THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF STATISTICS

An Evaluation of Stein Thinning for the Post-Processing of Markov Chain Monte Carlo Linear Regression Output

MICHAEL LICATA
SPRING 2024

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Computational Data Science
with honors in Statistics

Reviewed and approved* by the following:

Stephen Berg
Assistant Professor of Statistics
Thesis Supervisor

John Hannan
Associate Professor of Computer Science
Associate Department Head of Computer Science
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

In recent years many fields of research have turned to Markov Chain Monte Carlo (MCMC) sampling methods in order to better understand populations and systems that have probability distributions that are intractable, or impossible to determine in polynomial time. When using MCMC methods, more complicated systems require larger MCMC Chains to compute accurate estimates, which causes the output to be difficult or impossible to utilize within given CPU constraints. To solve this problem, post-processing methods have been developed to decrease the size of these outputs while not sacrificing their MSE, bias or variance. Chief among these methods are Stein Thinning and Zero Variance Control Variates (ZVCV), which are both evaluated in this paper and are compared with Basic or Vanilla MCMCM chains and a ground truth MCMC sample. It is determined that Stein Thinning has similar variance reduction power as ZVCV while maintaining lower bias and MSE. Stein Thinning's MSE, bias and variance are also found to be within a factor of 10 when compared to the raw MCMC chain that the Stein Thinned samples are computed from. While these results are promising, Stein Thinning fails to maintain these impressive results when the burn in tail is not removed before thinning the samples. These results, and the methods used to obtain them, are explored in depth and conclusions relevant to the field of Markov Chain Monte Carlo sampling are presented below.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I want to thank my Thesis Supervisor, Professor Stephen Berg, for all of his support throughout my research. Professor Berg always made himself available to meet, taught tons of mini lessons, and offered countless advice throughout this process. His mentorship was invaluable and I am truly grateful. I want to also thank my Honors Advisor, Professor John Hannan. Professor Hannan provided me with the trust and guidance I needed to pursue a Thesis outside of my major. I am grateful for this opportunity.

I finally want to thank my friends and family who supported me throughout this writing process and throughout my 4 years as a student within Schreyer Honors College. Without them none of this would have been possible.

Chapter 1

Introduction to Markov Chain Monte Carlo

1.1 Introduction

The use of Markov Chain Monte Carlo (MCMC), which is a process derived from the ideas of Bayesian statistics, has become increasingly popular in the past decade within various fields of research, such as psychology, cardiology, neuroscience, biology, and physics. One important use case of MCMC chains is to gain a stronger understanding of the distribution of β coefficients in Linear Regression equations. This paper aims to evaluate post-processing methods for MCMC output in the context of Linear Regression algorithms. Before delving into these post-processing procedures and the underlying theorems of MCMC, it is essential to gain a baseline understanding of Bayesian statistics, the Monte Carlo sampling method, and the theory behind Markov Chains.

1.2 Bayesian Statistics

To understand Bayesian statistics, it is helpful to begin with the main alternative approach, which is frequentist statistics. Frequentist statistics relies heavily on long-term analysis in order to understand the behavior and distribution of different populations. The most popular example is trying to determine the distribution, and thus the probability, of coin flips with two outcomes: heads and tails. In the frequentist approach, a coin flip would be simulated thousands of times, and a distribution of the outcomes could be plotted to learn about the mean and variance of the population of coin flips [**?**]. Rather than utilizing this long-term approach, Bayesian statistics uses as much prior information as possible in order to create a probability distribution of the population of interest. Looking back to the coin flip example described above, a Bayesian statistician would start with their prior beliefs about the coin, which would be resembled in a prior distribution of the coin flips. The prior distribution would be updated as a coin is actually flipped. The prior is then used to predict the actual outcome of the coin flip. [**?**].

The simple example outlined above is used to create a general understanding of what makes Bayesian statistics different from the common frequentist approach. The theory behind Bayesian

statistics, however, is a bit more complicated than a simple coin flip. The methods of Bayesian statistics come from the mind of Thomas Bayes and his famous Bayes theorem, which is provided below [**?**].

$$P(\mu|D) \propto P(D|\mu) * P(\mu) \tag{1.1}$$

$P(\mu|D)$ represents the conditional probability of a parameter or set of parameters (such as the mean or variance) given a dataset D. $P(\mu)$ represents the prior probability of μ, which is an a priori distribution that resembles all prior knowledge of parameter(s) μ. $P(D|\mu)$ is the conditional probability of the data D given a fixed value of the parameter(s) μ, which is also known as the likelihood function. The most important two pieces to Bayes theorem are those described above which are the prior distribution of the parameter(s) of interest and the likelihood function of the data given fixed parameter(s).

1.3 Monte Carlo Sampling

Now that a proper theoretical understanding of Bayes Theorem and Bayesian statistics has been developed, the Monte Carlo sampling method can be explored. A practical example for exploring this sampling method is known as Monte Carlo Integration. Monte Carlo Integration is a specific use case of the Monte Carlo method in which the expected value of a definite integral of a function can be estimated. This can be used to easily compute estimates for complex integrals when the function is known and the range, $(0, 1)$, is known. An example from the textbook, *Monte Carlo Statistical Methods* by Christan Robert and George Casella, is provided below and begins with a function called $h(x)$ [**?**].

$$\int_0^1 h(x)dx \tag{1.2}$$

Instead of computing the integral in Equation (1.2) by hand, it can be computed by generating

random iid values $X_1, X_2, ..., X_n$ in the range $(0, 1)$ using a given distribution, in this case $\mathcal{N}(0, 1)$ [**?**]. Then $h(X_1), h(X_2), ..., h(X_n)$ can be computed. This computation will find different estimates of the interval based on different values of $h(X_i)$ that are computed. The areas found by this computation can then be averaged to find an estimate for the integral. The integral, $\int_a^b h(x)dx$ is then estimated using the following formula, which computes the Monte Carlo average for the integral:

$$\frac{1}{n}\sum_{i=1}^{n} h(X_i) \tag{1.3}$$

In the above, it is understood that the Monte Carlo average will converge due to the Weak Law of Large Numbers. This theory states that the average, or mean value, of a set of random variables will converge in probability to the expected value as the sample of random variables approaches infinity. The expected value of a function, such as $h(x)$ which we looked at above, is mathematically represented as follows: $\hat{\mu} \xrightarrow{p} E[h(x)]$ as $n \to \infty$. Beyond understanding this convergence principle, it is also important to note that the variance of the resulting average will decrease as the size of the sample, n, increases. If the example above is extended, it is clear that this principle holds true. Supposed $X_1, ...X_n$ are an iid Monte Carlo sample from the distribution F with pdf f.

$$E_F\{g(X)\} = \int h(x)f(x)dx \tag{1.4}$$

for function h with finite expectation under F. A Monte Carlo estimator of $E_F\{h(X)\}$ is:

$$\hat{\mu} = \sum_{i=1}^{n} h(X_i)/n \tag{1.5}$$

To determine the variance of this estimator for $h(X)$ under F, find $Var(\hat{\mu})$, where C is the variance of $h(X)$ for $X \sim F$:

$$Var(\hat{\mu}) = \frac{1}{n} * \{\int h(x)^2 f(x)dx - [\int h(x)f(x)dx]^2\} = \frac{1}{n} * C \propto \frac{1}{n} \tag{1.6}$$

So it can be seen that the variance is proportional to $\frac{1}{n}$.

1.4 A Brief Introduction to Markov Chains

The last piece to MCMC simulations are Markov chains. A Markov chain, named after mathematician Andrey Markov, is a stochastic model that can be used to make predictions about a system. The most basic example of this is trying to predict the weather. If a system has access to a hundred days of weather for a specific town, it would be able to create a transition matrix that determines the probability of each type of weather, sunny, rainy, or windy, given the current state of the system. This process would make predictions about the next day of weather based only on knowledge of the current state of the weather. It would forget all of the previous states of weather that came before. This is an essential property of a Markov Chain, which is a concept known as the memory-less property.

1.5 MCMC and the Metropolis-Hastings Algorithm

By combining the ideas of Bayesian Statistics, Monte Carlo sampling, and Markov chains, Markov Chain Monte Carlo (MCMC) methods can be understood. The formal definition of MCMC, taken again from the *Monte Carlo Statistical Methods* textbook by Christan Robert and George Casella, is as follows:

Definition: A *Markov chain Monte Carlo* (MCMC) method for the simulation of a distribution f is any method producing an ergodic Markov chain $(X^{(t)})$ whose stationary distribution is f [**?**].

This definition can be explored in more depth by looking at one of the most popular MCMC algorithms, the Metropolis-Hastings (MH) algorithm.

This algorithm usually begins with a target distribution or probability density function (pdf),

$f(x)$, that is defined on a given sample space, S, from which samples can not be easily drawn [**?**]. This type of pdf is called intractable. To use the MH algorithm, a conditional density function, $g(x|y)$, also must be defined. In an ideal world, the function $g(x|y)$ is easy to draw samples from, and even better, is symmetric. The general idea of the algorithm is to develop a Markov chain that is both ergodic and stationary. An ergodic chain is one in which it is possible to get to every point from every other point. Stationary means that if $X^{(t)} \sim P$ then $X^{(t+1)} \sim P$ meaning that the chain will converge to P [**?**]. This can be accomplished by creating a transition kernel for the Markov chain that is based on the stationary distribution. The algorithm itself is fairly simple and best understood by looking at its steps:

1. Begin by selecting an arbitrary value $x^{(t)}$

2. Next, generate a value Y_t from the conditional probability function $g(y|x^{(t)})$. This is the step in which the Monte Carlo simulation takes place.

3. To determine the next value in the chain, $x^{(t+1)}$ a formula for the acceptance probability is used:

   - For asymmetric conditional distributions:

$$\rho(x^{(t)}, y) = \min\{\frac{f(y)}{f(x^{(t)})}\frac{g(x^{(t)}|y)}{g(y|x^{(t)})}, 1\} \tag{1.7}$$

   - For symmetric conditional distributions (which is preferred):

$$\rho(x^{(t)}, y) = \min\{\frac{g(x^{(t)}|y)}{g(y|x^{(t)})}, 1\} \tag{1.8}$$

4. Take the next value in the chain, $x^{(t+1)}$. This is the transition kernel of the Markov chain:

$$x^{(t+1)} = \begin{cases} Y_t & \text{with probability } \rho(x^{(t)}, y) \\ x^{(t)} & \text{with probability } 1 - \rho(x^{(t)}, y) \end{cases} \tag{1.9}$$

There are two important concepts to note in the above algorithm. The first is that the choice of the initial value can truly be arbitrary. Due to the memory-less property of a Markov chain, the first value will not impact the values thereafter [**?**]. While the beginning of the chain may not resemble the target distribution due to the choice of the initial value, there are methods to remove this portion of the chain, which will be explored later. The second important note is that while each value in the chain is dependent on the one before it, the values should all resemble a sample from the target distribution, P, though the sample will not be iid.

The motivation behind this thesis comes from the issues in current MCMC algorithms and their resulting chains when exploring complex distributions. The issue with these algorithms is that with complex distributions, it oftentimes takes hundreds of thousands of iterations in order to fully explore, and thus get an accurate simulation of, the target distribution P. In order to solve this issue, post-processing methods have been developed to take the output of an MCMC algorithm and decrease its size without increasing its bias or compromising its variance.

This paper will focus on the complex distributions that arise when determining the value(s) of β coefficients in linear regression models. There will be a review of common post-processing methods in which newer solutions, such as Stein-Thinning and the use of a Zero-Variance Control Variates, will be explored in depth and will be tested on linear regression models. The hope is that this research will provide future statisticians with the tools necessary to create efficient post-processing procedures for linear regression MCMC simulations.

Chapter 2

Literature Review of Post Processing Methods

2.1 Basic Post-Processing Methods

In order to solve some of the issues of MCMC algorithms discussed above, post-processing methods have been developed to take the output of an MCMC algorithm and shrink it to a reasonable size while attempting to improve either the bias, variance, or Mean-Squared Error (MSE) of the chain when compared with the ground truth values.

One such method is burn-in removal, which is where the value, N, at which the MCMC chain has converged to the target distribution is estimated. The points before N are then removed. Burn-in succeeds at removing the bias that could be caused by a bad initial value; however, it risks increasing variance as the number of values remaining may be very low [**?**].

Another proposed method is thinning, in which every k^{th} value is kept and the rest removed in order to remove the correlation between successive values of the chain. This method succeeds in reducing the autocorrelation in the resulting sample; however, it does not necessarily reduce the error of the output [**?**]. Thus, from the results of these two methods, it is clear that neither of them succeeds in successfully addressing the bias-variance trade off.

2.2 Stein-Thinning

This section aims to discuss an improvement in post-processing for MCMC chains and one of the most recent post-processing methods for MCMC output, which is called *Stein-Thinning*. In order to understand the motivation for Stein-Thinning and its predecessor, Kernel-Thinning, this section will begin by taking a look at the field of computational cardiology.

To better predict disease progression in the human heart and understand responses to therapy in non-invasive or damaging ways, researchers aim to develop a sort of digital twin of the heart [**?**]. In order to do this, researchers would have to be able to create an accurate simulation of the heart. This can be done by starting small. The heartbeats in an entire human heart are “coordinated by the calcium signaling in heart cells” [**?**]. If researchers start by simulating the calcium signaling in

heart cells, they could then build an upstream simulation of the heart using many of these smaller simulations. A strong simulation of the overall heartbeats in a human heart would allow researchers to better understand the health of the heart [**?**].

In order to create even the low-level model of calcium signaling, MCMC chains with tens of thousands of values are needed. This means that creating a full model of a human heart would take large quantities of these simulations, leading to thousands of hours of CPU time to create the actual simulation. The time-based limitations of this model could be improved if there was a way of getting equally as accurate but much smaller MCMC chains to simulate the calcium signaling in heart cells. This is where a post-processing method such as Stein-Thinning finds its motivation.

The goal of Stein-Thinning is to take an MCMC output and compress it into as small of a sample as possible, while still equally as accurately summarizing the target distribution, P. Before discussing Stein-Thinning itself, a theoretical understanding of discrepancy functions and the kernel Stein-Discrepency is required.

"A discrepancy is defined as a bivariate function D such that $D(P, P) = 0$ for all distributions P, and $D(P, Q) > 0$ for all $P \neq Q$." [**?**]. This essentially measures the dissimilarity between 2 distributions. One important property of a discrepancy function is called convergence control. A discrepancy function, D, as defined above, has convergence control when the function $D(P, Q_M) \to 0$, where Q_M is an output chain with M values and P is the target distribution, implies that Q_M converges to P, but the way in which it does needs to be explained in the equation [**?**].

The goal, then, is to find a way to calculate the discrepancy between an intractable distribution P and our output distribution, in order to create an optimization problem in which we select the next best point from our output that further minimizes the discrepancy between our compressed sample and target distribution.

This is where the idea of the Stein Discrepancy comes into play. The Stein Discrepancy was defined first by Gorham and Mackey (2017) and relies heavily on Stein's Method [**?**] [**?**]. This method defines a set of sufficiently differentiable vector fields with dimension d, which will be

called F. A differential operator, $\mathcal{L}$ is also defined as which acts on F in a way that $\int_{R^d} \mathcal{L}_p f \; dP = 0 \; \forall f \in F$ [**?**]. With this selection of a vector field and differential operator, the Stein Discrepancy can be defined as:

$$D_{\mathcal{L}_P F}(P, Q_M) = \sup_{f \in F} | \int_{R^d} \mathcal{L}_p f dQ| \tag{2.1}$$

From this definition, the kernel Stein Discrepancy (KSD) can be defined. Before explaining the derivation of this function, which is outlined in Gorham and Mackey (2017), it is necessary to define three terms used in the derivation [**?**]. First, a Hilbert space, $\mathcal{H}(k)$ is a vector space that is a complete inner product space. Second $\nabla \cdot$ is the divergence operator in R^d. Third, $G := \{g : R^d \rightarrow R^d | \sum_{i=1}^{d} ||g_i||^2_{\mathcal{H}(k)} \leq 1\}$ "is the unit ball in a Cartesian product of" a reproducing kernel Hilbert space (RKHS) [**?**]. When defining the KSD, $\mathcal{L}_p$ is the Langevin Stein Operator $\mathcal{L}_p g := p^{-1} \nabla \cdot (pg)$ [**?**] [**?**]. It then can be found that the kernel of $\mathcal{L}_p g$ is:

$$\begin{aligned} k_P(x, y) := \nabla_x \cdot \nabla_y k(x, y) + \langle \nabla_x k(x, y), \nabla_y log(p(y)) \rangle \\ + \langle \nabla_y k(x, y), \nabla_x log(p(x)) \rangle + k(x, y) \langle \nabla_x log p(x)), \nabla_y log(p(y)) \rangle \end{aligned} \tag{2.2}$$

As can be seen in Equation (2.2), one of the main requirements of the KSD is to compute $\nabla log(p)$ which is the gradient of the log of the probability distribution, p, of the target distribution P. There are certain methods of computing an MCMC chain where $\nabla log(p)$ is computed as a byproduct, which is something that will be returned to in Chapter 3.

The benefit of using the Stein Discrepancy in the post-processing of an MCMC output is that it allows for the computation of the kernel Stein Discrepancy function, $D(P, Q_M)$ mentioned above and also has convergence control on the discrepancy function. These benefits of using the Stein Discrepancy create an optimization problem which is called Stein Thinning [**?**]. The goal of this optimization problem is outlined by Riabiz (2022) and is to greedily produce a list of indexes, π, for which the kernel Stein Discrepancy is minimized [**?**]. The values, j of the index set π are

selected based on the KSD defined in Equation (2.2). The equation used to select the j values for an m length MCMC chain is based on the prior $j-1$ values and is defined below:

$$\pi(j) \in \underset{i \in \{1,\ldots,N\}}{\operatorname{argmin}} \frac{k_P(X_i, X_i)}{2} + \sum_{j'=1}^{j-1} k_P(X_{\pi(j')}, X_i) \tag{2.3}$$

To compute an N-length Stein-thinned chain from an M-length MCMC output, the complexity of the above equation would be $O(NM^2)$ or less if states are repeated. As defined by Riabiz (2022), the benefits of such an optimization algorithm are that (a) the KSD will guarantee that Q_M will weakly converge to P, (b) the algorithm directly addresses the bias-variance trade off that was not addressed by burn-in removal or k-thinning, (c) the algorithm performs burn-in automatically through its selection process, and (d) the algorithm can correct for systematic bias in the output through its selection process [**?**] [**?**]. As defined by Wenliang (2020), this algorithm will see issues when the dimensionality of P is too high $(d > 100)$ and when P has some far-off or hard-to-find distribution regions [**?**]. These issues, however, will not be explored in this paper as we analyze P with low dimension and with fairly condensed distribution regions.

2.3 Zero-Variance Control Variates

The second post-processing method that will be evaluated within this paper will be the use of Zero-Variance Control Variates (ZVCV) on MCMC output. This method aims to minimize the variance of MCMC estimations. ZVCV will be compared to Stein-Thinning to get a better understanding of how best to reduce the variance of MCMC output in a Linear Regression setting. Thus, a brief overview of ZVCV will be provided before discussing the research methods of this paper.

In this section, the goal is to reduce the variance of a vanilla MCMC estimator, $\hat{\mu}$, by applying ZVCV. To start, the equation for a simple control variate is provided below:

$$\hat{\mu}_{CV} = \frac{1}{N}\sum_{i=1}^{N}[f(\theta_i) - \tilde{f}(\theta_i)] + \int_{\Omega}\tilde{f}(\theta)\pi(\theta)d\theta \tag{2.4}$$

Where $f(\theta)$ is the target distribution and $\tilde{f}(\theta)$ would be a function that is similar to the target, which could be the output of a MCMC chain [**?**] [**?**]. The three necessary properties of the above control variate equation are as follows:

1. The ability to evaluate $\int_{\Omega}\tilde{f}(\theta)\pi(\theta)d\theta$
2. The ability to compute $\tilde{f}$ at all values of θ
3. The variance of $(f - \tilde{f})$ being significantly less that the variance of f

These properties can be satisfied when using the Langevin Stein Operator, $\mathcal{L}$, which is discussed in the above section as a necessary portion of the kernel Stein Discrepancy [**?**]. The way this is done is to base $\tilde{f}(\theta)$ off of the equation below:

$$\tilde{f}(\theta) = \mathcal{L}g(\theta) \tag{2.5}$$

Where $g(\theta)$ is a function selected by the user. In this, $\mathcal{L}$ has zero expectation under the probability density function, π, of the target [**?**]. Then Equation (2.4) can be simplified to the following:

$$\hat{\mu}_{CV} = \frac{1}{N}\sum_{i=1}^{N}[f(\theta_i) - \mathcal{L}g(\theta_i)] \tag{2.6}$$

The next important step is to select the function g, for which their are many proposed methods. The strategy that is utilized in the methods section of this paper is to choose $\mathcal{G}$ to be a "class of Qth order polynomials" [**?**]. The formulas below outline how this is done. To start, $g(\theta)$ can be represented as a polynomial function, $P(\theta)$ where:

$$\tilde{f}(\theta) = \mathcal{L}P(\theta) \tag{2.7}$$

The polynomial can then be simplified as a sum of monomial terms, which can be done in this case because the Stein Operator that is used is a linear differential operator [**?**][**?**]. This is captured below:

$$\tilde{f}(\theta) = \sum_{j=1}^{J} \beta_j \mathcal{L}P(\theta) \tag{2.8}$$

This can be further simplified as follows:

$$\tilde{f}(\theta) = \beta^T \mathbf{x}(\theta) \tag{2.9}$$

Where β are the coefficients of the polynomial and **x** is a vector that contains terms involving θ and the log gradient of the probability density function of the target, $\nabla_\theta log\pi(\theta)$ [**?**]. This looks like, and is very similar to the idea of linear regression as the goal of this is to compute a function, $\tilde{f}$, which is as similar as possible to the target function, f [**?**]. This essentially performs linear regression on f to compute $\tilde{f}$. Currently, the approach to estimate β involves using Ordinary Least Squares Regression (OLS) by the following equation [**?**][**?**]:

$$(\hat{\beta}) \in \underset{\substack{a\in\mathbb{R} \\ \beta\in\mathbb{R}^J}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} [f(\theta_i) - \beta^T \mathbf{x}(\theta_i)] \tag{2.10}$$

The final estimator from the use of ZVCV with the selection of $\mathcal{G}$ being the set of Qth order polynomials is as follows:

$$\hat{\mu}_{ZVCV} = \frac{1}{N} \sum_{i=1}^{N} [f(\theta_i) - \hat{\beta}^T \mathbf{x}(\theta_i)] \tag{2.11}$$

While the mathematics behind this theory are fairly complex, its implementation is very simple and will be reviewed in the next section. The most important piece of ZVCV to understand is that it utilized the Langevin Stein Operator in order to use zero variance control variates for MCMC output. This will be used to determine how well Stein-Thinning lowers variance in its thinned output.

Chapter 3

Methodology

3.1 Methodology Focus

The remainder of this paper will assess the performance of Stein Thinning as a post-processing method for MCMC Linear Regression output. Linear regression is one of the most popular and common Machine Learning techniques. Given a continuous dataset that, based on data exploration, seems to be linear, a linear regression model aims to determine the specific relationship between the independent and dependent variables of the dataset. Given a data set with a vector of independent variables, $\mathbf{x}$, and a vector of dependent variables, $\mathbf{y}$, a linear regression model aims to determine a vector of coefficients, β, such that, $\mathbf{y} = \alpha + \mathbf{x} * \beta$, where α is the intercept.

The goal of using MCMC in a linear regression setting is to generate a posterior distribution of the coefficients, β, to predict their value. In order to develop accurate posterior distributions of these coefficients, it is necessary to build massive MCMC chains, which are computationally inefficient to work with. For this reason, Stein Thinning will be applied to the output chains and the results will be compared to the raw chains and to estimates obtained using Zero-Variance Control Variates.

3.2 Linear Regression Data

The dataset used within this research is a Graduate School Admissions dataset created by Akshay Dattatray Khare on Kaggle [**?**]. This basic linear regression dataset aims to predict a student's chances of getting admitted into higher education following undergraduate studies. There are 400 entries in the dataset, each with 8 numeric variables and one binary variable. A description of each variable is provided below in Table 3.1:

Variable	Description
GRE Score	Student's score on the GRE Exam (0 to 340)
TOEFL Score	Student's score on the TOEFL Exam (0 to 120)
University Rating	Rating of the University being applied to (out of 5)
SOP	Ranking of student's Statement of Purpose (out of 5)
LOR	Ranking of student's Letter of Recommendation (out of 5)
CGPA	Student's Cumulative Undergraduate GPA
Serial Number	ID of each Student
Research	Binary variable indicating whether a student did (1) or did not (0) perform research
Chance of Admit	Percent chance of being admitted into the desired university graduate program (0 to 1)

Table 3.1: Description of Features in Graduate School Admissions Dataset

This dataset's response variable, and thus the variable of interest, is "Chance of Admit". In this paper, Stein Thinning for MCMC linear regression aims to be tested on 2- and 4-variable linear regression models. For this reason, the dataset will be condensed into two new datasets, which contain 2 and 4 explanatory variables and the response variable. This is in order to decrease the complexity of the required model and get a better understanding of the performance of Stein Thinning. The explanatory variables are chosen using feature selection, which is performed by doing LASSO (Least Absolute Shrinkage and Selection Operator) regression [**?**]. LASSO regression is performed using the `glmnet` package in R [**?**]. The output of performing LASSO Regression on the above data is provided below in both Figure 3.1 and Table 3.2.

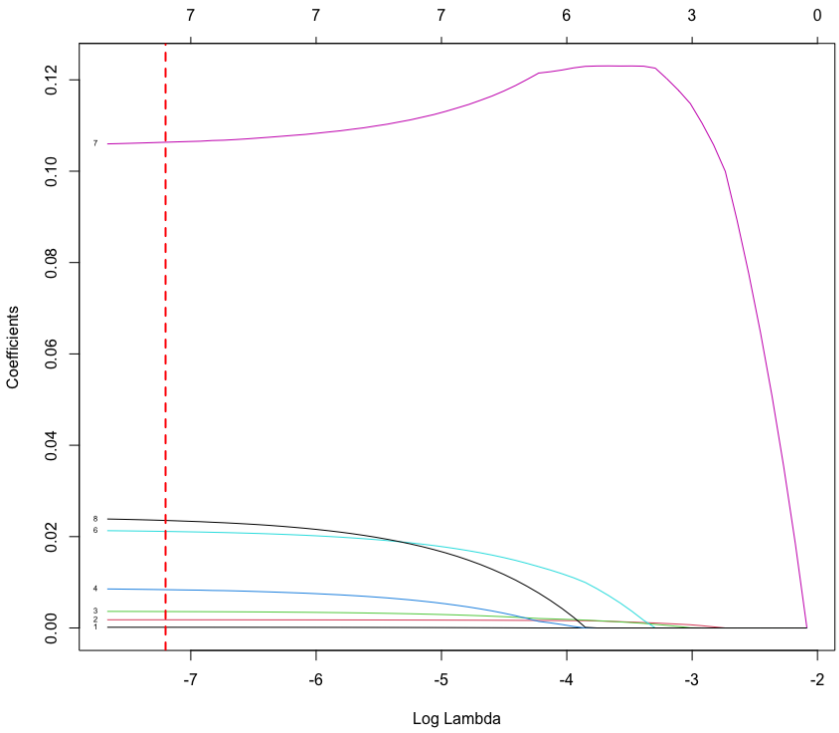

Figure 3.1: Graph showing the predictive power of each variable for the best lambda value found during cross validation

Variable	**Predictive Power**
Serial.No.	0.0001510178
GRE.Score	0.0017895667
TOEFL.Score	0.0035954292
University.Rating	0.0083849461
SOP	0.0
LOR	0.0211374393
CGPA	0.1063382407
Research	0.0235382003

Table 3.2: Predictive power of each variable found during LASSO Regression

It can then be seen that through feature selection, the 4 variables with the most predictive power are: “CGPA,” “Research,” “LOR,” and “University.Rating”. These will be used to create the 2 and 4-variable datasets used in the remainder of this paper. After selecting the variables of interest, the explanatory and response variables will be standardized to a mean of 0 and a standard deviation of 1. This is because we are not interested in the values of the data itself but rather in how the β

coefficients impact the data. This will make the output information regarding the β coefficients much easier to interpret.

3.3 Vanilla MCMC Function

The Vanilla MCMC function, or just the regular MCMC function, that is used is written in R utilizing Stan, which is a probabilistic programming language. The function itself is based on the Stan user guide and the recommendations of priors provided within that guide. The function is comprised of various components: the model code, the MCMC chain itself, and an extraction function, all of which will be detailed in this section

The model code is comprised of three sections: the description of the data, the parameters, and the model itself. In the data section, the explanatory variables, x, are defined as N x K matrices, whereas the response variable, y, is defined as a 1 x N vector. The parameters of the model are the constant, α, the 1 x K vector, β, and the random noise constant, σ. The variables N and K are defined outside of the model in a vector that is used in combination with the model code and is based on the input data frame. N is defined as the observations in the data frame and K is defined by the features of the data. The model section of the model code defines the prior distribution put on the β coefficients and the likelihood function for the MCMC. The prior distribution for the β coefficients is a Student's t-distribution, $t_v(3, 0, 0.5)$, which is the prior that is recommended for regression models with little prior information by the Stan developers, which they base off of Ghosh (2018) [**?**]. The likelihood function follows a $N(\mathbf{x} * \beta + \alpha, \sigma)$ distribution. This model code, as a whole, is based on the recommended choices outlined by the Stan User Guide and the Prior Choice Recommendations created by the Stan developers. The code for this model can be found in the Appendix of this paper.

After defining the model and the data that makes up that model, these pieces can be used to develop a `fit` object using the `stanfit` class. The method used to do this is called `stan` and takes four arguments: `model_code`, `data`, `iter`, and `chains`. In this case, the model code

and data are defined above, and the number of iterations, also known as the length of the chain, will be determined below. The chains will always be set to 1 as we do not need multiple chains.

Once a `fit` object has been produced, the Stan function `extract` is called in order to grab the MCMC sampled points from the `fit` object. This function takes three arguments: `object`, `permuted`, and `inc_warmup`. The `object` is the `fit` object developed above. The argument, `permuted`, takes a boolean and dictates whether the warmup samples in the chain should be dispersed with the rest of the samples. We will be consistently setting this to `False`. The `inc_warmup` argument simply asks if the warm up samples should be included, which will be changed for different runs in this paper.

3.4 Ground Truth

When testing a new post-processing method for an MCMC chain, it is important to be able to calculate the MSE, bias, and variance of the new chain when compared to the ground truth values. In this case, it is not possible to determine the actual ground truth values of the beta coefficients. In order to get the best estimate of these values, an MCMC chain will be computed with 1,000,000 values. The resulting chain will be averaged for each of the β coefficients and will be compared to the Stein Thinned values. This is the best method of understanding the performance of the stein-thinned chains.

3.5 Stein Thinning Function

While the ground truth chain will be computed using 1,000,000 values, the Stein thinned chains will be computed using just 10,000 values. This aims to show the power of Stein Thinning in extracting value from smaller samples. In order to actually perform Stein Thinning on the MCMC Model discussed in section 3.2, an R package called `R.stein.thinning` will be used [**?**]. This package was created by Riabiz (2022), and all of the methods within the package are based on their research, which was discussed previously in section 2.2 [**?**].

Two steps are used to thin the resulting samples from the MCMC function based on the Stein Thinning approach. The first step is to compute $\nabla log(p(a_i))$ for each sample in the chain $a_1, ..., a_n$. This is the gradient of the log of the probability distribution, p, of the target distribution P for each sample a_i. This is required because $\nabla log(p(a_i))$ is a necessary component of the KSD equation, which can be seen in Equation (10) of Section 2.2. This can be done using the `grad_log_prob` function from the `stan` package. This takes in the `fit` object computed earlier and produces the log posterior gradients, which is exactly what we need to compute the KSD.

The second step of the Stein Thinning approach using `R.stein.thinning` is using the `thin` function to perform the optimization algorithm on which Stein Thinning is based. This function takes in the sampled points extracted in the MCMC function, the log posterior gradients, and the number of points, `m`, desired in the thinned sample. This will produce the Stein Thinned sample of points of length, `m`, which can then be averaged and compared to the ground truth values and Vanilla MCMC values in order to determine the success of Stein Thinning for Linear Regression algorithms.

3.6 Zero-Variance Control Variate Function

The ZVCV values will be computed based on the same 10,000 value chains as the Stein thinned samples discussed above. These values are computed utilizing the research done by L.F. South et al. (2023) which they compiled into a R Package called `ZVCV` [**?**].

The computation of the ZVCV values relies on the `zvcv` function within the previously mentioned R Package. This function takes in the values of an MCMC chain that are being estimated, the full sample set from the MCMC chain, and the log posterior gradient values of the sample set. The last set of values is easily computed using the `grad_log_prob` function from the `stan` package as mentioned in the previous section. Once these three values are collected through running MCMC and the needed `stan` functions, the `zvcv` function can be run. This function outputs a set of values, but the important one for the purposes of this paper is the `expectation` value.

This is the estimate found using ZVCV as described in Section 2.3. This output is the mean ZVCV value for the MCMC chain.

Chapter 4

Results

4.1 Testing Procedure

Now that the code for the LASSO Regression, MCMC function, Stein Thinning function, and the ZVCV function have been outlined, the procedures for actually testing Stein Thinning on MCMC Linear regression output can be explored. To start, there are two files which contain two different Stein Thinning, ZVCV and MCMC functions for 2 variable and 4 variable Linear Regression models. In each of these files, the Stein and ZVCV functions take in a Stein sample size and boolean parameter. The sample size will dictate the size of the thinned sample, and the boolean will dictate whether or not to include the warm up in the MCMC sample prior to thinning it and computing the ZVCV. The values for the sample variable, `samp`, will be `{50, 100, 200}`, and the values for the `bool` variable will be `{True, False}`.

Each combination of `samp` and `bool` is one combination, denoted c. $\mathcal{C}$ is comprised of a list of combinations where, $\mathcal{C} = [(50, True), (50, False), ...(500, True), (500, False)]$. Each combination will be run $n = 100$ times and for each sample $i = 1, ..., n$ the average $\hat{\beta}_i$ values will be computed and denoted, $\hat{\mu_i}$.

The variables μ_i will represent the ground truth values within this simulation. These are the average values of each β_i found in the MCMC chain of one million values. The variables $\hat{\mu}_{s_i}$ represent the average values of the β_i found in each Stein Thinning iteration. The variable $\bar{\mu}_{s_c}$ represent the average values of the $\hat{\mu}_{s_i}$ found in each Stein Thinning combination. Similarly, $\hat{\mu}_{z_i}$ will represent the average values of the β_i found in each ZVCV iteration and $\bar{\mu}_{z_c}$ will represent the average values of the $\hat{\mu}_{z_i}$ found in each ZVCV combination. Finally, $\hat{\mu}_{v_i}$ will represent the average values of the β_i found in each Vanilla MCMC iteration and $\bar{\mu}_{v_c}$ will represent the average values of the $\hat{\mu}_{v_i}$ found in each Vanilla MCMC combination.The ground truth will be computed first, followed by iterative testing of the Stein Thinning, ZVCV and Vanilla MCMC functions.

Once the samples have been collected from the 10,000 value MCMC chains for Vanilla MCMC, Stein Thinning and ZVCV, these values will be averaged, as described above, and will be compared with the values obtained from the ground truth 1 million value MCMC chain. The bias, variance

and MSE for each $\bar{\mu}_i$ will be computed and used to compare the three approaches to the ground truth and to one another.

4.2 2 Variable Linear Regression Results

Before diving into the results, it is important to understand the true distribution of the variables of interest, namely β_1 and β_2 for 2 variable linear regression. The ground truth distributions of these variables are displayed in Figure 4.1 and Figure 4.2 below:

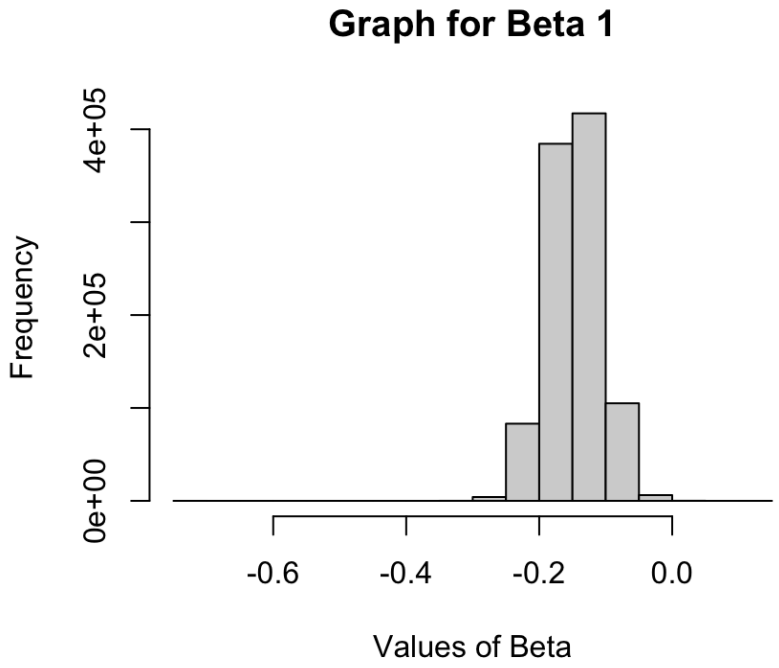

Figure 4.1: Histogram produced from the ground truth 1 million value MCMC Chain revealing the ground truth distribution of β_1

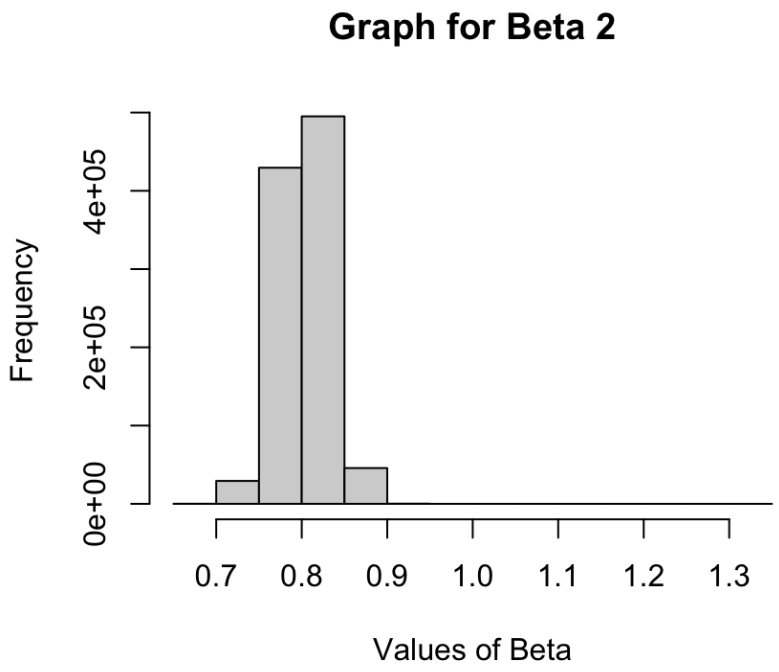

Figure 4.2: Histogram produced from the ground truth 1 million value MCMC Chain revealing the ground truth distribution of β_2

In the figures above, it can be seen that both β_1 and β_2 have approximate Normal distributions with low σ value. β_1 seems to have a μ value of approximately -0.15 and β_2 looks to have a μ value of about 0.8.

After determining the ground truth distributions, samples are obtained for the 2 variable linear regression model. The results below were computed for comparing Vanilla MCMC with 10,000 values to the ground truth MCMC with 1 million values.

ID	Bias	Variance	MSE	Stein Samples	Warmup
mu1	-8.500488e-04	9.942935e-07	1.716876e-06	50.00	TRUE
mu2	-2.006885e-03	1.777343e-06	5.804931e-06	50.00	TRUE
mu1	-3.451081e-04	2.291963e-07	3.482958e-07	50.00	FALSE
mu2	-2.010111e-03	1.404073e-06	5.444620e-06	50.00	FALSE
mu1	-7.788042e-04	9.253719e-07	1.531908e-06	100.00	TRUE
mu2	-2.130445e-03	2.062372e-06	6.601168e-06	100.00	TRUE
mu1	-3.483306e-04	3.246394e-07	4.459736e-07	100.00	FALSE
mu2	-1.917068e-03	1.436293e-06	5.111443e-06	100.00	FALSE
mu1	-7.256607e-04	1.281104e-06	1.807688e-06	200.00	TRUE
mu2	-2.191595e-03	1.741988e-06	6.545077e-06	200.00	TRUE
mu1	-3.635124e-04	2.768039e-07	4.089451e-07	200.00	FALSE
mu2	-1.812228e-03	1.447855e-06	4.732025e-06	200.00	FALSE

Table 4.1: Results From 2 Variable Linear Regression Vanilla MCMC Chains Compared With Ground Truth Data

The results in Table 4.1 were as expected. The inclusion of a warm up and the stein sample size did not effect the results, and they shouldn't have. These columns are meant for comparison purposed only, and will be utilized later on. In this case, since the comparison is between large chain and small chain MCMC, the results should contain very low bias, variance and MSE, across the board which the table above reveals. This mainly just reveals that the code is working as promised.

Next, the ZVCV are applied to the Vanilla MCMC chains from the results above, and Table 4.2 is produced.

ID	Bias	Variance	MSE	Stein Samples	Warmup
mu1	-1.010414e-01	5.929394e-02	6.950331e-02	50.00	TRUE
mu2	-2.912503e-02	1.881362e-02	1.966189e-02	50.00	TRUE
mu1	2.445493e-03	1.990674e-05	2.588717e-05	50.00	FALSE
mu2	1.125610e-02	1.492807e-05	1.416278e-04	50.00	FALSE
mu1	-1.690017e-01	5.517169e-02	8.373326e-02	100.00	TRUE
mu2	-4.683220e-02	2.494597e-02	2.713922e-02	100.00	TRUE
mu1	2.288499e-03	2.543244e-05	3.066966e-05	100.00	FALSE
mu2	1.098419e-02	1.266502e-05	1.333175e-04	100.00	FALSE
mu1	-1.604658e-01	6.881058e-02	9.455987e-02	200.00	TRUE
mu2	-3.032864e-02	4.082739e-02	4.174722e-02	200.00	TRUE
mu1	2.220888e-03	2.460727e-05	2.953962e-05	200.00	FALSE
mu2	1.066118e-02	1.409679e-05	1.277576e-04	200.00	FALSE

Table 4.2: Results From 2 Variable Linear Regression MCMC Chains Post-Processed With Zero Variance Control Variates Compared With Ground Truth Data

The above table is the result of attempting to use the MCMC results and produce average values for “mu1” and “mu2” with minimum variance when compared to the ground truth. It can be seen in the data that, with this goal in mind, the ZVCV approach performs quite well in reducing the variance, however, performs significantly better in the case where the warm up samples are not included in the MCMC output. Not including the warm up, or setting “Warmup” to “FALSE” is equivalent to performing burn in removal on the sample before producing the MCMC chain. Thus, these results make sense as the values removed during burn in would be the values produced as the algorithm searches for the ground truth distribution. As can be seen in Figures 4.1 and 4.2, the distributions have one hill, so once the algorithm finds the distribution it will stay their. Thus when burn in is removed better results are produced for ZVCV.

After the Vanilla MCMC and ZVCV results are produced, Stein Thinning results are computed and the output is displayed in table 4.3 below:

ID	Bias	Variance	MSE	Stein Samples	Warmup
mu1	-1.838074e-01	9.152684e-02	1.253120e-01	50.00	TRUE
mu2	-4.807169e-02	1.427989e-01	1.451098e-01	50.00	TRUE
mu1	-4.605959e-04	1.067062e-05	1.088277e-05	50.00	FALSE
mu2	-2.103955e-03	5.029856e-05	5.472518e-05	50.00	FALSE
mu1	-1.519135e-01	8.728036e-02	1.103581e-01	100.00	TRUE
mu2	-1.070638e-01	1.376168e-01	1.490795e-01	100.00	TRUE
mu1	-7.362941e-04	7.681230e-06	8.223359e-06	100.00	FALSE
mu2	-2.482687e-03	2.950685e-05	3.567059e-05	100.00	FALSE
mu1	-1.490659e-01	9.762930e-02	1.198500e-01	200.00	TRUE
mu2	-5.348578e-02	1.309750e-01	1.338357e-01	200.00	TRUE
mu1	-3.513470e-04	6.707580e-06	6.831025e-06	200.00	FALSE
mu2	-1.529068e-03	3.496777e-05	3.730582e-05	200.00	FALSE

Table 4.3: Results From 2 Variable Linear Regression MCMC Chains Post-Processed With Stein Thinning Compared With Ground Truth Data

As can be seen in the data above Stein Thinning, has lower or near equivalent variance when compared to the ZVCV results and performs a factor of 10 worse then the raw samples for MSE, Bias and Variance. Being that the sample sizes for Stein Thinning are 50, 100 and 200 compared to the 10,000 in the raw sample, this is better than expected. These results are visualized in the two plots provided below. They show Vanilla MCMC, Stein Thinning, and ZVCV mean-squared error results for 50, 100 and 200 stein samples for both μ_1 and μ_2.

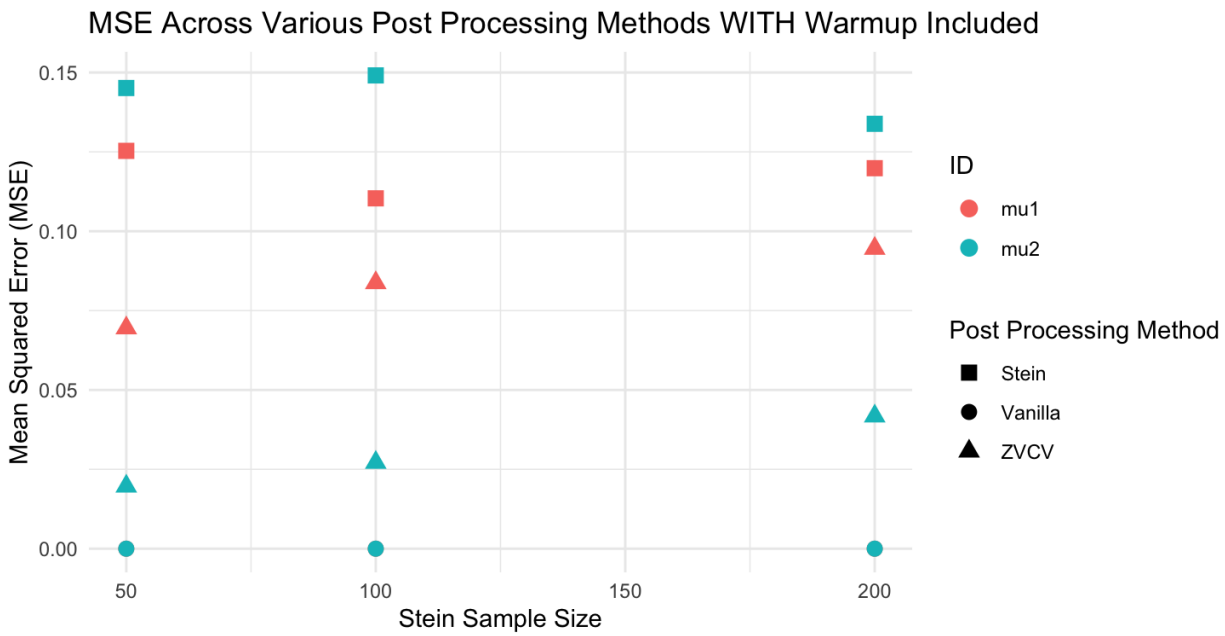

Figure 4.3: Dot Plot which visualizes the results of Vanilla MCMC, Stein Thinning and ZVCV for 2 variable linear regression where the warm up is included.

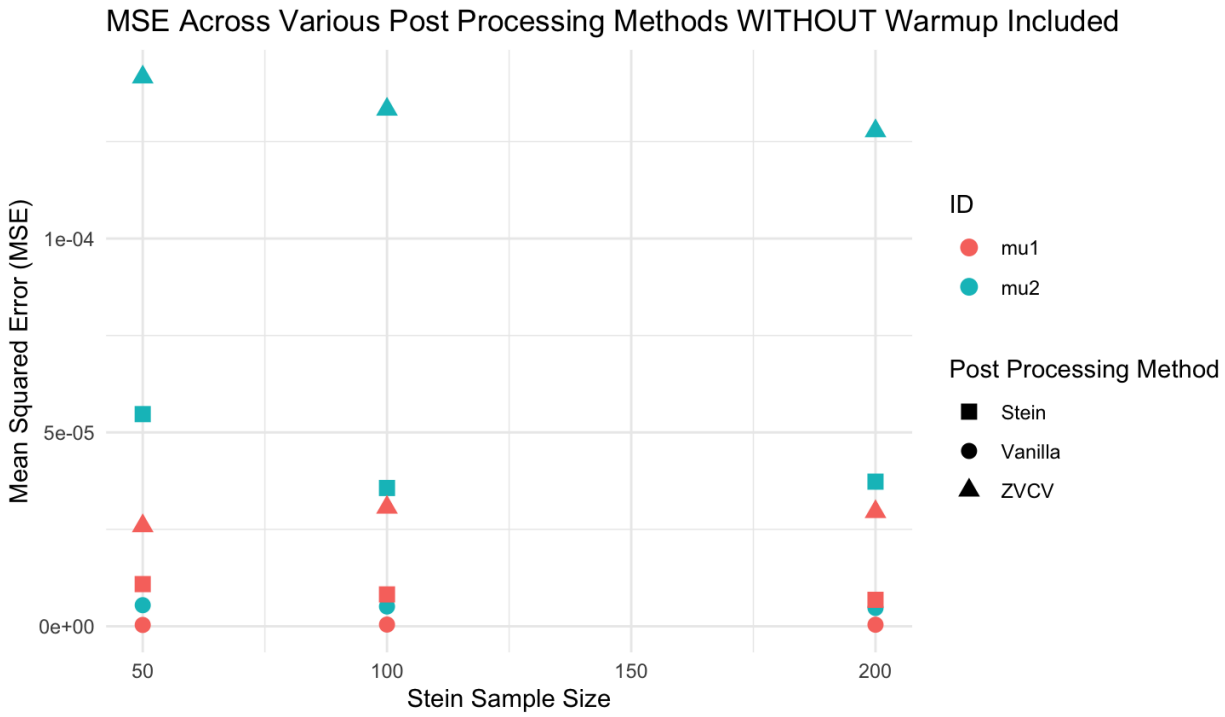

Figure 4.4: Dot Plot which visualizes the results of Vanilla MCMC, Stein Thinning and ZVCV for 2 variable linear regression where the warm up is NOT included.

In Table 4.3 and Figures 4.3 and 4.4 we also see that Stein Thinning performs significantly

better when the "Warmup" variable is set to "FALSE", meaning the warmup is NOT included. This is the same as the reasoning above. When Stein Thinning is performed on a MCMC that does not include the warm up, the Stein Thinned samples will not include points from outside of the ground truth distribution. This idea can be visualized in the two graphs below, which show Stein Thinning compared to the ground truth samples for 2 variable linear regression. In this case, 100 Stein Thinned points are extracted from a 10,000 value MCMC chain and plotted over the ground truth MCMC chain. Two plots are provided, one where the warmup was included in the MCMC chain and one where it was not.

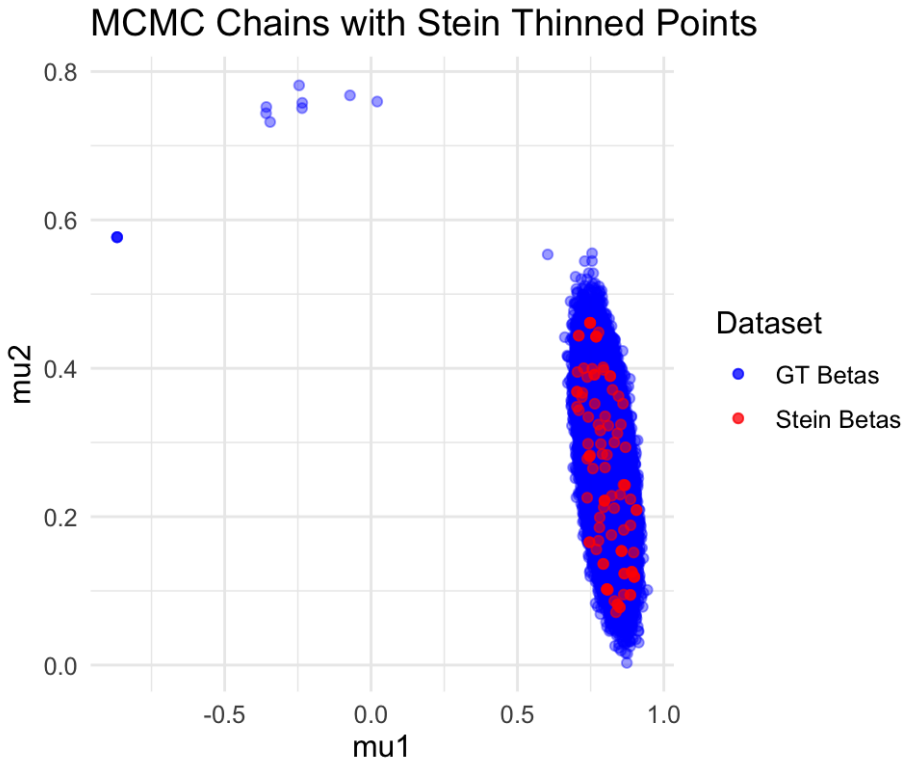

Figure 4.5: Scatter Plot that shows 100 Stein Thinned points produced without the warm up included in the original MCMC Chain plotted in RED over the 1 million Ground Truth points plotted in BLUE

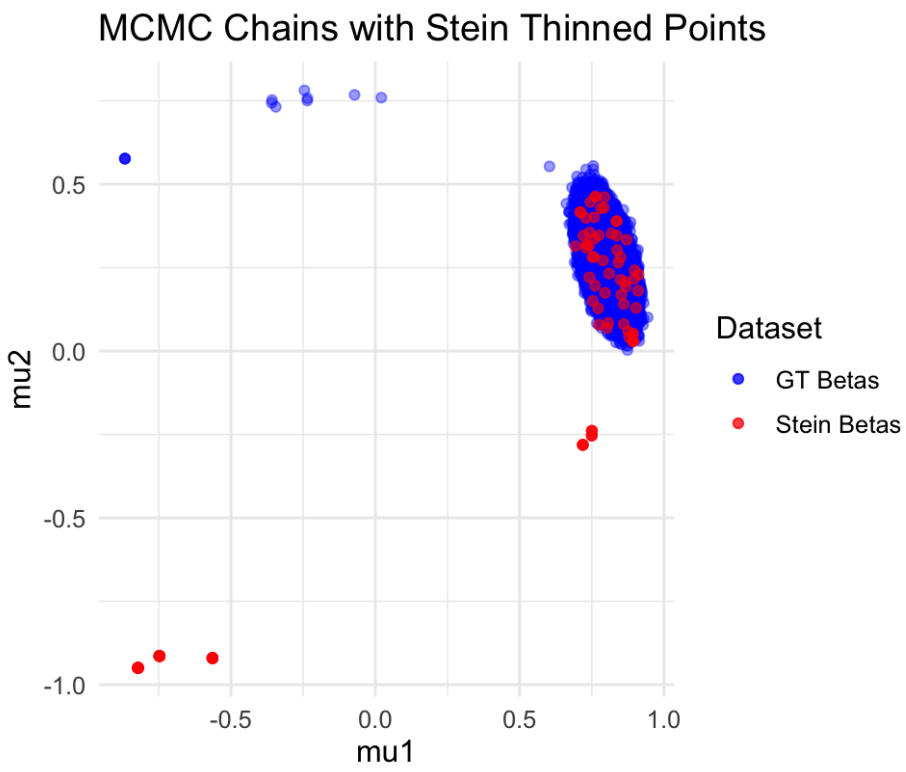

Figure 4.6: Scatter Plot that shows 100 Stein Thinned points produced with the warm up included in the original MCMC Chain plotted in RED over the 1 million Ground Truth points plotted in BLUE

In Figure 4.6 the Stein Thinned points have a burn in trail that is very dissimilar from the ground truth burn in trail, which is what leads to the higher values of Bias, Variance and MSE for the samples where the warm up *is* included. In Figure 4.5 the Stein Thinned points only cover the true distribution, which is why these samples produce such low values for Bias, Variance and MSE. To understand this idea even better, it helps to look at the stein thinned points versus the Vanilla MCMC points which is displayed in two plots below.

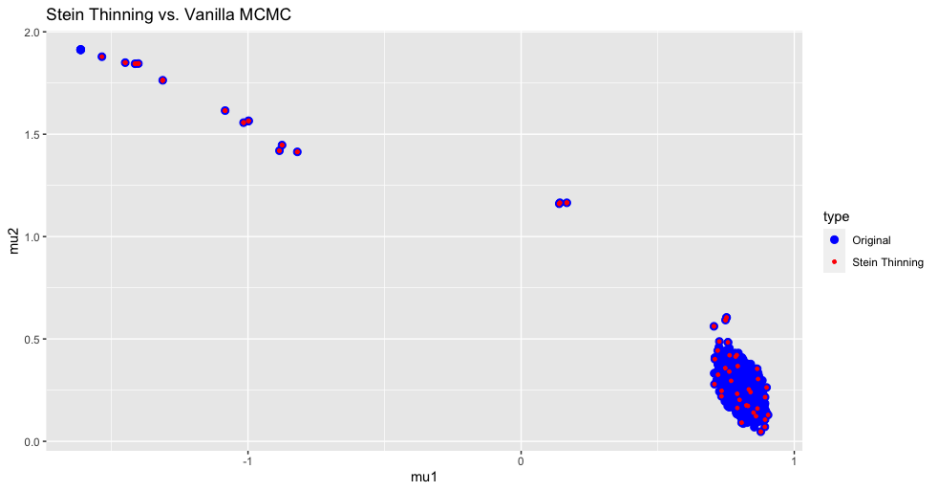

Figure 4.7: Scatter Plot that shows 100 Stein Thinned points produced with the warm up included plotted over the 10,000 Vanilla MCMC points

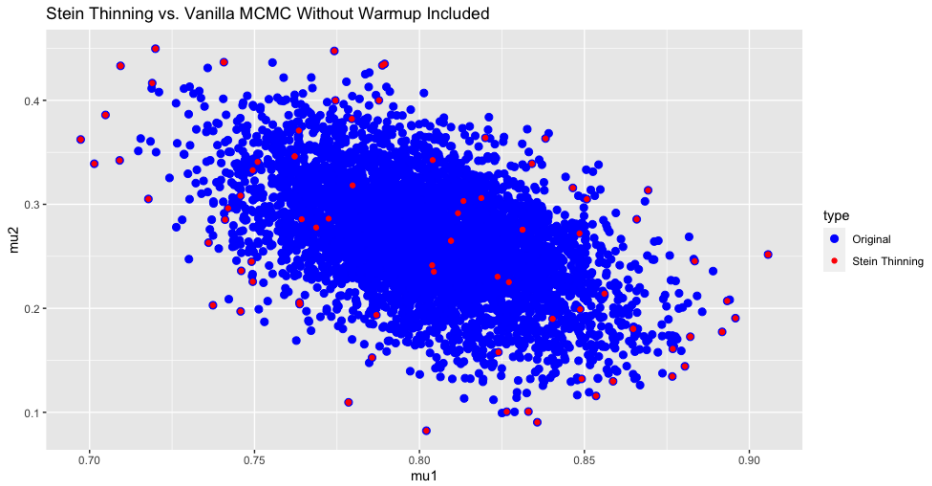

Figure 4.8: Scatter Plot that shows 100 Stein Thinned points produced without the warm up included plotted over the 10,000 Vanilla MCMC points

These plots reveal in better detail the points that the Stein Thinning algorithm selects from the burn in tail when the warm up samples are included in the initial 10,000 Vanilla MCMC chain. While Stein Thinning does not include all of the burn-in points, it includes enough to impact the MSE when compared to the ground truth chain, which has a different tail then the Vanilla MCMC chain. These plots both provide better insight into the MSE discrepancies that are seen in the results and reveal the effectiveness of Stein Thinning in reducing a 10,000 point plot to one of just 100 points.

4.3 4 Variable Linear Regression Results

The results for the 4 variable linear regression model were extremely similar to the results for the 2 variable linear regression model. For this reason, the results will not be explored as thoroughly in this section. Below in Table 4.4, Table 4.5, and Table 4.6 are the results for Vanilla MCMC, ZVCV and Stein Thinning respectively:

ID	Bias	Variance	MSE	Stein Samples	Warmup
mu1	-1.777650e-03	2.531867e-06	5.691905e-06	50.00	TRUE
mu2	-2.430845e-03	4.360022e-06	1.026903e-05	50.00	TRUE
mu3	-4.138680e-04	2.047716e-06	2.219003e-06	50.00	TRUE
mu4	1.129302e-03	2.397637e-06	3.672960e-06	50.00	TRUE
mu1	-1.512691e-03	6.437010e-07	2.931935e-06	50.00	FALSE
mu2	-1.703302e-03	6.583313e-07	3.559567e-06	50.00	FALSE
mu3	1.471860e-04	5.025235e-07	5.241872e-07	50.00	FALSE
mu4	1.164511e-03	3.420839e-07	1.698169e-06	50.00	FALSE
mu1	-1.883405e-03	2.852143e-06	6.399359e-06	100.00	TRUE
mu2	-1.744169e-03	4.371033e-06	7.413160e-06	100.00	TRUE
mu3	-4.680152e-04	1.752017e-06	1.971056e-06	100.00	TRUE
mu4	1.005528e-03	2.007839e-06	3.018926e-06	100.00	TRUE
mu1	-1.344505e-03	8.783446e-07	2.686038e-06	100.00	FALSE
mu2	-1.887622e-03	8.124386e-07	4.375554e-06	100.00	FALSE
mu3	1.222203e-04	4.803457e-07	4.952835e-07	100.00	FALSE
mu4	1.106547e-03	4.970698e-07	1.721516e-06	100.00	FALSE
mu1	-1.881893e-03	2.390101e-06	5.931622e-06	200.00	TRUE
mu2	-2.554878e-03	3.824351e-06	1.035175e-05	200.00	TRUE
mu3	-3.232455e-04	2.083628e-06	2.188116e-06	200.00	TRUE
mu4	1.244862e-03	1.673957e-06	3.223638e-06	200.00	TRUE
mu1	-1.218780e-03	4.569291e-07	1.942355e-06	200.00	FALSE
mu2	-1.797516e-03	6.785250e-07	3.909591e-06	200.00	FALSE
mu3	1.339254e-04	4.616969e-07	4.796329e-07	200.00	FALSE
mu4	9.547948e-04	2.536259e-07	1.165259e-06	200.00	FALSE

Table 4.4: Results from 4 Variable Linear Regression Vanilla MCMC Chains Compared with Ground Truth Data

ID	Bias	Variance	MSE	Stein Samples	Warmup
mu1	-9.607060e-02	5.143351e-02	6.066307e-02	50.00	TRUE
mu2	-1.610989e-02	7.835928e-02	7.861881e-02	50.00	TRUE
mu3	-4.345124e-02	4.974548e-02	5.163349e-02	50.00	TRUE
mu4	-8.315899e-03	3.322165e-02	3.329080e-02	50.00	TRUE
mu1	3.821114e-03	2.092526e-05	3.552617e-05	50.00	FALSE
mu2	1.151045e-02	1.240733e-05	1.448977e-04	50.00	FALSE
mu3	7.810795e-04	1.251207e-04	1.257308e-04	50.00	FALSE
mu4	-1.604534e-02	2.299883e-04	4.874412e-04	50.00	FALSE
mu1	-9.873425e-02	6.473701e-02	7.448546e-02	100.00	TRUE
mu2	6.527119e-03	1.112585e-01	1.113011e-01	100.00	TRUE
mu3	-5.867018e-02	4.405541e-02	4.749760e-02	100.00	TRUE
mu4	-1.090142e-03	4.607112e-02	4.607231e-02	100.00	TRUE
mu1	3.018555e-03	2.225699e-05	3.136866e-05	100.00	FALSE
mu2	1.111120e-02	1.531831e-05	1.387770e-04	100.00	FALSE
mu3	2.426775e-03	1.301135e-04	1.360028e-04	100.00	FALSE
mu4	-1.321044e-02	1.792124e-04	3.537281e-04	100.00	FALSE
mu1	-1.085522e-01	7.971575e-02	9.149932e-02	200.00	TRUE
mu2	-7.097481e-02	5.355695e-02	5.859437e-02	200.00	TRUE
mu3	-4.582703e-02	5.740861e-02	5.950873e-02	200.00	TRUE
mu4	2.329783e-02	4.076124e-02	4.130403e-02	200.00	TRUE
mu1	3.434795e-03	2.153681e-05	3.333463e-05	200.00	FALSE
mu2	1.107382e-02	1.846371e-05	1.410932e-04	200.00	FALSE
mu3	-4.787137e-04	1.573488e-04	1.575780e-04	200.00	FALSE
mu4	-1.344021e-02	1.486648e-04	3.293042e-04	200.00	FALSE

Table 4.5: Results from 4 Variable Linear Regression MCMC Chains Post-Processed with Zero Variance Control Variates Compared with Ground Truth Data

ID	Bias	Variance	MSE	Stein Samples	Warmup
mu1	-1.202405e-01	1.168097e-01	1.312675e-01	50.00	TRUE
mu2	-1.534913e-01	2.301343e-01	2.536939e-01	50.00	TRUE
mu3	-9.600361e-02	7.928202e-02	8.849872e-02	50.00	TRUE
mu4	1.298396e-02	1.013074e-01	1.014760e-01	50.00	TRUE
mu1	-9.735052e-04	1.596769e-05	1.691540e-05	50.00	FALSE
mu2	-2.022443e-03	3.086641e-05	3.495669e-05	50.00	FALSE
mu3	-2.328612e-04	1.016564e-05	1.021987e-05	50.00	FALSE
mu4	9.415071e-04	9.179918e-06	1.006635e-05	50.00	FALSE
mu1	-1.485675e-01	1.239745e-01	1.460468e-01	100.00	TRUE
mu2	2.090965e-02	2.229120e-01	2.233492e-01	100.00	TRUE
mu3	-1.219804e-01	7.735063e-02	9.222986e-02	100.00	TRUE
mu4	9.155775e-03	8.998549e-02	9.006932e-02	100.00	TRUE
mu1	-1.530580e-03	1.579258e-05	1.813526e-05	100.00	FALSE
mu2	-2.234988e-03	2.073857e-05	2.573374e-05	100.00	FALSE
mu3	5.704850e-04	1.136148e-05	1.168693e-05	100.00	FALSE
mu4	1.110121e-03	8.120180e-06	9.352549e-06	100.00	FALSE
mu1	-1.481631e-01	1.077802e-01	1.297325e-01	200.00	TRUE
mu2	-1.465972e-01	1.998645e-01	2.213552e-01	200.00	TRUE
mu3	-9.016573e-02	8.418856e-02	9.231842e-02	200.00	TRUE
mu4	4.181293e-02	8.045121e-02	8.219953e-02	200.00	TRUE
mu1	-1.566469e-03	1.307303e-05	1.552685e-05	200.00	FALSE
mu2	-1.085136e-03	2.222901e-05	2.340653e-05	200.00	FALSE
mu3	-3.902791e-04	8.841503e-06	8.993821e-06	200.00	FALSE
mu4	1.329109e-03	9.446193e-06	1.121272e-05	200.00	FALSE

Table 4.6: Results from 4 Variable Linear Regression MCMC Chains Post-Processed with Stein Thinning Compared with Ground Truth Data

As can be seen, all three approaches performed nearly identically as they did in the 2 variable linear regression case. The same behavior occurred when the warm up samples were included for both Stein Thinning and ZVCV in this case as well. It is difficult to create a visualization for this phenomena in the 4 variable case, as the data has over three dimensions. For that reason the R Package, `GGally` is used in order to create a matrix of scatter plots to visualize the various combinations of μ_1, μ_2, μ_3 and μ_4. The two figures are displayed below:

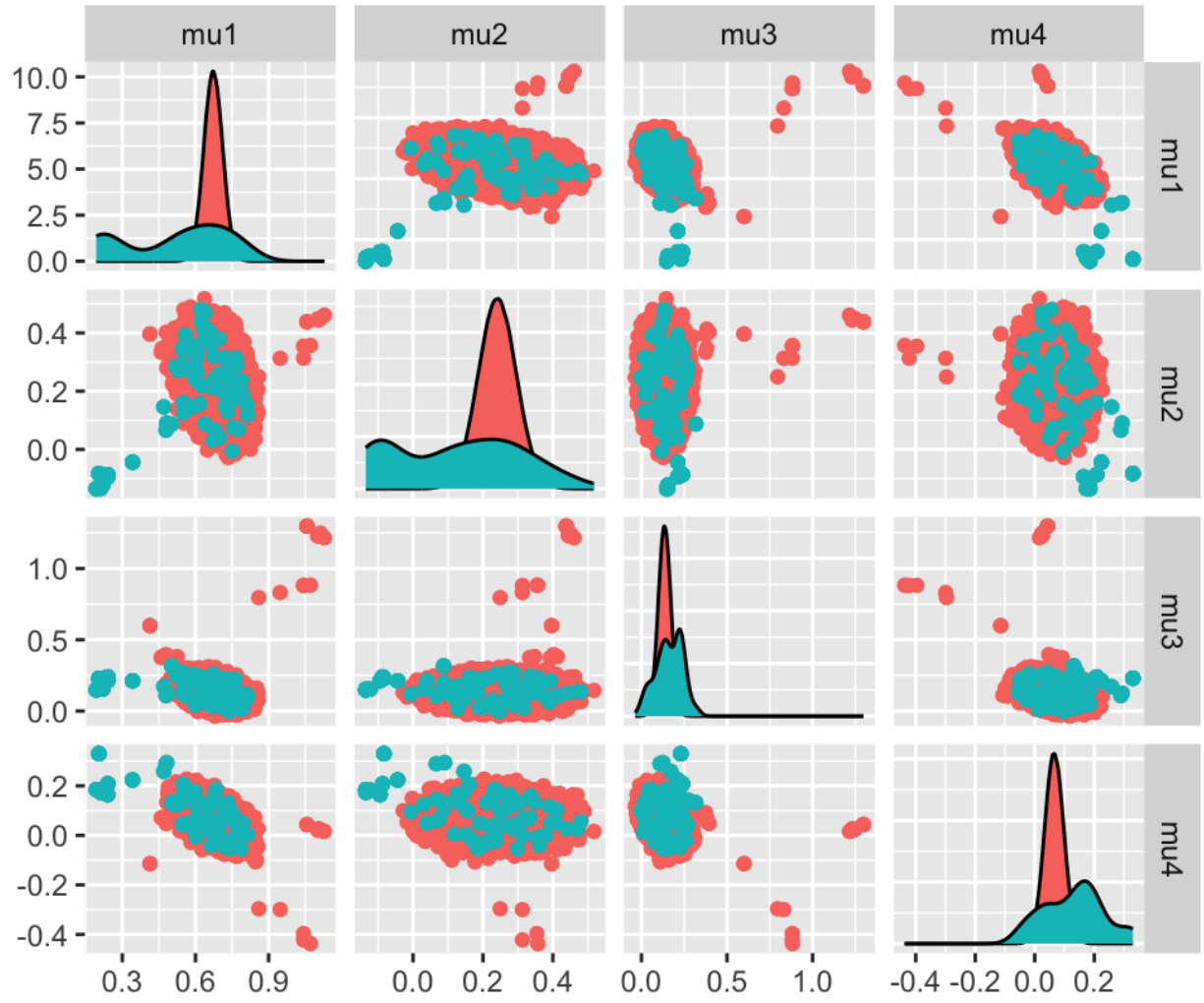

Figure 4.9: Scatter Plot that shows 100 Stein Thinned points produced with the warm up included in the original MCMC Chain plotted in ORANGE over the 1 million Ground Truth points plotted in BLUE.

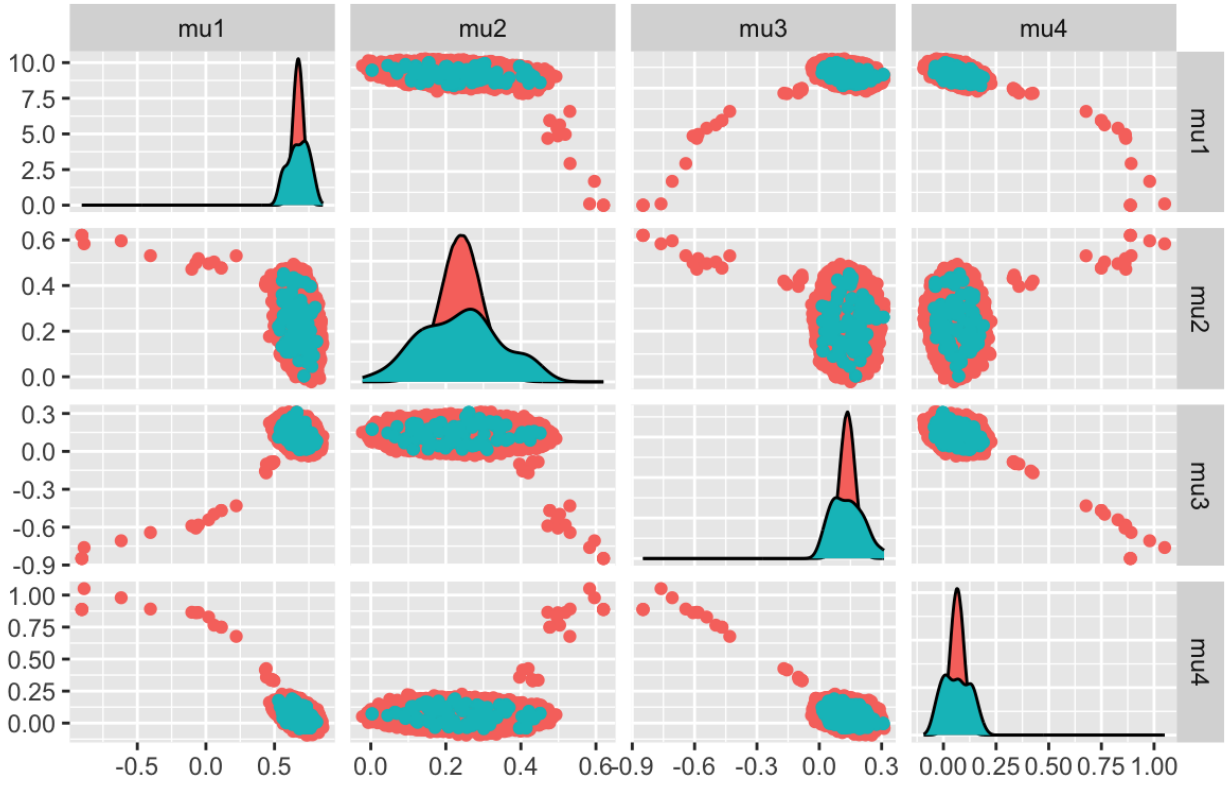

Figure 4.10: Scatter Plot that shows 100 Stein Thinned points produced without the warm up included in the original MCMC Chain plotted in ORANGE over the 1 million Ground Truth points plotted in BLUE.

Figures 4.9 and 4.10 show two different types of plots. The diagonal entries reveal the distribution of the given μ_i value for the ground truth sample compared to the distribution of the μ_i for the stein sample. The non diagonal entries show scatter plots that reveal the relationships between the four different μ_i values. These plots include the stein samples in **blue** over the ground truth samples in **orange**. In Figure 4.9 the viewer can see that in each scatter plot, the ground truth and stein samples have very different tails, which is identical to the 2 variable case and reveals why the MSE is so high in these cases. In Figure 4.10 the viewer can see that the stein thinned samples have *no* tail, revealing why the MSE is so low in these cases. These results are near identical to the plots that were seen in the 2 variable cases revealing a similar rationale for the worsened performance when warm up is included.

Chapter 5

Conclusions and Future Work

5.1 Conclusions and Future Work

In this paper, one of the most recently developed post-processing methods for MCMC output, Stein Thinning, is tested for MCMC Linear Regression output. The results of this testing are compared to Vanilla MCMC, ZVCV and Ground Truth MCMC values. We found that Stein Thinning obtains near equal variance when compared to ZVCV, which shows that the variance reduction of Stein Thinning lives up to its promises. We also found that when the warm up sample is not included in the MCMC sample to be thinned, Stein Thinning performs phenomenally against the ground truth samples, with MSE, Bias and Variance values that are of the factor 10^{-6}.

One of the main issues that was found when using Stein Thinning for the β distributions is that the thinning procedure did not automatically remove the burn in as it was expected to do. We found that when the warm up was included in the small MCMC chain that was to be thinned, the Stein Thinning procedure left many points from the tail of the chain in the thinned sample, which hurt the resulting MSE, Bias and Variance. Graphs were provided that revealed that the differing tails between the Stein Thinned chain and the ground truth chain were the result of these issues.

We have concluded that this is most likely a result of the package that was used and the fact that different MCMC chains were thinned then were used in the ground truth sampling. In future work it is necessary that the burn in removal feature of Stein Thinning be more closely assessed. It is important to understand how to best use Stein Thinning and its associated packages in practice, as this method is gaining traction in various important fields of research.

The three main findings of this research are summarized here. (1) Stein Thinning revealed incredible variance reduction capabilities, even when compared to ZVCV, known for variance reduction in this setting. (2) Stein Thinning showed a remarkable ability to address the bias-variance trade off as it reduced both within a factor of 10 of the raw sample of 10,000 values. (3) When using the Stein Thinning package in R, the choice to not include the warm up seems to work significantly better when the ground truth values are *also* MCMC chains. Conclusion (3) needs to be explored in more depth in future research.

Bibliography

[1] R. Van de Schoot, D. Kaplan, J. Denissen, J. B. Asendorpf, F. J. Neyer, and M. A. Van Aken, "A gentle introduction to bayesian analysis: Applications to developmental research," *Child development*, vol. 85, no. 3, pp. 842–860, 2014.

[2] BAYES, "An essay towards solving a problem in the doctrine of chances," *Biometrika*, vol. 45, no. 3-4, pp. 296–315, 1958.

[3] C. P. Robert, G. Casella, and G. Casella, *Monte Carlo statistical methods*, vol. 2. Springer, 1999.

[4] C. P. Robert, G. Casella, C. P. Robert, and G. Casella, "The metropolis—hastings algorithm," *Monte Carlo statistical methods*, pp. 267–320, 2004.

[5] L. F. South, M. Riabiz, O. Teymur, and C. J. Oates, "Postprocessing of mcmc," *Annual Review of Statistics and Its Application*, vol. 9, pp. 529–555, 2022.

[6] L. Mackey, "Advances in distribution compression," 2023. https://stanford.edu/ lmackey/papers/thinning-slides.pdf.

[7] J. Gorham and L. Mackey, "Measuring sample quality with kernels," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1292–1301, PMLR, 06–11 Aug 2017.

[8] C. Stein, "A bound for the error in the normal approximation to the distribution of a sum of dependent random variables," in *Proceedings of the Sixth Berkeley Symposium on Mathemat-*

ical Statistics and Probability, Volume 2: Probability Theory, vol. 6, pp. 583–603, University of California Press, 1972.

[9] M. Riabiz, W. Y. Chen, J. Cockayne, P. Swietach, S. A. Niederer, L. Mackey, and C. J. Oates, "Optimal thinning of mcmc output," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 84, no. 4, pp. 1059–1081, 2022.

[10] L. K. Wenliang and H. Kanagawa, "Blindness of score-based methods to isolated components and mixing proportions," *arXiv preprint arXiv:2008.10087*, 2020.

[11] L. South, "Monte carlo variance reduction using stein operators," in *14th International Conference in Monte Carlo Quasi-Monte Carlo Methods in Scientific Computing*, Quennsland University of Technology, August 2020.

[12] L. F. South, C. J. Oates, A. Mira, and C. Drovandi, "Regularized zero-variance control variates," *Bayesian Analysis*, vol. 18, no. 3, pp. 865–888, 2023.

[13] A. D. Khare, "Data for admission in the university," 2023. https://www.kaggle.com/datasets/akshaydattatraykhare/data-for-admission-in-the-university.

[14] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.

[15] V. Fonti and E. Belitser, "Feature selection using lasso," *VU Amsterdam research paper in business analytics*, vol. 30, pp. 1–25, 2017.

[16] J. Ghosh, Y. Li, and R. Mitra, "On the use of cauchy prior distributions for bayesian logistic regression," 2018.

Appendix - R Code

```
library("stein.thinning")
library(mcmc)
library(rstan)
library(plotly)
library(ZVCV)

# Creating Linear Regression Data Sets scaled for the prior dist
adm_data <- read.csv("/Users/michaellicata/Documents/SeniorYear/Thesis/Final Documents/adm_data.csv")
mydata4 <- adm_data[c("Chance.of.Admit", "CGPA", "Research")]
continuous_cols <- c("Chance.of.Admit", "CGPA")
mydata4[continuous_cols] <- scale(mydata4[continuous_cols])

load("/Users/michaellicata/Documents/SeniorYear/Thesis/Final Documents/2var/lr2gt.RData")

# Function that computes zero variate control variate for MCMC output
perform_zvcv <- function (smp, scr) {
  integrand <- smp[, 2:5]
  # ZV-CV
  temp <- zvcv(
    integrand,
    smp,
    scr
```

```
  )

  # Return Expectation Values for Beta1 and Beta2
  return(temp$expectation)
}

# Function that stein thins a similar mcmc output as above
stein_thin <- function(df, iter, bool, samp) {
  # Linear Regression Model
  mc <- "
    data {
      int<lower=0> N;
      int<lower=0> K;
      matrix[N, K] X;
      vector[N] y;
    }

    parameters {
      real alpha;
      vector[K] beta;
      real<lower=0> sigma;
    }

    model {
      beta ~ student_t(3,0,0.5);
      y ~ normal(X * beta + alpha, sigma);
    }

    "
  data_list <- list(N = nrow(df),
                    K = ncol(df) - 1,
                    X = as.matrix(df[, 2:5]),
                    y = df[, 1])
```

```
56
57
58    fit <- stan(model_code=mc, data=data_list, iter=iter, chains=1)
59
60    # Extract sampled points
61    smp <- extract(fit, permuted=FALSE, inc_warmup=bool)
62    smp <- smp[,,1:6]
63
64    # get gradients
65    scr <- t(apply(smp, 1, function(x) rstan::grad_log_prob(fit, x)))
66
67    zvcv_output <- perform_zvcv(smp, scr)
68
69    # Obtain a subset of {samp} points
70    idx <- thin(smp, scr, samp)
71    samples <- smp[idx, , drop = FALSE]
72
73    return(list(stein_data = samples, raw_data = smp, zvcv_data = zvcv_output))
74  }
75
76  # extracting mu values from df and putting them into a list
77  get_mus <- function(data) {
78    betas <- as.data.frame(data[, 2:5])
79    mu_is = colMeans(betas)
80    means_list = list()
81    for (i in 1:4) {
82      means_list[i] = mu_is[i]
83    }
84    return(mu_is)
85  }
86
87  # Stein Thinned MCMC Data with n set to number of test cases
88  # samp is set to number of values in thinned sample
```

```
89  stein_process <- function(data, iter, bool, samp, n) {
90
91    # Create initial data frames
92    stein_mu_hats <- data.frame()
93    raw_mu_hats <- data.frame()
94    zvcv_mu_hats <- data.frame()
95
96    for (i in 1:n) {
97
98      # Get results
99      mcmc_results <- stein_thin(data, iter, bool, samp)
100
101     # Seperate results
102     stein_data <- mcmc_results$stein_data
103     raw_data <- mcmc_results$raw_data
104     zvcv_data <- mcmc_results$zvcv_data
105
106     zvcv_mu_is <- as.list(zvcv_data)
107     stein_mu_is <- get_mus(stein_data)
108     raw_mu_is <- get_mus(raw_data)
109
110     # Add results to respective dfs
111     stein_mu_hats <- rbind(stein_mu_hats, stein_mu_is)
112     raw_mu_hats <- rbind(raw_mu_hats, raw_mu_is)
113     zvcv_mu_hats <- rbind(zvcv_mu_hats, zvcv_data)
114   }
115   colnames(stein_mu_hats) <- c("mu_hat1", "mu_hat2", "mu_hat3", "mu_hat4")
116   colnames(raw_mu_hats) <- c("mu_hat1", "mu_hat2", "mu_hat3", "mu_hat4")
117   colnames(zvcv_mu_hats) <- c("mu_hat1", "mu_hat2", "mu_hat3", "mu_hat4")
118   return(list(stein_beta_mus = stein_mu_hats, raw_beta_mus = raw_mu_hats, zvcv
        _beta_mus = zvcv_mu_hats))
119 }
120
```

```
# Stein Sample Size - samp = {50, 100, 200}
# Test Size - n = 100
# Iterations of MCMC - iter = 10,000
# Include Warmup - bool = {True, False}

# Define Global Variables
n <- 100
iter <- 10000
bools <- list(TRUE, FALSE)
samps <- list(50, 100, 200)

# Create results dataframes
stein_results_df <- data.frame()
raw_results_df <- data.frame()
zvcv_results_df <- data.frame()

# Helper function to add results to the result dataframe based on beta values
    obtained
add_results_df <- function(beta_mus) {
  df <- data.frame()
  for (i in 1:4) {
    id <- paste("mu", i, sep="")

    # Bias, Variance and MSE calculations
    mu_bias <- 1/n * sum(beta_mus[,i] - ground_truth_beta_mus[,i])
    mu_var <- var(beta_mus[,i])
    mu_mse <- mu_bias^2 + mu_var

    mu_results <- list(id, mu_bias, mu_var, mu_mse, samp, bool)
    df <- rbind(df, mu_results)
  }
  colnames(df) <- c("ID", "Bias", "Variance", "MSE", "Stein Samples", "Warmup"
    )
```

```
  return(df)
}

# Result Collection
for (samp in samps) {
  for (bool in bools) {
    stein_thin_results <- stein_process(mydata4, iter, bool, samp, n)

    stein_beta_mus <- stein_thin_results$stein_beta_mus
    raw_beta_mus <- stein_thin_results$raw_beta_mus
    zvcv_beta_mus <- stein_thin_results$zvcv_beta_mus

    raw_results_df <- rbind(raw_results_df, add_results_df(raw_beta_mus))
    stein_results_df <- rbind(stein_results_df, add_results_df(stein_beta_mus)
    )
    zvcv_results_df <- rbind(zvcv_results_df, add_results_df(zvcv_beta_mus))
  }
}
```

Listing 5.1: 4 Variable Linear Regression Code

```
library("stein.thinning")
library(mcmc)
library(rstan)
library(plotly)
library(ZVCV)

# Creating Linear Regression Data Sets scaled for the prior dist
adm_data <- read.csv("/Users/michaellicata/Documents/SeniorYear/Thesis/Final
    Documents/adm_data.csv")
mydata4 <- adm_data[c("Chance.of.Admit", "CGPA", "Research")]
continuous_cols <- c("Chance.of.Admit", "CGPA")
mydata4[continuous_cols] <- scale(mydata4[continuous_cols])

```

```
load("/Users/michaellicata/Documents/SeniorYear/Thesis/Final Documents/2var/
    lr2gt.RData")

# Function that computes zero variate control variate for MCMC output
perform_zvcv <- function (smp, scr) {
  integrand <- smp[, 2:3]
  # ZV-CV
  temp <- zvcv(
    integrand,
    smp,
    scr
  )

  # Return Expectation Values for Beta1 and Beta2
  return(temp$expectation)
}

# Function that stein thins a similar mcmc output as above
stein_thin <- function(df, iter, bool, samp) {
  # Linear Regression Model
  mc <- "
    data {
      int<lower=0> N;
      int<lower=0> K;
      matrix[N, K] X;
      vector[N] y;
    }

    parameters {
      real alpha;
      vector[K] beta;
      real<lower=0> sigma;
    }
```

```
45
46      model {
47        beta ~ student_t(3,0,0.5);
48        y ~ normal(X * beta + alpha, sigma);
49      }
50
51      "
52    data_list <- list(N = nrow(df),
53                      K = ncol(df) - 1,
54                      X = as.matrix(df[, 2:3]),
55                      y = df[, 1])
56
57
58    fit <- stan(model_code=mc, data=data_list, iter=iter, chains=1)
59
60    # Extract sampled points
61    smp <- extract(fit, permuted=FALSE, inc_warmup=bool)
62    smp <- smp[,,1:4]
63
64    # get gradients
65    scr <- t(apply(smp, 1, function(x) rstan::grad_log_prob(fit, x)))
66
67    zvcv_output <- perform_zvcv(smp, scr)
68
69    # Obtain a subset of {samp} points
70    idx <- thin(smp, scr, samp)
71    samples <- smp[idx, , drop = FALSE]
72
73    return(list(stein_data = samples, raw_data = smp, zvcv_data = zvcv_output))
74  }
75
76  # extracting mu values from df and putting them into a list
77  get_mus <- function(data) {
```

```
  betas <- as.data.frame(data[, 2:3])
  mu_is = colMeans(betas)
  means_list = list()
  for (i in 1:2) {
    means_list[i] = mu_is[i]
  }
  return(mu_is)
}

# Stein Thinned MCMC Data with n set to number of test cases
# samp is set to number of values in thinned sample
stein_process <- function(data, iter, bool, samp, n) {

  # Create initial data frames
  stein_mu_hats <- data.frame()
  raw_mu_hats <- data.frame()
  zvcv_mu_hats <- data.frame()

  for (i in 1:n) {

    # Get results
    mcmc_results <- stein_thin(data, iter, bool, samp)

    # Seperate results
    stein_data <- mcmc_results$stein_data
    raw_data <- mcmc_results$raw_data
    zvcv_data <- mcmc_results$zvcv_data

    zvcv_mu_is <- as.list(zvcv_data)
    stein_mu_is <- get_mus(stein_data)
    raw_mu_is <- get_mus(raw_data)

    # Add results to respective dfs
```

```
    stein_mu_hats <- rbind(stein_mu_hats, stein_mu_is)
    raw_mu_hats <- rbind(raw_mu_hats, raw_mu_is)
    zvcv_mu_hats <- rbind(zvcv_mu_hats, zvcv_data)
  }
  colnames(stein_mu_hats) <- c("mu_hat1", "mu_hat2")
  colnames(raw_mu_hats) <- c("mu_hat1", "mu_hat2")
  colnames(zvcv_mu_hats) <- c("mu_hat1", "mu_hat2")
  return(list(stein_beta_mus = stein_mu_hats, raw_beta_mus = raw_mu_hats, zvcv
    _beta_mus = zvcv_mu_hats))
}

# Stein Sample Size - samp = {50, 100, 200}
# Test Size - n = 100
# Iterations of MCMC - iter = 10,000
# Include Warmup - bool = {True, False}

# Define Global Variables
n <- 100
iter <- 10000
bools <- list(TRUE, FALSE)
samps <- list(50, 100, 200)

# Create results dataframes
stein_results_df <- data.frame()
raw_results_df <- data.frame()
zvcv_results_df <- data.frame()

# Helper function to add results to the result dataframe based on beta values
    obtained
add_results_df <- function(beta_mus) {
  df <- data.frame()
  for (i in 1:2) {
    id <- paste("mu", i, sep="")
```

```
142
143     # Bias, Variance and MSE calculations
144     mu_bias <- 1/n * sum(beta_mus[,i] - ground_truth_beta_mus[,i])
145     mu_var <- var(beta_mus[,i])
146     mu_mse <- mu_bias^2 + mu_var
147
148     mu_results <- list(id, mu_bias, mu_var, mu_mse, samp, bool)
149     df <- rbind(df, mu_results)
150   }
151   colnames(df) <- c("ID", "Bias", "Variance", "MSE", "Stein Samples", "Warmup"
      )
152   return(df)
153 }
154
155 # Result Collection
156 for (samp in samps) {
157   for (bool in bools) {
158     stein_thin_results <- stein_process(mydata4, iter, bool, samp, n)
159
160     stein_beta_mus <- stein_thin_results$stein_beta_mus
161     raw_beta_mus <- stein_thin_results$raw_beta_mus
162     zvcv_beta_mus <- stein_thin_results$zvcv_beta_mus
163
164     raw_results_df <- rbind(raw_results_df, add_results_df(raw_beta_mus))
165     stein_results_df <- rbind(stein_results_df, add_results_df(stein_beta_mus)
      )
166     zvcv_results_df <- rbind(zvcv_results_df, add_results_df(zvcv_beta_mus))
167   }
168 }
```

Listing 5.2: 2 Variable Linear Regression Code

```
1 library(glmnet)
2
```

```
adm_data <- read.csv("adm_data.csv")

x <- model.matrix(Chance.of.Admit ~ . -1, data=adm_data) # -1 to exclude
    intercept
y <- adm_data$Chance.of.Admit

lassoModel <- glmnet(x, y, alpha=1)
plot(lassoModel)

cvModel <- cv.glmnet(x, y, alpha=1)
bestLambda <- cvModel$lambda.min

plot(cvModel$glmnet.fit, xvar = "lambda", label = TRUE)
abline(v=log(bestLambda), col="red", lwd=2, lty=2)

coef(cvModel, s=bestLambda)
```

Listing 5.3: LASSO Regression Code