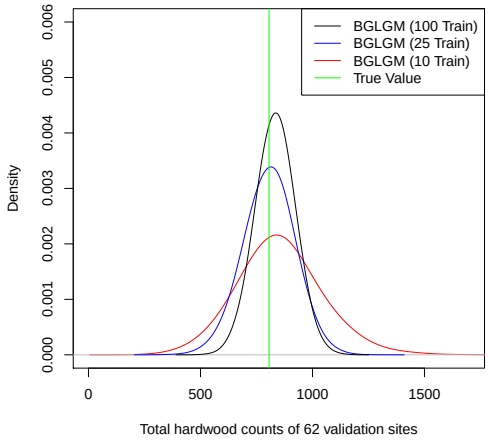
(c) BGLGM / Stratified



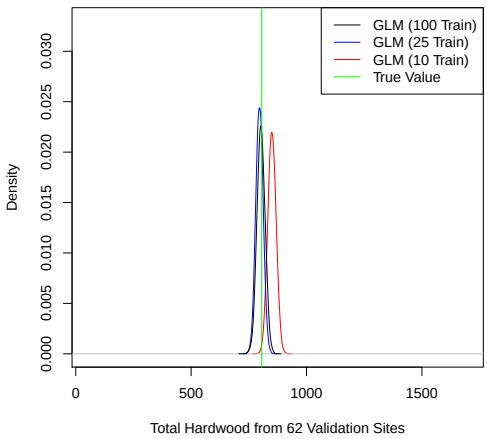
(d) GLM / Stratified

Figure A.3: Comparing BGLGM posterior distributions of total number of hardwood trees to the frequentist distributions from GLM - simulation 3.

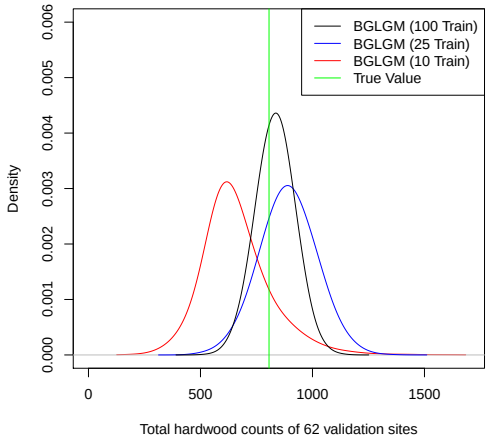
Simulation 4 - Random & Stratified Comparisons



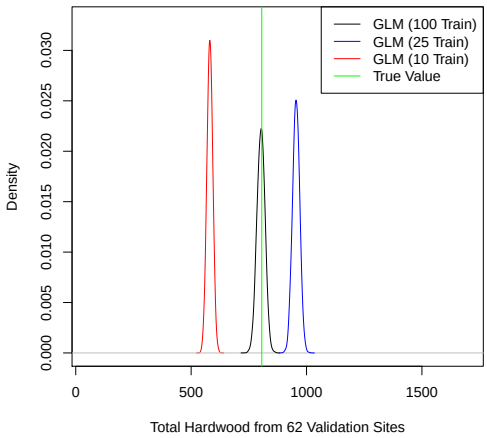
(a) BGLGM / Radnom



(b) GLM / Random



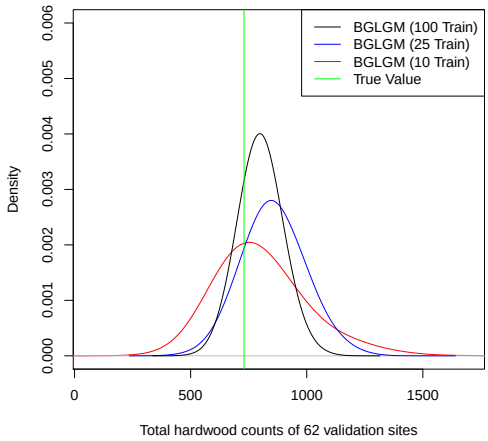
(c) BGLGM / Stratified



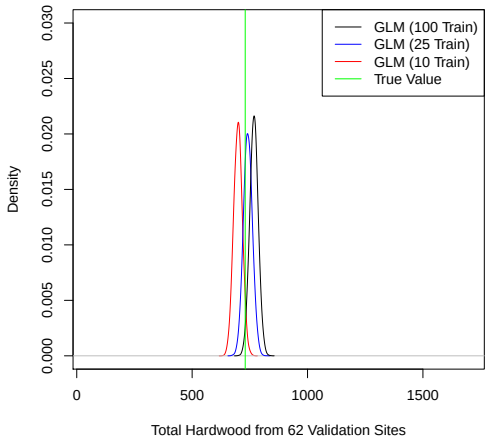
(d) GLM / Stratified

Figure A.4: Comparing BGLGM posterior distributions of total number of hardwood trees to the frequentist distributions from GLM - simulation 4.

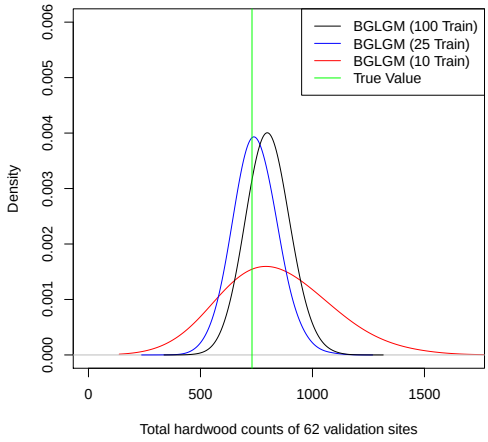
Simulation 5 - Random & Stratified Comparisons



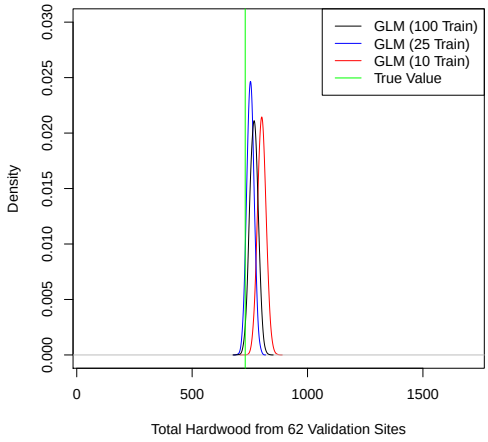
(a) BGLGM / Random



(b) GLM / Random



(c) BGLGM / Stratified

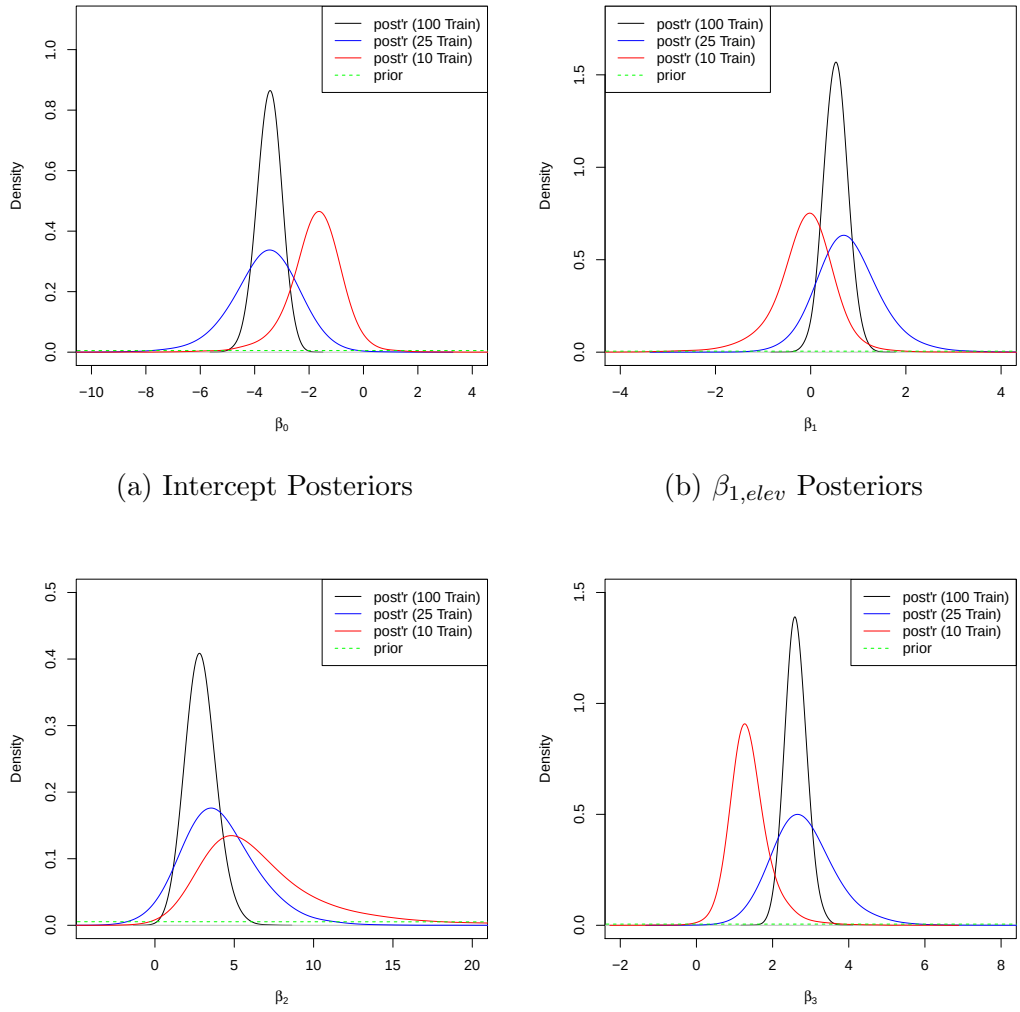


(d) GLM / Stratified

Figure A.5: Comparing BGLGM posterior distributions of total number of hardwood trees to the frequentist distributions from GLM - simulation 5.

A.0.2 Posterior & Prior of model parameters

Simulation 1 - Stratified sampling



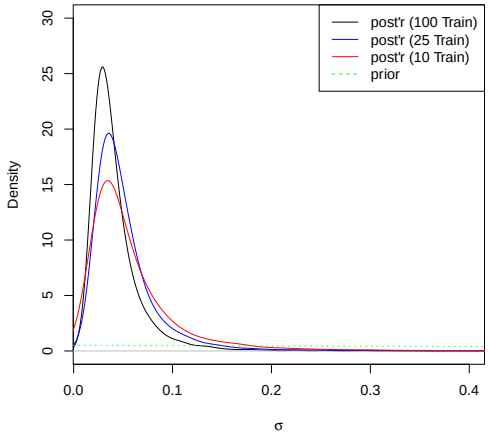
(a) Intercept Posteriors

(b) $\beta_{1,elev}$ Posteriors

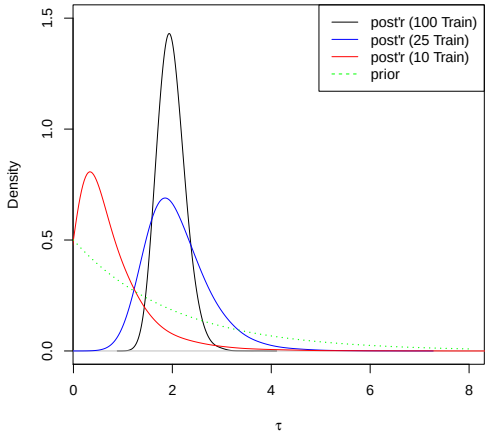
(c) $\beta_{2,veg-}$ Posteriors

(d) $\beta_{3,veg+}$ Posteriors

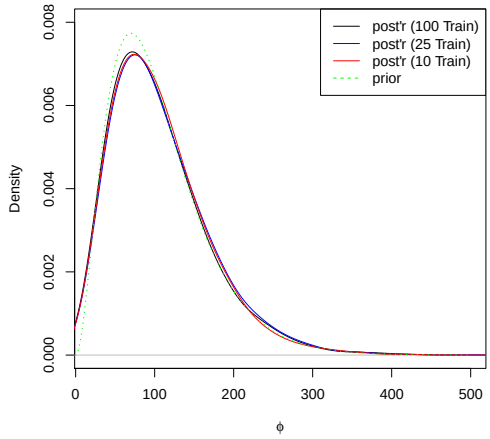
Figure A.6: Priors and posteriors of β's from simulation 1 - stratified sampling.



(a) σ Posteriors



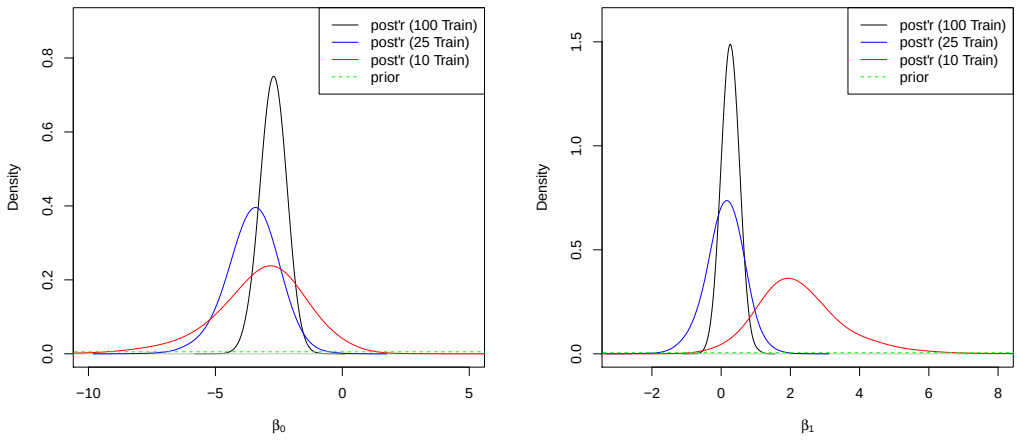
(b) τ Posteriors



(c) ϕ Posteriors

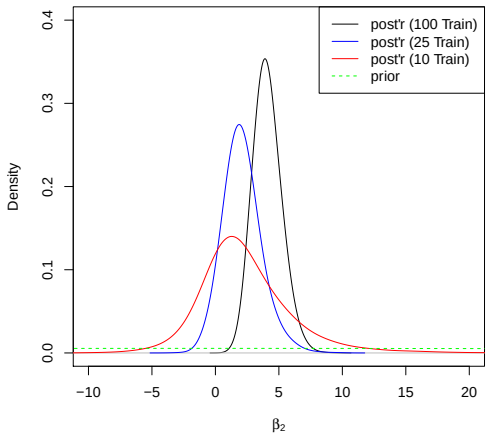
Figure A.7: Priors and posteriors of σ, τ, and ϕ from simulation 1 - stratified sampling.

Simulation 2 - Random sampling

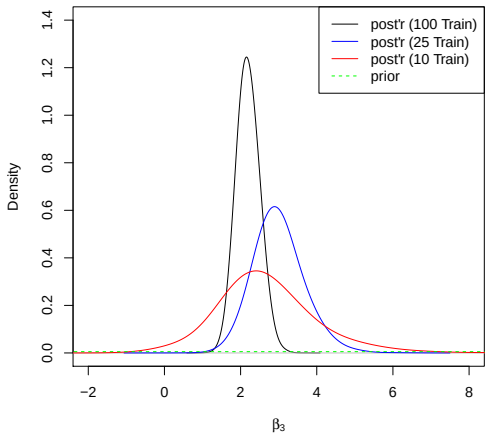


(a) Intercept Posteriors

(b) $\beta_{1,elev}$ Posteriors

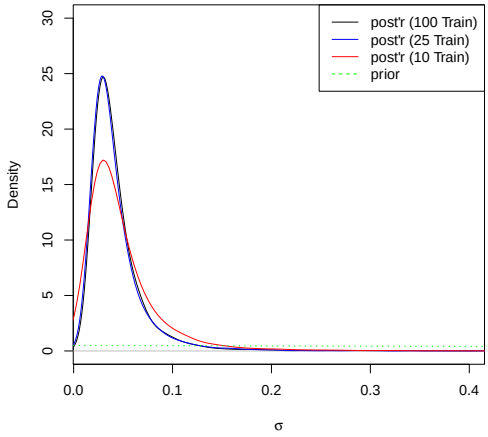


(c) $\beta_{2,veg-}$ Posteriors

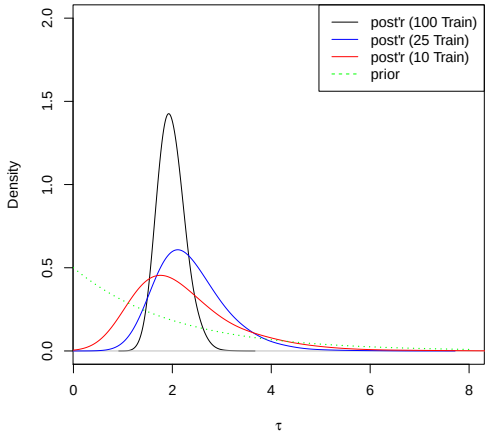


(d) $\beta_{3,veg+}$ Posteriors

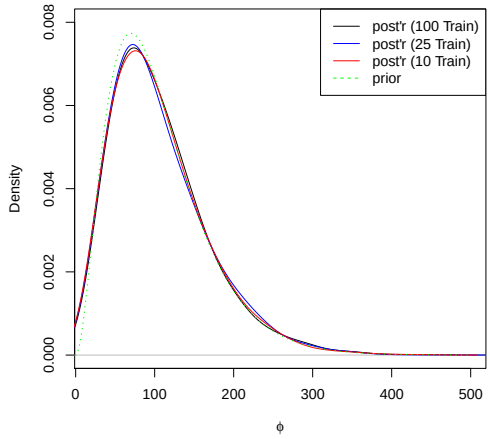
Figure A.8: Priors and posteriors of β's from simulation 2 - random sampling.



(a) σ Posteriors



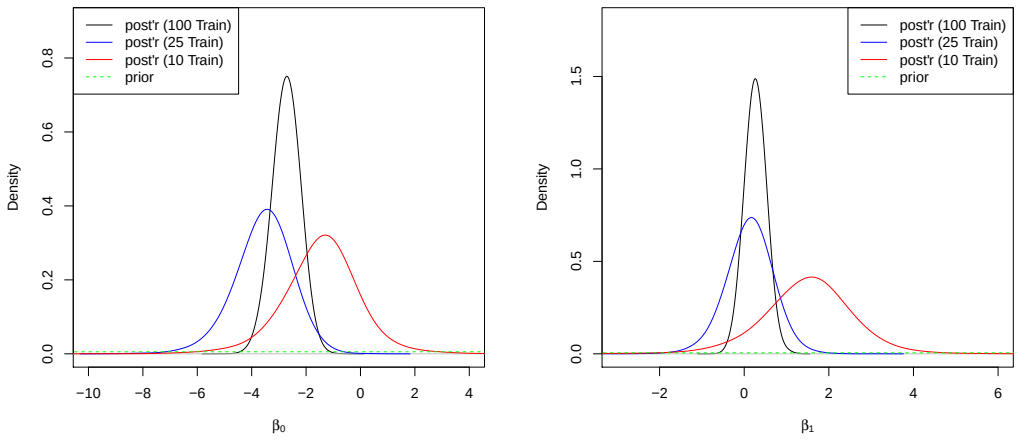
(b) τ Posteriors



(c) ϕ Posteriors

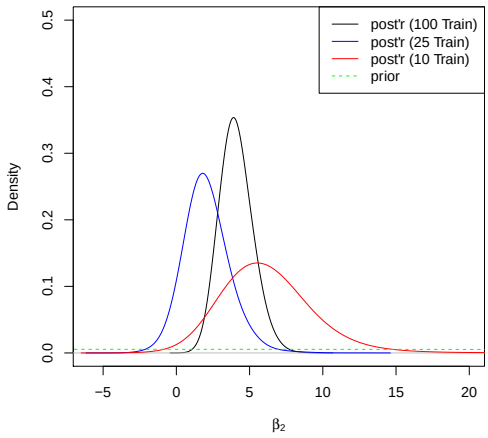
Figure A.9: Priors and posteriors of σ, τ, and ϕ from simulation 2 - random sampling.

Simulation 2 - Stratified sampling

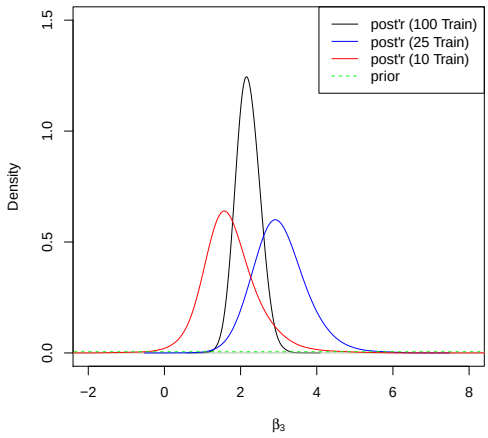


(a) Intercept Posteriors

(b) $\beta_{1,elev}$ Posteriors

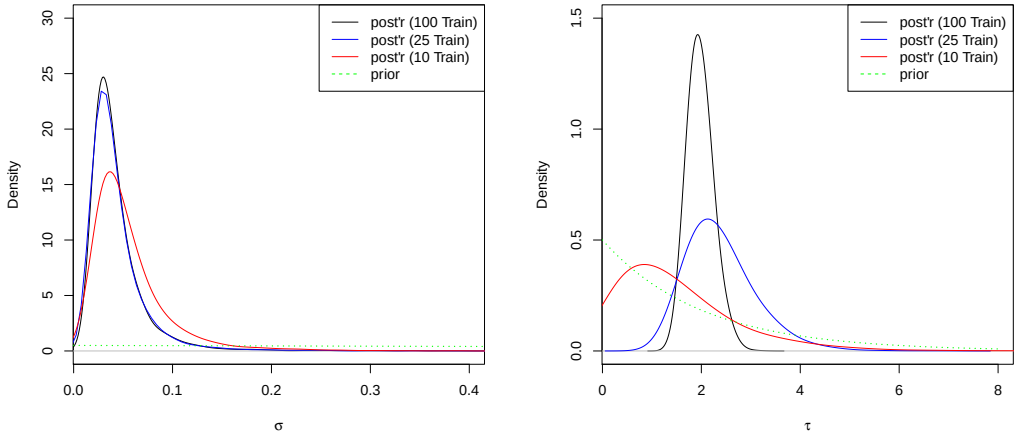


(c) $\beta_{2,veg-}$ Posteriors



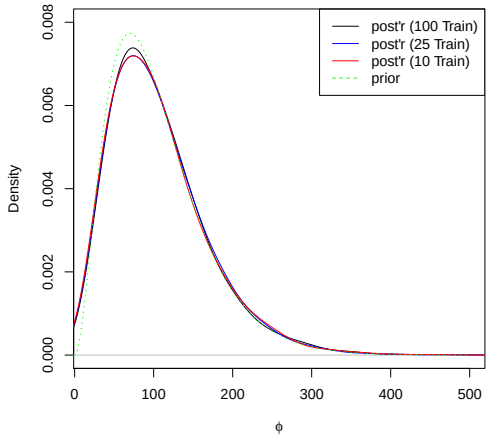
(d) $\beta_{3,veg+}$ Posteriors

Figure A.10: Priors and posteriors of β's from simulation 2 - stratified sampling.



(a) σ Posteriors

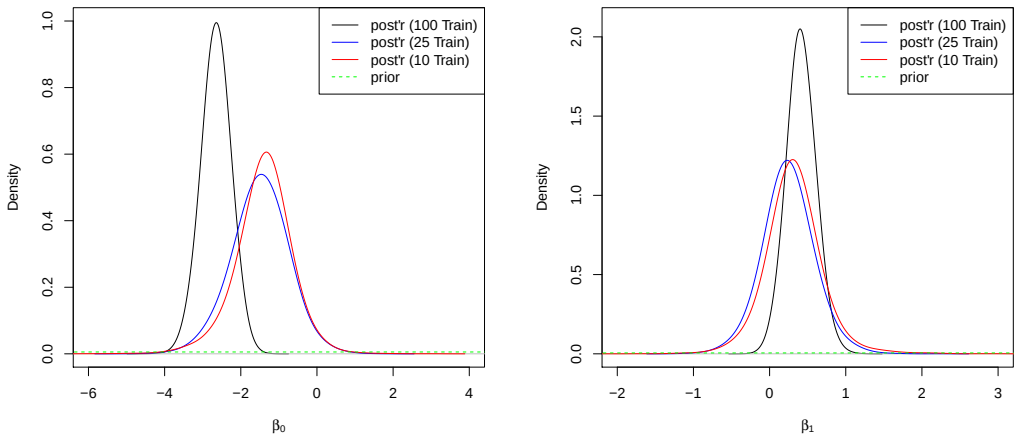
(b) τ Posteriors



(c) ϕ Posteriors

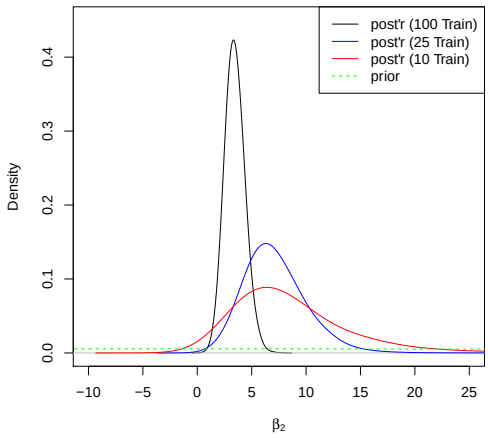
Figure A.11: Priors and posteriors of σ, τ, and ϕ from simulation 2 - stratified sampling.

Simulation 3 - Random sampling

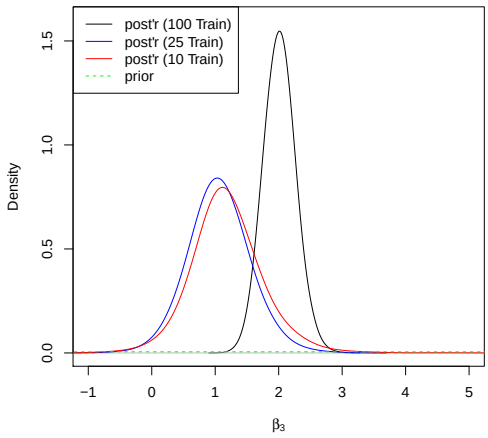


(a) Intercept Posteriors

(b) $\beta_{1,elev}$ Posteriors

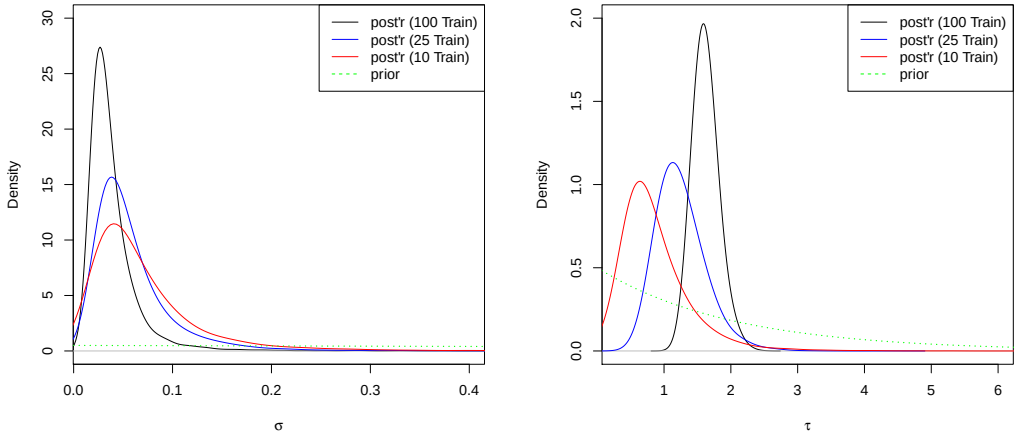


(c) $\beta_{2,veg-}$ Posteriors



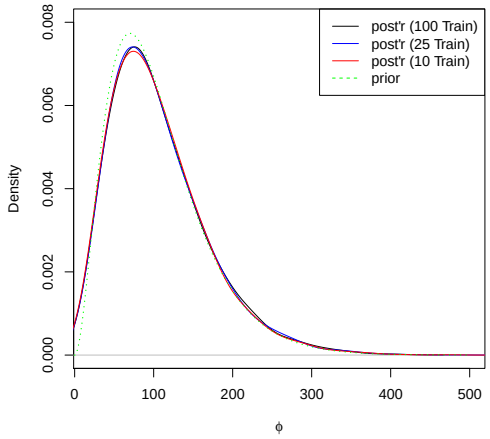
(d) $\beta_{3,veg+}$ Posteriors

Figure A.12: Priors and posteriors of β's from simulation 3 - random sampling.



(a) σ Posteriors

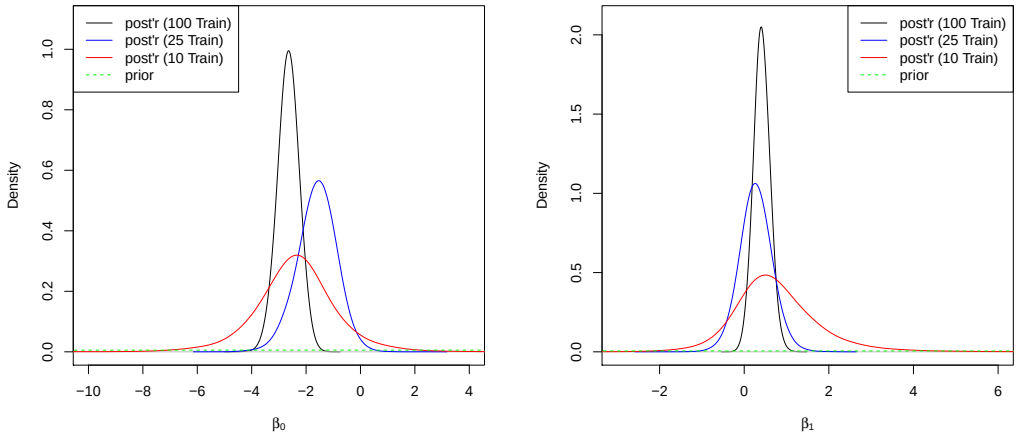
(b) τ Posteriors



(c) ϕ Posteriors

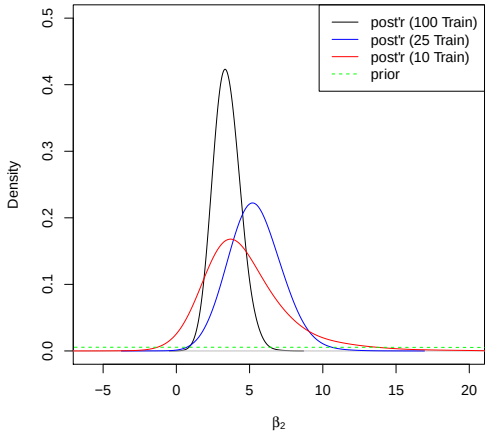
Figure A.13: Priors and posteriors of σ, τ, and ϕ from simulation 3 - random sampling.

Simulation 3 - Stratified sampling

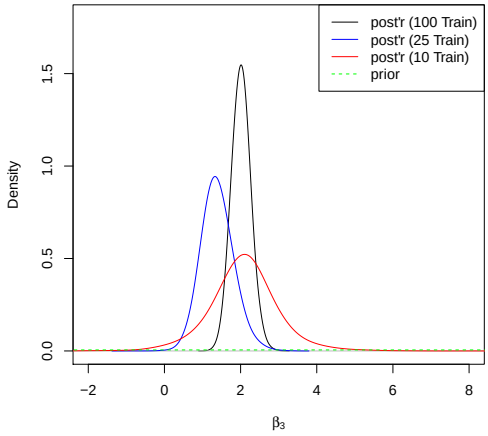


(a) Intercept Posteriors

(b) $\beta_{1,elev}$ Posteriors

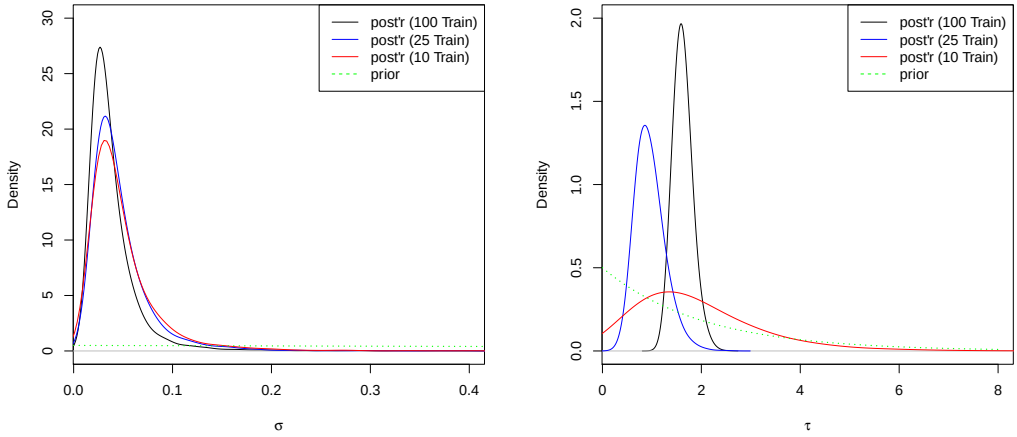


(c) $\beta_{2,veg-}$ Posteriors



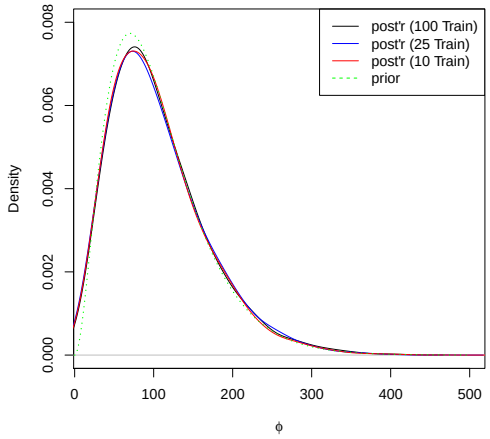
(d) $\beta_{3,veg+}$ Posteriors

Figure A.14: Priors and posteriors of β's from simulation 3 - stratified sampling.



(a) σ Posteriors

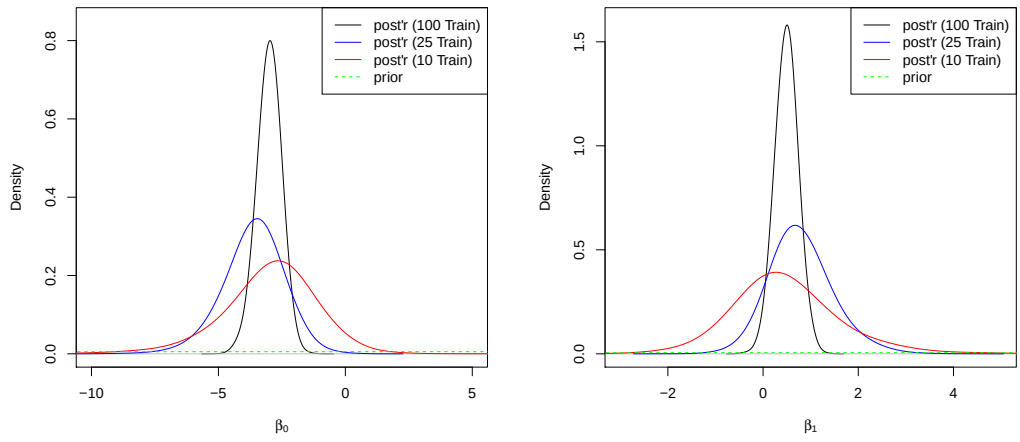
(b) τ Posteriors



(c) ϕ Posteriors

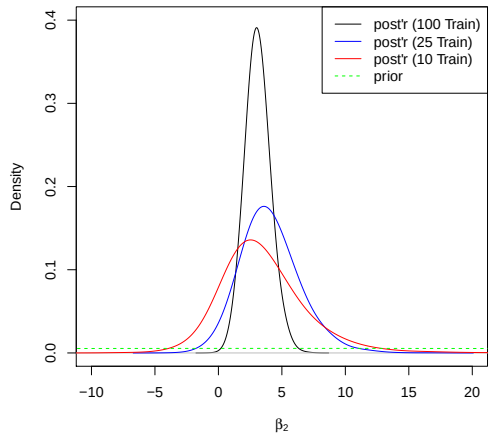
Figure A.15: Priors and posteriors of σ, τ, and ϕ from simulation 3 - stratified sampling.

Simulation 4 - Random sampling

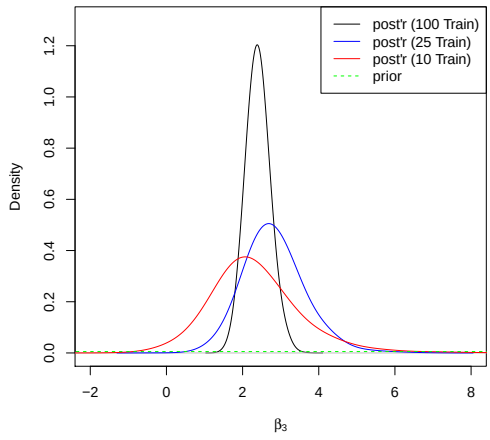


(a) Intercept Posteriors

(b) $\beta_{1,elev}$ Posteriors

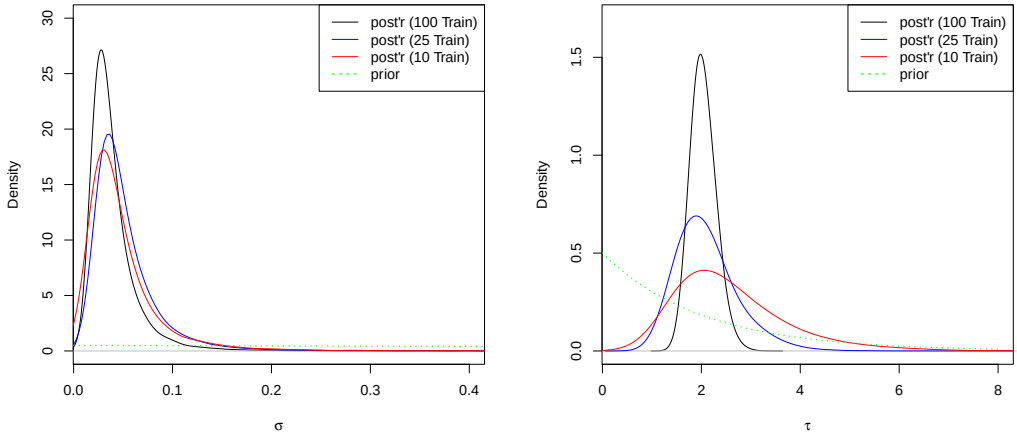


(c) $\beta_{2,veg-}$ Posteriors



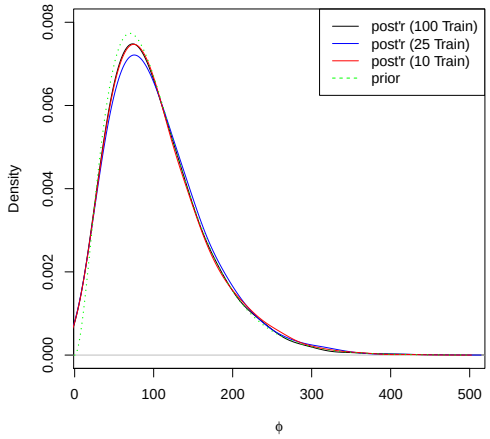
(d) $\beta_{3,veg+}$ Posteriors

Figure A.16: Priors and posteriors of β's from simulation 4 - random sampling.



(a) σ Posteriors

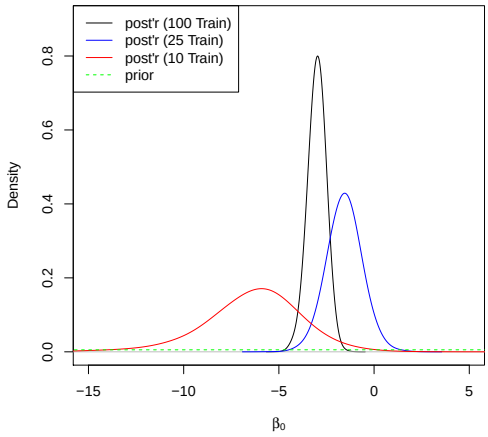
(b) τ Posteriors



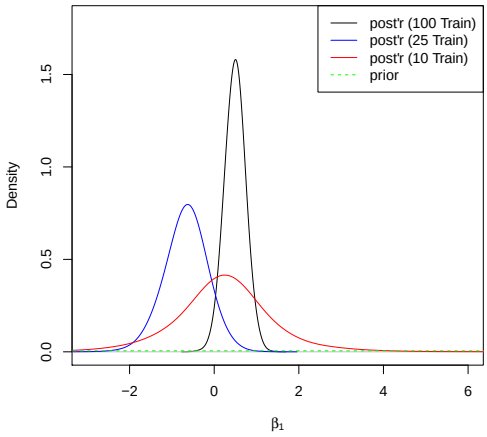
(c) ϕ Posteriors

Figure A.17: Priors and posteriors of σ, τ, and ϕ from simulation 4 - random sampling.

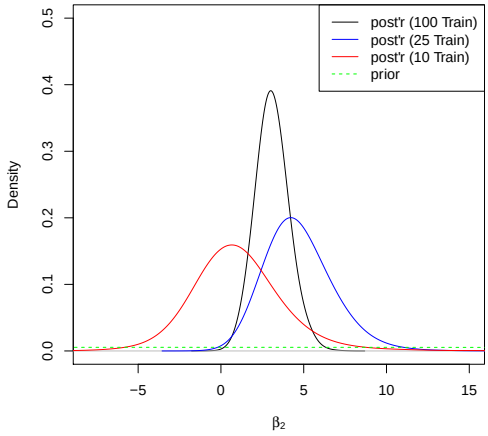
Simulation 4 - Stratified sampling



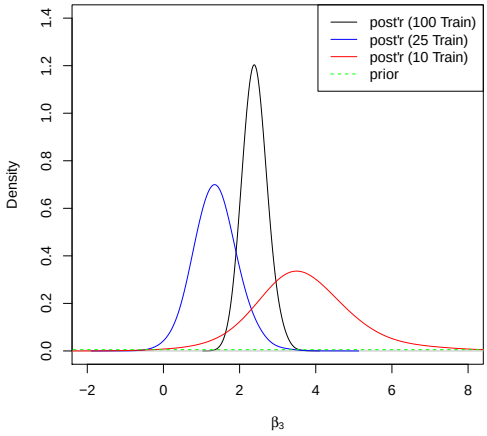
(a) Intercept Posteriors



(b) $\beta_{1,elev}$ Posteriors

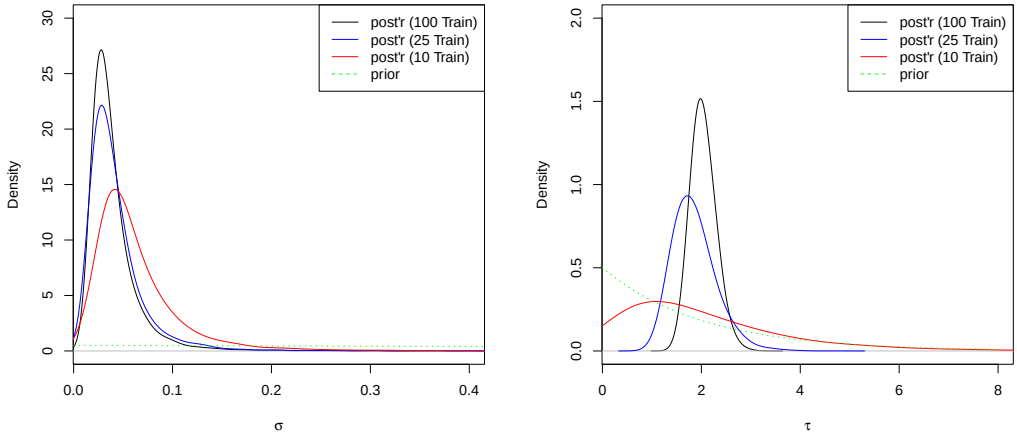


(c) $\beta_{2,veg-}$ Posteriors



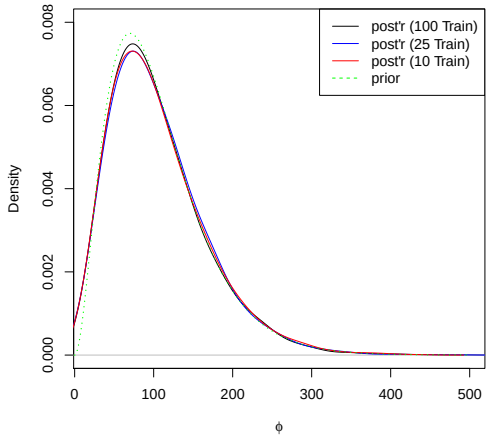
(d) $\beta_{3,veg+}$ Posteriors

Figure A.18: Priors and posteriors of β's from simulation 4 - stratified sampling.



(a) σ Posteriors

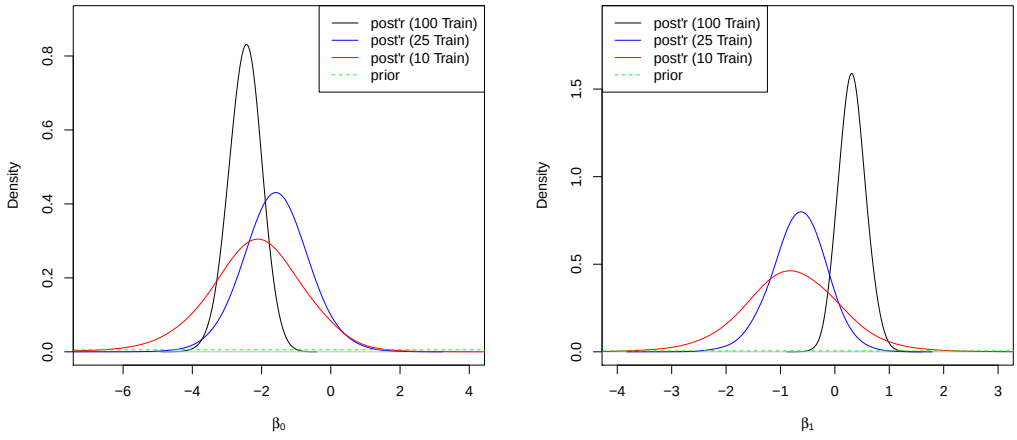
(b) τ Posteriors



(c) ϕ Posteriors

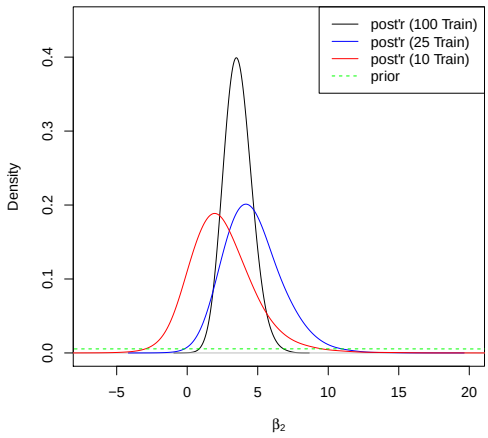
Figure A.19: Priors and posteriors of σ, τ, and ϕ from simulation 4 - stratified sampling.

Simulation 5 - Random sampling

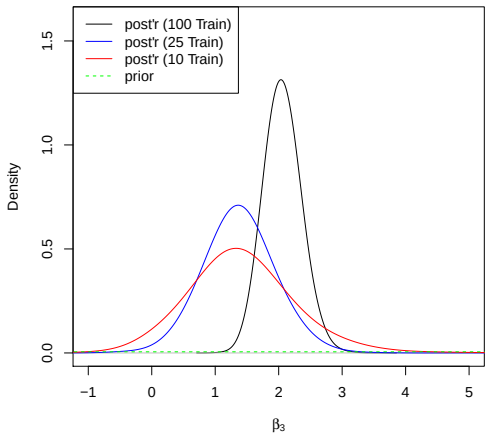


(a) Intercept Posteriors

(b) $\beta_{1,elev}$ Posteriors

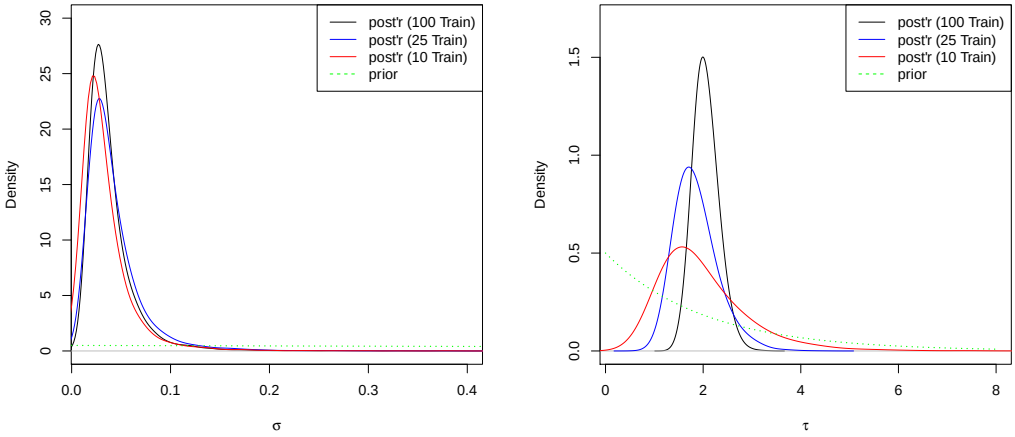


(c) $\beta_{2,veg-}$ Posteriors



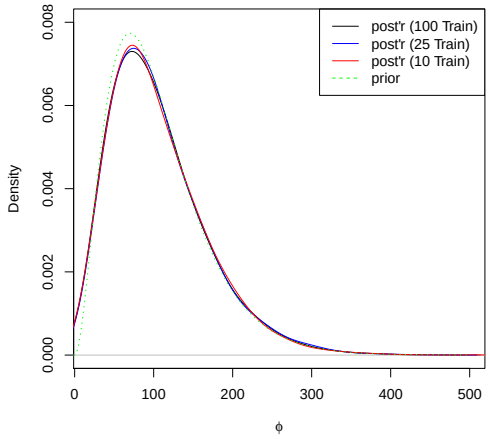
(d) $\beta_{3,veg+}$ Posteriors

Figure A.20: Priors and posteriors of β's from simulation 5 - random sampling.



(a) σ Posteriors

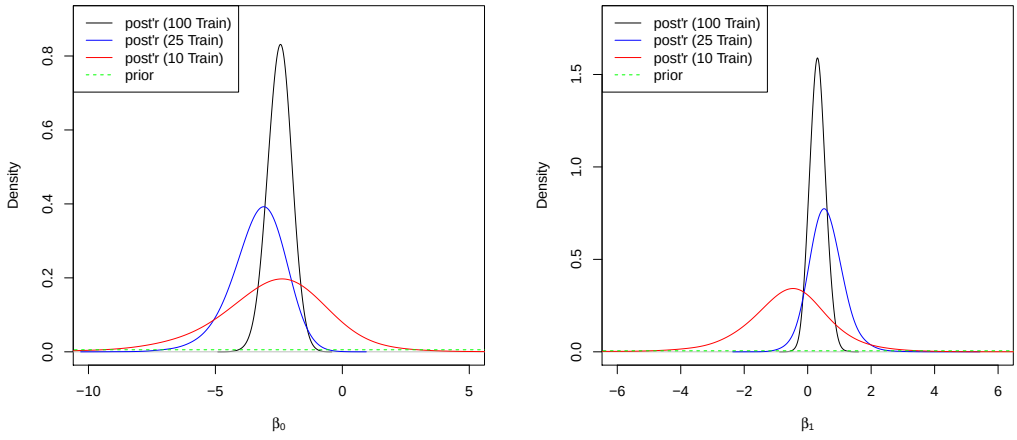
(b) τ Posteriors



(c) ϕ Posteriors

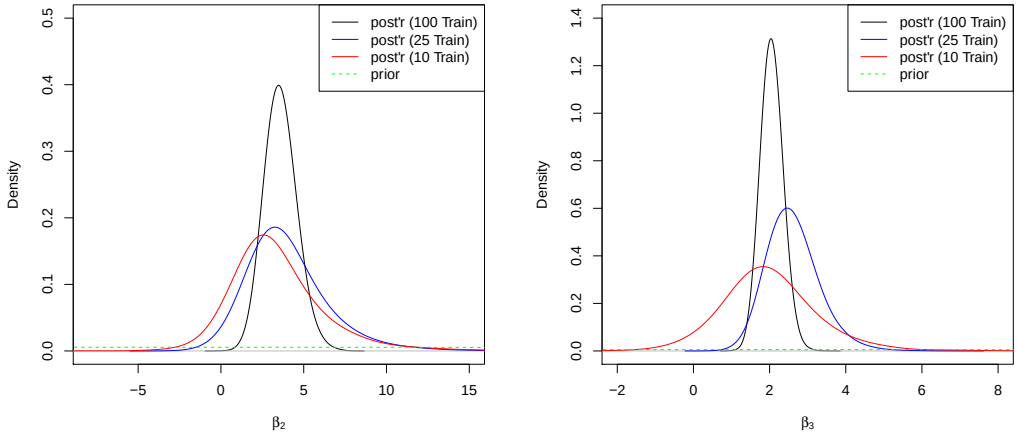
Figure A.21: Priors and posteriors of σ, τ, and ϕ from simulation 5 - random sampling.

Simulation 5 - Stratified sampling



(a) Intercept Posteriors

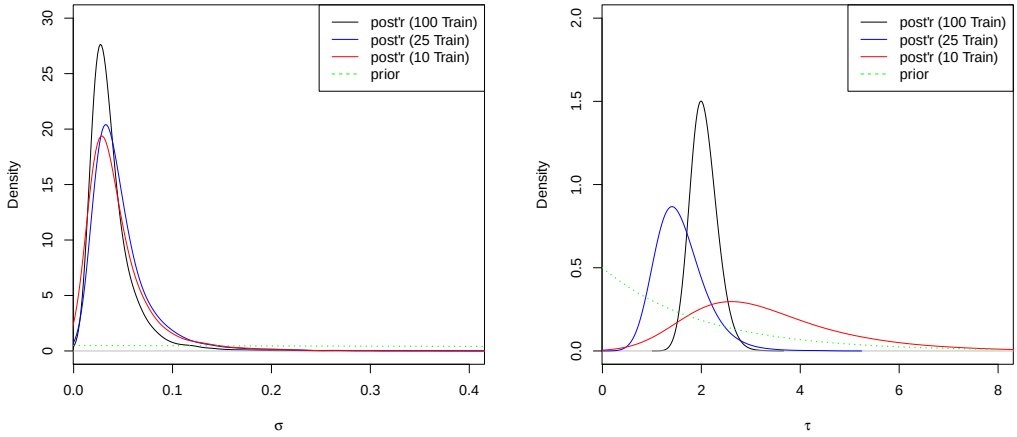
(b) $\beta_{1,elev}$ Posteriors



(c) $\beta_{2,veg-}$ Posteriors

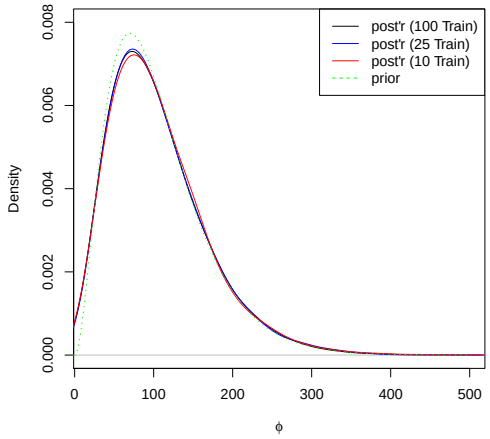
(d) $\beta_{3,veg+}$ Posteriors

Figure A.22: Priors and posteriors of β's from simulation 5 - stratified sampling.



(a) σ Posteriors

(b) τ Posteriors

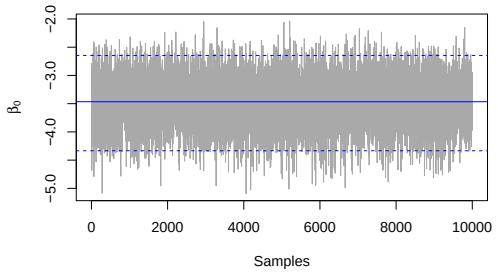


(c) ϕ Posteriors

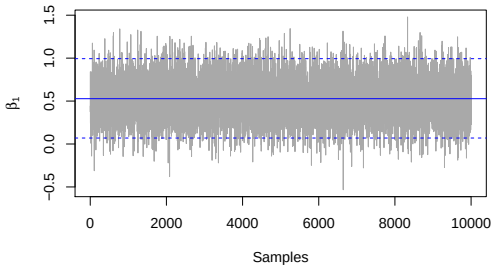
Figure A.23: Priors and posteriors of σ, τ, and ϕ from simulation 5 - stratified sampling.

A.0.3 MCMC Trace Plots

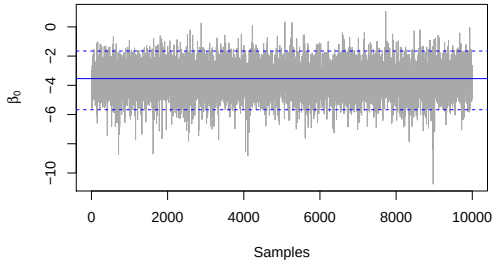
Simulation 1 - Random sampling



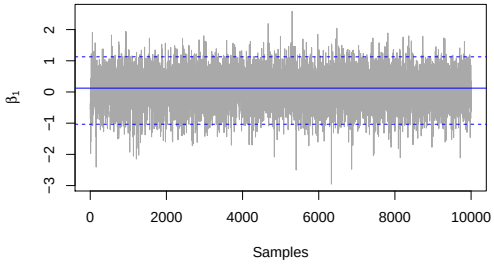
(a) 100 training sites — Intercept



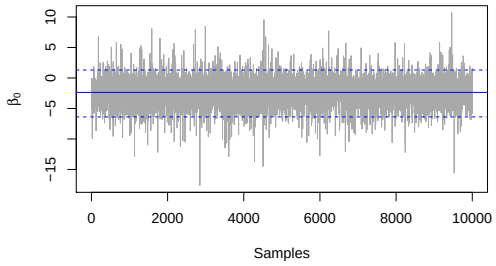
(b) 100 training sites — β_1



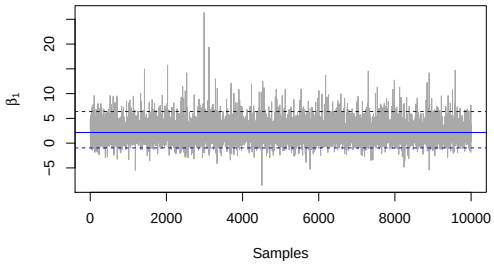
(c) 25 training sites — Intercept



(d) 25 training sites — β_1

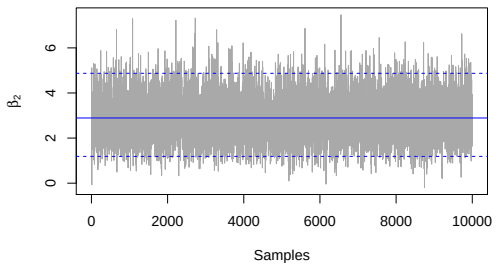


(e) 10 training sites — Intercept

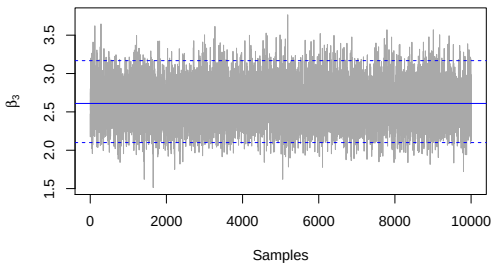


(f) 10 training sites — β_1

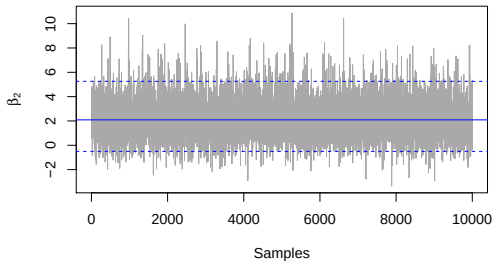
Figure A.24: MCMC Trace plots of β_0 and β_1 from simulation 1 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



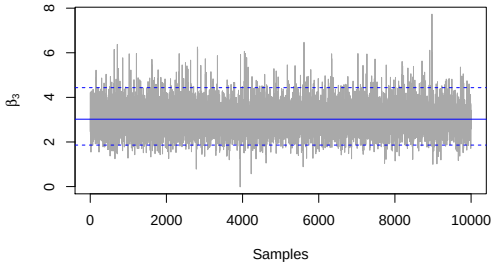
(a) 100 training sites — β_2



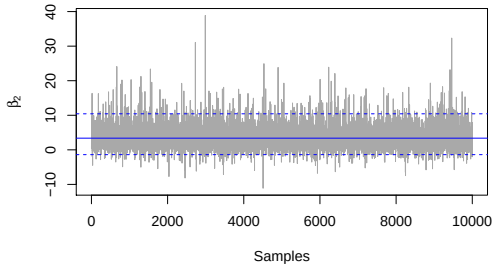
(b) 100 training sites — β_3



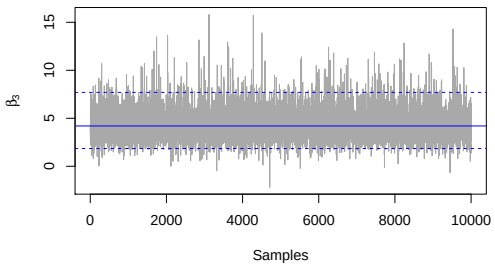
(c) 25 training sites — β_2



(d) 25 training sites — β_3

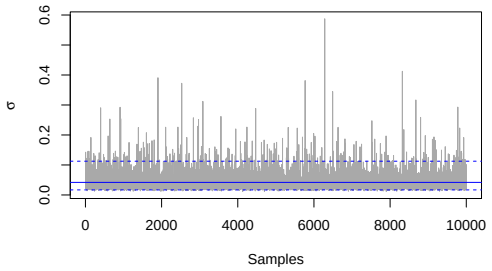


(e) 10 training sites — β_2

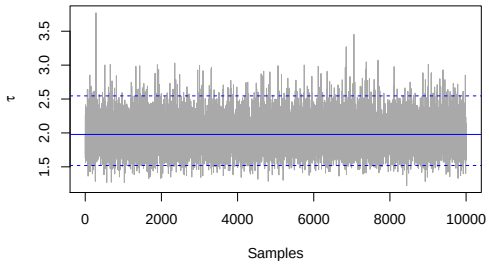


(f) 10 training sites — β_3

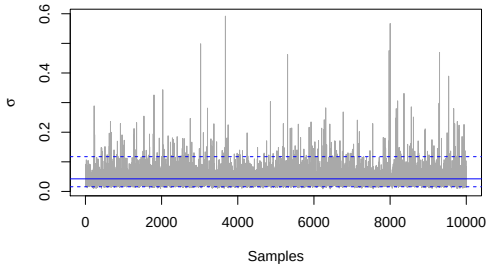
Figure A.25: MCMC Trace plots of β_2 and β_3 from simulation 1 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



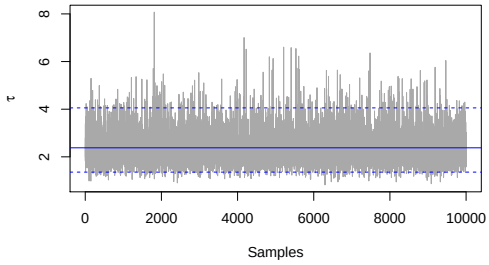
(a) 100 training sites — σ



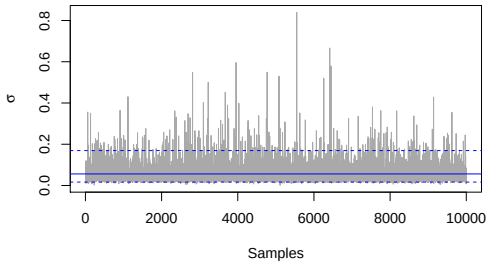
(b) 100 training sites — τ



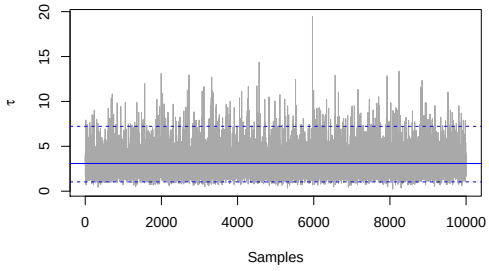
(c) 25 training sites — σ



(d) 25 training sites — τ

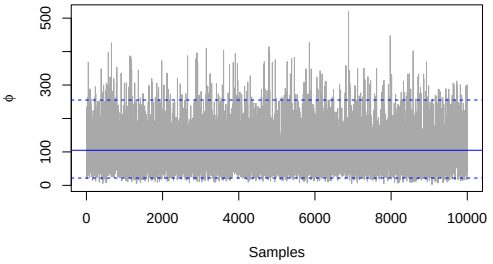


(e) 10 training sites — σ

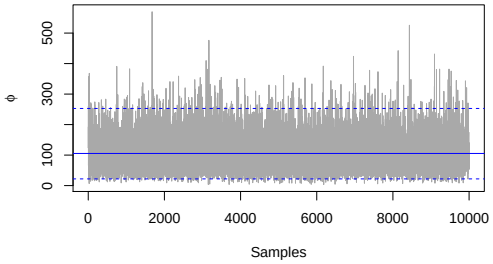


(f) 10 training sites — τ

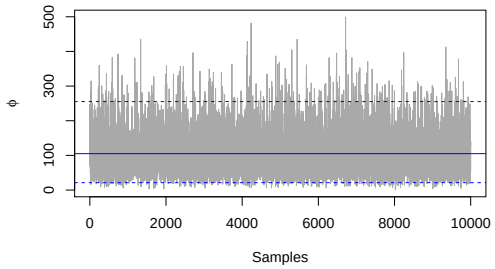
Figure A.26: MCMC Trace plots of σ and τ from simulation 1 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



(a) 100 training sites — ϕ



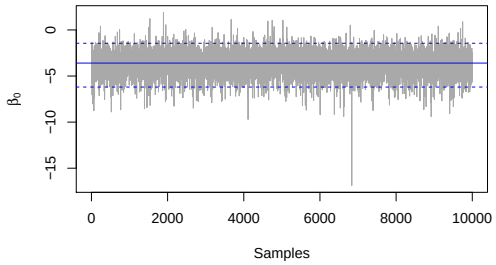
(b) 25 training sites — ϕ



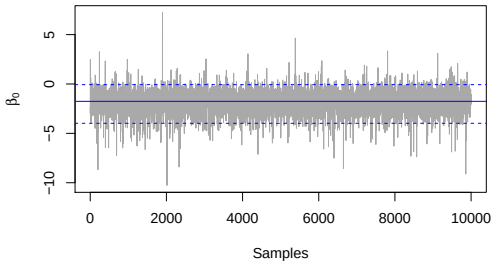
(c) 10 training sites — ϕ

Figure A.27: MCMC Trace plots of ϕ from simulation 1 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.

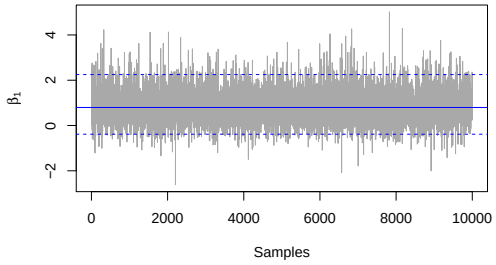
Simulation 1 - Stratified sampling



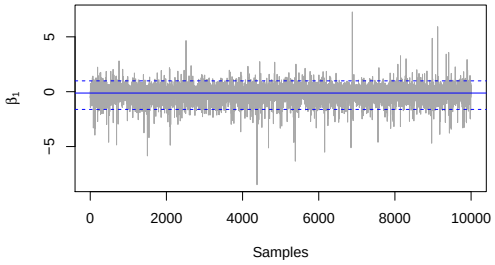
(a) 25 training sites — Intercept



(b) 10 training sites — Intercept

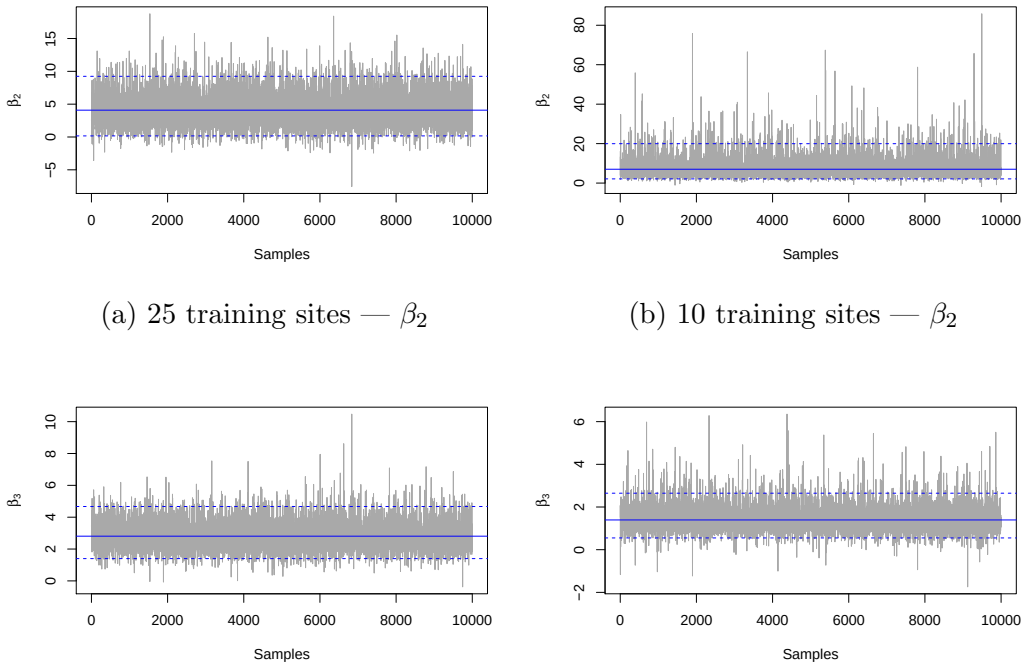


(c) 25 training sites — β_1



(d) 10 training sites — β_1

Figure A.28: MCMC Trace plots of β_0 and β_1 from simulation 1 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



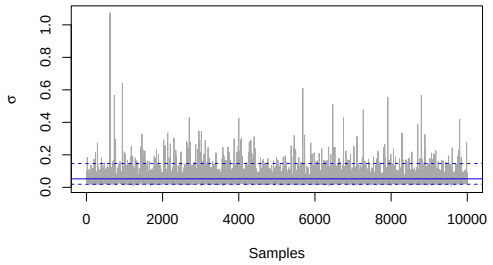
(a) 25 training sites — β_2

(b) 10 training sites — β_2

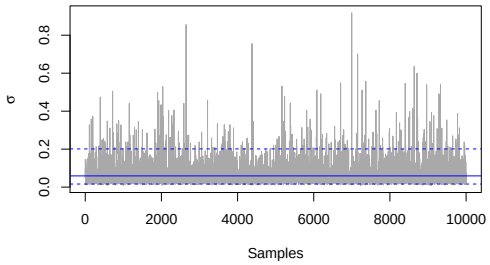
(c) 25 training sites — β_3

(d) 10 training sites — β_3

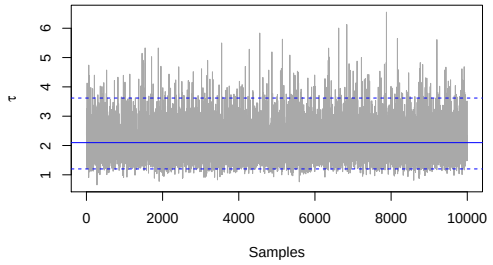
Figure A.29: MCMC Trace plots of β_2 and β_3 from simulation 1 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



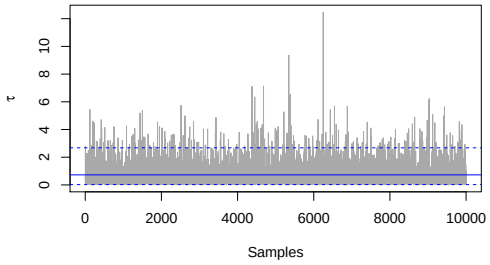
(a) 25 training sites — σ



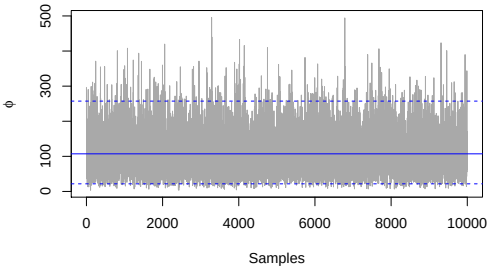
(b) 10 training sites — σ



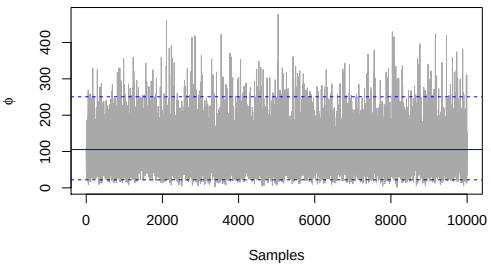
(c) 25 training sites — τ



(d) 10 training sites — τ



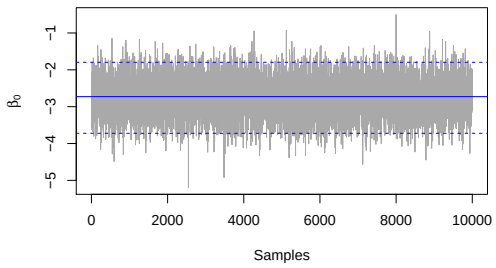
(e) 25 training sites — ϕ



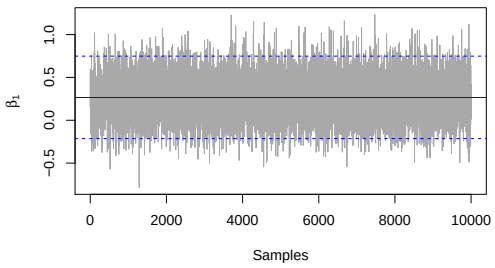
(f) 10 training sites — ϕ

Figure A.30: MCMC Trace plots of σ, τ and ϕ from simulation 1 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.

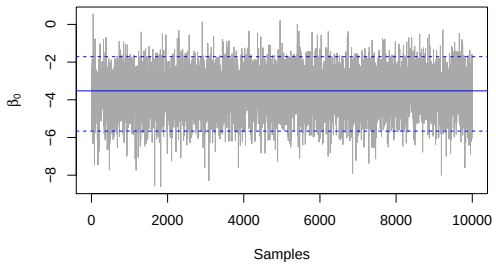
Simulation 2 - Random sampling



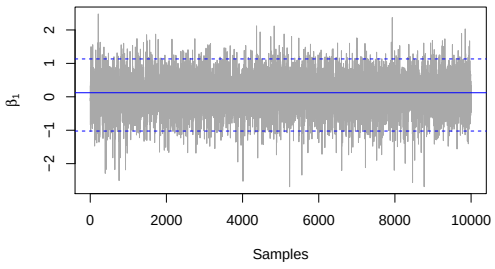
(a) 100 training sites — Intercept



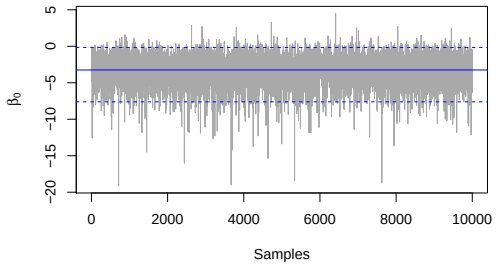
(b) 100 training sites — β_1



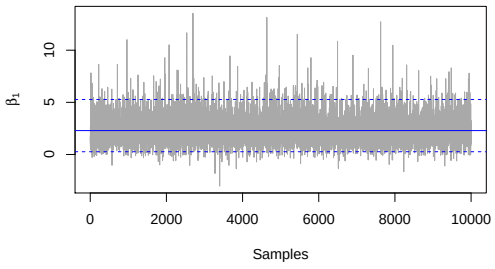
(c) 25 training sites — Intercept



(d) 25 training sites — β_1

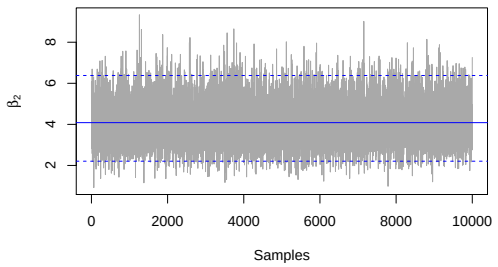


(e) 10 training sites — Intercept

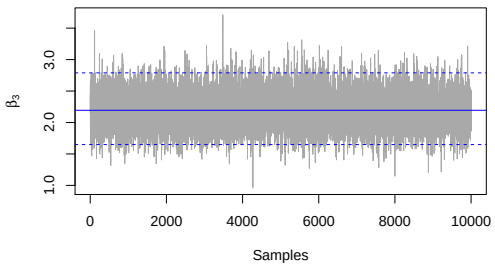


(f) 10 training sites — β_1

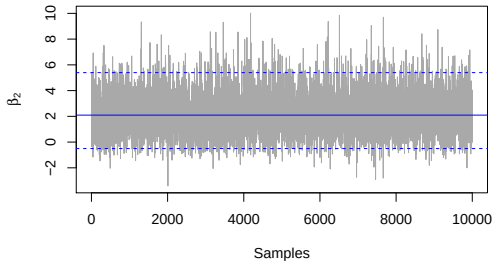
Figure A.31: MCMC Trace plots of β_0 and β_1 from simulation 2 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



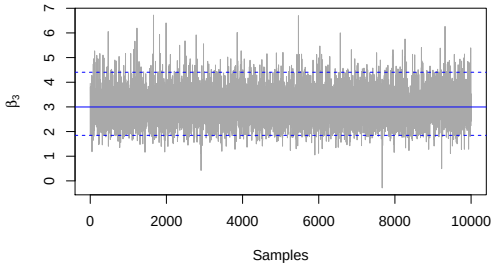
(a) 100 training sites — β_2



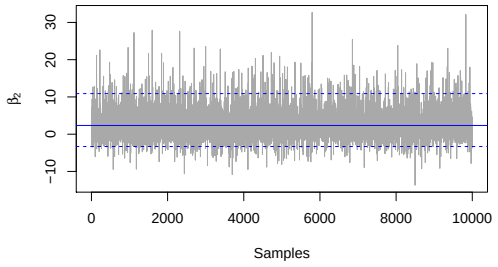
(b) 100 training sites — β_3



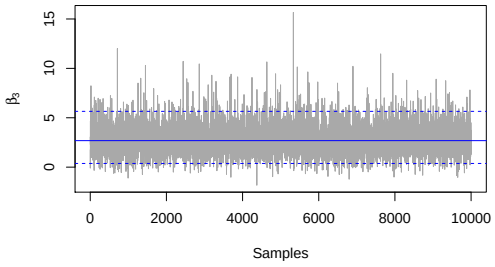
(c) 25 training sites — β_2



(d) 25 training sites — β_3

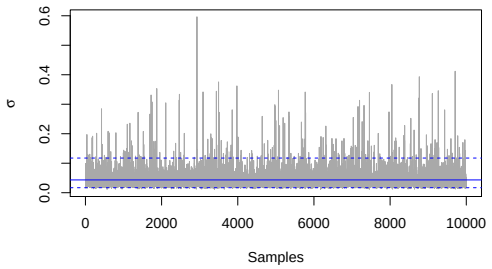


(e) 10 training sites — β_2

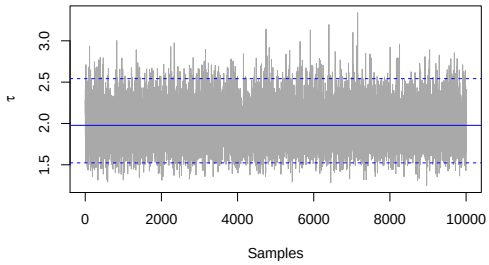


(f) 10 training sites — β_3

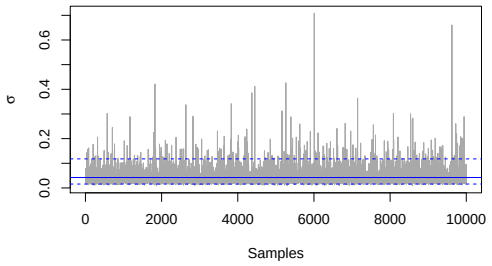
Figure A.32: MCMC Trace plots of β_2 and β_3 from simulation 2 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



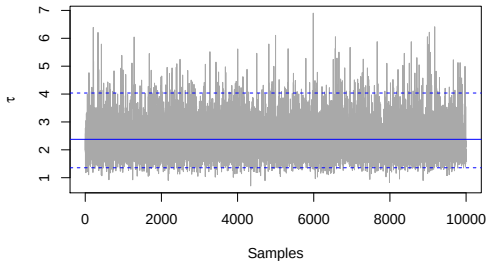
(a) 100 training sites — σ



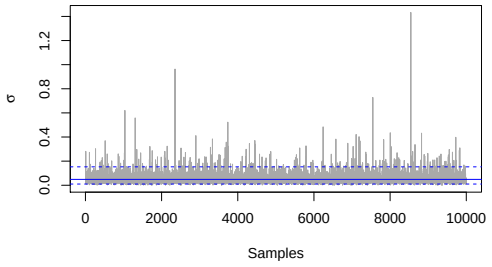
(b) 100 training sites — τ



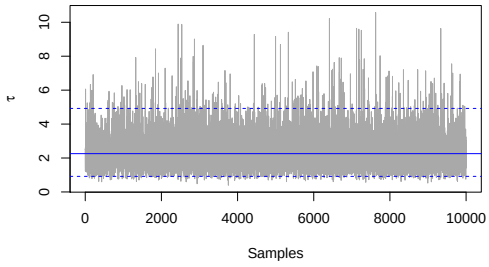
(c) 25 training sites — σ



(d) 25 training sites — τ

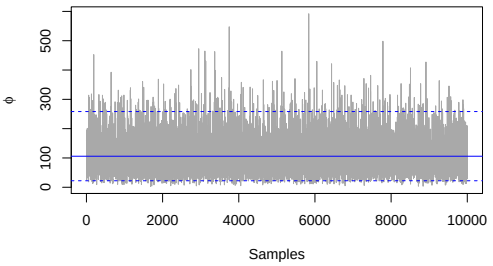


(e) 10 training sites — σ

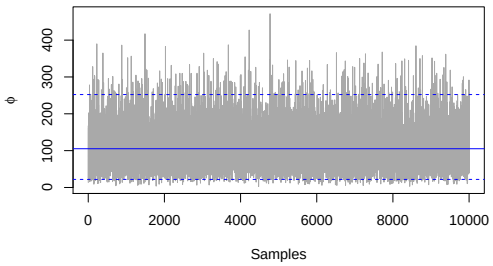


(f) 10 training sites — τ

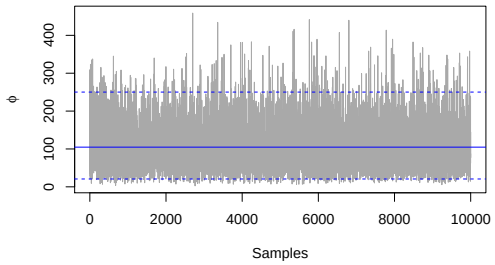
Figure A.33: MCMC Trace plots of σ and τ from simulation 2 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



(a) 100 training sites — ϕ



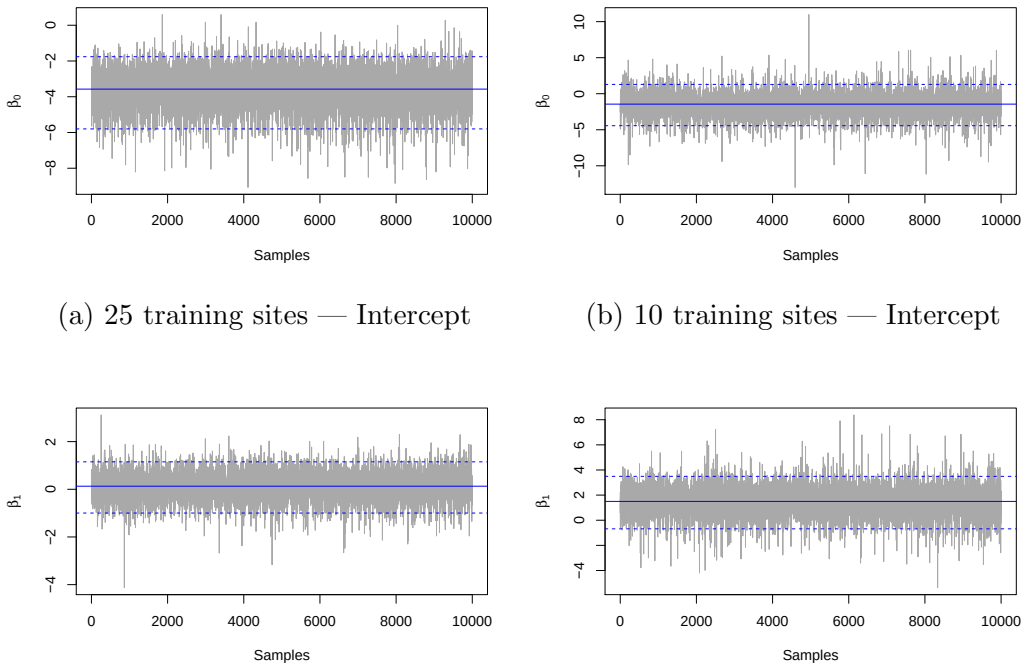
(b) 25 training sites — ϕ



(c) 10 training sites — ϕ

Figure A.34: MCMC Trace plots of ϕ from simulation 2 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.

Simulation 2 - Stratified sampling



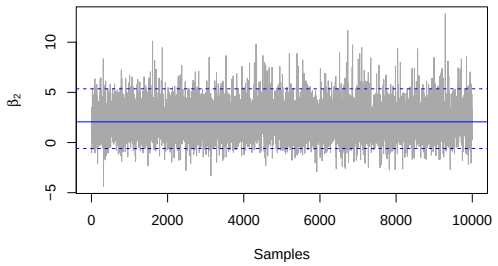
(a) 25 training sites — Intercept

(b) 10 training sites — Intercept

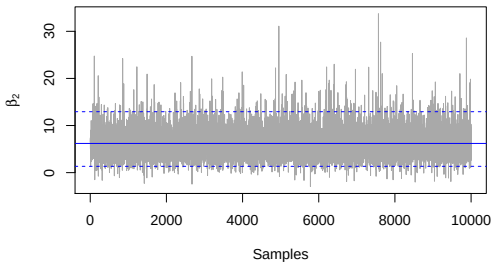
(c) 25 training sites — β_1

(d) 10 training sites — β_1

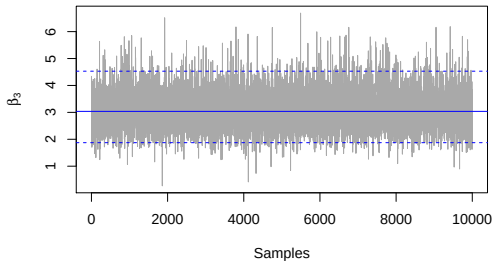
Figure A.35: MCMC Trace plots of β_0 and β_1 from simulation 2 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



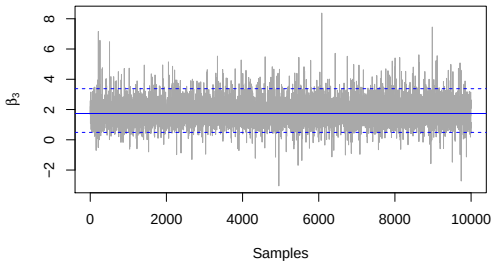
(a) 25 training sites — β_2



(b) 10 training sites — β_2

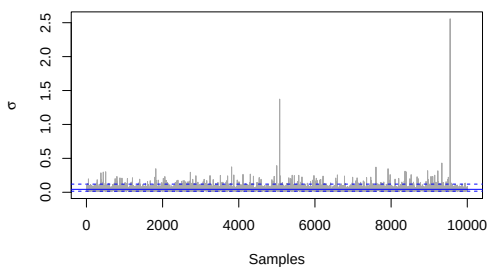


(c) 25 training sites — β_3

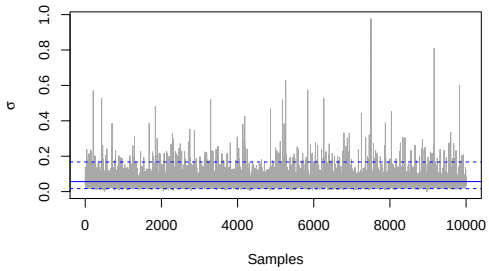


(d) 10 training sites — β_3

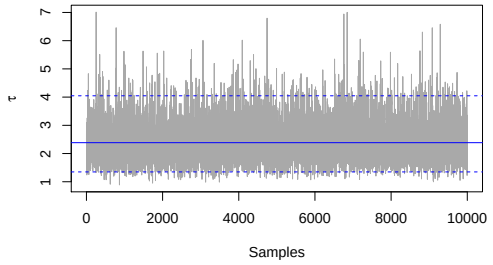
Figure A.36: MCMC Trace plots of β_2 and β_3 from simulation 2 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



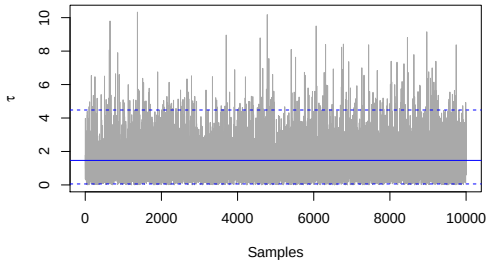
(a) 25 training sites — σ



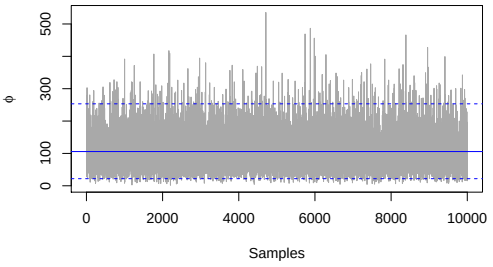
(b) 10 training sites — σ



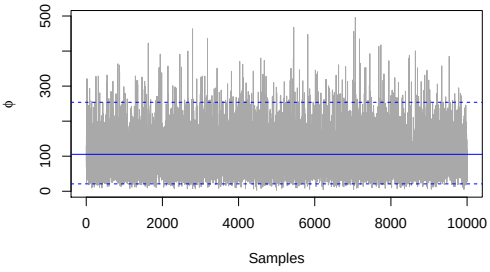
(c) 25 training sites — τ



(d) 10 training sites — τ



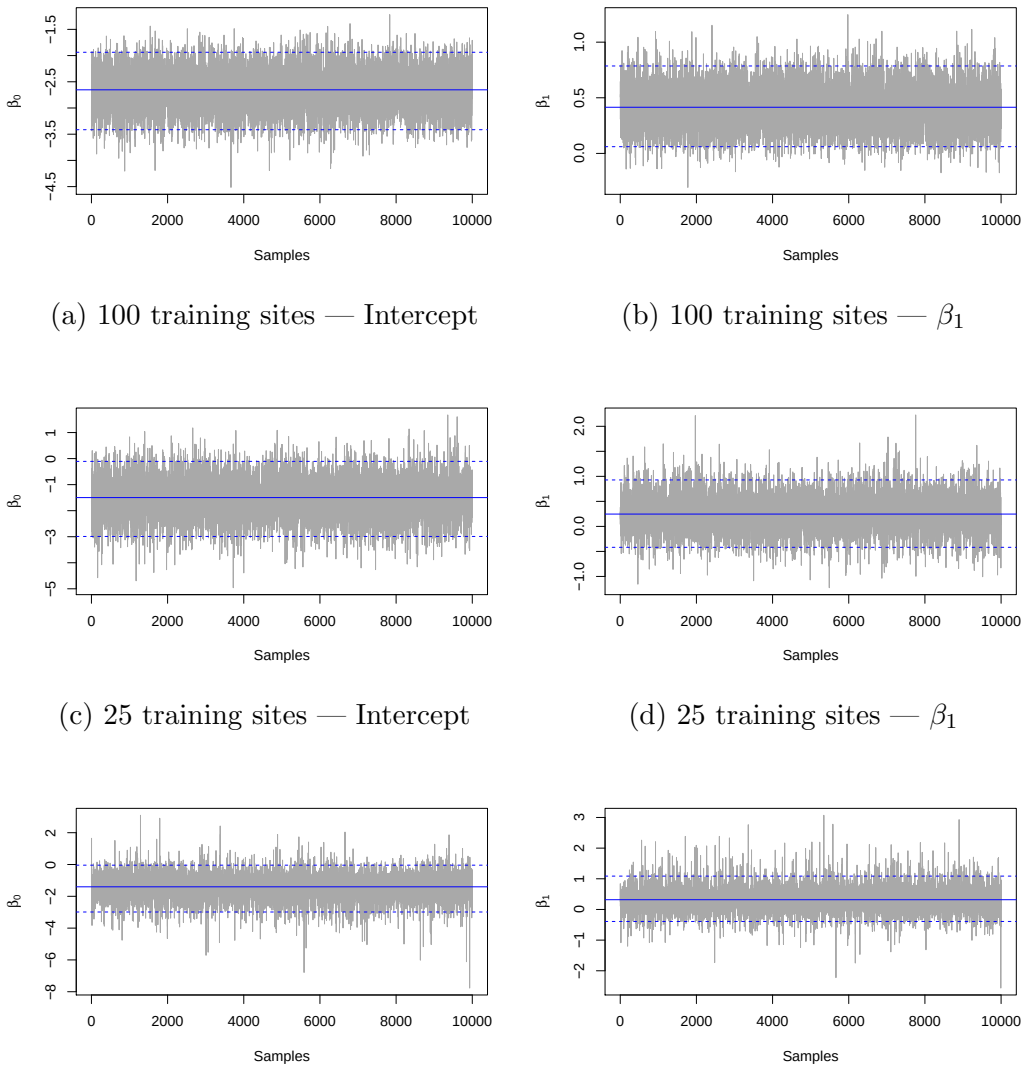
(e) 25 training sites — ϕ



(f) 10 training sites — ϕ

Figure A.37: MCMC Trace plots of σ, τ and ϕ from simulation 2 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.

Simulation 3 - Random sampling



(a) 100 training sites — Intercept

(b) 100 training sites — β_1

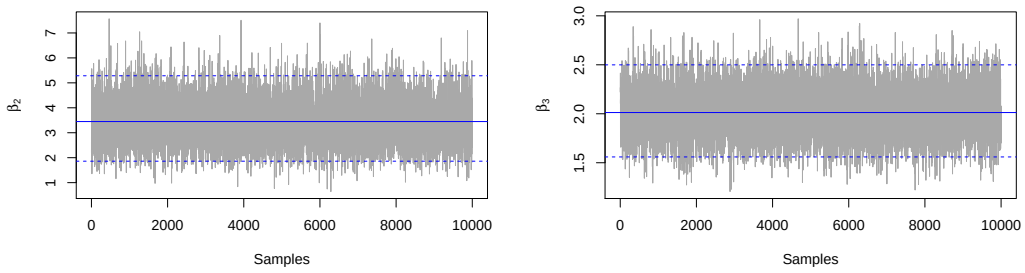
(c) 25 training sites — Intercept

(d) 25 training sites — β_1

(e) 10 training sites — Intercept

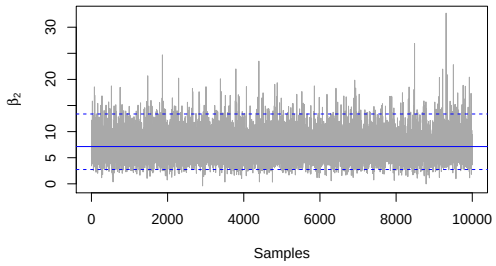
(f) 10 training sites — β_1

Figure A.38: MCMC Trace plots of β_0 and β_1 from simulation 3 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.

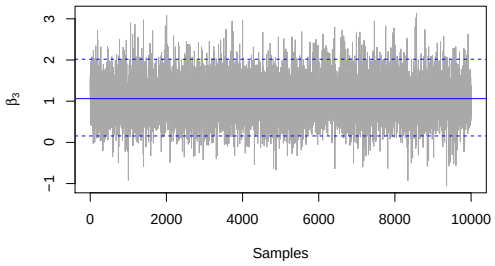


(a) 100 training sites — β_2

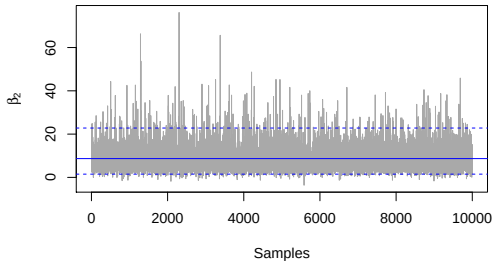
(b) 100 training sites — β_3



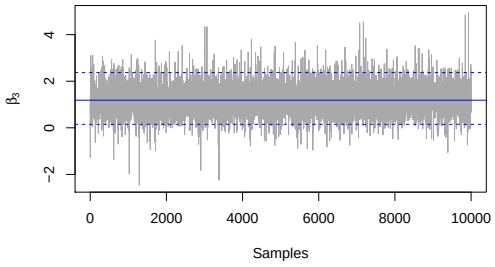
(c) 25 training sites — β_2



(d) 25 training sites — β_3

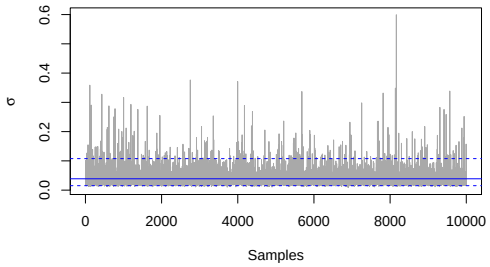


(e) 10 training sites — β_2

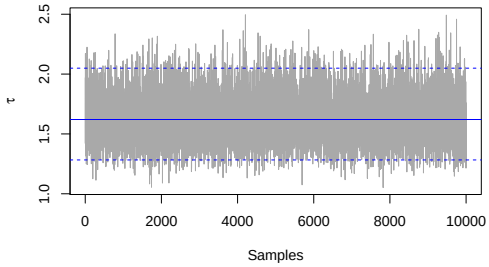


(f) 10 training sites — β_3

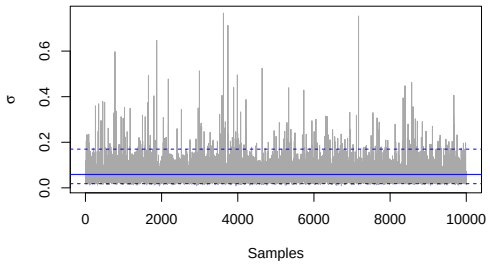
Figure A.39: MCMC Trace plots of β_2 and β_3 from simulation 3 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



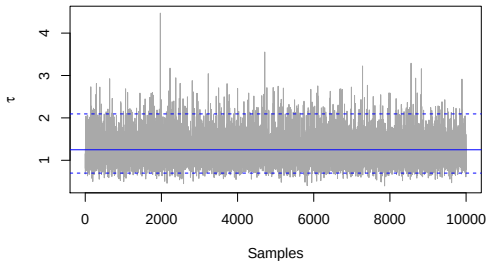
(a) 100 training sites — σ



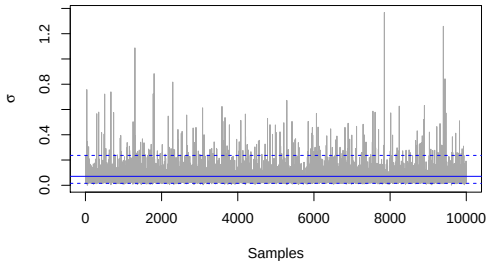
(b) 100 training sites — τ



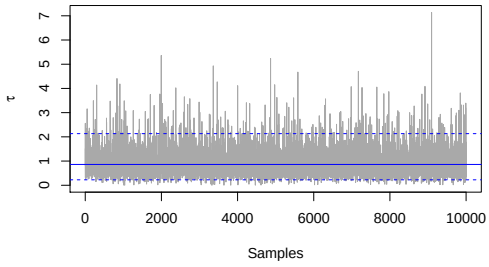
(c) 25 training sites — σ



(d) 25 training sites — τ

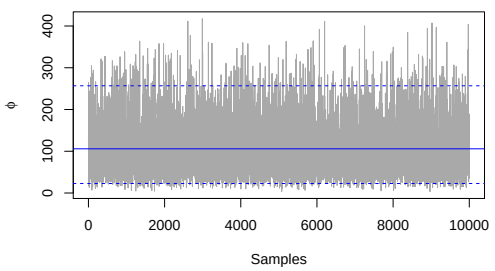


(e) 10 training sites — σ

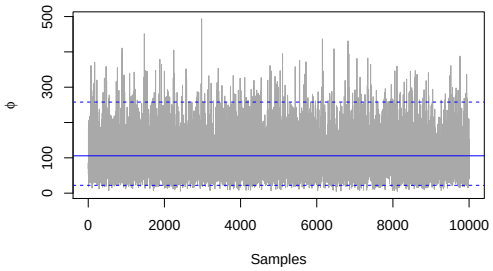


(f) 10 training sites — τ

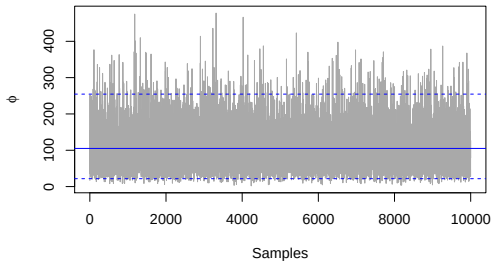
Figure A.40: MCMC Trace plots of σ and τ from simulation 3 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



(a) 100 training sites — ϕ



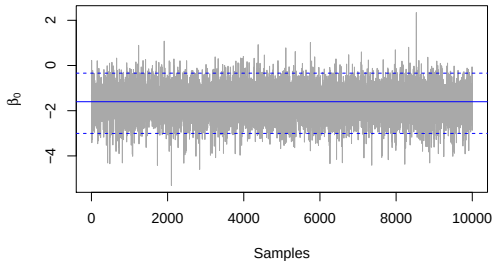
(b) 25 training sites — ϕ



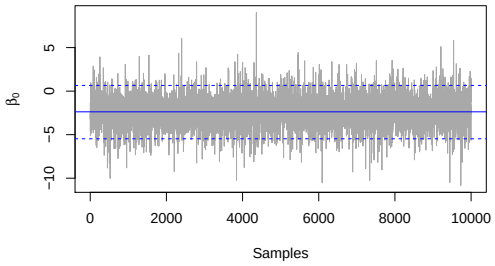
(c) 10 training sites — ϕ

Figure A.41: MCMC Trace plots of ϕ from simulation 3 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.

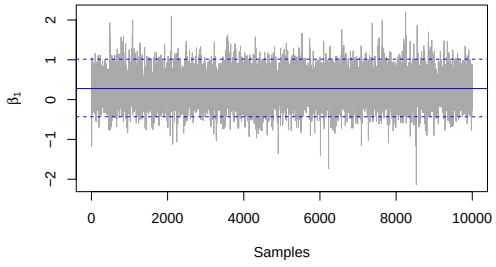
Simulation 3 - Stratified sampling



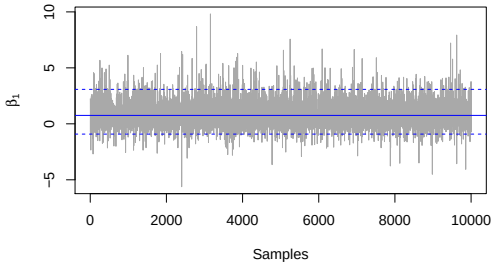
(a) 25 training sites — Intercept



(b) 10 training sites — Intercept

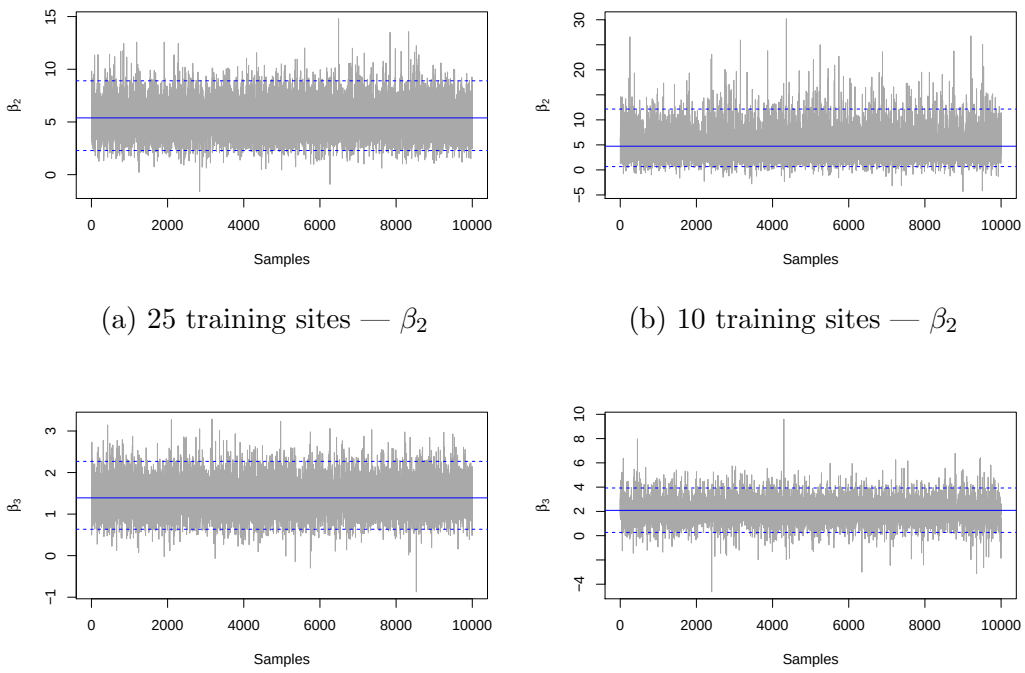


(c) 25 training sites — β_1



(d) 10 training sites — β_1

Figure A.42: MCMC Trace plots of β_0 and β_1 from simulation 3 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



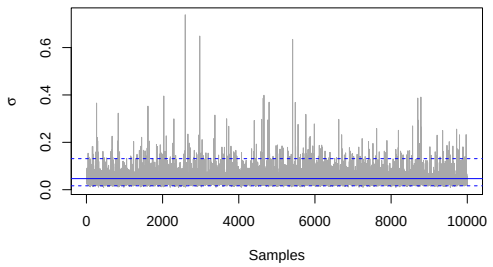
(a) 25 training sites — β_2

(b) 10 training sites — β_2

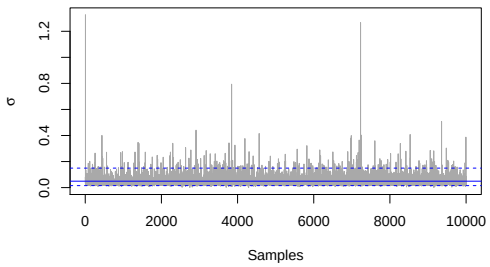
(c) 25 training sites — β_3

(d) 10 training sites — β_3

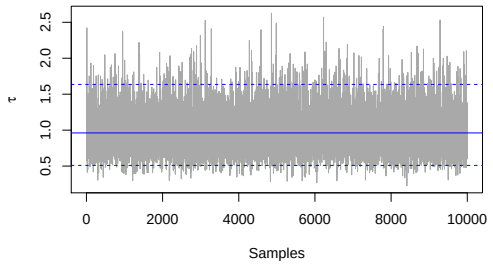
Figure A.43: MCMC Trace plots of β_2 and β_3 from simulation 3 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



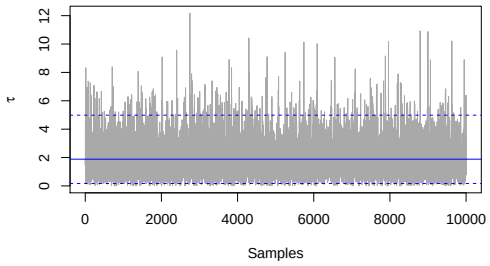
(a) 25 training sites — σ



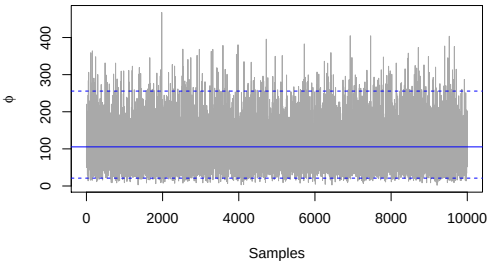
(b) 10 training sites — σ



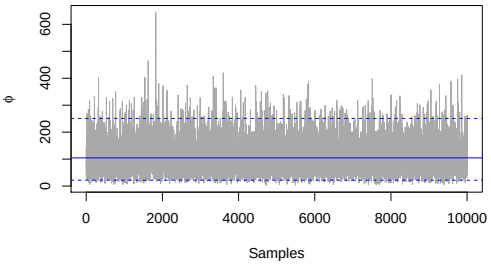
(c) 25 training sites — τ



(d) 10 training sites — τ



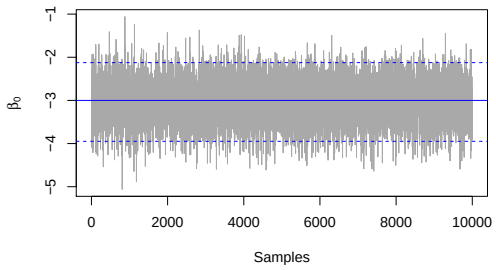
(e) 25 training sites — ϕ



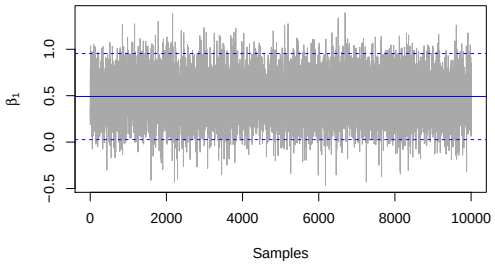
(f) 10 training sites — ϕ

Figure A.44: MCMC Trace plots of σ, τ and ϕ from simulation 3 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.

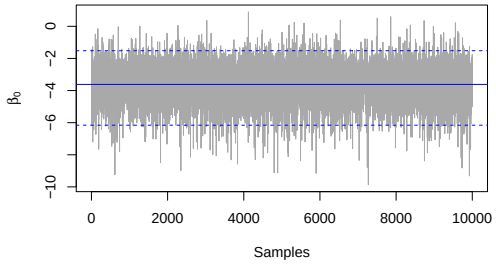
Simulation 4 - Random sampling



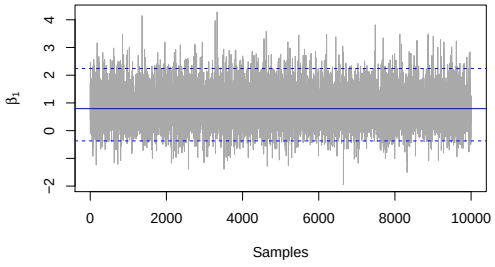
(a) 100 training sites — Intercept



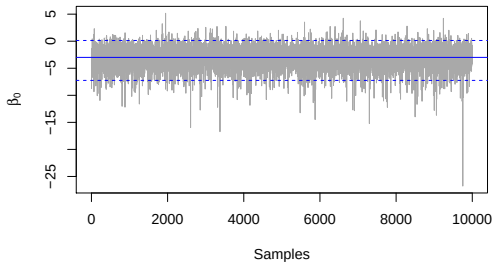
(b) 100 training sites — β_1



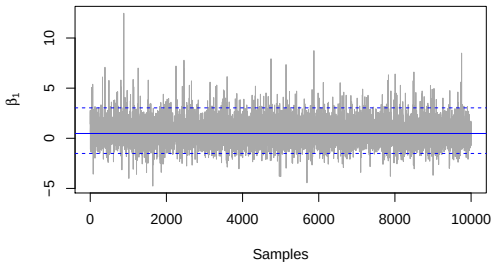
(c) 25 training sites — Intercept



(d) 25 training sites — β_1

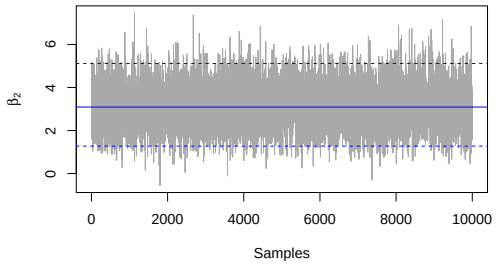


(e) 10 training sites — Intercept

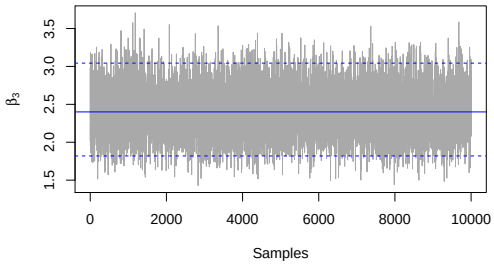


(f) 10 training sites — β_1

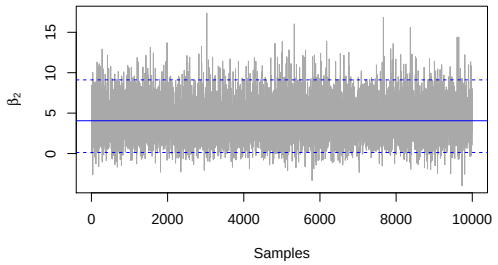
Figure A.45: MCMC Trace plots of β_0 and β_1 from simulation 4 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



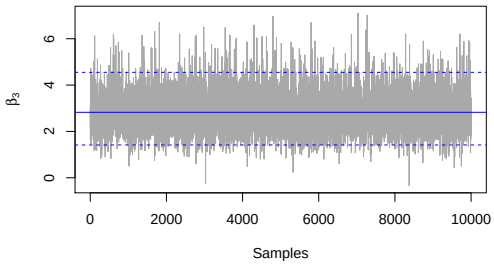
(a) 100 training sites — β_2



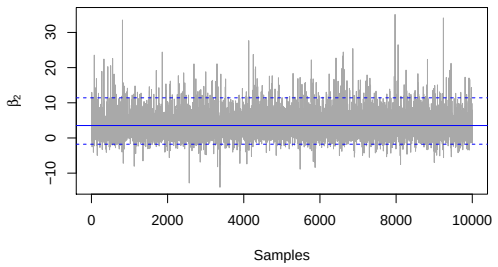
(b) 100 training sites — β_3



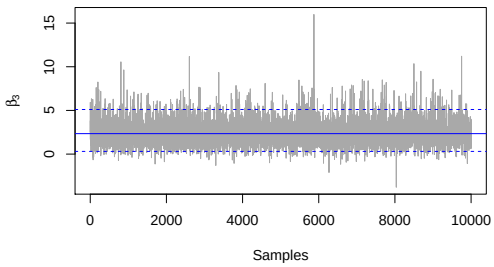
(c) 25 training sites — β_2



(d) 25 training sites — β_3

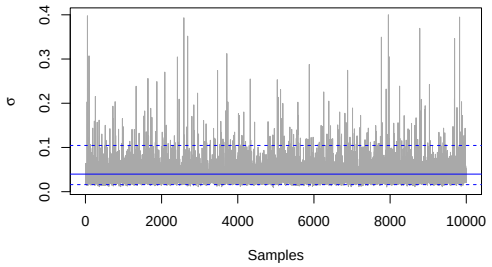


(e) 10 training sites — β_2

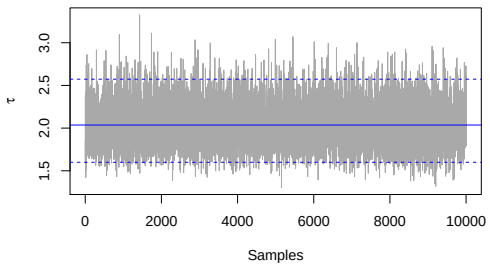


(f) 10 training sites — β_3

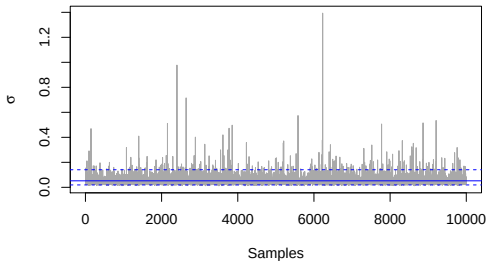
Figure A.46: MCMC Trace plots of β_2 and β_3 from simulation 4 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



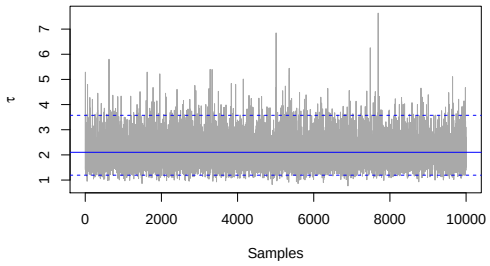
(a) 100 training sites — σ



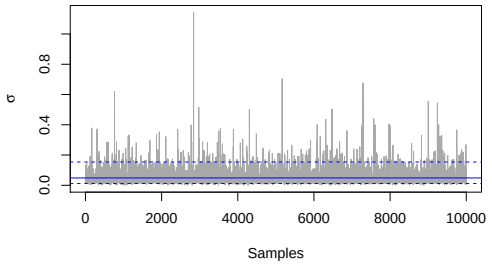
(b) 100 training sites — τ



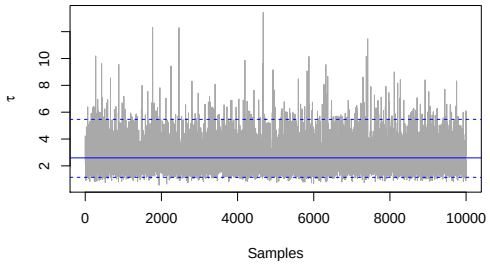
(c) 25 training sites — σ



(d) 25 training sites — τ

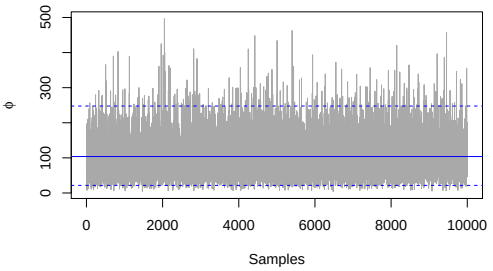


(e) 10 training sites — σ

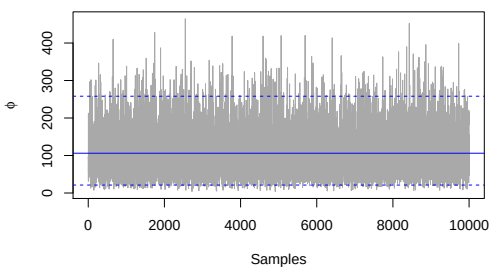


(f) 10 training sites — τ

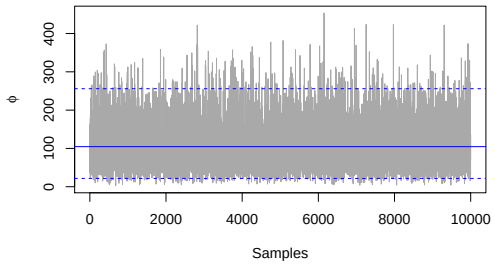
Figure A.47: MCMC Trace plots of σ and τ from simulation 4 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



(a) 100 training sites — ϕ



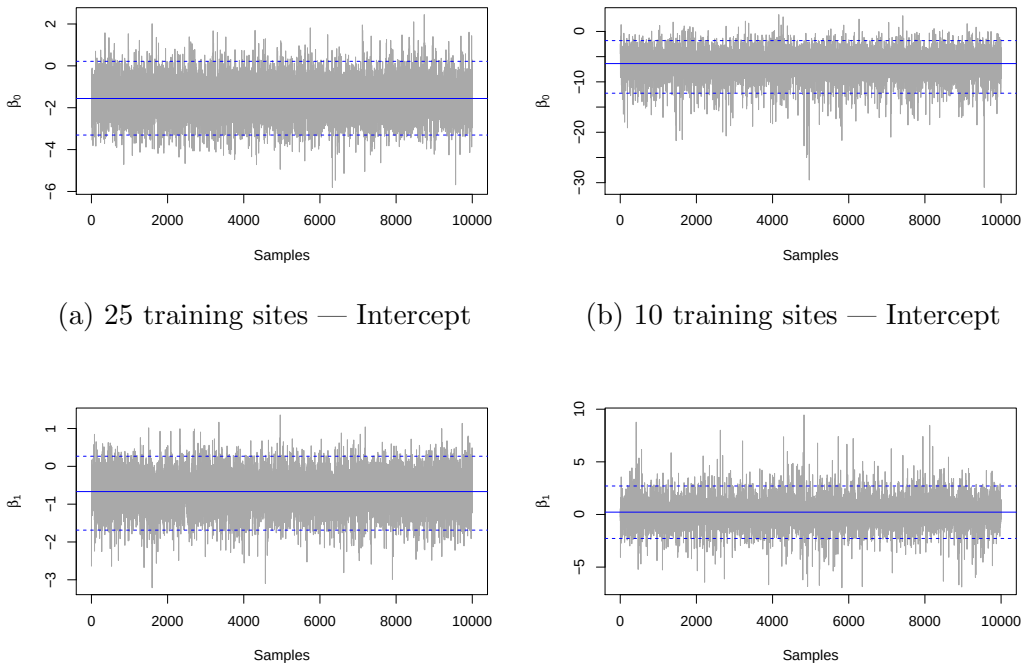
(b) 25 training sites — ϕ



(c) 10 training sites — ϕ

Figure A.48: MCMC Trace plots of ϕ from simulation 4 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.

Simulation 4 - Stratified sampling



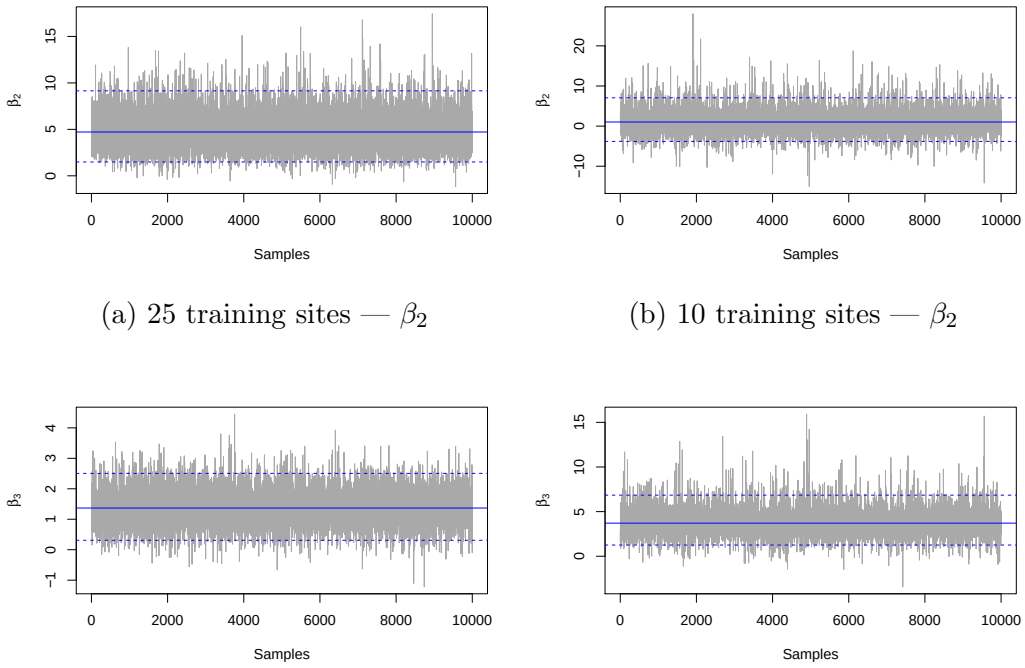
(a) 25 training sites — Intercept

(b) 10 training sites — Intercept

(c) 25 training sites — β_1

(d) 10 training sites — β_1

Figure A.49: MCMC Trace plots of β_0 and β_1 from simulation 4 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



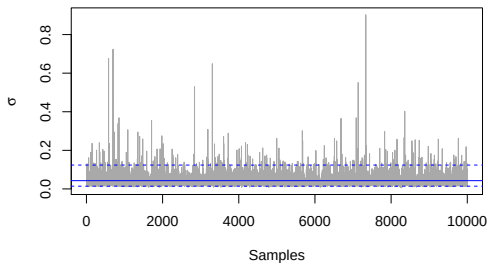
(a) 25 training sites — β_2

(b) 10 training sites — β_2

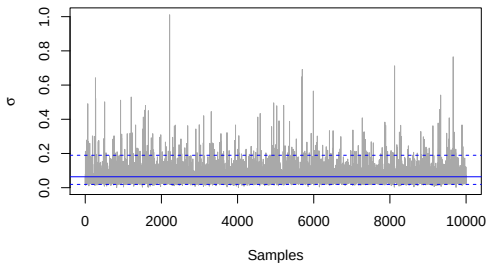
(c) 25 training sites — β_3

(d) 10 training sites — β_3

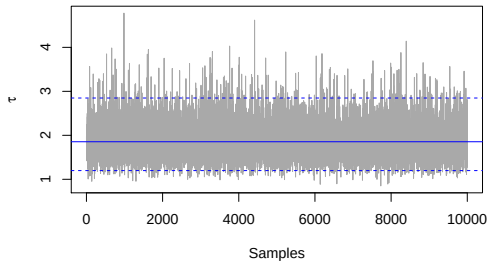
Figure A.50: MCMC Trace plots of β_2 and β_3 from simulation 4 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



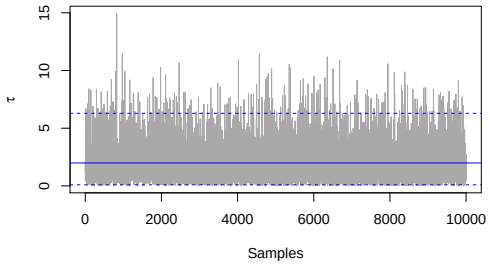
(a) 25 training sites — σ



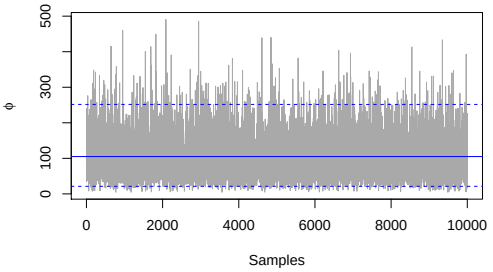
(b) 10 training sites — σ



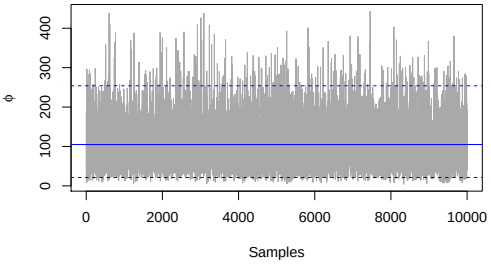
(c) 25 training sites — τ



(d) 10 training sites — τ



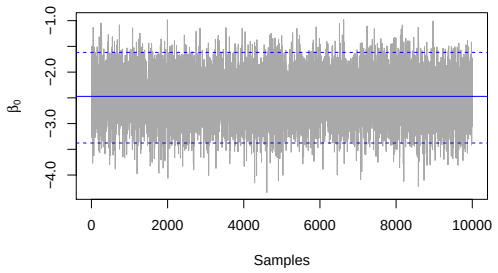
(e) 25 training sites — ϕ



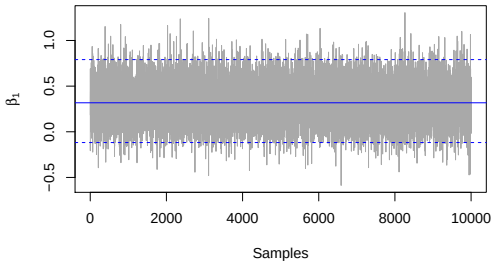
(f) 10 training sites — ϕ

Figure A.51: MCMC Trace plots of σ, τ and ϕ from simulation 4 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.

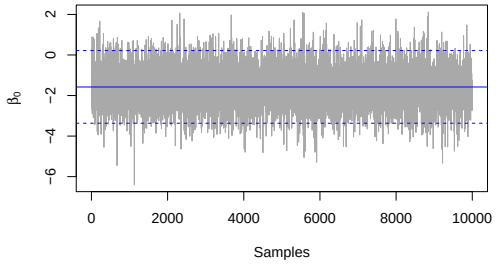
Simulation 5 - Random sampling



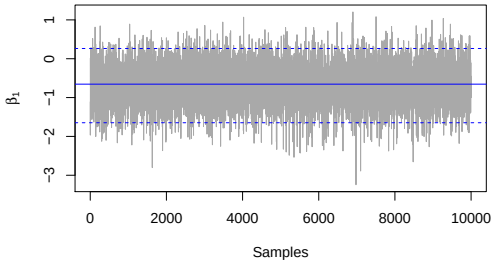
(a) 100 training sites — Intercept



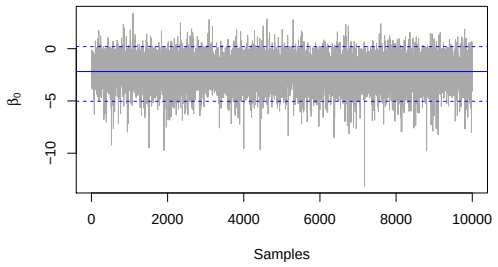
(b) 100 training sites — β_1



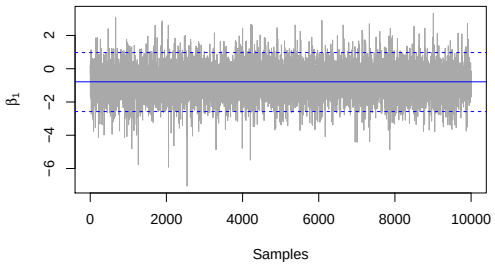
(c) 25 training sites — Intercept



(d) 25 training sites — β_1

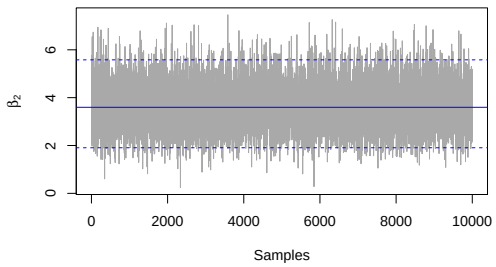


(e) 10 training sites — Intercept

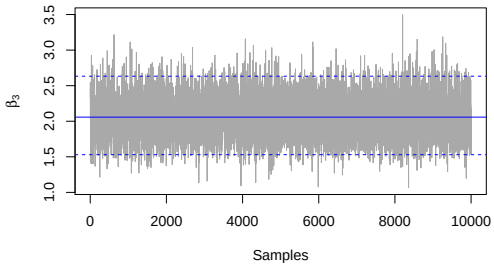


(f) 10 training sites — β_1

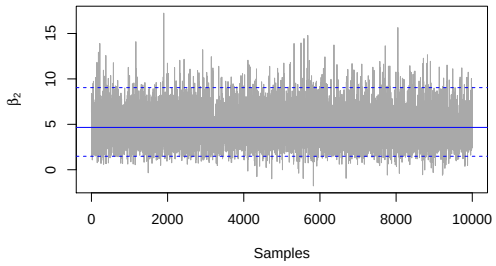
Figure A.52: MCMC Trace plots of β_0 and β_1 from simulation 5 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



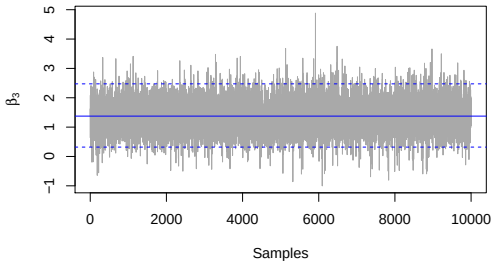
(a) 100 training sites — β_2



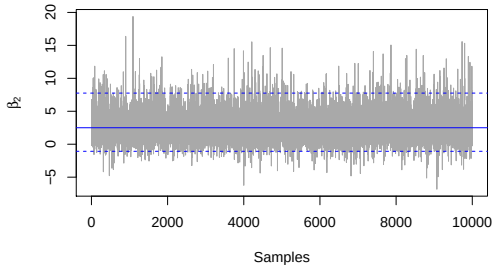
(b) 100 training sites — β_3



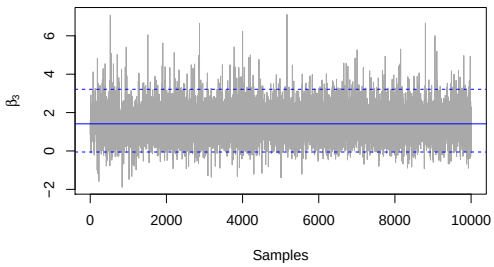
(c) 25 training sites — β_2



(d) 25 training sites — β_3

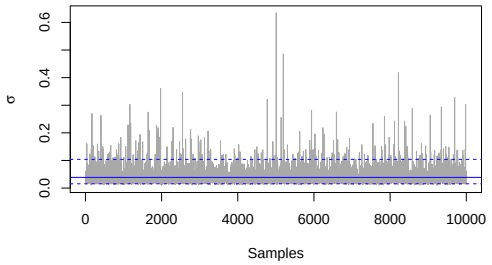


(e) 10 training sites — β_2

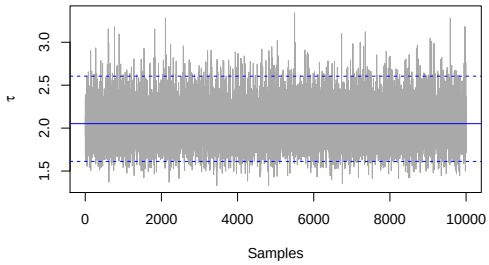


(f) 10 training sites — β_3

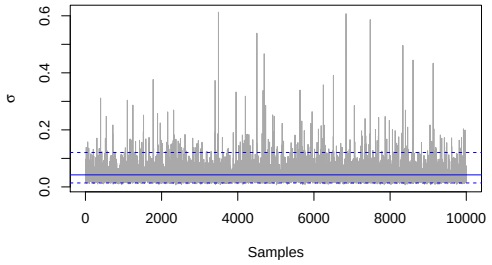
Figure A.53: MCMC Trace plots of β_2 and β_3 from simulation 5 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



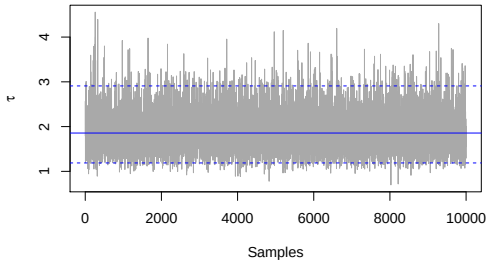
(a) 100 training sites — σ



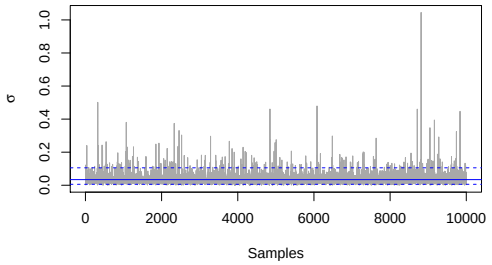
(b) 100 training sites — τ



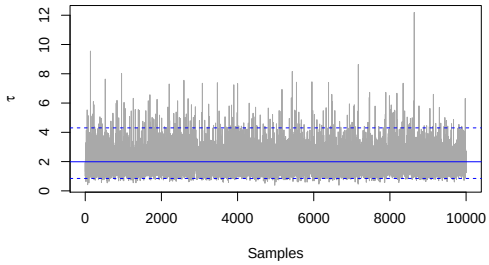
(c) 25 training sites — σ



(d) 25 training sites — τ

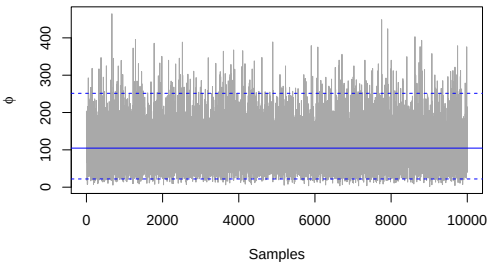


(e) 10 training sites — σ

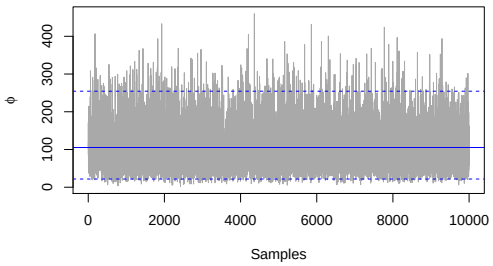


(f) 10 training sites — τ

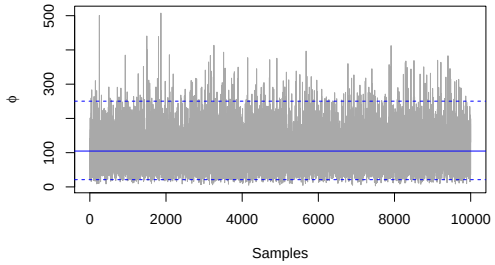
Figure A.54: MCMC Trace plots of σ and τ from simulation 5 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



(a) 100 training sites — ϕ



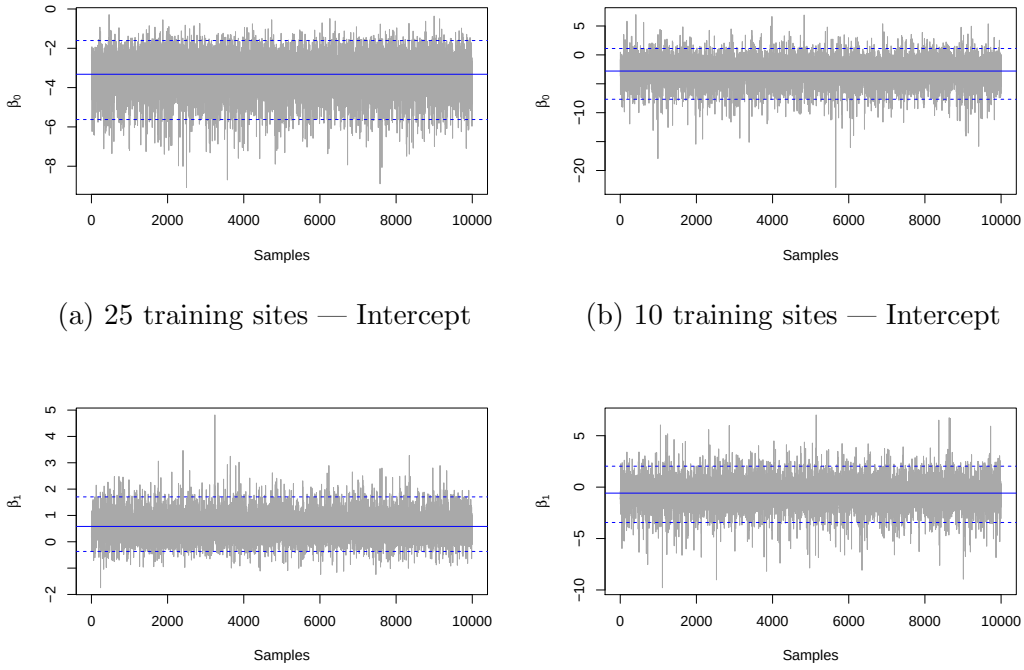
(b) 25 training sites — ϕ



(c) 10 training sites — ϕ

Figure A.55: MCMC Trace plots of ϕ from simulation 5 (random sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.

Simulation 5 - Stratified sampling



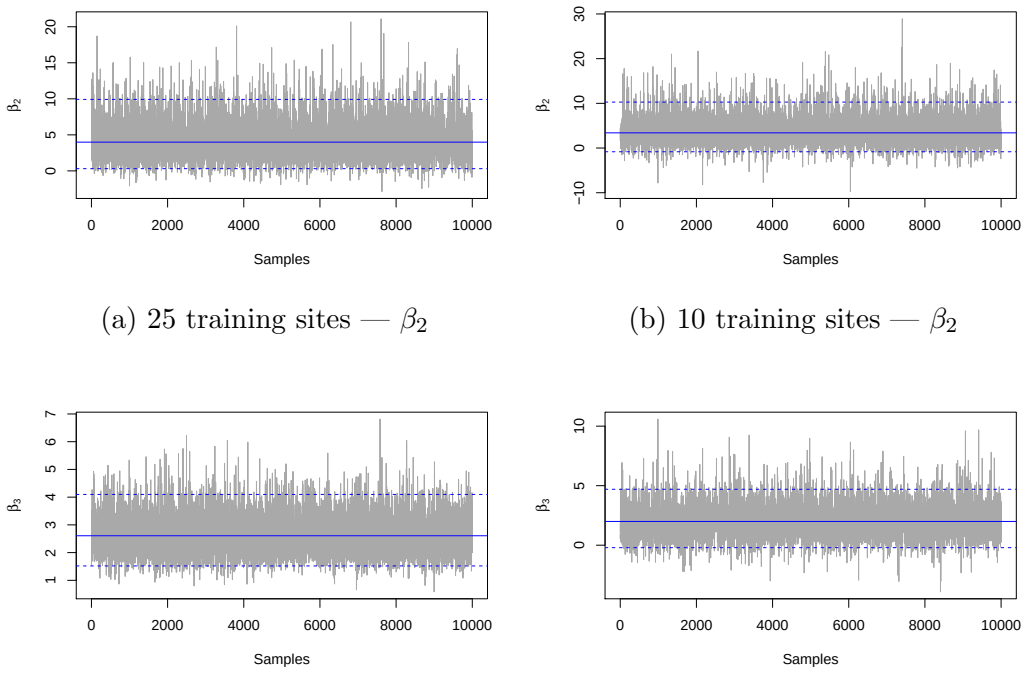
(a) 25 training sites — Intercept

(b) 10 training sites — Intercept

(c) 25 training sites — β_1

(d) 10 training sites — β_1

Figure A.56: MCMC Trace plots of β_0 and β_1 from simulation 5 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



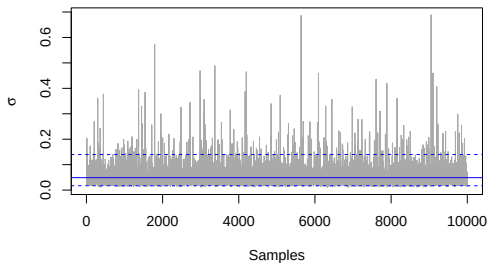
(a) 25 training sites — β_2

(b) 10 training sites — β_2

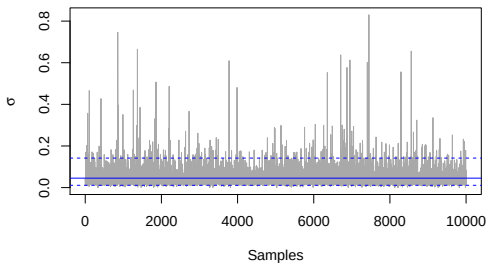
(c) 25 training sites — β_3

(d) 10 training sites — β_3

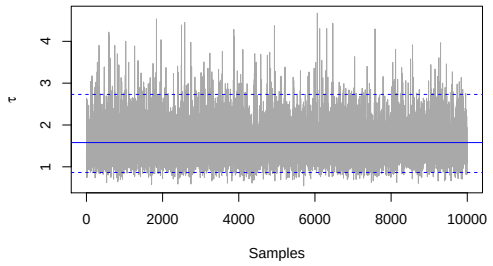
Figure A.57: MCMC Trace plots of β_2 and β_3 from simulation 5 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.



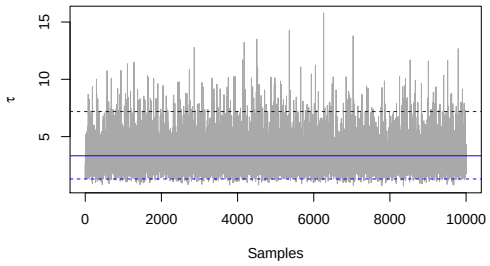
(a) 25 training sites — σ



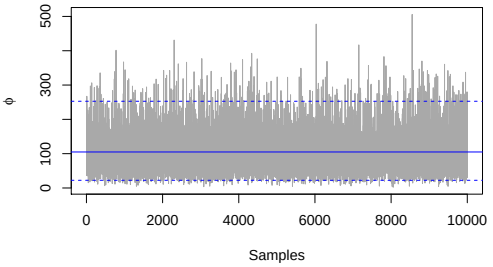
(b) 10 training sites — σ



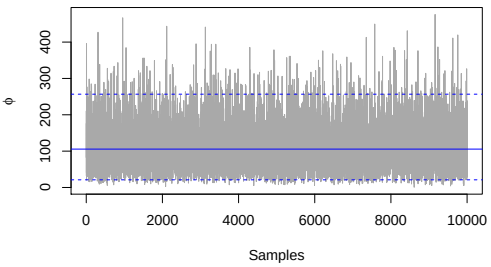
(c) 25 training sites — τ



(d) 10 training sites — τ



(e) 25 training sites — ϕ



(f) 10 training sites — ϕ

Figure A.58: MCMC Trace plots of σ, τ and ϕ from simulation 5 (stratified sampling), with their corresponding mean, 2.5%, and 97.5% quantiles.

Bibliography

Juan J Abellan, Daniela Fecht, Nicky Best, Sylvia Richardson, and David J Briggs. Bayesian analysis of the multivariate geographical distribution of the socio-economic environment in england. *Environmetrics*, 18(7):745–758, 2007.

Rémi Bardenet, Arnaud Doucet, and Chris Holmes. On Markov chain Monte Carlo methods for tall data. *The Journal of Machine Learning Research*, 18(1):1515–1557, 2017.

Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

José M Bernardo and Adrian FM Smith. Bayesian Theory, 2001.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Michael GB Blum. Approximate Bayesian computation: a nonparametric perspective. *Journal of the American Statistical Association*, 105(491):1178–1187, 2010.

S. Brooks, A. Gelman, G. L. Jones, and X. Meng, editors. *Handbook of Markov Chain Monte Carlo*. Taylor & Francis, 2011.

Stephen Brooks. Markov chain Monte Carlo method and its application. *Journal of the Royal Statistical Society: series D (the Statistician)*, 47(1):69–100, 1998.

Patrick E. Brown. Model-Based Geostatistics the Easy Way. *Journal of Statistical Software*, 63(12):1–24, 2015. URL `http://www.jstatsoft.org/v63/i12/`.

Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.

Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian treed models. *Machine Learning*, 48(1-3):299–320, 2002.

Hugh A Chipman, Edward I George, and Robert E McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, pages 266–298, 2010.

Oksana A Chkrebtii, Andrew Hoegh, Reihaneh Entezari, Radu V Craiu, Jeffrey S Rosenthal, Abdolreza Mohammadi, Maurits Kaptein, Luca Martino, Rafael B Stern, Francisco Louzada, et al. Contributed Discussion on Article by Pratola. *Bayesian Analysis*, 11(3):929–943, 2016.

Ole F Christensen, Gareth O Roberts, and Martin Sköld. Robust Markov chain Monte Carlo methods for spatial generalized linear mixed models. *Journal of Computational and Graphical Statistics*, 15(1):1–17, 2006.

Radu V Craiu and Jeffrey S Rosenthal. Bayesian computation via Markov Chain Monte Carlo. *Annual Review of Statistics and Its Application*, 1:179–201, 2014.

Noel Cressie. *Statistics for Spatial Data*. John Wiley & Sons, 1993.

Paul J Curran and Peter M Atkinson. Geostatistics and remote sensing. *Progress in Physical Geography*, 22(1):61–78, 1998.

Peter J. Diggle and Paulo J. Ribeiro. *Model-based Geostatistics*. Springer Series in Statistics. Springer, 3 2007. ISBN 0387329072 978-0387329079.

Peter J Diggle, JA Tawn, and RA Moyeed. Model-based geostatistics. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47(3):299–350, 1998.

Christopher C Drovandi, Anthony N Pettitt, and Anthony Lee. Bayesian indirect inference using a parametric auxiliary model. *Statistical Science*, 30(1):72–95, 2015.

Reihaneh Entezari, Patrick E Brown, and Jeffrey S Rosenthal. Bayesian Spatial Analysis of Hardwood Tree Counts in Forests via MCMC. *arXiv preprint arXiv:1807.01239*, 2018a.

Reihaneh Entezari, Radu V Craiu, and Jeffrey S Rosenthal. Likelihood inflating sampling algorithm. *Canadian Journal of Statistics*, 46(1):147–175, 2018b.

Jerome H Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.

Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian Data Analysis*, volume 2. CRC press Boca Raton, FL, 2014.

Emanuele Giorgi and Peter J Diggle. PrevMap: an R package for prevalence mapping. *Journal of Statis*, 2017.

Emanuele Giorgi, Daniela K Schlüter, and Peter J Diggle. Bivariate geostatistical modelling of the relationship between loa loa prevalence and intensity of infection. *Environmetrics*, 2017.

W Keith Hastings. Monte Carlo sampling methods using Markov Chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Adam Kapelner and Justin Bleich. bartMachine: Machine Learning with Bayesian Additive Regression Trees. *arXiv preprint arXiv:1312.2171*, 2013.

Kathryn Blackmond Laskey and James W Myers. Population Markov Chain Monte Carlo. *Machine Learning*, 50(1-2):175–196, 2003.

Chris Loken, Daniel Gruner, Leslie Groer, Richard Peltier, Neil Bunn, Michael Craig, Teresa Henriques, Jillian Dempsey, Ching-Hsing Yu, Joseph Chen, et al. SciNet: lessons learned from building a power-efficient top-20 system and data centre. In *Journal of Physics: Conference Series*, volume 256, page 012026. IOP Publishing, 2010.

David JC MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

Dougal Maclaurin and Ryan P Adams. Firefly Monte Carlo: Exact MCMC with Subsets of Data. In *Uncertainty in Artificial Intelligence (UAI)*, pages 543–552, 2014.

Georges Matheron. *Traité de géostatistique appliquée. 1 (1962)*, volume 1. Editions Technip, 1962.

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.

Sean P Meyn and Richard L Tweedie. *Markov Chains and Stochastic Stability*. Springer Science & Business Media, 2012.

Jesper Møller, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen. Log gaussian cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482, 1998.

Radford M Neal. Probabilistic inference using Markov Chain Monte Carlo methods. 1993.

Willie Neiswanger, Chong Wang, and Eric Xing. Asymptotically exact, embarrassingly parallel MCMC. *arXiv preprint arXiv:1311.4780*, 2013.

Matthew T Pratola. Efficient Metropolis–Hastings proposal mechanisms for Bayesian regression tree models. *Bayesian Analysis*, 11(3):885–911, 2016.

Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

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 Richard L Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

Gareth O Roberts, Andrew Gelman, and Walter R Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120, 1997.

Jeffrey S Rosenthal. Parallel computing and Monte Carlo algorithms. *Far East Journal of Theoretical Statistics*, 4(2):207–236, 2000.

Jeffrey S Rosenthal. *A First Look at Rigorous Probability Theory*. World Scientific Publishing Company, 2006.

Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B (statistical methodology)*, 71(2):319–392, 2009.

Steven L Scott, Alexander W Blocker, Fernando V Bonassi, H Chipman, E George, and R McCulloch. Bayes and big data: The consensus Monte Carlo algorithm. In *EFaBBayes 250 conference*, volume 16, 2013.

Benjamin A Shaby and Brian J Reich. Bayesian spatial extreme value analysis to assess the changing risk of concurrent high temperatures across large portions of european cropland. *Environmetrics*, 23(8):638–648, 2012.

Michael L Stein. Interpolation of Spatial Data: Some Theory for Kriging. *Springer-Verlag, New York*, 1999.

US Bureau of Census. 2013 ACS 1-YEAR PUMS data. URL `http://www.census.gov/programs-surveys/acs/data/pums.html`.

Xiangyu Wang and David B Dunson. Parallelizing MCMC via Weierstrass sampler. *arXiv preprint arXiv:1312.4605*, 2013.

Darren J Wilkinson. Parallel bayesian computation. *Statistics Textbooks and Monographs*, 184:477, 2006.

Yuhong Wu, Håkon Tjelmeland, and Mike West. Bayesian CART: Prior specification and posterior simulation. *Journal of Computational and Graphical Statistics*, 16 (1):44–66, 2007.

Hao Zhang. Inconsistent estimation and asymptotically equal interpolations in model-based geostatistics. *Journal of the American Statistical Association*, 99(465):250–261, 2004.