UC Irvine
UC Irvine Electronic Theses and Dissertations

Title
Scalable Target Marketing: Distributed Markov Chain Monte Carlo for Bayesian Hierarchical Models

Permalink
https://escholarship.org/uc/item/6405n55p

Author
Bumbaca, Federico

Publication Date
2018

Copyright Information
This work is made available under the terms of a Creative Commons Attribution-NonCommercial-NoDerivatives License, available at https://creativecommons.org/licenses/by-nc-nd/4.0/

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

IRVINE

Scalable Target Marketing:
Distributed Markov Chain Monte Carlo for Bayesian Hierarchical Models

DISSERTATION

submitted in partial satisfaction of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in Management

by

Federico Bumbaca

Dissertation Committee:

Professor Imran Currim, Chair

Professor Sanjeev Dewan

Associate Professor John Turner

Professor Rajeev Tyagi

2018

© 2018 Federico Bumbaca

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Professor Imran Currim, for his academic and professional guidance throughout my doctoral studies. I would also like to thank my committee members, Professors Sanjeev Dewan, John Turner, and Rajeev Tyagi, for their enthusiasm and support.

In addition, I thank Professors Sanjog Misra at the University of Chicago and Peter E. Rossi at the University of California in Los Angeles, for introducing me to the big data opportunity in marketing research and their guidance. Financial support for my doctoral studies was provided by the University of California, Irvine.

CURRICULUM VITAE

Federico Bumbaca

2018 Ph.D. in Management

University of California, Irvine

ABSTRACT OF THE DISSERTATION

Scalable Target Marketing:

Distributed Markov Chain Monte Carlo for Bayesian Hierarchical Models

By

Federico Bumbaca

Doctor of Philosophy in Management

University of California, Irvine, 2018

Professor Imran Currim, Chair

I propose a distributed Markov chain Monte Carlo (MCMC) algorithm for estimating Bayesian hierarchical models when the number of cross-sectional units is very large and the objects of interest are the unit-level parameters. The two-stage algorithm is asymptotically exact, retains the flexibility of a standard MCMC algorithm, and is easy to implement. The algorithm constructs an estimator of the posterior predictive distribution of the unit-level parameters in the first stage, and uses the estimator as the prior distribution in the second stage for the unit-level draws. Both stages are embarrassingly parallel. I demonstrate the algorithm with simulated data from a hierarchical logit model and show that it is faster and more efficient than a single machine algorithm by at least an order of magnitude. For a relatively small number of observations per cross-sectional unit, the algorithm is both faster and has better mixing properties than the standard hybrid Gibbs sampler. I illustrate my approach with data on 1,100,000 donors to a charitable organization, and simulations with up to 100 million units.

1 Introduction

Many problems in marketing and economics, and in particular target marketing, require unit-specific decisions such as which online advertisement to show, what prices to charge, or which promotion to offer. These targeted strategies are becoming popular due to the availability of very large data sets. These panels allow for unit-level inferences which are the basis of optimal unit-level strategies. These panels are characterized by a very large number of cross-sectional units, N ($> 1,000,000$) and a small number of observations per unit, T (< 50). The small number of observations per unit require the sharing of information across units for inference procedures.

Bayesian hierarchical models are well suited for panel data problems in which unit-level parameters are desired (Rossi and Allenby 1993). The unit-level parameters are associated with each unit's data points, whereas the model's common parameters specify the population distribution of the unit-level parameters and characterize their uncertainty. Typically, a hybrid Markov chain Monte Carlo (MCMC) algorithm (Rossi, Allenby, and McCulloch 2005) running on a single processor is used to conduct inference in Bayesian hierarchical models. A common MCMC algorithm is the hybrid Gibbs sampler which alternates between draws of the unit-level parameters, given the previous iteration's common parameters draw, and a draw of the common parameters, given the previous iteration's unit-level parameter draws. The process continues until the combined draws are from their joint posterior distribution.

Estimating Bayesian hierarchical models with panel datasets that have a very large N are a challenge for existing algorithms. When the number of units is very large, simulation from unit-level posterior distributions is not feasible within a reasonable amount of time on a single processor, due to processor performance. Using only a sample of the cross-sectional units to reduce computational burden does not produce individual level parameter estimates for all units.

Distributing the data and simulations across multiple machines is another approach to addressing the resource limitation problem. In particular, when estimating a Bayesian hierarchical model with a hybrid Gibbs algorithm, an easy-to-implement distributed algorithm is to simulate the common parameter draws on the master machine and to distribute the simulation of the unit-level

parameter draws across multiple worker machines. The limitation of this approach is that each parallel process needs an updated common parameter draw for each iteration of the unit-level parameter draws (and conversely, each common parameter draw requires updated unit-level parameter draws). The problem is that communicating the common parameter draws to each parallel process (and communicating the unit-level parameter draws to the master process) on each iteration is prohibitively expensive across multiple computers (Scott et al. 2016). My own simulations find that although there is an improvement in effective sample size generated per unit of computing, it is marginal compared to the fully distributed approach I introduce in this article. Results are available from the authors.

Bardenet, Doucet and Holmes (2015) broadly classify applications of MCMC to Bayesian inference when N is big into two groups, distributed and subsampling algorithms. Embarrassingly parallel distributed approaches parallelize computation in a distributed computing environment while minimizing the communication of data across machines. Subsampling algorithms reduce the number of individual data point likelihood evaluations on a single computer. Although these big N algorithms were not conceived for hierarchical models, I may adapt them albeit at significant cost. The crux of the matter is that Bayesian hierarchical models do not easily lend themselves to embarrassingly parallel estimation when the objects of interest are both the unit-level and the common parameters, or to subsampling when the number of observations per unit is not very large. I elaborate in Section 2 why current methods don't work with hierarchical models.

I propose a distributed MCMC algorithm for hierarchical models that simulates draws from the unit-level posterior distributions. It consists of two stages: (i) an MCMC algorithm constructs an estimator for the posterior predictive distribution of the unit-level parameters, and (ii) an independence Metropolis-Hastings or direct sampling algorithm simulates unit-specific parameter draws, using the first stage estimator as the prior. The method is asymptotically exact. The algorithm's estimator of the unit-level posterior distribution is an asymptotically unbiased estimator of the single machine unit-level posterior, as the number of units per shard approaches infinity. Further, the algorithm retains the central ideas and flexibility of any standard MCMC algorithm (for example,

the hybrid Gibbs sampler) for which any prior may be stipulated, such as a mixture of distributions. While my focus is not on recovering draws of the common parameters, I may easily simulate them by conditioning on the available unit-level draws.

The algorithm performs well: for non-conjugate posterior distributions, unit-level posterior densities converge to those of the "gold standard" single machine hybrid Gibbs algorithm much more efficiently (as measured by effective sample size per unit of computing). For small T the proposed algorithm estimates a Bayesian hierarchical model more than an order of magnitude faster (in execution time for a given number of MCMC iterations) and more efficiently than the single machine hybrid Gibbs algorithm. For large T the algorithm is still faster by an order of magnitude but efficiency decreases to about an order of magnitude greater. For conjugate posterior distributions the performance of the proposed algorithm is independent of the number of observations per unit. I test the algorithm on a large cluster of computers for estimating models with up to 100 million units. This is particularly important when the high precision that a big N method enables is critical to the application of interest. To further boost speed and efficiency I propose a subsampling algorithm which samples the data in the first stage.

2 Related Literature

The literature emphasizes Bayesian inference for large data sets that do not have a panel structure. In this section, I review this literature and explain why these methods are difficult to extend to the hierarchical setting without compromising the accuracy or the computational gains that can be realized from a distributed processing environment.

The general hierarchical model can be written:

$$y_{it} \sim p(y_{it}|\beta_i) \quad for\ i = 1,...,N\ and\ t = 1,...,T_i \tag{2.1}$$

$$\beta_i \sim p(\beta_i|\theta) \quad for\ i = 1,...,N \tag{2.2}$$

$$\theta \sim p(\theta|\tau) \tag{2.3}$$

$p(y_{it}|\beta_i)$ is the probability of observing y_{it} at time t for unit i, N is the number of cross-sectional units, T_i is the number of observations for unit i, and $\{\beta_i\}$ are the unit-level parameters. There is a standard two-stage prior with $p(\beta_i|\theta)$ as the first-stage. I refer to θ as common parameters with prior density $p(\theta|\tau)$ and hyper-parameters τ. Although T_i is unit-dependent, I let $T_i = T$ for notational simplicity.

Bardenet, Doucet and Holmes' (2015) discussion of distributed algorithms focuses on those of Scott et al. (2016) and Neiswanger, Wang, and Xing (2014). These algorithms estimate non-hierarchical models in an embarrassingly parallel manner. Scott et al. (2016) suggests a two-stage extension to estimate hierarchical models. For each of the two stages, Scott et al. (2016) partitions the full data $Y = \{y_{it}\}$ into S shards such that all of the observations for a unit are in the same shard $Y_s = \{y_{it}\}_{i \in I_s}$, where I_s is a vector indicating the randomly allocated units to shard s. The first stage simulates both the common θ and unit-level $\{\beta_i\}$ parameter draws with S parallel MCMC simulations (each shard on a separate worker machine), discards the unit-level parameter draws, and algorithmically combines the S collections of common parameter draws for each iteration of the MCMC algorithm. The second stage draws the unit-specific parameters in an embarrassingly parallel manner (each shard of units on a separate worker machine), given the synthesized common

parameter draws from the first stage.

They express the posterior for the common parameters in the first stage as the product of S subposteriors.

$$p(\theta|Y,\tau) \propto \prod_s p(\theta|Y_s,\tau) \tag{2.4}$$

$$\propto \prod_s \left\{ p(Y_s|\theta)\, p(\theta|\tau)^{1/S} \right\} \tag{2.5}$$

$$\propto \prod_s \left\{ \left[\prod_{i\in I_s} p(\beta_i|\theta) \prod_t p(y_{it}|\beta_i) \right] p(\theta|\tau)^{1/S} \right\} \tag{2.6}$$

$$\propto \prod_s \left\{ \prod_{i\in I_s} p(\beta_i|\theta)\, p(\theta|\tau)^{1/S} \right\} \tag{2.7}$$

where the full data prior for θ is $p(\theta|\tau) = \prod_{s=1}^{S} p(\theta|\tau)^{1/S}$. For each of the S parallel processes, they simulate the common parameter draws from $p(\theta|Y_s,\tau) \propto \prod_{i\in I_s} p(\beta_i|\theta)\, p(\theta|\tau)^{1/S}$, given the $\{\beta_i\}_{i\in I_s}$ draws.

Given the S collections of common parameter draws, Scott et al. (2016) and Neiswanger, Wang, and Xing (2014) algorithmically synthesize a single collection of draws from the full data posterior distribution $p(\theta|Y,\tau)$ in different ways. On the r^{th} iteration of an MCMC simulation, Scott et al. (2016) synthesize the combined draw θ^r with a weighted average of the r^{th} draws of θ from the S processes, θ_s^r.

$$\theta^r = \left(\sum_s W_s \right)^{-1} \left(\sum_s W_s \theta_s^r \right) \tag{2.8}$$

where the weight $W_s = \Sigma_s^{-1}$ and $\Sigma_s = Var(\theta_s|Y_s,\tau)$.

I emphasize the primary limitations of the algorithm over secondary considerations[1,2]. Although the algorithm is exact if $p(\theta|Y_s,\tau)$ is normal, since $p(\theta|Y,\tau)$ must also be normal, it

[1] If some of the subposteriors are centered very far from other subposteriors their product may be poorly approximated (Bardenet, Doucet and Holmes 2015).

[2] If $p(\theta|\tau)$ is a conjugate distribution, $p(\theta|\tau)^{1/S}$ is not necessarily also a conjugate distribution. The modeler must either approximate $p(\theta,\tau)^{1/S}$ with a suitable conjugate distribution, or revert to a possibly less efficient Metropolis-Hastings algorithm for the simulation of θ draws.

is approximate otherwise. For precise target marketing I require exact estimates. Additionally, the population distribution of the unit-level parameters $p(\beta_i|\theta)$ must be a single component or unimodal distribution, due to the problem of combining draws from different multimodal distributions. For example, θ draws from mixture component k in shard 1 may lie in the region of a major mode whereas as θ draws from mixture component j in shard 2 may lie in the non-overlapping region of a minor mode. It would be inappropriate to algorithmically combine their draws since they are in different and unrelated regions. This factor is especially salient in the marketing context. A multimodal prior as the population distribution $p(\beta_i|\theta)$ allows a great deal of flexibility and the opportunity to discover new structure (Rossi, Allenby, and McCulloch 2005).

Neiswanger, Wang, and Xing's (2014) distributed (nonparametric density product estimation) MCMC algorithm has an advantage over that of Scott et al.'s (2016) in that it does not require a normal posterior or normal subposteriors for exactness. It is asymptotically exact for any posterior. To synthesize the combined draws, they first use kernel density estimation with a normal kernel to construct S shard-specific density estimators $\hat{p}(\theta|Y_s,\tau)$ for the densities of each shard's parameters.

$$\hat{p}(\theta|Y_s,\tau) = \frac{1}{R}\sum_{r_s}\phi\left(\theta\left|\theta_s^{r_s}, h^2 I_d\right.\right) \tag{2.9}$$

where h is a smoothing parameter, and $\theta_s^{r_s}$ is the r_s^{th} draw of θ for shard s.

The full data posterior density estimator is the product of these S shard-level estimators.

$$\hat{p}(\theta|Y,\tau) \propto \prod_s \hat{p}(\theta|Y_s,\tau) \tag{2.10}$$

$$= \frac{1}{R^S}\prod_s\sum_{r_s}\phi\left(\theta\left|\theta_s^{r_s}, h^2 I_d\right.\right) \tag{2.11}$$

$$\propto \sum_{r_1}\cdots\sum_{r_S} w_{\{r_1,\ldots,r_S\}}\phi\left(\theta\left|\bar{\theta}_{\{r_1,\ldots,r_S\}}, \frac{h^2}{S}I_d\right.\right) \tag{2.12}$$

where $\bar{\theta}_{\{r_1,\ldots,r_S\}} = \frac{1}{S}\sum_s \theta_s^{r_s}$ and $w_{\{r_1,\ldots,r_S\}} = \prod_s \phi\left(\theta_s^{r_s}\left|\bar{\theta}_{\{r_1,\ldots,r_S\}}, h^2 I_d\right.\right)$. $p(\theta|Y,\tau)$ is a mixture of R^S normal densities with unnormalized mixture weights $w_{\{r_1,\ldots,r_S\}}$. The synthesized draw for iteration

r, θ^r, is drawn from $p(\theta|Y,\tau)$ in two steps: (i) draw a mixture component $\{r_1,\ldots,r_S\}$ with an independence Metropolis with Gibbs sampler (Neiswanger, Wang, and Xing 2014), and (ii) draw θ^r from this mixture component.

The algorithm has several fundamental limitations as well as secondary concerns[1,2]. The computational complexity of the algorithm is quadratic with S (the authors also suggest an alternative algorithm whose complexity is linear with S), suggesting that its complexity may materially impact the computational advantages of a distributed approach for large values of S. Since the method is based on kernel density estimation I do not expect it to scale well as the dimension of the parameter space grows, due to the curse of dimensionality. Finally, the bound on the mean-squared-error of the approximated posterior increases exponentially with the number of shards S (Bardenet, Doucet and Holmes 2015).

Bardenet, Doucet and Holmes (2015) make brief mention of distributed algorithms that avoid multiplying the S subposteriors. They combine subposteriors by their barycenter (Srivastava et al. 2015) or median (Minsker et al. 2014). The challenge with these combinations is that their statistical meaning is unclear. Although Minsker et al.'s (2014) median estimate is robust to outliers it has the potential of losing valuable information (Bardenet, Doucet and Holmes 2015).

Bardenet, Doucet and Holmes (2015) also present an overview of subsampling-based algorithms for non-hierarchical models. Subsampling algorithms run on a single computer with all of the data in memory. They reduce the number of individual data point likelihood evaluations that are necessary at each MCMC iteration. I may estimate Bayesian hierarchical models with subsampling methods in primarily two ways: (i) subsampling the unit-level draws $\{\beta_i\}$ to draw the common parameters θ, and (ii) subsampling a unit's T observations to draw β_i.

In a typical MCMC algorithm that alternates between draws of $\{\beta_i\}$ and θ, and when N is huge, it may be advantageous to subsample the $\{\beta_i\}$ draws to compute the log-likelihood and ratio for making an acceptance decision for the proposal draw for θ. Although there are computational gains to be made at each iteration of the θ draws if N is extremely large, say millions, I argue that these savings are minuscule compared to the amount of computation required to draw the N

unit-level draws at each iteration. This is especially true if $p(\beta_i|\theta)$ and $p(\theta|\tau)$ are conjugate, in which case the θ draws are already extremely fast.

The opportunity for substantial computational gains seems most appropriate for subsampling the unit-level data for the unit-level draws for non-conjugate posteriors. However, when applied in my context of drawing unit-level parameters in a hierarchical model when T is not large, it is doubtful whether subsampling algorithms that are designed for very large T may be of value[3] .

[3]Bardenet, Doucet and Holmes' (2015) improved confidence sampler is designed for subsampling in cases of very large sample size (they consider subsampling samples of size 1,000 from datasets with up to 10,000,000 observations).

3 The Proposed Algorithm

I guide the reader through the development of the proposed algorithm by presenting a sequence of three algorithms.

3.1 The Model

The standard Bayesian hierarchical model (2.1 - 2.3) induces a joint posterior distribution (3.1) of the model parameters $\{\beta_i\}$ and θ

$$p(\{\beta_i\}, \theta | Y, \tau) \propto p(\theta|\tau) \prod_i \left[p(\beta_i|\theta) \prod_t p(y_{it}|\beta_i) \right] \tag{3.1}$$

where $Y = \{y_{it}\}$ is the full data of observed outcomes and covariates (I omit the covariates for notational convenience). The posterior full conditional densities are

$$\beta_i | \theta, y_i \propto p(\beta_i|\theta) \prod_t p(y_{it}|\beta_i) \text{ for } i = 1, ..., N \tag{3.2}$$

$$\theta | \{\beta_i\}, \tau \propto p(\theta|\tau) \prod_i p(\beta_i|\theta) \tag{3.3}$$

I assume that $p(\beta_i|\theta)$ and $p(y_{it}|\beta_i)$ are not conjugate: $p(\beta_i|\theta, y_i)$ is a non-conjugate distribution. I discuss the simpler conjugate case in Section 3.5.4.

3.2 Benchmark Algorithm $\mathscr{A}_1$

Algorithm $\mathscr{A}_1$ is a single machine benchmark algorithm that draws $\{\beta_i\}$ and θ from the joint posterior (3.1). $\mathscr{A}_1$ may be a MCMC algorithm, a direct sampling algorithm, or any other algorithm that draws from the model's posterior distribution. The β_i and θ draws are from their respective posterior marginal densities, $p(\beta_i|Y, \tau)$ and $p(\theta|Y, \tau)$. The densities $p(\beta_i|Y, \tau)$ for $i = 1, ..., N$ represent "truth" against which subsequent algorithms are compared. A typical implementation of $\mathscr{A}_1$ is a hybrid Gibbs algorithm that alternates between draws from (3.2) and (3.3). Since $p(\beta_i|\theta, y_i)$

is non-conjugate, I need a Metropolis-Hastings algorithm to draw β_i. If $p(\theta\,|\tau)$ and $p\,(\beta_i\,|\theta\,)$ are conjugate, as is commonly the case, $p\,(\theta\,|\{\beta_i\}\,,\tau)$ is a conjugate distribution from which I may sample θ directly.

3.3 Equivalent Algorithm $\mathscr{A}_2$

To motivate the single machine algorithm $\mathscr{A}_2$ I integrate out θ from (3.2) and derive an explicit expression for the posterior marginal density of β_i, $p\,(\beta_i\,|Y,\tau)$. For the moment I assume that the posterior marginal density of θ, $p\,(\theta\,|Y,\tau)$, is known.

$$p\,(\beta_i\,|Y,\tau) = \int p\,(\beta_i\,|\theta,y_i)\,p\,(\theta\,|Y,\tau)\,d\theta \tag{3.4}$$

$$\propto \int p\,(\beta_i\,|\theta\,)\prod_t p\,(y_{it}\,|\beta_i\,)\,p\,(\theta\,|Y,\tau)\,d\theta \tag{3.5}$$

$$= \int p\,(\beta_i\,|\theta\,)\,p\,(\theta\,|Y,\tau)\,d\theta\prod_t p\,(y_{it}\,|\beta_i\,) \tag{3.6}$$

$$= \mathbb{E}_{\theta|Y,\tau}\left[p\,(\beta_i\,|\theta\,)\right]\prod_t p\,(y_{it}\,|\beta_i\,) \tag{3.7}$$

where $\mathbb{E}_{\theta|Y,\tau}\left[p\,(\beta_i\,|\theta\,)\right] = \int p\,(\beta_i\,|\theta\,)\,p\,(\theta\,|Y,\tau)\,d\theta$.

To get an intuitive sense for (3.7) I may interpret $\mathbb{E}_{\theta|Y,\tau}\left[p\,(\beta_i\,|\theta\,)\right]$ as the posterior predictive density[4] of β_i - the density of β_i before observing y_i, given $p\,(\theta\,|Y,\tau)$. "posterior" refers to the dependence on the posterior $p\,(\theta\,|Y,\tau)$, and "predictive" refers to the prediction of β_i before observing y_i. Although y_i is included in Y, it influences β_i only indirectly through $p\,(\theta\,|Y,\tau)$. To emphasize that $\mathbb{E}_{\theta|Y,\tau}\left[p\,(\beta_i\,|\theta\,)\right]$ is a density function of β_i, I denote it as $p_{\theta|Y,\tau}\,(\beta_i)$. Rewriting (3.7) by replacing $\mathbb{E}_{\theta|Y,\tau}\left[p\,(\beta_i\,|\theta\,)\right]$ with $p_{\theta|Y,\tau}\,(\beta_i)$, simplifying $\prod_t p\,(y_{it}\,|\beta_i\,)$ to $p\,(y_i\,|\beta_i\,)$, and nor-

[4]My definition differs from but is closely related to that of Gelman et al. (2014) (Appendix: Theorem 1).

malizing

$$p(\beta_i|Y,\tau) \propto \mathbb{E}_{\theta|Y,\tau}[p(\beta_i|\theta)]\prod_t p(y_{it}|\beta_i) \tag{3.8}$$

$$= p_{\theta|Y,\tau}(\beta_i)\, p(y_i|\beta_i) \tag{3.9}$$

$$p(\beta_i|Y,\tau) = \frac{p_{\theta|Y,\tau}(\beta_i)\, p(y_i|\beta_i)}{p(y_i)} \tag{3.10}$$

$p(\beta_i|Y,\tau)$ may be interpreted as a consequence of Bayes' theorem (3.10) in which $p_{\theta|Y,\tau}(\beta_i)$ is a highly informative prior distribution for β_i before observing y_i, $p(y_i|\beta_i)$ is the likelihood of y_i occurring given β_i, $p(y_i)$ is a normalization constant, and $p(\beta_i|Y,\tau)$ is the posterior distribution of β_i after observing y_i.

Ideally, I would like to draw β_i from (3.9 - 3.10) but I need $p_{\theta|Y,\tau}(\beta_i)$, which is a mathematical object. I replace the 'expectation' operator in $p_{\theta|Y,\tau} = \mathbb{E}_{\theta|Y,\tau}[p(\beta_i|\theta)]$ with the 'average' operator to construct the estimator $\dot{p}_{\theta|Y,\tau}(\beta_i)$

$$\dot{p}_{\theta|Y,\tau}(\beta_i) = \frac{1}{R}\sum_r p(\beta_i|\theta^r) \tag{3.11}$$

where θ^r is the r^{th} draw of θ from $p(\theta|Y,\tau)$, and R is the total number of draws[5]. I replace $p_{\theta|Y,\tau}(\beta_i)$ in (3.9) with $\dot{p}_{\theta|Y,\tau}(\beta_i)$ to draw β_i from $\dot{p}(\beta_i|\{\theta^r\},Y,\tau)$

$$\dot{p}(\beta_i|\{\theta^r\},Y,\tau) \propto \dot{p}_{\theta|Y,\tau}(\beta_i)\prod_t p(y_{it}|\beta_i) \tag{3.12}$$

$$= \frac{1}{R}\sum_r p(\beta_i|\theta^r)\prod_t p(y_{it}|\beta_i) \tag{3.13}$$

I denote the posterior marginal density induced by $\dot{p}(\beta_i|\{\theta^r\},Y,\tau)$ as $\dot{p}(\beta_i|Y,\tau)$. Algorithms $\mathscr{A}_1$ and $\mathscr{A}_2$ are equivalent: $\dot{p}(\beta_i|Y,\tau) = p(\beta_i|Y,\tau)$ (Appendix: Theorems 2 and 3).

Finally, to derive (3.12 - 3.13) I assumed that draws from $p(\theta|Y,\tau)$ are available to construct $\dot{p}_{\theta|Y,\tau}(\beta_i)$. This suggests a two-stage algorithm for $\mathscr{A}_2$. Stage one constructs $\dot{p}_{\theta|Y,\tau}(\beta_i)$ and stage

[5] I assume that R is sufficiently large that the Strong Law of Large Numbers guarantees convergence of $\dot{p}_{\theta|Y,\tau}(\beta_i)$ to $p_{\theta|Y,\tau}(\beta_i)$ (Robert and Casella 2010).

two draws $\{\beta_i\}$. More specifically, stage one implements algorithm $\mathscr{A}_1$ to draw $\{\beta_i\}$ and θ from (3.1), discards the $\{\beta_i\}$ draws, and keeps the θ draws to construct $\dot{p}_{\theta|Y,\tau}(\beta_i)$ using (3.11). Stage two draws β_i from $\dot{p}(\beta_i|\{\theta^r\},Y,\tau)$ (3.12) for $i = 1,...,N$.

Although it is inefficient to discard the $\{\beta_i\}$ draws in stage one only to redraw them in stage two with no apparent benefit, both sets of draws are from $p(\beta_i|Y,\tau)$, the purpose of $\mathscr{A}_2$ is to motivate $\mathscr{A}_3$ through the introduction of $\dot{p}_{\theta|Y,\tau}(\beta_i)$. This peculiarity of $\mathscr{A}_2$ goes away in $\mathscr{A}_3$.

3.4 Distributed Algorithm $\mathscr{A}_3$

Algorithm $\mathscr{A}_3$ is my proposed algorithm (Figures 3.1 and 3.2), an embarrassingly parallel implementation of $\mathscr{A}_2$.

3.4.1 Stage One

The first stage constructs an estimator of $\dot{p}_{\theta|Y,\tau}(\beta_i)$ in an embarrassingly parallel manner by first implementing algorithm $\mathscr{A}_1$ on mutually exclusive subsets of the data, each on a separate machine. I partition the full data Y into S shards such that all of the data for unit i are in the same shard: $Y_s = \{y_{it}\}_{i\in I_s}$, $s = 1,...,S$, and I_s is a vector indicating the randomly allocated units to shard s. The joint subposterior distribution of the model parameters for shard s follows (3.1)

$$p\left(\{\beta_i\}_{i\in I_s},\theta_s|\tau,Y_s\right) \propto p(\theta|\tau)\prod_{i\in I_s}\left[p(\beta_i|\theta)\prod_t p(y_{it}|\beta_i)\right] \tag{3.14}$$

For each parallel process, following (3.11), the $\{\beta_i\}_{i\in I_s}$ draws are discarded, and the θ_s draws are used to construct the subposterior predictive distribution of β_i for shard s, which I denote as $\dot{p}_{\theta|Y_s,\tau}(\beta_i)$

$$\dot{p}_{\theta|Y_s,\tau}(\beta_i) = \frac{1}{R}\sum_r p(\beta_i|\theta_s^r) \tag{3.15}$$

where θ_s^r is the r^{th} draw of θ for shard s, and R is the total number of draws[5]. To construct an

Figure 3.1: Algorithm $\mathscr{A}_3$: The proposed algorithm (for non-conjugate posterior)

Stage One: Draw $\beta \sim \ddot{p}_{\theta|Y,\tau}(\beta)$
Input: Y, S
Output: $\{\beta^r\}$

1. Divide Y into S shards
   (a) $I_s = \{i : z_i = s\}$, $s = 1,...,S$, where $p(z_i = s) = \frac{1}{S}$, and $|I_s| = \frac{N}{S}$
   (b) $Y_s = \{y_{it}\}_{i \in I_s}$, $s = 1,...,S$
2. Run S parallel simulations ($s = 1,...,S$)
   (a) for $r = 1$ to R
      i. Draw $\{\beta_i^r\}_{i \in I_s}, \theta_s^r \,|\tau, Y_s \propto p(\theta\,|\tau) \prod_{i \in I_s} [p(\beta_i\,|\theta) \prod_t p(y_{it}\,|\beta_i)]$ (Algorithm $\mathscr{A}_1$)
   (b) keep $\{\theta_s^r\}$ and discard $\{\beta_i^r\}_{i \in I_s}$
   (c) for $r = 1$ to R/S
      i. Draw $z^r \sim Multinomial\left(n = 1, p = \left\{\frac{1}{R},...,\frac{1}{R}\right\}\right)$, $z^r \in \{1,...,R\}$
      ii. Draw $\beta_s^r \sim p(\beta_s^r \,|\theta_s^{z^r})$
3. Collect the β_s^r draws and shuffle
   (a) $\{\beta^r\} \leftarrow shuffle(\{\beta_s^r\})$

Stage Two: Draw $\{\beta_i\}$
Input: Y, S, $\{\beta^r\}$
Output: $\{\beta_i^r\}$

1. Divide Y into S shards
   (a) $I_s = \{i : z_i = s\}$, $s = 1,...,S$, where $p(z_i = s) = \frac{1}{S}$, and $|I_s| = \frac{N}{S}$
   (b) $Y_s = \{y_{it}\}_{i \in I_s}$, $s = 1,...,S$
2. Run S parallel independence Metropolis-Hastings simulations ($s = 1,...,S$)
   (a) $\beta_i^1 = \beta^1$, for $i \in I_s$
   (b) for $r = 2$ to R
      i. $\alpha_i^r = min\left\{\frac{\prod_t p(y_{it}|\beta^r)}{\prod_t p(y_{it}|\beta_i^{r-1})}, 1\right\}$, for $i \in I_s$
      ii. $\beta_i^r = \begin{cases} \beta^r & wp\ \alpha_i^r \\ \beta_i^{r-1} & wp\ 1-\alpha_i^r \end{cases}$, for $i \in I_s$

Figure 3.2: A Practitioner's Guide

1. Estimate C_0
   (a) Choose a sufficiently large $N' \ll N$, but not too large, say $N' = 10,000$
   (b) Choose a sufficiently small $S' > 1$, say $S' = 3$
   (c) Choose a sufficiently large R for MCMC convergence and mixing
   (d) Run Algorithm $\mathscr{A}_1$ to construct $\dot{p}_{\theta|Y,\tau}(\beta_i) = \frac{1}{R}\sum_r p(\beta_i|\theta^r)$
   (e) Run Algorithm $\mathscr{A}_3$ Stage One to construct $\ddot{p}_{\theta|Y,\tau}(\beta_i) = \frac{1}{SR}\sum_s\sum_r p(\beta_i|\theta_s^r)$
   (f) Estimate $\varepsilon^2 \approx \sup_{\beta_{ik}, k\in\{1,\ldots,d\}}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_{ik}) - \dot{p}_{\theta|Y,\tau}(\beta_{ik})\right|^2\right]$
   (g) Approximate $C_0 \approx \left(\frac{S'^2+1}{S'N'R\varepsilon^2}\right)$
2. Choose $\varepsilon^2_{max} = \sup_{\beta_i}\mathbb{E}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right]$
3. Estimate $S_{max} \approx \begin{cases} \lfloor C_0 NR\varepsilon^2_{max}\rfloor & \text{if } S^2_{max} \gg 1 \\ \left\lfloor \frac{C_0}{2}\left(NR\varepsilon^2_{max} + \sqrt{(NR\varepsilon^2_{max})^2 - 4C_0^{-2}}\right)\right\rfloor & \text{otherwise} \end{cases}$
4. Choose $S \leq S_{max}$
5. If $S = S_{max}$ run Algorithm $\mathscr{A}_3$
6. If $S < S_{max}$
   (a) Estimate $p \approx \begin{cases} \sqrt{\frac{S}{S_{max}}} & \text{if } S^2 \gg p^2 \text{ and } S^2_{max} \gg 1 \\ \sqrt{\frac{S^2}{C_0 SNR\varepsilon^2_{max} - 1}} & \text{otherwise} \end{cases}$
   (b) Run Algorithm $\mathscr{A}'_3$

estimator of $\dot{p}_{\theta|Y,\tau}(\beta_i)$ I average the S shard-level estimators $\dot{p}_{\theta|Y_s,\tau}(\beta_i)$, $s = 1,...,S$.

$$\ddot{p}_{\theta|Y,\tau}(\beta_i) = \frac{1}{S}\sum_s \dot{p}_{\theta|Y_s,\tau}(\beta_i) \tag{3.16}$$

$$= \frac{1}{SR}\sum_s\sum_r p(\beta_i|\theta_s^r) \tag{3.17}$$

$\ddot{p}_{\theta|Y,\tau}(\beta_i)$ is an asymptotically unbiased estimator of $\dot{p}_{\theta|Y,\tau}(\beta_i)$ as the number of units per shard approaches infinity: $\lim_{N/S\to\infty}\mathbb{E}\left[\ddot{p}_{\theta|Y,\tau}(\beta_i)\right] = \dot{p}_{\theta|Y,\tau}(\beta_i)$ (Appendix: Theorems 4 and 5).
$\ddot{p}_{\theta|Y,\tau}(\beta_i)$ has better variance properties than $\dot{p}_{\theta|Y_s,\tau}(\beta_i)$: its limit distribution has a smaller variance by a factor of S (Appendix: Theorem 4). For this reason I discard these first stage $\{\beta_i\}_{i\in I_s}$ draws in favor of draws derived from $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ in the second stage.

A distinctive feature of the proposed algorithm is that it does not algorithmically combine the $\{\theta_s^r\}$ draws to synthesize single machine $\{\theta^r\}$ draws, as required by Scott et al. (2016) and Neiswanger, Wang, and Xing (2014). I use the $\{\theta_s^r\}$ draws to construct $\ddot{p}_{\theta|Y,\tau}(\beta_i)$, a mathematical procedure that does not require computation. Since the computational complexity of constructing and drawing from $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ is independent of the number of shards, the computational complexity of the algorithm is as well.

3.4.2 Stage Two

The second stage of the proposed algorithm draws $\{\beta_i\}$ in an embarrassingly parallel manner. As in the first stage I partition the full data Y into S shards, one shard per machine. To draw β_i using $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ from the first stage, I replace $\dot{p}_{\theta|Y,\tau}(\beta_i)$ in (3.12) with $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ to draw from

$$\ddot{p}(\beta_i|\{\theta^r\},Y,\tau) \propto \ddot{p}_{\theta|Y,\tau}(\beta_i)\prod_t p(y_{it}|\beta_i) \text{ for } i\in I_s \tag{3.18}$$

$$= \frac{1}{SR}\sum_s\sum_r p(\beta_i|\theta_s^r)\prod_t p(y_{it}|\beta_i) \text{ for } i\in I_s \tag{3.19}$$

where $\{\theta_s^r\}$ denotes the collection of θ draws from the first stage. I denote the posterior marginal distribution induced by $\ddot{p}(\beta_i|\{\theta^r\},Y,\tau)$ as $\ddot{p}(\beta_i|Y,\tau)$. Algorithm $\mathscr{A}_3$ is an asymptotically unbi-

ased estimator of $\mathscr{A}_2$ as the number of units per shard approaches infinity:
$\lim_{N/S\to\infty}\mathbb{E}\left[\ddot{p}\left(\beta_i\left|Y,\tau\right.\right)\right]=p\left(\beta_i\left|Y,\tau\right.\right)$ (Appendix: Theorem 6).

I propose an independence Metropolis-Hastings algorithm to draw from (3.18-3.19). More precisely, the proposal and target densities are $\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)$ and $\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)\prod_t p\left(y_{it}\left|\beta_i\right.\right)$, respectively. I simulate R proposal draws from $\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)$. Given the previous iteration's draw β_i^{r-1}, I accept proposal draw β_i^r with probability

$$\alpha = \min\left\{1,\frac{\ddot{p}_{\theta|Y,\tau}\left(\beta_i^r\right)\prod_t p\left(y_{it}\left|\beta_i^r\right.\right)}{\ddot{p}_{\theta|Y,\tau}\left(\beta_i^{r-1}\right)\prod_t p\left(y_{it}\left|\beta_i^{r-1}\right.\right)}\frac{\ddot{p}_{\theta|Y,\tau}\left(\beta_i^{r-1}\right)}{\ddot{p}_{\theta|Y,\tau}\left(\beta_i^r\right)}\right\} \tag{3.20}$$

$$= \min\left\{1,\frac{\prod_t p\left(y_{it}\left|\beta_i^r\right.\right)}{\prod_t p\left(y_{it}\left|\beta_i^{r-1}\right.\right)}\right\} \tag{3.21}$$

a very fast computation.

When T is not large an independence Metropolis-Hastings algorithm is especially attractive: $\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)$, a highly informative prior, has a large influence on the unit-specific posterior density since the unit likelihood $\prod_t p\left(y_{it}\left|\beta_i\right.\right)$ is relatively flat, and therefore draws from $\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)$ are expected to have a high acceptance rate. I do not need a more computationally intensive random walk algorithm. Furthermore, under mild conditions, an independence Metropolis-Hastings algorithm converges uniformly rather than geometrically or worse for a random walk algorithm (Robert and Casella 2010). However, for large T each unit's likelihood may be more sharply peaked and influenced less by the prior. Proposal draws from $\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)$ are therefore expected to have lower acceptance rates when T is large.

3.5 Algorithm $\mathscr{A}_3$ Additional Details

3.5.1 Communication Overhead

Algorithm $\mathscr{A}_3$, as described, implies that I communicate R draws of θ_s^r from each parallel process in stage one to the master machine to construct $\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)$, from which I simulate R draws of β_i in stage two. To decrease communication overhead between parallel processes and the master

machine, I simulate the stage two draws from $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ in parallel in stage one, rather than in stage two. To simulate R draws from $\ddot{p}_{\theta|Y,\tau}(\beta_i) = \frac{1}{S}\sum_s \dot{p}_{\theta|Y_s,\tau}(\beta_i)$ in stage one: (i) simulate $\frac{R}{S}$ draws from $\dot{p}_{\theta|Y_s,\tau}(\beta_i)$ in parallel on each of the S machines, (ii) communicate the draws to the master machine, and (iii) combine and shuffle. To simulate a draw from $\dot{p}_{\theta|Y_s,\tau}(\beta_i) = \frac{1}{R}\sum_r p(\beta_i|\theta_s^r)$: draw a multinomial distributed indicator vector with parameter vector $\left(\frac{1}{R},...,\frac{1}{R}\right)$ to determine the active component r, and draw from $p(\beta_i|\theta_s^r)$.

I reduce first stage communication overhead from each parallel process to the master machine from R draws of θ_s^r to $\frac{R}{S}$ draws of β_i. For example, if β_i is a d-dimensional vector and $p(\beta_i|\theta)$ is the normal distribution, θ represents the mean and covariance parameters for a normal distribution which requires $d + \frac{d(d+1)}{2}$ elements. I reduce communication overhead from $R\left(d + \frac{d(d+1)}{2}\right)$ to $\frac{R}{S}d$, a reduction by a factor of dS.

3.5.2 Maximum Number of Shards

Although $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ is an asymptotically unbiased estimator of $\dot{p}_{\theta|Y,\tau}(\beta_i)$ as $N_s = N/S$ approaches infinity, for finite N_s $\mathbb{E}\left[\ddot{p}_{\theta|Y,\tau}(\beta_i)\right] \neq \dot{p}_{\theta|Y,\tau}(\beta_i)$. Even though the bias goes away as N_s approaches infinity, it is of practical importance to bound the error for finite N_s. For the practitioner the object of interest is the maximum number of shards for partitioning the data as a function of an error bound. I define the error bound as the maximum expected squared error[6] between $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ and $\dot{p}_{\theta|Y,\tau}(\beta_i)$: $\varepsilon_{max}^2 = \sup_\beta\left[\mathbb{E}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right]\right]$. It follows (Appendix: Theorems 7 and 8) that the maximum number of shards is

$$S_{max} \approx \left\lfloor C_0 NR\varepsilon_{max}^2 \right\rfloor \text{ for } S_{max}^2 \gg 1 \tag{3.22}$$

where I may estimate $C_0 = \left\{\sup_{\beta_i}\left[\nabla p(\beta_i|\theta)^T I_\theta^{-1} \nabla p(\beta_i|\theta)\right]\right\}^{-1}$ empirically (Appendix: Theorem 8 and Appendix B: Figure 3.2). I_θ is the Fisher information matrix evaluated at the true value of θ. The proposed algorithm scales with Fisher information, the amount of information that Y

[6] I choose the maximum expected squared error instead of the expected total variation distance for analytical convenience.

carries about θ.

3.5.3 Stage One Subsampling Algorithm $\mathscr{A}_3'$

If $S < S_{max}$, possibly due to a limitation in the number of available machines in the computing environment, then $N_s = \frac{N}{S} > \frac{N}{S_{max}} = N_{s_{max}}$ and $\sup_\beta \left[\mathbb{E}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right]\right] < \varepsilon_{max}^2$. That is, N_s is larger than it needs to be for $\sup_\beta \left[\mathbb{E}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right]\right] = \varepsilon_{max}^2$. I may reduce computation and communication overhead with a modification to the first stage by sampling Y with probability p prior to dividing the data into S shards. The optimal stage one subsampling rate (Appendix: Theorem 9) is

$$p \approx \sqrt{\frac{S}{S_{max}}} \qquad \text{for } S^2 \gg p^2 \quad \text{and} \quad S_{max}^2 \gg 1 \tag{3.23}$$

If $p \ll 1$ I may expect substantial savings in first stage computation. The proposed algorithm with stage one subsampling is described in Figures 3.2 and 3.3.

3.5.4 $p(\beta_i|\theta)$ and $p(y_{it}|\beta_i)$ are Conjugate

For the proposed algorithm, I assumed that $p(\beta_i|\theta)$ and $p(y_{it}|\beta_i)$ are not conjugate. If they are conjugate, I may sample the β_i draws in stage two directly from a mixture whose components are a conjugate distribution. To draw from (3.19): (i) draw multinomial distributed indicator vectors with parameter vectors $\left(\frac{1}{S},...,\frac{1}{S}\right)$ and $\left(\frac{1}{R},...,\frac{1}{R}\right)$ to determine the indices $\{s,r\}$, and (ii) draw β_i from the conjugate distribution $p(\beta_i|\theta_s^r)\prod_t p(y_{it}|\beta_i)$. The performance of this second stage implementation is independent of the number of observations T per unit because I avoid using the independence Metropolis-Hastings algorithm.

3.5.5 Common Parameters

The shard-specific θ draws from the first stage of the proposed algorithm (3.14) are from $p(\theta|Y_s,\tau)$, not $p(\theta|Y,\tau)$. If the researcher is interested in the common parameters, I may simulate draws

Figure 3.3: Algorithm $\mathscr{A}_3'$ Stage One: The proposed algorithm with subsampling

Stage One: Draw $\beta \sim \ddot{p}_{\theta|Y,\tau}(\beta)$
Input: Y, S, p
Output: $\{\beta^r\}$

1. Sample Y with probability $p \in (0,1)$
   (a) $I_i \sim Bernoulli(p)$, $i = 1,...,N$
   (b) Define $I^p = \{i : I_i = 1\}$, $N^p = \sum_i I_i$,
   (c) $Y^p = \{y_{it}\}_{i \in I^p}$
2. Divide Y^p into S shards
   (a) $I_s^p = \{i \in I^p : z_i = s\}$, $s = 1,...,S$, where $p(z_i = s) = \frac{1}{S}$, and $\left|I_s^p\right| = \frac{N^p}{S}$
   (b) $Y_s^p = \{y_{it}\}_{i \in I_s^p}$, $s = 1,...,S$
3. Run S parallel simulations ($s = 1,...,S$)
   (a) for $r = 1$ to R
      i. Draw $\{\beta_i^r\}_{i \in I_s^p}, \theta_s^r \left|\tau, Y_s^p \propto p(\theta|\tau) \prod_{i \in I_s^p} [p(\beta_i|\theta) \prod_t p(y_{it}|\beta_i)]\right.$ (Algorithm $\mathscr{A}_1$)
   (b) keep $\{\theta_s^r\}$ and discard $\{\beta_i^r\}_{i \in I_s^p}$
   (c) for $r = 1$ to R/S
      i. Draw $z^r \sim Multinomial\left(n = 1, p = \left\{\frac{1}{R},...,\frac{1}{R}\right\}\right)$, $z^r \in \{1,...,R\}$
      ii. Draw $\beta_s^r \sim p(\beta \left|\theta_s^{z^r}\right.)$
4. Collect the β_s^r draws and shuffle
   (a) $\{\beta^r\} \leftarrow shuffle(\{\beta_s^r\})$

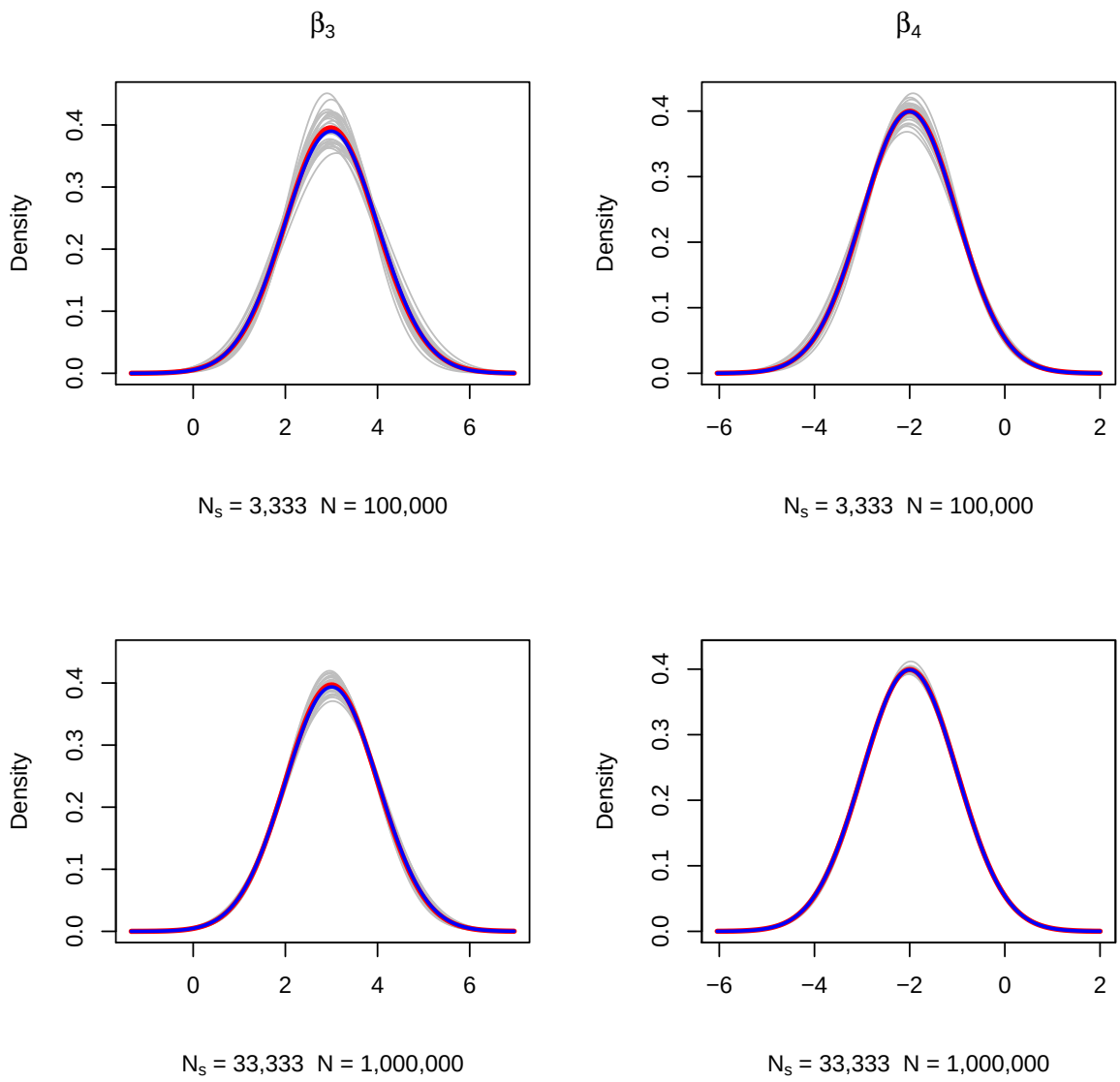
from (3.3) using the second stage unit-level draws $\{\beta_i\}$. If $p(\beta_i|\theta)$ and $p(\theta|\tau)$ are conjugate, as is commonly the case, I may sample the θ draws directly from a conjugate distribution.

4 Simulation

I demonstrate the proposed algorithm using simulated data to estimate a hierarchical multinomial logit model with four choice alternatives and four response parameters, $\beta_i' = (\beta_{i1}, \beta_{i2}, \beta_{i3}, \beta_{i4})$. The first three response parameters are alternative-specific intercepts, and the fourth parameter is the coefficient for a common covariate (for example, price). For data generation I draw each unit's response parameter β_i from a normal distribution with mean $\mu' = (1, 2, 3, -2)$ and covariance Σ equal to the identity matrix. For estimation I impose a normal prior for β_i and a normal inverse Wishart prior for $\theta = \{\mu, \Sigma\}$.

I run the simulation on a 32-core Linux computer, each core representing a single machine. The benchmark algorithm $\mathscr{A}_1$ is a random walk hybrid Gibbs sampler implemented with the R *bayesm* library function `rhierMnlMixture` (Rossi 2015). The independence Metropolis-Hastings algorithm in the second stage of $\mathscr{A}_3$ is implemented in R and C++. Parallelism is implemented with the R *parallel* library function `mclapply`. Due to the memory limitations of my computer, both algorithms return the posterior draws for a random sample of 1,000 cross-sectional units.

Figure 4.1: Convergence of marginals of posterior predictive density estimators



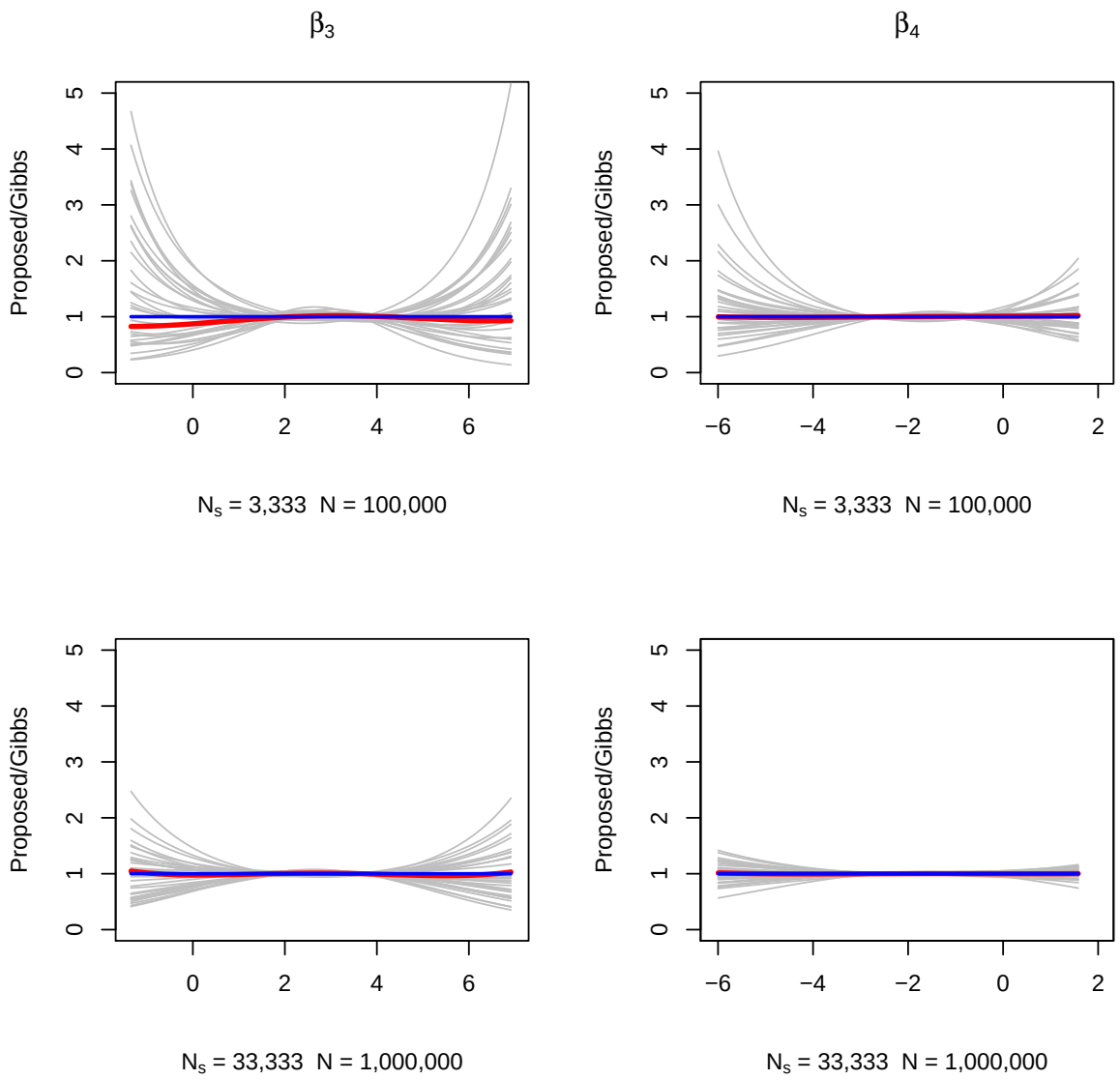
$T = 5$ observations per unit, $S = 30$ shards, and $R = 10{,}000$ MCMC iterations including 2,000 iterations for burn-in ($\dot{p}_{\theta|Y_s,\tau}(\beta_i)$ in grey, $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ in red, $\dot{p}_{\theta|Y,\tau}(\beta_i)$ in blue)

4.1 Convergence

I first evaluate the convergence of $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ to $\dot{p}_{\theta|Y,\tau}(\beta_i)$. Figure 4.1 compares plots of the marginals of $\dot{p}_{\theta|Y,\tau}(\beta_i)$, the S shard-level estimators $\dot{p}_{\theta|Y_s,\tau}(\beta_i)$, and $\ddot{p}_{\theta|Y,\tau}(\beta_i)$. I consider two

elements of β_i: one of the three intercepts β_{i3}, and the coefficient of the alternative-specific covariate β_{i4}. I find that $\ddot{p}_{\theta|Y,\tau}(\beta_i) \approx \dot{p}_{\theta|Y,\tau}(\beta_i)$ at $N_s = 3,333$. $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ converges to $\dot{p}_{\theta|Y,\tau}(\beta_i)$ at a faster rate than $\dot{p}_{\theta|Y_s,\tau}(\beta_i)$ since most of the shard-level marginal densities $\dot{p}_{\theta|Y_s,\tau}(\beta_i)$ do not converge at $N_s = 3,333$ and many do not converge at $N_s = 33,333$. This finding is consistent with the theoretically larger variance of $\dot{p}_{\theta|Y_s,\tau}(\beta_i)$ by a factor of S. Plots of the ratio of marginals are consistent with these findings (Figure 4.2).

Figure 4.2: Convergence of marginals of posterior predictive density estimators



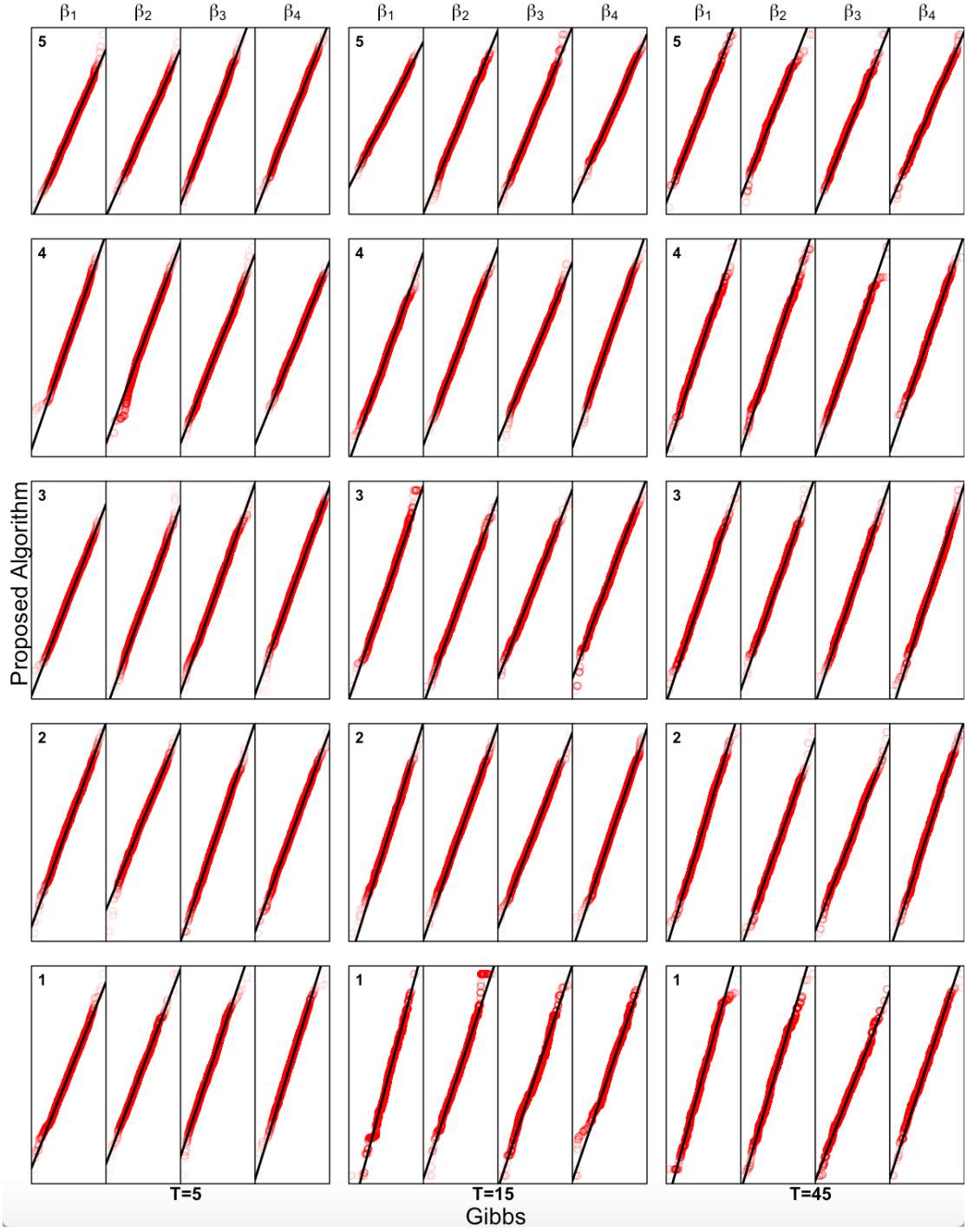
$T = 5$ observations per unit, $S = 30$ shards, and $R = 10,000$ MCMC iterations including 2,000 iterations for burn-in ($\frac{\dot{p}_{\theta|Y_s,\tau}(\beta_i)}{\dot{p}_{\theta|Y,\tau}(\beta_i)}$ in grey, $\frac{\ddot{p}_{\theta|Y,\tau}(\beta_i)}{\dot{p}_{\theta|Y,\tau}(\beta_i)}$ in red, $\frac{\dot{p}_{\theta|Y,\tau}(\beta_i)}{\dot{p}_{\theta|Y,\tau}(\beta_i)} = 1$ in blue)

I next evaluate the convergence of unit-level posterior densities $\ddot{p}(\beta_i|Y,\tau)$ to $p(\beta_i|Y,\tau)$. The Q-Q plots in Figure 4.3 compare quantiles of the β_i draws from the proposed algorithm with those of the benchmark for a random sample of five cross-sectional units, for $T = 5$, 15, and 45 observations

per unit, and $N_s = 3,333$ units per shard ($N = 100,000$). For convenience in interpretation I plot the 45-degree line. Qualitatively, I find excellent convergence for $T = 5$ and good convergence for $T = 15$ and $T = 45$ (for example, see unit 1). The diminishment is due to the independence Metropolis-Hastings algorithm in the second stage. As I increase T unit-level posterior densities narrow. Narrower posteriors decrease acceptance rates. Fewer accepted draws reduce effective sample sizes, increase bias, and negatively impact convergence.

The random walk of the benchmark algorithm for the Metropolis-Hastings draws causes acceptance rates to increase slightly as T increases (from 20.2% at $T = 5$ to 22.9% at $T = 45$). For the independence Metropolis-Hastings algorithm in the second stage of the proposed algorithm acceptance rates decrease with T (from 36.4% at $T = 5$ to 5.0% at $T = 45$). The random walk algorithm adapts to the shape of each unit's posterior whereas the independence algorithm draws from a fixed proposal distribution that is independent of any unit's posterior. To compensate for this effect I may increase the number of proposal draws for those units that have a large number of observations and low acceptance rates.

Figure 4.3: Convergence of unit-level posteriors: Q-Q plots ($N_s = 3,333$)



Q-Q plots of unit-specific draws from the proposed algorithm and the single machine hybrid Gibbs algorithm, for a random sample of five units and $T = 5$, 15, and 45 observations per unit. $N_s = 3,333$ units per shard, $N = 100,000$ units, $S = 30$ shards, and $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in

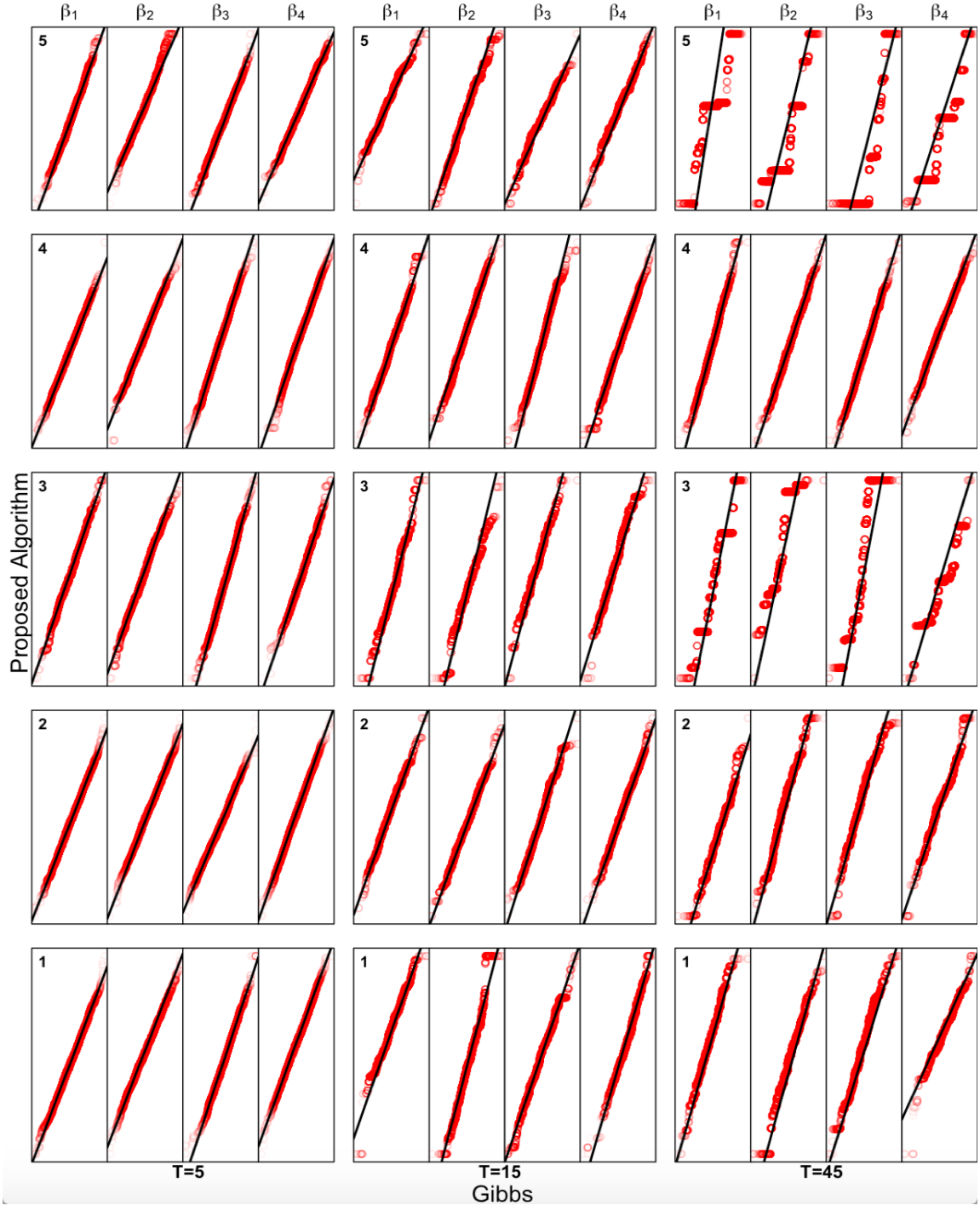
Table 4.1: Convergence of unit-level posterior densities: Q-Q Correlations ($N_s = 3,333$)

T	β_1			β_2			β_3			β_4		
	1%ile	5%ile	Median	1%ile	5%ile	Median	1%ile	5%ile	Median	1%ile	5%ile	Median
5	.994	.998	.999	.991	.997	.999	.994	.998	.999	.992	.998	.999
15	.982	.993	.999	.977	.993	.999	.976	.993	.999	.978	.994	.999
45	.912	.976	.997	.926	.974	.997	.909	.976	.997	.936	.979	.998

Correlation percentiles of unit-specific draw quantiles from the single machine hybrid Gibbs algorithm and the proposed algorithm, for a random sample of 1,000 units. $N = 100,000$ units, $S = 30$ shards, and $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in

Since correlation is a measure of the linear relationship between two variables I may quantify the linear relationship between quantiles of the draws of the proposed algorithm and those of the benchmark. A correlation that is very close to one indicates an exact linear relationship (a straight line) and excellent convergence. Table 4.1 presents the 1^{st} and 5^{th} correlation percentiles, and the median correlation of unit-level draws for a random sample of 1,000 cross-sectional units. Median correlations are excellent (.997 - .999) across all values of T and suggest an exact linear relationship. However in the tails I see some decline especially at larger values of T. For $T = 5$, the 5^{th} percentiles are excellent (.997 - .998) and the 1^{st} percentiles decrease slightly (.991 - .994). For $T = 15$ the 5^{th} percentiles remain excellent (.993 - .994) but the 1^{st} percentiles indicate poor convergence (.976 - .982). For $T = 45$ both the 5^{th} percentiles (.974 - .979) and the 1^{st} percentiles (.909 - .936) suggest poor convergence.

Figure 4.4: Convergence of unit-level posteriors: Q-Q plots ($N_s = 33,333$)



Q-Q plots of unit-specific draws from the proposed algorithm and the single machine hybrid Gibbs algorithm, for a random sample of five units and $T = 5$, 15, and 45 observations per unit. $N_s = 33,333$ units per shard, $N = 1,000,000$ units, $S = 30$ shards, and $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in

Table 4.2: Convergence of unit-level posterior densities: Q-Q correlations ($N_s = 33,333$ units per shard)

T	β_1			β_2			β_3			β_4		
	1%ile	5%ile	Median	1%ile	5%ile	Median	1%ile	5%ile	Median	1%ile	5%ile	Median
5	.993	.998	.999	.995	.997	.999	.995	.998	.999	.994	.998	.999
15	.980	.994	.999	.978	.993	.999	.979	.993	.999	.982	.995	.999
45	.916	.973	.997	.919	.974	.997	.918	.970	.997	.900	.973	.997

Correlation percentiles of unit-specific draw quantiles from the single machine hybrid Gibbs algorithm and the proposed algorithm, for a random sample of 1,000 units. $N_s = 33,333$ units per shard, $N = 1,000,000$ units, $S = 30$ shards, and $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in

Figure 4.4 and Table 4.2 present convergence results for $N_s = 33,333$ ($N = 1,000,000$). The correlations for $N_s = 3,333$ (Table 4.1) agree closely with those for $N_s = 33,333$ (Table 4.2). The reason why correlations do not increase further at $N_s = 33,333$ is because convergence occurs at $N_s \approx 3,333$: once $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ converges to $\dot{p}_{\theta|Y,\tau}(\beta_i)$ at $N_s \approx 3,333$ any further increase in N_s negligibly decreases the maximum expected squared error between $\dot{p}_{\theta|Y,\tau}(\beta_i)$ and $\ddot{p}_{\theta|Y,\tau}(\beta_i)$. Although several units in Figure 4.4 (see units 3 and 5 for $T = 45$) converge rather poorly I surmise that this may be due to the small sample of units chosen for the figures. The reason why these particular units converge poorly is because T is large and their posteriors are likely in the tails of the predictive density.

4.2 Performance

To quantify the performance of the proposed algorithm relative to the benchmark I consider four metrics: execution time, effective sample size (ESS), ESS per minute, and the ratio of ESS/minute of the proposed algorithm to the hybrid Gibbs sampler. The effective sample size for correlated draws is the size of an i.i.d. sample with the same variance as the simulated draws (Robert and Casella, 2010). ESS per minute is therefore a measure of the amount of information obtained from posterior draws per unit of computing time. It quantifies the efficiency of an MCMC chain.

Table 4.3: Performance of the proposed algorithm

T	Algorithm	$N_s = 3,333$ $N = 100,000$				$N_s = 33,333$ $N = 1,000,000$			
		Time	ESS	ESS/min	$\frac{\text{ESS}_{\text{Proposed}}/\text{min}}{\text{ESS}_{\text{Gibbs}}/\text{min}}$	Time	ESS	ESS/min	$\frac{\text{ESS}_{\text{Proposed}}/\text{min}}{\text{ESS}_{\text{Gibbs}}/\text{min}}$
5	Gibbs	158	1,141	7.2	38.0	1,553	1,145	.7	37.6
	Proposed	12	3,301	273.7		117	3,254	27.7	
15	Gibbs	203	1,170	5.8	12.4	2,140	1,172	.5	13.2
	Proposed	19	1,362	72.2		182	1,309	7.2	
45	Gibbs	388	1,212	3.1	4.0	3,649	1,209	.3	3.3
	Proposed	36	444	12.4		384	426	1.1	

Performance metrics are for a random sample of 1,000 units. Ratios relative to the single machine hybrid Gibbs algorithm are in parentheses. For the subsampling algorithm, $N_s = 3,333$ units per shard in the first stage (first stage sampling rate $p = 10\%$), $N_s = 33,333$ units per shard in the second stage. $S = 30$ shards, and $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in

For small T ($T = 5$) the proposed algorithm dominates the single machine hybrid Gibbs algorithm (Table 4.3) in two respects: (i) distributed processing decreases execution time by an order of magnitude, and (ii) the independence Metropolis-Hastings algorithm produces less correlated draws, resulting in an ESS that is about three times larger. For $N = 1,000,000$ the proposed algorithm takes two hours versus 26 hours for the benchmark. Efficiency (ESS per minute) relative to the benchmark is larger by a factor of thirty-seven.

For moderate and large T($T = 15$ and 45) the proposed algorithm remains an order of magnitude faster but ESS declines, especially for large T. The decline is attributed to the expected decrease in acceptance rates. The proposed algorithm takes three hours for moderate T (versus 36 for the benchmark) and 6.5 hours for large T (versus 2.5 days for the benchmark). Efficiency gains relative to the benchmark decrease to a factor of thirteen for moderate T and to 3-4 times for large T.

4.3 Scalability with N

I evaluate the scalability of the proposed algorithm for N up to 100 million units on a large cluster with a parallel distributed file system. Each Linux node has 24 cores. I implement parallelism

with scripts and C++ and I optimize the I/O intensive parts of the R code. For each run, $T = 5$ observations and $R = 20,000$ MCMC iterations, including 4,000 iterations for burn-in. I keep every 10^{th} draw for each of the N units. I set the number of shards S so that $N_s = 33,333$ or larger.

Table 4.4: Scalability with N

Units N	Shards S	Units per Shard $N_s = \frac{N}{S}$	Total Execution Time in minutes (I/O time)[1]
1 million	30	33,333	61 (0)
10 million	300	33,333	76 (2)
100 million	1,728[2]	57,870	162 (21)

Scalability testing is implemented on a large cluster with a parallel distributed file system. $T = 5$ observations, $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in and keeping every 10^{th} draw.

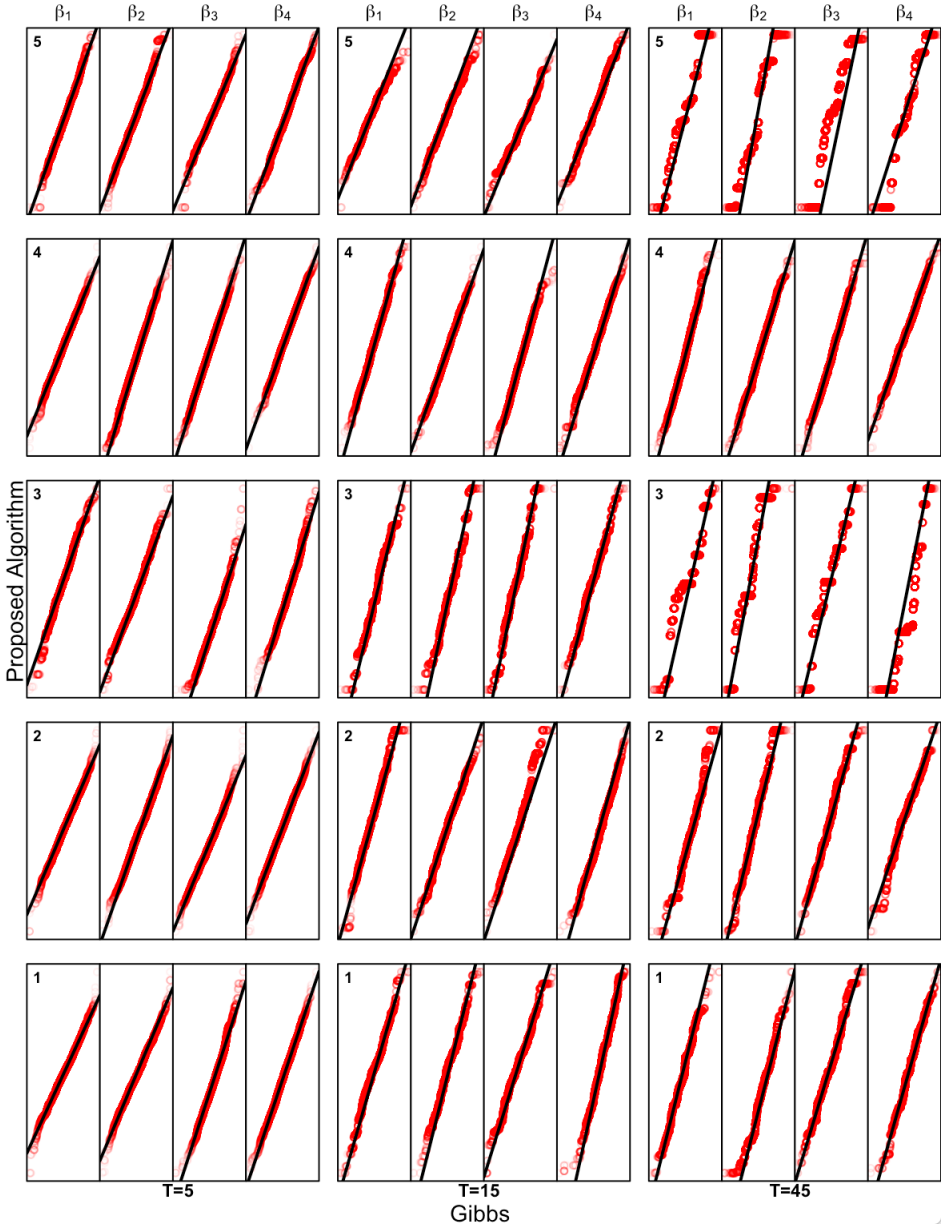
1. I/O time is the amount of time in minutes that is used for transferring data to and from each node for each stage. It is included in the total execution time.
2. The cluster limits the number of cores that a single application may use at one time to 1,728

For $N = 1$ million units and $T = 5$ the proposed algorithm (Table 4.4) runs about twice as fast (61 minutes) on the large cluster compared to the 32-core Linux computer (117 minutes). For $N = 10$ million the proposed algorithm runs in 76 minutes, a relatively small increase from 61 minutes for a ten-fold increase in N. Communication overhead increases to 2 minutes due to the ten-fold increase in the number of file transfers between worker machines and the master computer. For $N = 100$ million I need 3,000 shards to maintain $N_s = 33,333$, however the cluster limits us to 1,728 cores. Execution time increases to 162 minutes, still a relatively modest amount of time for a ten-fold increase in N. Communication overhead increases to 21 minutes due to the even larger number of file transfers. Although I do not compare the efficiency of the proposed algorithm to the benchmark, I expect that efficiency gains also increase approximately linearly with N provided that the computing environment is not limited in the number of available cores.

4.4 Subsampling in Stage One

Since $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ converges to $\dot{p}_{\theta|Y,\tau}(\beta_i)$ at $N_s \approx 3,333$ I may subsample Y at rate $p = 10\%$ in the first stage (algorithm $\mathscr{A}_3'$). For $N = 1$ million units and $S = 30$ shards, $N_s = pN/S = 3,333$. I do not find any difference in first stage convergence between $N_s = 3,333$ and $N_s = 33,333$ (Figure 4.5 and Table 4.5, as compared to Figure 4.4 and Table 4.2).

Figure 4.5: Convergence of unit-level posterior densities (with stage one subsampling): Q-Q plots



Q-Q plots of unit-specific draws from the proposed subsampling algorithm and the single machine hybrid Gibbs algorithm, for a random sample of five units and $T = 5$, 15, and 45 observations per unit. $N_s = 3,333$ units per shard in the first stage, $N_s = 33,333$ units per shard in the second stage, $N = 1,000,000$ units, $S = 30$ shards, and $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in

Table 4.5: Convergence of unit-level posterior densities (with stage one subsampling): Q-Q correlations

T	β_1			β_2			β_3			β_4		
	1%ile	5%ile	Median	1%ile	5%ile	Median	1%ile	5%ile	Median	1%ile	5%ile	Median
5	.992	.996	.999	.994	.997	.999	.990	.996	.999	.992	.9978	.999
15	.971	.993	.999	.972	.992	.999	.974	.993	.999	.980	.995	.999
45	.908	.970	.997	.913	.968	.997	.916	.971	.997	.887	.971	.998

Correlation percentiles of unit-specific draw quantiles from the single machine hybrid Gibbs algorithm and the proposed algorithm with stage one subsampling, for a random sample of 1,000 units. $N_s = 3,333$ units per shard in the first stage, $N_s = 33,333$ units per shard in the second stage, $N = 1,000,000$ units, $S = 30$ shards, and $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in

Subsampling improves performance and efficiency substantially. On the 32-core Linux computer for small T, execution time decreases and efficiency increases by a factor of four, resulting in an efficiency gain of over two orders of magnitude relative to the benchmark (Table 4.6). For moderate and large T relative efficiency more than doubles.

Table 4.6: Performance of the proposed algorithm (with stage one subsampling)

T	Algorithm	$N_s = 33,333$ $N = 1,000,000$			
		Time (minutes)	ESS	ESS/minute	$\frac{\text{ESS}_{\text{Proposed}}/\text{min}}{\text{ESS}_{\text{Gibbs}}/\text{min}}$
5	Proposed	117	3,254	27.7	37.6
	Subsampling*	27	3,294	122.0	165.4
15	Proposed	182	1,309	7.2	13.2
	Subsampling*	68	1,318	19.3	35.3
45	Proposed	384	426	1.1	3.3
	Subsampling*	174	430	2.5	7.5

Performance metrics are for a random sample of 1,000 units. For the proposed algorithm with first stage subsampling, $N_s = 3,333$ units per shard in the first stage (first stage sampling rate $p = 10\%$), $N_s = 33,333$ units per shard in the second stage. $S = 30$ shards, and $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in

On the large cluster (Table 4.7) for $N = 1$ and 10 million units, subsampling reduces execution time by a factor of four to 13 minutes and 19 minutes, respectively. For $N = 100$ million units execution time is halved from 162 minutes to 78 minutes. Communication overhead is not affected by subsampling.

Table 4.7: Scalability with N (with stage one subsampling)

Units N	Shards S	Units per Shard $N_s = \frac{N}{S}$	Algorithm	Total Execution Time in minutes (I/O time)[1]
1 million	30	33,333	Proposed Subsampling[2]	61 (0) 13 (0)
10 million	300	33,333	Proposed Subsampling	76 (2) 19 (3)
100 million	1,728[3]	57,870	Proposed Subsampling	162 (21) 78 (19)

Scalability testing is implemented on a large cluster with a parallel distributed file system. $T = 5$ observations, $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in and keeping every 10^{th} draw.

1. I/O time is the amount of time in minutes that is used for transferring data to and from each node for each stage. It is included in the total execution time.
2. For the proposed algorithm with stage one subsampling $p = 10\%$.
3. The cluster limits the number of cores that a single application may use at one time to 1,728

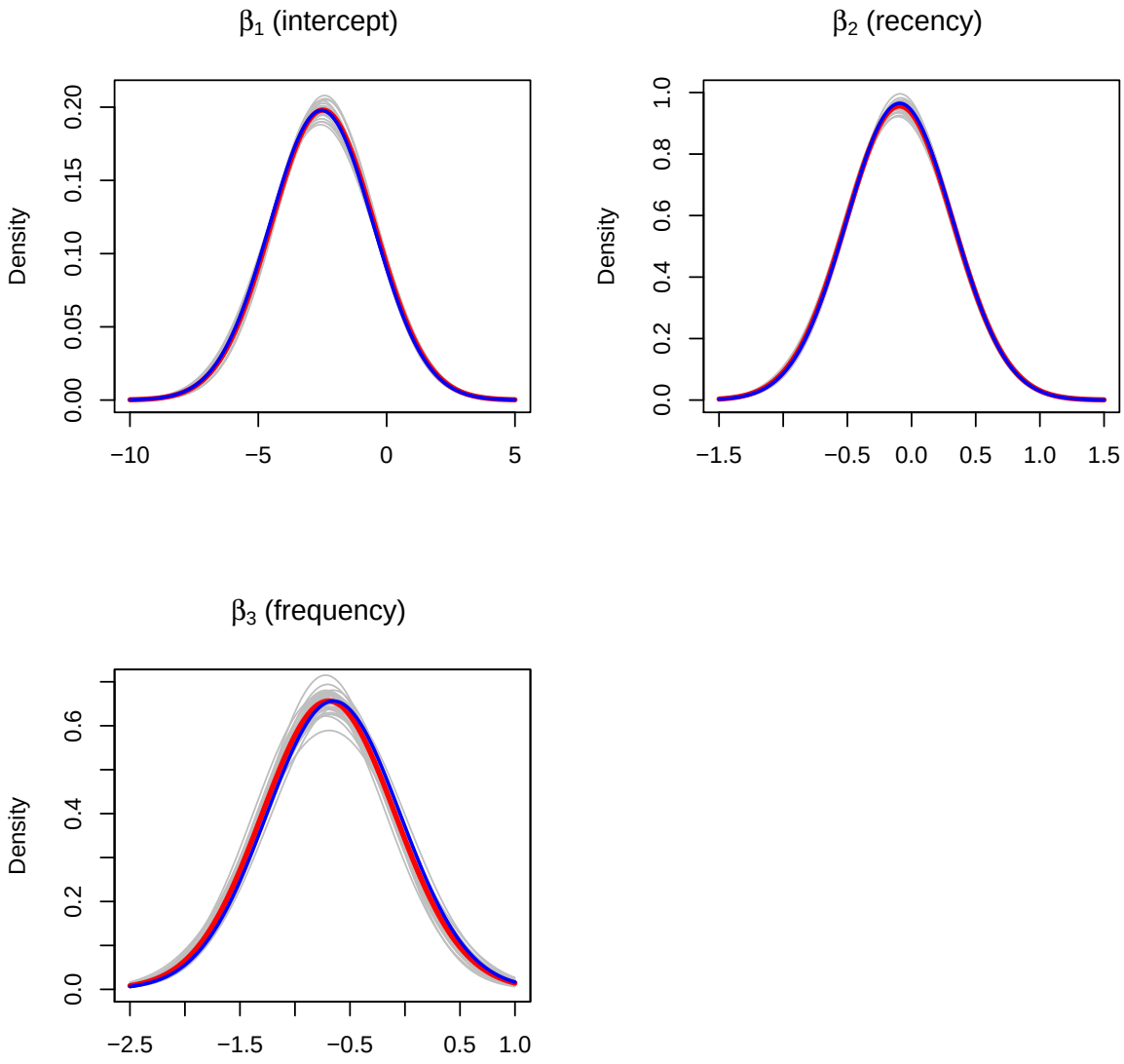
5 Application: Donor Response

I illustrate the proposed algorithm by modeling donor response to solicitations using data from a nonprofit charitable organization. Since the number of donors is very large, the need for an efficient estimation method is clear.

The data from a leading US nonprofit organization (Malthouse 2009) contains the donation and solicitation histories for about 1.1 million donors, 3 million donations, and 28 million solicitations for donations over a fifteen-year period (1992-2006). Data for each donor is collected beginning from the date of her first donation. The total number of donors increases linearly over time. I remove incomplete histories, reducing the number of cross-sectional units to 1,088,269. Donors make an average of 2.8 donations (standard deviation is 4.3) with a mean interdonation time of 362 days (standard deviation is 262 days). The 25^{th}, 50^{th}, and 75^{th} percentiles of the total number of donations per donor after their initial donation are 0, 0, and 2, respectively. The majority of donors only donate once or at most several times before "lapsing" (Feng 2014). Only about seven percent of all solicitations after the first donation result in donations. The 25^{th}, 50^{th}, and 75^{th} percentiles of the number of solicitations per donor are 12, 21, and 31, respectively.

The dependent variable represents whether or not a donor responds to a solicitation, given that she has donated at least once. As covariates I use an intercept, the number of days since her last donation (recency), and the number of past donations (frequency). The average of past donation amounts does not influence donor response. I log transform the data and estimate a hierarchical binomial logit model of solicitation response with my proposed algorithm and the single machine hybrid Gibbs algorithm. I use a normal prior for β_i and a normal inverse Wishart prior for $\theta = \{\mu, \Sigma\}$, the mean and covariance matrix of the normal prior.

Figure 5.1: Donor response: Convergence of marginals of posterior predictive density estimators

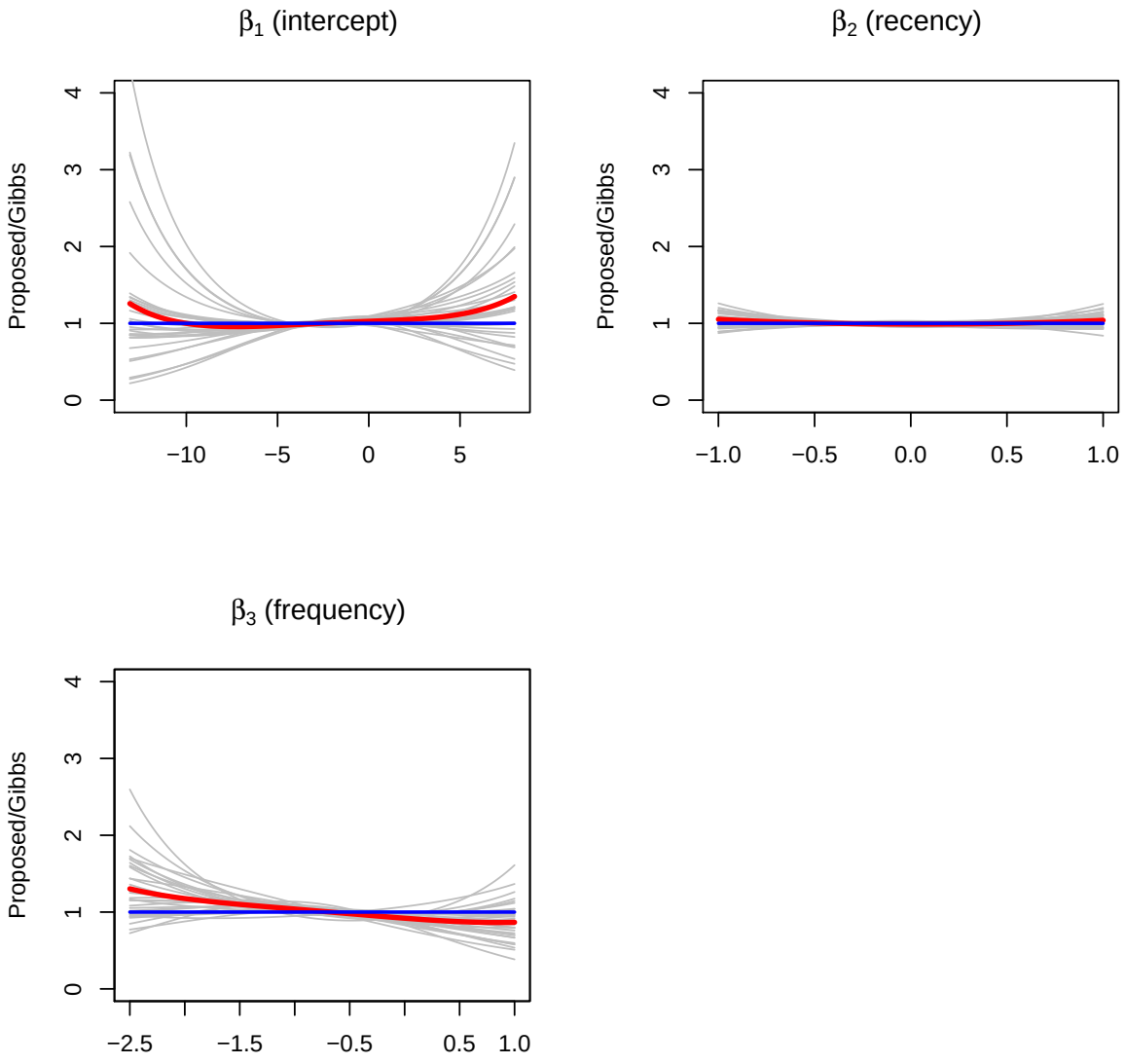


$S = 30$ shards, $N_s \approx 36,000$ units per shard, and $R = 40,000$ MCMC iterations including 5,000 iterations for burn-in ($\dot{p}_{\theta|Y_s,\tau}(\beta_i)$ in grey, $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ in red, $\dot{p}_{\theta|Y,\tau}(\beta_i)$ in blue)

Figure 5.1 plots the marginals of $\dot{p}_{\theta|Y,\tau}(\beta_i)$, $\dot{p}_{\theta|Y_s,\tau}(\beta_i)$, and $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ for $S = 30$ shards ($N_s \approx 36,000$). Marginal posterior predictive densities converge well. Although plots of the ratio of marginals (Figure 5.2) show that β_1 (intercept) and β_3 (frequency) do not converge as well as β_2 (recency), especially in the tails, unit-level posterior densities for a random sample of 24 units

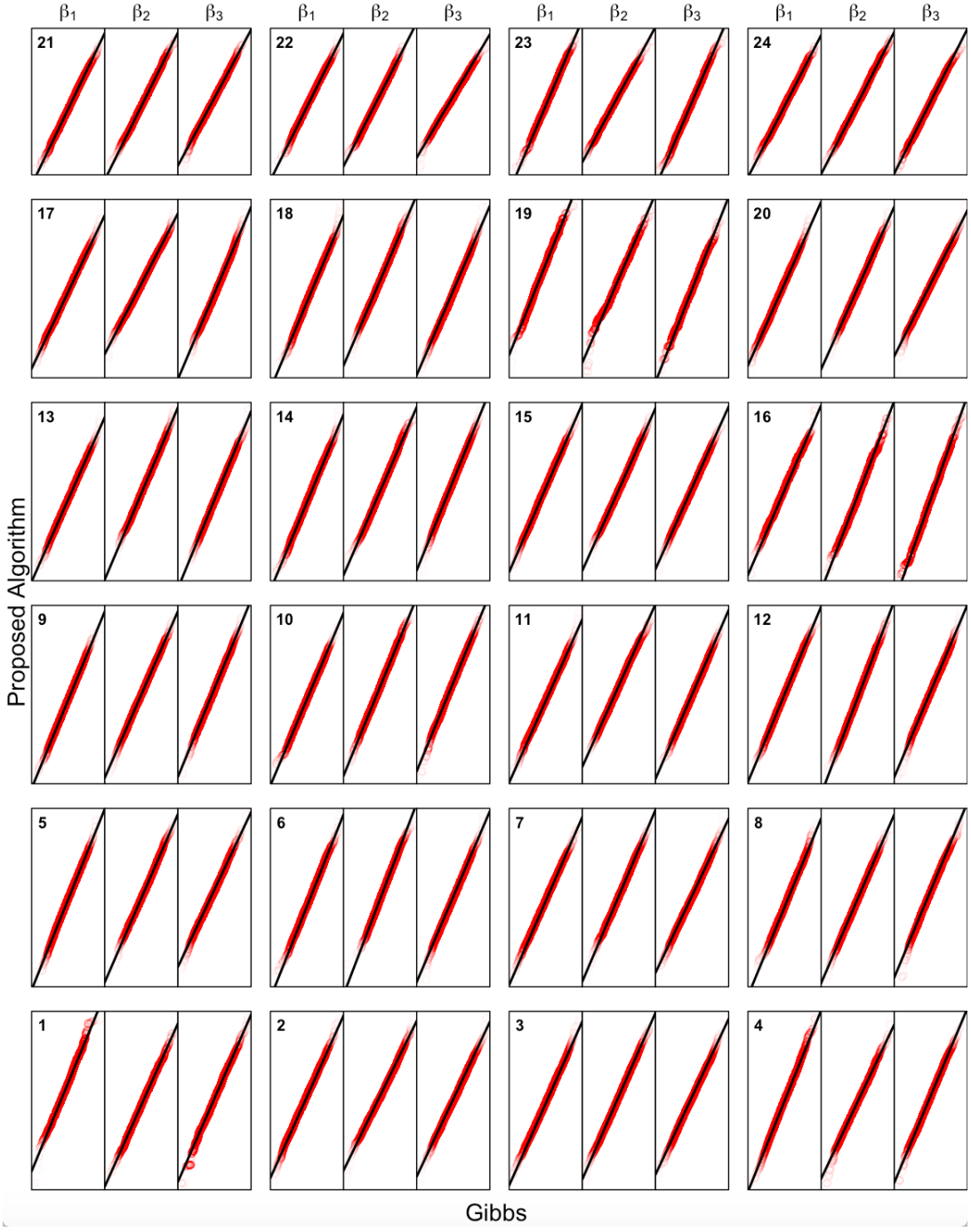
converge well (Figure 5.3). Presumably, this is because there are a sufficiently large number of observations per unit (median of 21 observations per unit) in the likelihood to influence the posterior far enough away from the posterior predictive density. Correlation percentiles of unit-specific draw quantiles for a random sample of 1,000 cross-sectional units confirm my qualitative findings (the 1^{st} and 5^{th} percentiles and the median are 995, .999, and 1.0).

Figure 5.2: Donor response: Convergence of marginals of posterior predictive density estimators



$S = 30$ shards, $N_s \approx 36,000$ units per shard, and $R = 40,000$ MCMC iterations including 5,000 iterations for burn-in ($\frac{\dot{p}_{\theta|Y_s,\tau}(\beta_i)}{\dot{p}_{\theta|Y,\tau}(\beta_i)}$ in grey, $\frac{\ddot{p}_{\theta|Y,\tau}(\beta_i)}{\dot{p}_{\theta|Y,\tau}(\beta_i)}$ in red, $\frac{\dot{p}_{\theta|Y,\tau}(\beta_i)}{\dot{p}_{\theta|Y,\tau}(\beta_i)} = 1$ in blue)

Figure 5.3: Donor response: Q-Q plots showing convergence of unit-level posteriors



Q-Q plots of unit-specific draws from the proposed algorithm and the single machine hybrid Gibbs algorithm, for a random sample of twenty-four units. $N_s \approx 36,000$ units per shard, $N = 1,088,269$ units, $S = 30$ shards, and $R = 40,000$ MCMC iterations including 5,000 iterations for burn-in

The proposed algorithm takes about 6 hours versus 3 days and 5 hours for the benchmark. Acceptance rates are 54% and 23% for the proposed algorithm and the benchmark, respectively. Effective sample sizes (ESS) are 15,389 and 3,178, respectively (out of 40,000 MCMC iterations including 5,000 for burn-in), and ESS per minute are 40.15 and 0.69, respectively. The proposed algorithm is more efficient than the single machine hybrid Gibbs algorithm by a factor of 58, significantly better than the 13-fold efficiency gain with simulated data for $T = 15$ (Table 4.3). This may be due to the simulated dataset's high variability compared to the more realistic low variability of the donation dataset. The donation dataset's smaller Fisher information, due to its lower variability, induces a flatter likelihood function even though there are a median of 21 observations per unit for the donation dataset and only 15 for the comparable simulation dataset. Higher acceptance rates (54% versus 17%), values of ESS (15,389 versus 1,309), and efficiency gains (58 versus 13) are a consequence. In practice, the simulation dataset's high Fisher information may not be representative of real datasets, which suggests that the efficiency gain of the proposed algorithm is expected to be markedly higher than cited using the simulation dataset in this article.

5.1 Scalability with S

The constant C_0 in (3.22) characterizes the proposed algorithm's inherent scalability which is dependent on Y and the model. I estimate C_0 for the donation and simulation datasets with the aid of small scale simulations (Appendix: Theorem 8 and Table 5.1). Simulations take 1 hour for the donation dataset and 20 minutes for the simulation dataset. Our estimates of C_0 are off by 10-20%. Comparing estimates of C_0 for the donation ($C_0 = 7.980 \times 10^{-7}$) and simulation ($C_0 = 2.278 \times 10^{-4}$) datasets, measures of their respective Fisher information about θ, the simulation data carries about three orders of magnitude more information about the common parameters θ than does the donation data. The proposed algorithm scales better with the simulation dataset.

Table 5.1: Scalability with S: Estimation of C_0

Dataset	C_0 Estimation[1] with $N' \ll N$		Squared Error[2]	
	Actual Maximum Squared Error ε^2	C_0	Maximum Expected Squared Error ε^2_{max}	Actual Maximum Squared Error ε^2_{actual}
Simulation	9.143×10^{-5}	2.278×10^{-4}	8.24×10^{-6}	1.04×10^{-5}
Donation	1.193×10^{-2}	7.980×10^{-7}	9.88×10^{-4}	8.98×10^{-4}

1. To estimate C_0 for the simulation dataset, I first construct $\dot{p}_{\theta|Y,\tau}(\beta_i)$ and $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ with sufficiently large $N' \ll N$ and sufficiently small $S' > 1$. I choose $N' = 10,000$ units, $S' = 3$ shards, $R = 16,000$ MCMC iterations after burn-in (4,000). I approximate the maximum expected squared error as the actual maximum squared error $\varepsilon^2 \approx \sup_{\beta_{ik}, k \in \{1,\ldots,d\}} \left[\left| \ddot{p}_{\theta|Y,\tau}(\beta_{ik}) - \dot{p}_{\theta|Y,\tau}(\beta_{ik}) \right|^2 \right]$ and approximate $C_0 \approx \left(\frac{S'^2+1}{S'N'R\varepsilon^2} \right)$. For the donation dataset I use $N' = 10,000$ units, $S' = 3$ shards, $R = 35,000$ MCMC iterations after burn-in (5,000) to estimate ε^2 and calculate C_0.
2. I solve for the maximum expected squared error ε^2_{max} for $S_{max} = 30$ because I are limited to a maximum of 30 cores in my computing environment: $\varepsilon^2_{max} \approx \left(\frac{S^2_{max}+1}{S_{max}NR} \right) C_0^{-1}$. For the simulation dataset $N = 1,000,000$ units, $S = 30$ shards, $R = 16,000$ MCMC iterations after burn-in (4,000). For the donation dataset $N = 1,088,269$ units, $S = 30$ shards, $R = 35,000$ MCMC iterations after burn-in (5,000). The actual maximum squared error is estimated by constructing $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ and using $\varepsilon^2_{actual} \approx \sup_{\beta_{ik}, k \in \{1,\ldots,d\}} \left[\left| \ddot{p}_{\theta|Y,\tau}(\beta_{ik}) - \dot{p}_{\theta|Y,\tau}(\beta_{ik}) \right|^2 \right]$.

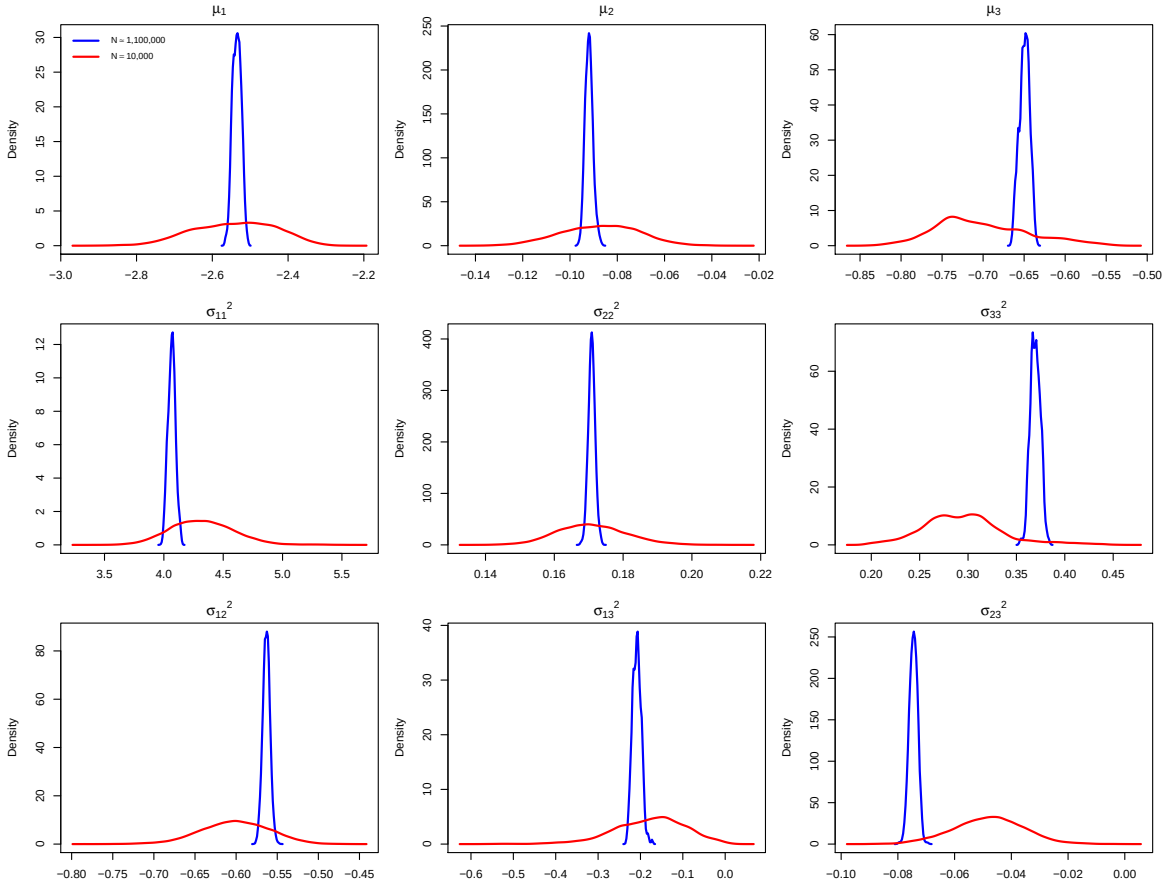
I conclude that although low Fisher information datasets are more efficient in terms of ESS per unit of computing, they limit the scalability of the proposed algorithm with S. I expect that for low Fisher information datasets S_{max} is more likely to limit scalability than the number of available computers. For high Fisher information datasets the number of available computers is more likely to limit scalability. In the latter case I may increase performance by subsampling in the first stage to accommodate the limited number of available computers, while keeping within a given error bound.

5.2 Subsampling Units

Although the focus of this article is to propose a scalable algorithm for estimating all of the unit-level parameters in Bayesian hierarchical models, it is interesting to examine the effect of a sub-

sampling[7]. I show that inferences from a subsample of units are very different from the those based on all of the data. Even large subsamples can provide incorrect inferences.

Figure 5.4: Donor response: Marginal posterior densities of μ and Σ (for moderate and big N)



$R = 40,000$ MCMC iterations including 5,000 iterations for burn-in

I consider a moderate subsample ($N = 10,000$) which can easily run on a single processor MCMC implementation. I compare inferences about the common parameters for this subsample with the full sample of data ($N \approx 1,100,000$). In my hierarchical binary logit model of donor response, I assume that unit-level response parameters have a multivariate normal distribution, $\beta_i \sim N(\mu, \Sigma)$. The common parameter θ consists of a mean vector μ and the unique elements of the covariance

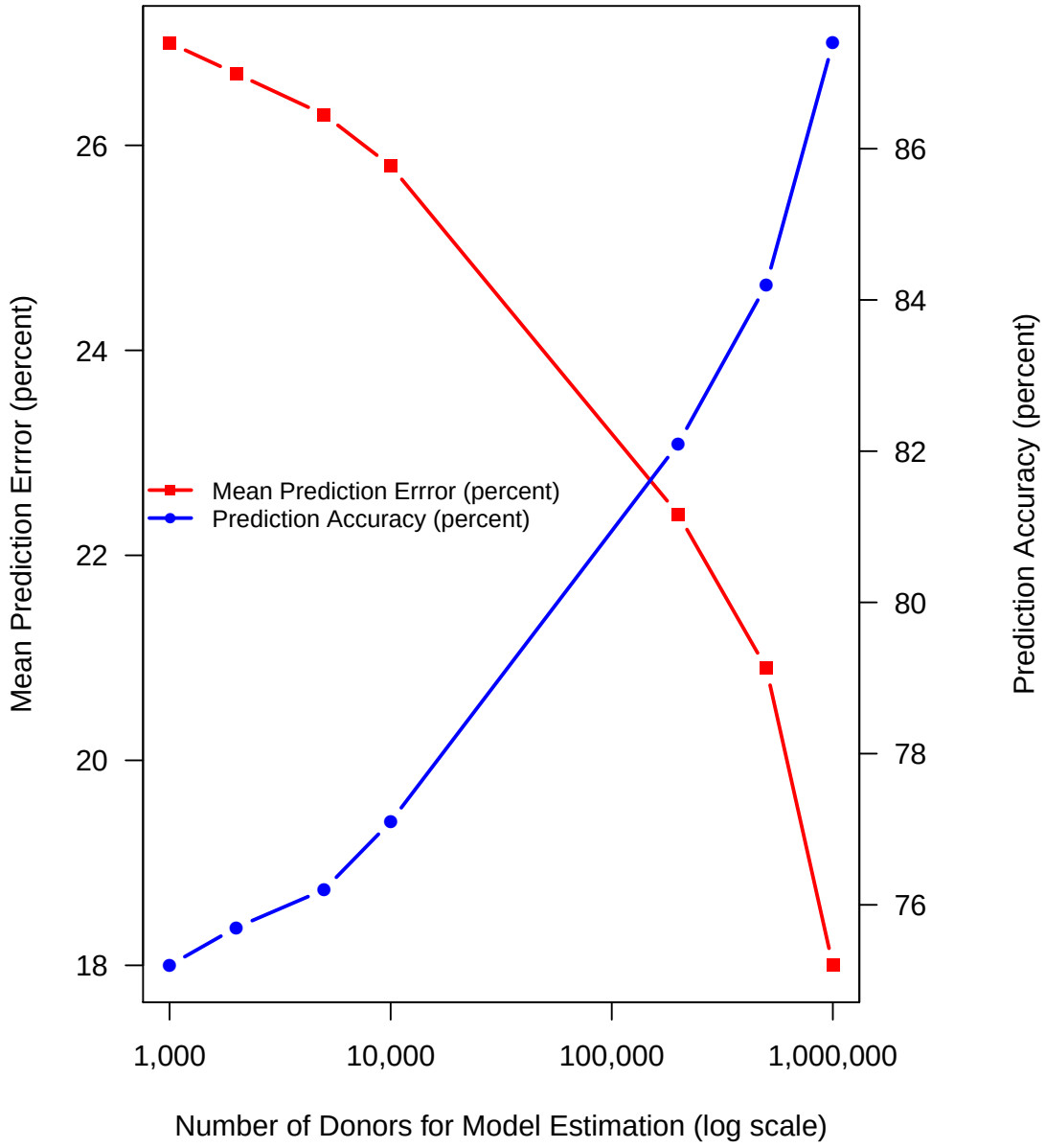
[7]Subsampling in this context refers to subsampling the cross-sectional units for purposes of estimating the common parameters, whereas in the context of the proposed algorithm it is for purposes of constructing an estimator of the posterior predictive density.

matrix Σ. I plot marginal posterior densities (Figure 5.4) of the common parameter draws for moderate and big N. Standard deviations for all elements of $\{\mu, \Sigma\}$ decrease by a factor of about 10, as expected when increasing the sample size by 100. More interestingly, there is a clear separation of the posterior means for several elements of μ and Σ between moderate and big N. Multivariate tests for the equality of means are significant[8]. This suggests that a big N approach to parameter estimation has the added benefit of different and presumably more accurate parameter estimates and characterization of heterogeneity, and that the proposal to subsample units may provide misleading inferences.

To illustrate the managerial relevance of different parameter estimates with increasing subsample size I consider a prediction task. I estimate a hierarchical binomial logit model of solicitation response to predict the probability of response for a focal subsample of 1,000 random donors with at least 16 observations (solicitations). The first 15 observations are used for model estimation and the remaining for prediction. The prediction period is donor-dependent, it is not a fixed time interval. I estimate the model with an increasing number of donors from 1,000 to 1 million. In all cases the first 1,000 donors is the focal subsample used for prediction. For the remaining donors beyond the first 1,000, at most 15 observations are used for model estimation. A model with an intercept term and three covariates predicts well: time since last donation, number of donations in the current year, and total donation amount in the current year.

[8] Krishnamoorthy and Yu's (2004) modified Nel and van der Merwe multivariate test for the equality of means are significant. p-values for μ and the unique elements of Σ are 4.866×10^{-16} and 1.346×10^{-35}, respectively.

Figure 5.5: Donor response: Predicted response probability



Mean prediction error is the mean absolute deviation between the predicted probability of response and the observed probability of response (0 or 1). Prediction accuracy is the proportion of correct predictions, wherein a prediction probability greater than fifty-percent is interpreted as a positive predicted response.

Figure 5.5 plots the mean prediction error[9] and prediction accuracy[10] as a function of the number of donors for model estimation. Mean prediction error decreases from 27% when estimating a model with 1,000 donors, to 18% with 1 million donors. Prediction accuracy increases from 75% to 87%. A 12% increase in efficiency can be substantial when managing a very large donor pool.

[9]Mean prediction error is the mean absolute deviation between the predicted probability of response and the observed probability of response (0 or 1).

[10]Prediction accuracy is the proportion of correct predictions, wherein a prediction probability greater than fifty-percent is interpreted as a positive predicted response.

6 Conclusions

I propose a distributed MCMC algorithm for estimating Bayesian hierarchical models when the number of units is very large (big N) and the objects of interest are the unit-level parameters. The method is asymptotically exact, retains the flexibility of any standard MCMC algorithm to accommodate any prior, has a computational complexity that is independent of the number of shards, does not impose any distributional assumptions on posteriors, has lean communication requirements, and is easy to implement using existing MCMC packages.

For small T, the proposed algorithm dominates the performance of the single machine hybrid Gibbs algorithm in two respects. It is more computationally efficient by distributing its processing, and it is more algorithmically efficient by simulating draws that are less correlated. This double-win produces an overall efficiency gain of at least an order of magnitude (for $N = 1,000,000$) relative to the single machine hybrid Gibbs algorithm. To boost performance further, a modification to the proposed algorithm subsamples the data in the first stage. Using simulated data with $N = 1,000,000$ and $T = 5$, the single machine hybrid Gibbs algorithm takes 26 hours to run, the proposed algorithm needs 2 hours, and the proposed algorithm with stage one subsampling runs in 30 minutes. For larger T, the algorithm still dominates the single machine hybrid Gibbs algorithm in the sense of delivering about an order of magnitude greater effective sample size per unit of computing even though the mixing properties are not as favorable.

I apply the proposed algorithm to a panel of one million donors to model donor response to solicitations. The efficiency gain, as measured by effective sample size per unit of computing, is markedly higher than that of my simulated examples. This is due to the comparatively low Fisher information of the donor dataset. In general, I expect that for real applications efficiency gains of the proposed algorithm are greater than those cited for the simulated examples in this article.

The proposed algorithm may be implemented on a multicore computer or a cluster of computers. I have demonstrated its scalability with simulated and real panels of one million units on a multicore computer, a pedestrian environment. The potential for scalability is even greater on a large cluster. My simulations suggest that efficiency gains increase approximately linearly with N.

With 100,000,000 units and utilizing a 1,728-processor cluster, only a few hours of computation time is required to undertake unit-level parameter inference.

7 References

Bardenet, Remi, Arnaud Doucet, and Chris Holmes (2015), "On Markov chain Monte Carlo methods for tall data" arXiv:1505.02827

Beaumont, Mark A. (2003), "Estimation of Population Growth or Decline in Genetically Monitored Populations", *Genetics, 164,* 1139–1160.

Feng, Shanfei (2014), "Getting Lapsed Donors Back: An Empirical Investigation of Relationship Management in the Post-Termination Stage", J*ournal of Non-Profit and Public Sector Marketing*, 26 (2), 127-141.

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin (2014*), Bayesian Data Analysis*, Third Edition, CRC Press.

Grimmett, Geoffrey and David Stirzaker (2007), *Probability and Random Processes*, Third Edition, Oxford.

Krishnamoorthy, K. and Jianqi Yu (2004), "Modified Nel and Van der Merwe test for the multivariate Behrens-Fisher problem", *Statistics & Probability Letters*, 66 (2), 161–169.

Le Cam, Lucien, and Grace Lo Yang (2000), *Asymptotics in Statistics: Some Basic Concepts*. Springer-Verlag.

Malthouse, Edward C. (2009), "The Results from the Lifetime Value and Customer Equity Modeling Competition", *Journal of Interactive Marketing*, 23, 272-275.

Minsker, Stanislav, Sanvesh Srivastava, Lizhen Lin, and David B. Dunson (2014), "Scalable and Robust Bayesian Inference via the Median Posterior", *Proceedings of the 31st International Conference on Machine Learning.*

Neiswanger, Willie, Chong Wang, and Eric Xing (2014), "Asymptotically exact, embarrassingly parallel MCMC", *Proceedings of the 30th International Conference on Uncertainty in Artificial Intelligence.*

Robert, Christian P., and George Casella (2010), *Monte Carlo Statistical Methods*, Second Edition, Springer.

Rossi, Peter E. (2015), *bayesm: Bayesian Inference for Marketing/MicroEconometrics*, 3.0 ed.

Rossi, Peter E., and Greg M. Allenby (1993), "A Bayesian Approach to Estimating Household Parameters", *Journal of Marketing Research*, 30 (2), 171-182.

Rossi, Peter E., Greg M. Allenby and Robert McCulloch (2005), *Bayesian Statistics and Marketing*, John Wiley & Sons.

Scott, Steven L., Alexander W. Blocker, Fernando V. Bonassi, Hugh A. Chipman, Edward I. George, and Robert E. McCulloch (2016) "Bayes and Big Data: The Consensus Monte Carlo Algorithm*", International Journal of Management Science and Engineering Management*, 11 (2), 78-88.

Srivastava, Sanvesh, Volkan Cevher, Quoc Tran-Dinh, and David B. Dunson (2015), "WASP: Scalable Bayes via Barycenters of Subset Posteriors*", Proceedings of the 18th International Conference on Artificial Intelligence and Statistics.*

Vaart, A. W. van der (1998), *Asymptotic Statistics*, Cambridge University Press.

8 Appendix: Theorems

In the following proofs, $\dot{p}_{\theta|Y,\tau}(\beta_i)$, $\dot{p}_{\theta|Y_s,\tau}(\beta_i)$ and $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ are estimators of posterior predictive densities constructed from convergent MCMC chains. $Y = \{Y_s\}_{s=1}^{S}$ is the data for all N units, Y_s is the data in shard s for $N_s = \frac{N}{S}$ units, and $S > 1$ is the number of shards. $\beta_i \in \mathbb{R}^d$ where d is the dimension of β_i.

Theorem 1. *Gelman et al.'s (2014) posterior predictive density of β_i is $\mathbb{E}_{\theta|Y_{-i},\tau}[p(\beta_i|\theta)]$*

Proof. Gelman et al.'s (2014) definition of the posterior predictive distribution of y_i is the density of y_i after observing Y_{-i}, where Y_{-i} excludes y_i, an unknown observable. I extend this idea to the unit-level parameter β_i in a Bayesian hierarchical setting: the posterior predictive distribution of β_i is the density of β_i after observing Y_{-i}.

$$\begin{aligned} p(\beta_i|Y_{-i},\tau) &= \int p(\beta_i|\theta)\, p(\theta|Y_{-i},\tau)\, d\theta \\ &= \mathbb{E}_{\theta|Y_{-i},\tau}[p(\beta_i|\theta)] \end{aligned}$$

□

Theorem 2. $\mathbb{E}\left[\dot{p}_{\theta|Y,\tau}(\beta_i)\right] = \mathbb{E}_{\theta|Y,\tau}[p(\beta_i|\theta)]$ *for* $\beta_i \in \mathbb{R}^d$

Proof. The proof is inspired by Beaumont (2003).

$$\begin{aligned} \mathbb{E}\left[\dot{p}_{\theta|Y,\tau}(\beta_i)\right] &= \mathbb{E}_{\{\theta^r\}|Y,\tau}\left[\frac{1}{R}\sum_{r=1}^{R} p(\beta_i|\theta^r)\right] \\ &= \int \frac{1}{R}\sum_{r=1}^{R} p(\beta_i|\theta^r)\, p(\{\theta^r\}|Y,\tau)\, d\{\theta^r\} \\ &= \int\cdots\int \frac{1}{R}\sum_{r=1}^{R} p(\beta_i|\theta^r)\prod_{j=1}^{R} p\left(\theta^j|Y,\tau\right) d\theta^1 \ldots d\theta^R \\ &= \frac{1}{R}\sum_{r=1}^{R}\int p(\beta_i|\theta^r)\, p(\theta^r|Y,\tau)\, d\theta^r \prod_{j\neq r}\int p\left(\theta^j|Y,\tau\right) d\theta^j \\ &= \frac{1}{R}\sum_{r=1}^{R} \mathbb{E}_{\theta^r|Y,\tau}[p(\beta_i|\theta^r)] \qquad \int p\left(\theta^j|Y,\tau\right) d\theta^j = 1 \\ &= \mathbb{E}_{\theta|Y,\tau}[p(\beta_i|\theta)] \end{aligned}$$

□

Theorem 3. *$\dot{p}(\beta_i|Y,\tau) = p(\beta_i|Y,\tau)$ for $\beta_i \in \mathbb{R}^d$*

Proof.

$$\begin{aligned}
\dot{p}(\beta_i|Y,\tau) &= \int \dot{p}(\beta_i|\{\theta^r\},Y,\tau)\, p(\{\theta^r\}|Y,\tau)\, d\{\theta^r\} \\
&= \mathbb{E}_{\{\theta^r\}|Y,\tau}\left[\dot{p}(\beta_i|\{\theta^r\},Y,\tau)\right] \\
&= \mathbb{E}_{\{\theta^r\}|Y,\tau}\left[\dot{p}_{\theta|Y,\tau}(\beta_i)\prod_t p(y_{it}|\beta_i)\right] \\
&= \mathbb{E}_{\{\theta^r\}|Y,\tau}\left[\dot{p}_{\theta|Y,\tau}(\beta_i)\right]\prod_t p(y_{it}|\beta_i) \qquad p(y_{it}|\beta_i) \text{ is not stochastic} \\
&= \mathbb{E}_{\theta|Y,\tau}\left[p(\beta_i|\theta)\right]\prod_t p(y_{it}|\beta_i) \qquad \text{Theorem 2} \\
&= p(\beta_i|Y,\tau) \qquad \text{Equation 3.7}
\end{aligned}$$

□

Theorem 4. *The limit distributions of $\dot{p}_{\theta|Y_s,\tau}(\beta_i)$ and $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ for $\beta_i \in \mathbb{R}^d$ are:*

1. $\sqrt{N}\left(\dot{p}_{\theta|Y_s,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right) \to^P N\left(0, \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{R}\right) I_\theta^{-1} \nabla p(\beta_i|\theta)\right)$
2. $\sqrt{N}\left(\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right) \to^P N\left(0, \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{SR}\right) I_\theta^{-1} \nabla p(\beta_i|\theta)\right)$

Proof. **1.** limit distribution of $\sqrt{N}\left(\dot{p}_{\theta|Y_s,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right)$

1.1. limit distribution of $\sqrt{N}\left(\dot{p}_{\theta|Y,\tau}(\beta_i) - p(\beta_i|\theta)\right)$ for $\beta_i \in \mathbb{R}^d$

For purposes of deriving limit distributions, I take a frequentist view in that Y is a random sample from a distribution for some fixed, nonrandom, unknown parameter. I assume that N is large enough that standard asymptotics apply and that the Bernstein-von Mises theorem yields a good approximation to the posterior (Le Cam and Yang 2000; Vaart 1998). In particular, I assume that posterior distributions approach a normal distribution centered at the true parameter value with covariance equal to the inverse of the Fisher information matrix divided by N.

Let θ_N^r denote the r^{th} draw from the posterior distribution of θ using algorithm $\mathscr{A}_1$ with data Y for N units. Therefore $\sqrt{N}(\theta_N^r - \theta) \to^d N\left(0, I_\theta^{-1}\right)$ where I_θ is the Fisher information matrix at θ

for N units of data Y. Since θ is a unknown constant $\sqrt{N}(\theta_N^r - \theta) \to^P N\left(0, I_\theta^{-1}\right)$ by Vaart (1998) Theorem 2.7. For notational simplicity I dispense with the N subscript for θ^r.

I derive the limit distribution of $\sqrt{N}\left(\dot{p}_{\theta|Y,\tau}(\beta_i) - p(\beta_i|\theta)\right)$ by applying the multivariate delta method to $\sqrt{N}(\theta^r - \theta) \to^P N\left(0, I_\theta^{-1}\right)$ and using the transformation $\dot{p}_{\theta|Y,\tau}(\beta_i) = \frac{1}{R}\sum_r p(\beta_i|\theta^r)$. $\beta_i \in \mathbb{R}^d$

$$
\begin{aligned}
\sqrt{N}(\theta^r - \theta) &\to^P N\left(0, I_\theta^{-1}\right) \\
\sqrt{N}(p(\beta_i|\theta^r) - p(\beta_i|\theta)) &\to^P N\left(0, \nabla p(\beta_i|\theta)^T I_\theta^{-1} \nabla p(\beta_i|\theta)\right) \\
&\qquad \text{multivariate delta method} \\
\sqrt{N}\left(\sum_r p(\beta_i|\theta^r) - \sum_r p(\beta_i|\theta)\right) &\to^P N\left(0, \nabla p(\beta_i|\theta)^T R I_\theta^{-1} \nabla p(\beta_i|\theta)\right) \\
&\qquad \text{sum of R random variables} \\
\sqrt{N}\left(\frac{1}{R}\sum_r p(\beta_i|\theta^r) - p(\beta_i|\theta)\right) &\to^P N\left(0, \nabla p(\beta_i|\theta)^T (RI_\theta)^{-1} \nabla p(\beta_i|\theta)\right) \\
&\qquad \text{divide by R} \\
\sqrt{N}\left(\dot{p}_{\theta|Y,\tau}(\beta_i) - p(\beta_i|\theta)\right) &\to^P N\left(0, \nabla p(\beta_i|\theta)^T (RI_\theta)^{-1} \nabla p(\beta_i|\theta)\right)
\end{aligned}
$$

1.2. limit distribution of $\sqrt{N}\left(\dot{p}_{\theta|Y_s,\tau}(\beta_i) - p(\beta_i|\theta)\right)$ for $S > 1$, $\beta_i \in \mathbb{R}^d$

Similarly, for each shard of $N_s = \frac{N}{S}$ units of data Y_s, $\sqrt{N_s}(\theta^r - \theta) \to^P N\left(0, SI_\theta^{-1}\right)$, where $\frac{I_\theta}{S}$ is the Fisher information matrix at θ for $\frac{N}{S}$ units of data. I assume that $\frac{N}{S}$ is large enough so that the Bernstein-von Mises theorem applies. Therefore, following the reasoning in step 1.1,

$$
\begin{aligned}
\sqrt{N_s}\left(\dot{p}_{\theta|Y_s,\tau}(\beta_i) - p(\beta_i|\theta)\right) &\to^P N\left(0, \nabla p(\beta_i|\theta)^T S(RI_\theta)^{-1} \nabla p(\beta_i|\theta)\right) \\
\sqrt{N}\left(\dot{p}^s_{\theta|Y,\tau}(\beta_i) - p(\beta_i|\theta)\right) &\to^P N\left(0, \nabla p(\beta_i|\theta)^T S^2 (RI_\theta)^{-1} \nabla p(\beta_i|\theta)\right) \\
&\qquad \text{multiply by } \sqrt{S}
\end{aligned}
$$

1.3. limit distribution of $\sqrt{N}\left(\dot{p}_{\theta|Y_s,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right)$ for $S > 1$, $\beta_i \in \mathbb{R}^d$

Subtract $\sqrt{N}\left(\dot{p}_{\theta|Y,\tau}(\beta_i) - p(\beta_i|\theta)\right)$ from $\sqrt{N}\left(\dot{p}_{\theta|Y_s,\tau}(\beta_i) - p(\beta_i|\theta)\right)$

$$\begin{aligned}
\sqrt{N}\left(\dot{p}_{\theta|Y_s,\tau}(\beta_i) - p(\beta_i|\theta)\right) - \sqrt{N}\left(\dot{p}_{\theta|Y,\tau}(\beta_i) - p(\beta_i|\theta)\right) &\to^P N\Big(0, \nabla p(\beta_i|\theta)^T \left[S^2 (RI_\theta)^{-1} + (RI_\theta)^{-1}\right] \\
&\quad \times \nabla p(\beta_i|\theta)\Big) \\
\sqrt{N}\left(\dot{p}_{\theta|Y_s,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right) &\to^P N\left(0, \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{R}\right) I_\theta^{-1} \nabla p(\beta_i|\theta)\right)
\end{aligned}$$

2. limit distribution of $\sqrt{N}\left(\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right)$ for $\beta_i \in \mathbb{R}^d$

Apply the transformation $\ddot{p}_{\theta|Y,\tau}(\beta_i) = \frac{1}{S}\sum_s \dot{p}_{\theta|Y_s,\tau}(\beta_i)$

$$\begin{aligned}
\sqrt{N}\left(\dot{p}_{\theta|Y_s,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right) &\to^P N\left(0, \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{R}\right) I_\theta^{-1} \nabla p(\beta_i|\theta)\right) \\
&\quad \text{see step 1.3} \\
\sqrt{N}\left(\frac{1}{S}\sum_s \left(\dot{p}_{\theta|Y_s,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right)\right) &\to^P N\left(0, \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{SR}\right) I_\theta^{-1} \nabla p(\beta_i|\theta)\right) \\
&\quad \text{mean of S rand. vars.} \\
\sqrt{N}\left(\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right) &\to^P N\left(0, \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{SR}\right) I_\theta^{-1} \nabla p(\beta_i|\theta)\right)
\end{aligned}$$

□

Theorem 5. $\lim_{N\to\infty} \mathbb{E}\left[\ddot{p}_{\theta|Y,\tau}(\beta_i)\right] = \lim_{N/S\to\infty} \mathbb{E}\left[\ddot{p}_{\theta|Y,\tau}(\beta_i)\right] = \dot{p}_{\theta|Y,\tau}(\beta_i)$ *for* $\beta_i \in \mathbb{R}^d$

Proof. Let θ_N^r denote the r^{th} draw of θ from the posterior density $p(\theta|Y,\tau)$, where N is the number of units of data in Y. It is reasonable to assume that the sequence θ_1^r, θ_2^r,... of random vectors is uniformly integrable because I may choose any prior density $p(\theta|\tau)$ that appropriately restricts the amount of probability in the tails of posterior $p(\theta|Y,\tau)$. Since $\theta_N^r \to^P \theta$ (Theorem 4 step 1.1) and θ_N^r is uniformly integrable, it follows that $\theta_N^r \to^{L_1} \theta$ by Grimmett and Stirzaker (2007) Theorem 7.10(3). Therefore, for $\beta_i \in \mathbb{R}^d$, $\dot{p}_{\theta|Y,\tau}(\beta_i) \to^{L_1} p(\beta_i|\theta)$ by the same reasoning as in Theorem 4 (replace convergence in probability with L_1-convergence).

Similarly, by the above reasoning, $\theta_{N_s}^r \to^{L_1} \theta$ and $\dot{p}_{\theta|Y_s,\tau}(\beta_i) \to^{L_1} p(\beta_i|\theta)$. Again, following the same reasoning as in Theorem 4 (replace convergence in probability with L_1-convergence), it follows that $\ddot{p}_{\theta|Y,\tau}(\beta_i) \to^{L_1} \dot{p}_{\theta|Y,\tau}(\beta_i)$ and therefore that $\lim_{N\to\infty} \mathbb{E}\left[\ddot{p}_{\theta|Y,\tau}(\beta_i)\right] = \dot{p}_{\theta|Y,\tau}(\beta_i)$. For fixed S, $\lim_{N\to\infty} \mathbb{E}\left[\ddot{p}_{\theta|Y,\tau}(\beta_i)\right] = \lim_{N/S\to\infty} \mathbb{E}\left[\ddot{p}_{\theta|Y,\tau}(\beta_i)\right]$. □

Theorem 6. $\lim_{N\to\infty}\mathbb{E}\left[\ddot{p}\left(\beta_i|Y,\tau\right)\right]=\lim_{N/S\to\infty}\mathbb{E}\left[\ddot{p}\left(\beta_i|Y,\tau\right)\right]=p\left(\beta_i|Y,\tau\right)$ *for* $\beta_i\in\mathbb{R}^d$

Proof.

$$\begin{aligned}
\lim_{N\to\infty}\mathbb{E}\left[\ddot{p}\left(\beta_i|Y,\tau\right)\right] &= \mathbb{E}\left[\lim_{N\to\infty}\ddot{p}\left(\beta_i|Y,\tau\right)\right] \qquad \text{Dominated Convergence Theorem}\\
&= \mathbb{E}\left[\lim_{N\to\infty}\mathbb{E}\left[\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)\prod_t p\left(y_{it}|\beta_i\right)\right]\right]\\
&= \mathbb{E}\left[\lim_{N\to\infty}\mathbb{E}\left[\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)\right]\right]\prod_t p\left(y_{it}|\beta_i\right) \qquad p\left(y_{it}|\beta_i\right)\text{ is not stochastic}\\
&= \mathbb{E}\left[\dot{p}_{\theta|Y,\tau}\left(\beta_i\right)\right]\prod_t p\left(y_{it}|\beta_i\right) \qquad \text{Theorem 5}\\
&= \mathbb{E}_{\theta|Y,\tau}\left[p\left(\beta_i|\theta\right)\right]\prod_t p\left(y_{it}|\beta_i\right) \qquad \text{Theorem 2}\\
&= p\left(\beta_i|Y,\tau\right) \qquad \text{Equation 3.7}
\end{aligned}$$

where in the first line the Dominated Convergence Theorem applies because
$\ddot{p}\left(\beta_i|Y,\tau\right)=\mathbb{E}\left[\ddot{p}\left(\beta_i|Y,\tau\right)\prod_t p\left(y_{it}|\beta\right)\right]$ is a bounded density function - both
$\ddot{p}\left(\beta_i|Y,\tau\right)=\frac{1}{SR}\sum_s\sum_r p(\beta_i|\theta_s^r)$ and $p\left(y_{it}|\beta_i\right)$ are bounded density functions. For fixed S,
$lim_{N\to\infty}\mathbb{E}\left[\ddot{p}\left(\beta_i|Y,\tau\right)\right]=lim_{N/S\to\infty}\mathbb{E}\left[\ddot{p}\left(\beta_i|Y,\tau\right)\right]$. □

Theorem 7. *For large finite N, the expected squared error between* $\dot{p}_{\theta|Y_s,\tau}\left(\beta_i\right)$ *and* $\dot{p}_{\theta|Y,\tau}\left(\beta_i\right)$*, and between* $\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)$ *and* $\dot{p}_{\theta|Y,\tau}\left(\beta_i\right)$*, for* $\beta_i\in\mathbb{R}^d$*, are:*

1. $\mathbb{E}\left[\left|\dot{p}_{\theta|Y_s,\tau}\left(\beta_i\right)-\dot{p}_{\theta|Y,\tau}\left(\beta_i\right)\right|^2\right]\approx\left(\frac{S^2+1}{NR}\right)\nabla p\left(\beta_i|\theta\right)^T I_\theta^{-1}\nabla p\left(\beta_i|\theta\right)$
2. $\mathbb{E}\left[\left|\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)-\dot{p}_{\theta|Y,\tau}\left(\beta_i\right)\right|^2\right]\approx\left(\frac{S^2+1}{SNR}\right)\nabla p\left(\beta_i|\theta\right)^T I_\theta^{-1}\nabla p\left(\beta_i|\theta\right)$

Proof. I make rather strong assumptions for the purpose of approximating the expected squared error. I do not claim rigorous results. Given that
$\sqrt{N}\left(\dot{p}_{\theta|Y_s,\tau}\left(\beta_i\right)-\dot{p}_{\theta|Y,\tau}\left(\beta_i\right)\right)\to^P N\left(0,\nabla p\left(\beta_i|\theta\right)^T\left(\frac{S^2+1}{R}\right)I_\theta^{-1}\nabla p\left(\beta_i|\theta\right)\right)$ under the assumptions of Theorem 4, I additionally assume that a finite N is sufficiently large such that for $\beta_i\in\mathbb{R}^d$ I may justify the approximation

$$\dot{p}_{\theta|Y_s,\tau}\left(\beta_i\right)-\dot{p}_{\theta|Y,\tau}\left(\beta_i\right)\sim N\left(\frac{B_s\left(\beta_i\right)}{N},\nabla p\left(\beta_i|\theta\right)^T\left(\frac{S^2+1}{NR}\right)I_\theta^{-1}\nabla p\left(\beta_i|\theta\right)\right)$$

where $\frac{B_s(\beta_i)}{N}$ is a small non-zero bias. I assume that $\dot{p}_{\theta|Y_s,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)$ is approximately normally distributed with mean $\frac{B_s(\beta_i)}{N}$ and variance $\nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{NR}\right) I_\theta^{-1} \nabla p(\beta_i|\theta)$ for $\beta_i \in \mathbb{R}^d$, and

$$\begin{aligned}
\mathbb{E}\left[\left|\dot{p}_{\theta|Y_s,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right] &\approx Var\left(\dot{p}_{\theta|Y_s,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right) + \mathbb{E}\left[\dot{p}_{\theta|Y_s,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right]^2 \\
&= \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{NR}\right) I_\theta^{-1} \nabla p(\beta_i|\theta) + \left(\frac{B_s(\beta_i)}{N}\right)^2 \\
&\approx \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{NR}\right) I_\theta^{-1} \nabla p(\beta_i|\theta) \qquad \text{large N}
\end{aligned}$$

Similarly, given that $\sqrt{N}\left(\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right) \to^P N\left(0, \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{SR}\right) I_\theta^{-1} \nabla p(\beta_i|\theta)\right)$ under the assumptions of Theorem 4, I additionally assume that a finite N is sufficiently large such that for $\beta_i \in \mathbb{R}^d$ I may justify the approximation

$$\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i) \sim N\left(\frac{B(\beta_i)}{N}, \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{SNR}\right) I_\theta^{-1} \nabla p(\beta_i|\theta)\right)$$

where $\frac{B(\beta_i)}{N}$ is a small non-zero bias. Therefore

$$\begin{aligned}
\mathbb{E}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right] &\approx Var\left(\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right) + \mathbb{E}\left[\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right]^2 \\
&= \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{SNR}\right) I_\theta^{-1} \nabla p(\beta_i|\theta) + \left(\frac{B(\beta_i)}{N}\right)^2 \\
&\approx \nabla p(\beta_i|\theta)^T \left(\frac{S^2+1}{SNR}\right) I_\theta^{-1} \nabla p(\beta_i|\theta) \qquad \text{large N}
\end{aligned}$$

□

Theorem 8. *The maximum number of shards S_{max} for a given maximum expected squared error ε^2_{max}, and an empirical estimate for C_0 are:*

1. $S_{max} \approx \left\lfloor \frac{C_0}{2}\left(NR\varepsilon^2_{max} + \sqrt{(NR\varepsilon^2_{max})^2 - 4C_0^{-2}}\right)\right\rfloor \approx \left\lfloor C_0 NR\varepsilon^2_{max}\right\rfloor$ for $S^2_{max} \gg 1$, where $C_0 = \left\{\sup_\beta \left[\nabla p(\beta_i|\theta)^T I_\theta^{-1} \nabla p(\beta_i|\theta)\right]\right\}^{-1}$

2. $C_0 \approx \left(\frac{S'^2+1}{S'N'R}\right)\left\{\sup_{\beta_{ik},k\in\{1,\ldots,d\}}\left[\left|\ddot{p}_{\theta|Y,\tau}\left(\beta_{ik}\right)-\dot{p}_{\theta|Y,\tau}\left(\beta_{ik}\right)\right|^2\right]\right\}^{-1}$, where $N' \ll N$ is sufficiently large and $S > 1$ is sufficiently small

Proof. Denote the expected squared error between $\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)$ and $\dot{p}_{\theta|Y,\tau}\left(\beta_i\right)$ as $\varepsilon^2\left(\beta_i\right)$ for $\beta_i \in \mathbb{R}^d$. Therefore for $\beta_i \in \mathbb{R}^d$, $\varepsilon^2\left(\beta_i\right) = \mathbb{E}\left[\left|\ddot{p}_{\theta|Y,\tau}\left(\beta_i\right)-\dot{p}_{\theta|Y,\tau}\left(\beta_i\right)\right|^2\right] \approx \left(\frac{S^2+1}{SNR}\right)\nabla p\left(\beta_i|\theta\right)^T I_\theta^{-1}\nabla p\left(\beta_i|\theta\right)$ (Theorem 7). The maximum expected squared error is

$$\begin{aligned}
\varepsilon_{max}^2 &= \sup_{\beta_i}\left[\varepsilon^2\left(\beta_i\right)\right] \\
&\approx \sup_{\beta_i}\left[\left(\frac{S^2+1}{SNR}\right)\nabla p\left(\beta_i|\theta\right)^T I_\theta^{-1}\nabla p\left(\beta_i|\theta\right)\right] \\
&= \left(\frac{S^2+1}{SNR}\right)\sup_{\beta_i}\left[\nabla p\left(\beta_i|\theta\right)^T I_\theta^{-1}\nabla p\left(\beta_i|\theta\right)\right]
\end{aligned}$$

Define $C_0 = \left\{\sup_{\beta_i}\left[\nabla p\left(\beta_i|\theta\right)^T I_\theta^{-1}\nabla p\left(\beta_i|\theta\right)\right]\right\}^{-1}$ so that $\varepsilon_{max}^2 \approx \left(\frac{S^2+1}{SNR}\right)C_0^{-1}$. Solving for the maximum number of shards S_{max} subject to the maximum expected squared error ε_{max}^2

$$\begin{aligned}
S_{max} &\approx \left\lfloor \frac{NR\varepsilon_{max}^2 + \sqrt{\left(NR\varepsilon_{max}^2\right)^2 - 4C_0^{-2}}}{2C_0^{-1}} \right\rfloor \\
&\approx \left\lfloor C_0 NR\varepsilon_{max}^2 \right\rfloor \qquad S_{max}^2 \gg 1
\end{aligned}$$

$\nabla p\left(\beta_i|\theta\right)^T I_\theta^{-1}\nabla p\left(\beta_i|\theta\right)$ must be computed at the true value of θ for $\beta_i \in \mathbb{R}^d$. It is more convenient to empirically estimate $C_0 = \left\{\sup_{\beta}\left[\nabla p\left(\beta_i|\theta\right)^T I_\theta^{-1}\nabla p\left(\beta_i|\theta\right)\right]\right\}^{-1}$ for some small but sufficiently large $N' \ll N$ and a sufficiently small $S' > 1$, so that N'/S' is large enough that the

Bernstein-von Mises theorem applies.

$$\begin{aligned}
\varepsilon_{max}^2 &= \sup_{\beta_i}\left[\varepsilon^2(\beta_i)\right] \\
\left(\frac{S'^2+1}{S'N'R}\right)\sup_{\beta_i}\left[\nabla p(\beta_i|\theta)^T I_\theta^{-1}\nabla p(\beta_i|\theta)\right] &\approx \sup_{\beta_i}\left[\mathbb{E}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_i)-\dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right]\right] \\
\left(\frac{S'^2+1}{S'N'R}\right)C_0^{-1} &\approx \sup_{\beta_i}\left[\frac{1}{M}\sum_m\left|\ddot{p}^m_{\theta|Y,\tau}(\beta_i)-\dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right] \\
C_0 &\approx \left(\frac{S'^2+1}{S'N'R}\right)\left\{\sup_{\beta_i}\left[\frac{1}{M}\sum_m\left|\ddot{p}^m_{\theta|Y,\tau}(\beta_i)-\dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right]\right\}^{-1} \\
&\approx \left(\frac{S'^2+1}{S'N'R}\right)\left\{\sup_{\beta_i}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_i)-\dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right]\right\}^{-1} \quad \text{M=1}
\end{aligned}$$

where $\ddot{p}^m_{\theta|Y,\tau}(\beta_i)$ is the posterior predictive density estimator for the m^{th} random partitioning of data Y into S shards. For computational convenience, I let $M=1$.

Since $\beta_i \in \mathbb{R}^d$, for $d>1$ it may be computationally demanding to estimate $\sup_{\beta_i}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_i)-\dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right]$. Let $\beta_i=(\beta_{i1},...,\beta_{id})^T$ so that $\ddot{p}_{\theta|Y,\tau}(\beta_{ik})$ and $\dot{p}_{\theta|Y,\tau}(\beta_{ik})$ denote the β_{ik}, $k\in\{1,...,d\}$, marginals of $\ddot{p}_{\theta|Y,\tau}(\beta_i)$ and $\dot{p}_{\theta|Y,\tau}(\beta_i)$, respectively. I approximate $\sup_{\beta_i}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_i)-\dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right]$ with $C\sup_{\beta_{ik},k\in\{1,...,d\}}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_{ik})-\dot{p}_{\theta|Y,\tau}(\beta_{ik})\right|^2\right]$ where C is a proportionality constant that may be absorbed by C_0. Therefore

$$C_0 \approx \left(\frac{S'^2+1}{S'N'R}\right)\left\{\sup_{\beta_{ik},k\in\{1,...,d\}}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_{ik})-\dot{p}_{\theta|Y,\tau}(\beta_{ik})\right|^2\right]\right\}^{-1}$$

□

Theorem 9. *The optimal stage one subsampling rate is* $p\approx\sqrt{\frac{S}{S_{max}}}$ *for* $S^2\gg p^2$ *and* $S^2_{max}\gg 1$

Proof. Let p denote the first stage subsampling rate, and replace $N_s=\frac{N}{S}$ with $N_s=\frac{Np}{S}$ in Theorem 4 step 1.2 to show that

$$\sqrt{N}\left(\ddot{p}_{\theta|Y,\tau}(\beta_i)-\dot{p}_{\theta|Y,\tau}(\beta_i)\right)\to^P N\left(0,\nabla p(\beta_i|\theta)^T\left(\frac{S^2+p^2}{SRp^2}\right)I_\theta^{-1}\nabla p(\beta_i|\theta)\right)$$

Follow Theorem 7 to show that

$$\mathbb{E}\left[\left|\ddot{p}_{\theta|Y,\tau}(\beta_i) - \dot{p}_{\theta|Y,\tau}(\beta_i)\right|^2\right] \approx \nabla p(\beta_i|\theta)^T \left(\frac{S^2+p^2}{SNRp^2}\right) I_\theta^{-1} \nabla p(\beta_i|\theta)$$

Follow Theorem 8 to show that

$$\varepsilon_{max}^2 \approx \left(\frac{S^2+p^2}{SNRp^2}\right) C_0^{-1}$$

Solve for p

$$p \approx \sqrt{\frac{S^2}{C_0 SNR\varepsilon_{max}^2 - 1}}$$

Simplify using $S_{max} \approx C_0 NR\varepsilon_{max}^2$ (Theorem 8) and $S^2 \gg p^2$

$$p \approx \sqrt{\frac{S}{S_{max}}} \qquad S^2 \gg p^2 \text{ and } S_{max}^2 \gg 1$$

□