Distributed Markov Chain Monte Carlo for Bayesian Hierarchical Models

Federico (Rico) Bumbaca*
The Paul Merage School of Business
University of California - Irvine

Sanjog Misra†
Booth School of Business
University of Chicago

Peter E. Rossi‡
Anderson School of Management
University of California - Los Angeles

February 2017, revised May 7, 2017, revised June 3, 2017

Abstract

This article proposes 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. The algorithm is demonstrated with simulated data from a hierarchical logit model and is shown to be faster and more efficient (in effective sample size generated per unit of computing) 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. We illustrate our approach with data on 1,100,000 donors to a charitable organization, and simulations with up to 100 million units.

Keywords: distributed Markov chain Monte Carlo, Bayesian hierarchical model

*fbumbaca@uci.edu
†Sanjog.Misra@chicagobooth.edu
‡perossichi@gmail.com

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. Targeted strategies have become increasingly popular. This popularity is, in part, facilitated by the availability of large panel data sets. In principle, these panels allow for unit-level inferences which can form the basis of optimal customised unit-level strategies. Typically, these panels have a very large number of cross-sectional units, N ($> 1,000,000$), but much smaller numbers of observations per unit, T (< 50). The limited number of observations per unit necessitate the use of inference procedures that allow for information to be shared across units.

While it is well-recognized that Bayesian hierarchical models are ideally suited for panel data problems in which unit-level parameters are desired, panel datasets with very large N pose significant computing challenges for existing algorithms. Typically, inference in these models is conducted via a hybrid MCMC algorithm (Rossi, Allenby, and McCulloch, 2005) running on a single processor (see, for example, the `rhierMnlRwMixture` function in the R package, `bayesm`). When the number of units is very large, it is not feasible to simulate from unit-level posterior distributions within a reasonable amount of time, due to processor bottlenecks. Some suggest using only a sample of the cross-sectional units to reduce computational burden. Obviously, this approach does not produce individual level parameter estimates for all units and therefore cannot be used in a modern world characterized by customized and target marketing. Less obviously, we demonstrate that even relatively large subsamples of units can produce misleading inferences even about common parameters such as the degree of heterogeneity in the data.

Another approach to addressing the resource limitation problem is to distribute the data and simulations across multiple machines. In particular, when estimating a Bayesian hierarchical model, 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. However, the communication costs associated with transmitting data between machines on each iteration is prohibitively expensive and not practical (Scott et. al, 2016). Our

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 we introduce in this article. Results are available from the authors.

Bardenet et al. (2015) broadly classify Bayesian MCMC algorithms for big N into two groups, distributed and subsampling algorithms. Distributed approaches parallelize computation in a distributed computing environment, whereas 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, they may be adapted 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. We elaborate in Section 2 why current methods don't work well with hierarchical models.

We 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 for the construction of an estimator for the posterior predictive distribution of the unit-level parameters, and (ii) an independence Metropolis-Hastings or direct sampling algorithm for the simulation of unit-specific parameter draws, using the first stage estimator as the prior. The method is asymptotically exact in that it does not impose any distributional assumptions or approximations on the posterior of the unit-level parameters when the number of units allocated to each worker machine is sufficiently large. Further, it retains the central ideas and flexibility of any standard MCMC algorithm (e.g. hybrid Gibbs sampler) for which any prior may be stipulated, such as a mixture of distributions. While our focus is not on recovering draws of the common parameters, these draws may be easily simulated by conditioning on the available unit-level draws.

The algorithm performs well: for non-standard 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 minute). For small T, the proposed algorithm is more than an order of magnitude faster and more efficient 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 standard posterior distributions, gains are expected to be independent of the number of observations per unit. The algorithm is tested on a large-scale cluster computing system and is found to scale very well 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, we propose a subsampling algorithm which samples the data in the first stage.

The remainder of this paper is organized as follows. Section 2 discusses the related literature. In Section 3, we describe the proposed distributed MCMC algorithm for hierarchical models. Section 4 demonstrates the algorithm using simulated data by estimating a hierarchical logit model, and compares the results with those of the single machine hybrid Gibbs algorithm. In Section 5, we illustrate our method using a large panel of charitable donors. We conclude in Section 6.

2 Related Literature

There is a substantial literature on Bayesian inference for large data sets that do not have a panel structure. For extremely large N, it may be very costly to evaluate the full posterior and, therefore, a wide variety of proposals have been made that use some sort of distributed processing approach which breaks the problem down into subproblems, each one with a managable N. Still other proposals emphasize subsampling to reduce computational requirements. In this section, we will review these proposals and explain how these methods can be 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 generical hierarchical model can be written:

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

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

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

$p(y_{it}|\beta_i)$ 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, β_i are the unit-level parameters. There is a standard two-stage prior with $p(\beta_i|\theta)$ as the first-stage. We refer to θ as common parameters with prior density $p(\theta|\tau)$ and hyper-parameters τ. Although T_i is unit-dependent, we let $T_i = T$ for notational simplicity.

Bardenet et al.'s (2015) discussion of distributed algorithms focuses on those of Scott et al. (2016) and Neiswanger et al. (2014). These algorithms are designed to estimate single layer (not hierarchical) models in an embarrassingly parallel manner, and may be extended to hierarchical models with a two-stage approach, as suggested by Scott et al. (2016). For each of the two stages, the full data $Y = \{y_{it}\}$ is partitioned 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 units allocated 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.

The algorithmic combining of the common parameter draws is based on expressing the posterior for the common parameters as the product of S subposteriors. In simplified notation to more clearly illustrate the idea,

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

$$= \prod_s p(Y_s|\theta)\, p(\theta)^{1/S} \tag{2.5}$$

where the full data prior is $p(\theta) = \prod_{s=1}^{S} p(\theta)^{1/S}$. In our notation for model (2.1-2.3), $p(\theta|Y) = p(\theta, \{\beta_i\}|Y, \tau)$, $p(Y_s|\theta) = p\left(Y_s, \{\beta_i\}_{i \in I_s}|\theta\right) = \prod_{i \in I_s} p(\beta_i|\theta) \prod_t p(y_{it}|\beta_i)$, and $p(\theta) = p(\theta|\tau)$. We omit τ and β_i in (2.4-2.5) for additional clarity.

Given the S collections of common parameter draws (from the S worker machines), the ap-

proaches of Scott et al. (2016) and Neiswanger et al. (2014) differ in how they algorithmically synthesize a single collection of draws from the full data posterior distribution. On the rth iteration of an MCMC simulation, Scott et al. (2016) synthesize the combined draw θ^r with a weighted average of the rth draw of θ from each of the S shards, θ_s^r.

$$\theta^r = \left(\sum_s W_s \right)^{-1} \left(\sum_s W_s \theta_s^r \right) \tag{2.6}$$

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

Several comments about the algorithm are appropriate. First, if subposteriors have disjoint support, a posterior that is the product of subposteriors is poorly approximated (Bardenet et al., 2015). Second, if $p\left(\theta\right)$ is a known standard parametric distribution, it is not necessarily the case that $p\left(\theta\right)^{1/S}$ is also a known standard parametric distribution. The modeler must either approximate $p\left(\theta\right)^{1/S}$ with a suitable parametric distribution, or revert to a possibly less efficient Metropolis-Hastings algorithm for the simulation of θ draws. Third, although the algorithm is exact if $p\left(\theta \left| Y_s \right.\right)$ is normal (since $p\left(\theta \left| Y \right.\right)$ is also normal), it is approximate otherwise.

Forth, the algorithm requires that the parametric distribution for the unit-level parameters $p\left(\beta_i \left| \theta \right.\right)$ be limited to a single component distribution (e.g. normal), due to a problem akin to the label switching problem in Bayesian mixture modeling. Mixture component draws from different shards are independent and do not share a natural correspondence as a basis for combining their draws. For example, θ draws from mixture component k in shard 1 is independent of θ draws from mixture component j in shard 2. Component k in shard 1 may represent a high-probability region of θ whereas component j in shard 2 may represent a non-overlapping low-probability region. It would be inappropriate to algorithmically combine their draws since they represent different and unrelated components.

Neiswanger et al.'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 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 $\hat{p}(\theta|Y_s)$ for the densities of each shard's parameters.

$$p(\theta|Y_s) = \frac{1}{R}\sum_{r_s}\phi\left(\theta\left|\theta_s^{r_s}, h^2 I_d\right.\right) \tag{2.7}$$

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

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

$$\begin{aligned} p(\theta|Y) &\propto \prod_s p(\theta|Y_s) && (2.8)\\ &= \frac{1}{R^S}\prod_s\sum_{r_s}\phi\left(\theta\left|\theta_s^{r_s}, h^2 I_d\right.\right) && (2.9)\\ &\propto \sum_{r_1}\cdots\sum_{r_S} w_{\{r_1,\ldots,r_S\}}\phi\left(\theta\left|\bar{\theta}_{\{r_1,\ldots,r_S\}}, \frac{h^2}{S} I_d\right.\right) && (2.10)\end{aligned}$$

where $\bar{\theta}_{\{r_1,\ldots,r_S\}} = \frac{1}{S}\sum_s \theta_s^{r_s}$ and $w_{\{r_1,\ldots,r_S\}} = \prod_s \phi\left(\theta_s^{r_s}\left|\bar{\theta}_{\{r_1,\ldots,r_S\}}, h^2 I_d\right.\right)$. $p(\theta|Y)$ 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 $\hat{p}(\theta|Y)$ in two steps: (i) drawing a mixture component $\{r_1,\ldots,r_S\}$ with an independence Metropolis with Gibbs sampler (Neiswanger et al., 2014), and (ii) drawing θ^r from this mixture component.

The algorithm has several limitations. First and second, like Scott et al. (2016), if subposteriors have disjoint support, the posterior is poorly approximated, and $p(\theta)^{1/S}$ may not be a known standard parametric distribution. Third, 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 it's complexity may materially impact the computational advantages of a distributed approach for large values of S. Fourth, since the method is based on kernel density estimation, we do not expect it to scale well as the dimension of the parameter space becomes large, due to the curse of dimensionality. Fifth, the bound on the mean-squared-error of the approximated posterior explodes exponentially with the number of shards S (Bardenet et al., 2015).

Bardenet et al. (2015) also mention distributed algorithms that avoid multiplying the S subposteriors. Instead, subposteriors may be combined by their barycenter or median. The challenge

with these combinations is that their statistical meaning is unclear.

As we explain in Section 3, our proposed distributed algorithm does not suffer from any of the above limitations. Although the algorithm is asymptotically exact, the approximation error for finite N is quantified as a function of the number of shards S so that it is under the control of the practitioner.

Bardenet et al. (2015) also present an overview of subsampling-based algorithms for single layer models. Subsampling algorithms run on a single computer with all of the data in memory. Their computational benefit comes from a reduction in the number of individual data point likelihood evaluations that are necessary at each MCMC iteration. Subsampling methods may be applied to the estimation of Bayesian hierarchical models 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. Although there are computational gains to be made at each iteration of the θ draws if N is extremely large, say millions, we 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 $p\left(\beta_i \,|\theta\right)$ 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-standard posteriors. However, when applied in our context of drawing unit-level parameters in a hierarchical model with not large T, it is doubtful whether subsampling algorithms that are designed for very large T may be of value.[1]

[1]For target posteriors that may be well approximated by the Bernstein-von Mises approximation, Bardenet et al.'s (2015) improved confidence sampler is shown to exhibit excellent variance, mixing, and scalability properties. However, this algorithm 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). In our context, we are unlikely to have samples of size much larger than 50 for each of our many units. It is not clear how the Bardenet proposal could be adapted to the hierarchical setting as this would require subsampling units not observations within unit.

3 The Proposed Algorithm

The central idea of the proposed two-stage algorithm is to construct an estimator for the posterior predictive distribution of the unit-level parameters in the first stage, and to use the estimator as the prior distribution in the second stage. Both stages are embarrassingly parallel.

We proceed as follows. After defining the Bayesian hierarchical model, we guide the reader through the development of the proposed algorithm by presenting a sequence of three algorithms. Algorithm $\mathcal{A}_1$ is the standard Gibbs algorithm, a serial MCMC procedure. Algorithm $\mathcal{A}_2$ is equivalent to $\mathcal{A}_1$ in that they both share the same posterior for the unit-level parameters. It consists of two serial stages, an algorithm for constructing an unbiased estimator of the posterior predictive distribution in the first stage, and an algorithm for drawing the unit-level draws in the second stage. The advantage of $\mathcal{A}_2$ over $\mathcal{A}_1$ is that it is parallelizable. Algorithm $\mathcal{A}_3$ proposes an embarrassingly parallel implementation of $\mathcal{A}_2$ in the first stage to construct an asymptotically unbiased (approximate) estimator of the posterior predictive distribution, and in the second stage to draw the unit-level parameters.

3.1 The Model

For the reader's convenience, we reproduce the standard Bayesian hiearchical model here.

$$y_{it} \sim p\left(y_{it}\left|\beta_i\right.\right) \quad for\ i = 1, ..., N\ and\ t = 1, ..., T_i \tag{3.1}$$

$$\beta_i \sim p\left(\beta_i\left|\theta\right.\right) \quad for\ i = 1, ..., N \tag{3.2}$$

$$\theta \sim p(\theta\left|\tau\right.) \tag{3.3}$$

where $\{\beta_i\}$ are the unit-level parameters and θ represents the common parameters.

The joint posterior distribution of the model parameters $\{\beta_i\}$ and θ is

$$p\left(\{\beta_i\}, \theta\left|\tau, Y\right.\right) \propto p(\theta\left|\tau\right.) \prod_i \left[p\left(\beta_i\left|\theta\right.\right) \prod_t p\left(y_{it}\left|\beta_i\right.\right) \right] \tag{3.4}$$

where $Y = \cup_i \{y_{it}\}_{t=1}^{T}$ is the full data of observed outcomes and covariates (we omit the covariates

for notational convenience). The posterior full conditional densities are

$$\beta_i \,|\theta, y_i \quad \propto \quad p\left(\beta_i \,|\theta\right) \prod_t p\left(y_{it} \,|\beta_i\right) \quad for\ i = 1, ..., N \tag{3.5}$$

$$\theta \,|\{\beta_i\}, \tau \quad \propto \quad p(\theta \,|\tau) \prod_i p\left(\beta_i \,|\theta\right) \tag{3.6}$$

where $y_i = \{y_{it}\}_{t=1}^{T}$. We assume that $p(\beta_i \,|\theta)$ and $p\left(y_{it} \,|\beta_i\right)$ are not conjugate so that $p\left(\beta_i \,|\theta, y_i\right)$ is a nonstandard distribution. We discuss the simpler conjugate case in a separate section of the paper. We introduce the following notation. $p\left(\theta \,|Y\right)$ and $p\left(\beta_i \,|Y\right)$ are the posterior marginal distributions of the common and unit-level parameters given data Y, respectively.

3.2 Gibbs Algorithm $\mathcal{A}_1$

A standard (hybrid) Gibbs algorithm ($\mathcal{A}_1$) to sample from (3.5 - 3.6) iterates between draws of $\{\beta_i\}$ and θ: (i) $p\left(\beta_i \,|\theta, y_i\right)$ is nonstandard and thus necessitates a Metropolis-Hastings algorithm, and (ii) if $p(\theta \,|\tau)$ and $p\left(\beta_i \,|\theta\right)$ are conjugate, as is usually case, $p\left(\theta \,|\{\beta_i\}, \tau\right)$ is a standard distribution from which θ may be sampled directly. Of course, estimation of (3.4) may be implemented with other MCMC schemes. Our proposed algorithm is agnostic to the MCMC procedure used for estimating (3.4) and the parametric form of $p\left(\beta_i \,|\theta\right)$ (or whether it is a mixture).

$\mathcal{A}_1$ is not designed to fully take advantage of distributed processing capabilities for estimation purposes. For example, an obvious parallel estimation strategy for $\mathcal{A}_1$ is to estimate each β_i (or partition of β_is) given θ in parallel across multiple computing units. The limitation of this approach is that each parallel process needs an updated θ draw for each iteration of β_i draws (and conversely, each θ draw requires updated $\{\beta_i\}$ draws). The problem is that communicating the θ draws to each parallel process on each iteration is prohibitively expensive across multiple computers (Scott et. al, 2016).

3.3 Equivalent Algorithm $\mathcal{A}_2$

Since the objects of interest in this article are the unit-level parameters $\{\beta_i\}$, we may integrate out θ in (3.5) to draw from the posterior marginal density of β_i

$$\begin{aligned} p\left(\beta_i \,|Y\right) &\propto \int p\left(\beta_i \,|\theta\right) \prod_t p\left(y_{it} \,|\beta_i\right) p\left(\theta \,|Y\right) d\theta && (3.7) \\ &= \int p\left(\beta_i \,|\theta\right) p\left(\theta \,|Y\right) d\theta \prod_t p\left(y_{it} \,|\beta_i\right) && (3.8) \\ &= \mathbb{E}_\theta \left[p\left(\beta_i \,|\theta\right)\right] \prod_t p\left(y_{it} \,|\beta_i\right) && (3.9) \\ &= p\left(\beta_i \,|\{\beta\}\right) \prod_t p\left(y_{it} \,|\beta_i\right) && (3.10) \end{aligned}$$

where $p\left(\theta \,|Y\right)$ is the posterior marginal distribution of θ, and $\mathbb{E}_\theta \left[p\left(\beta_i \,|\theta\right)\right]$ is the posterior predictive density of β_i (Appendix: Theorems) which we denote as $p\left(\beta_i \,|\{\beta\}\right)$. The posterior predictive density of β_i is the density of β_i for a unit whose data y_i has not yet been observed, given $\{\beta_{j\neq i}\}$ (Gelman et al., 2014). The cost of drawing β_i from (3.10) is that we need $p\left(\beta_i \,|\{\beta\}\right)$, a mathematical object. Let $\dot{p}\left(\beta_i \,|\{\beta\}\right)$ denote the estimator of the posterior predictive density $p\left(\beta_i \,|\{\beta\}\right)$, which we construct as

$$\dot{p}\left(\beta_i \,|\{\beta\}\right) = \frac{1}{R} \sum_r p(\beta_i \,|\theta^r) \tag{3.11}$$

where θ^r is the rth draw of θ from $p\left(\theta \,|Y\right)$, and R is the total number of draws. $\dot{p}\left(\beta_i \,|\{\beta\}\right)$ is an unbiased estimator of $p\left(\beta_i \,|\{\beta\}\right)$ (Appendix: Theorems).

Since we cannot draw β_i from (3.10), we replace $p\left(\beta_i \,|\{\beta\}\right)$ in (3.10) with $\dot{p}\left(\beta_i \,|\{\beta\}\right)$ to draw from

$$\begin{aligned} \dot{p}\left(\beta_i \,|\{\theta^r\}, Y\right) &\propto \dot{p}\left(\beta_i \,|\{\beta\}\right) \prod_t p\left(y_{it} \,|\beta_i\right) && (3.12) \\ &= \frac{1}{R} \sum_r p(\beta_i \,|\theta^r) \prod_t p\left(y_{it} \,|\beta_i\right) && (3.13) \end{aligned}$$

where $\{\theta^r\}$ denotes the collection of R draws of θ from $p\left(\theta \,|Y\right)$.

$\dot{p}\left(\beta_i \left| \{\theta^r\}, Y \right.\right)$ is an unbiased estimator of $p\left(\beta_i \left| Y \right.\right)$, which follows from the unbiasedness of $\dot{p}\left(\beta_i \left| \{\beta\} \right.\right)$. As a consequence, Beaumont (2003) shows that the posterior marginal distribution $\dot{p}\left(\beta_i \left| Y \right.\right)$ induced by $\dot{p}\left(\beta_i \left| \{\theta^r\}, Y \right.\right)$ equals $p\left(\beta_i \left| Y \right.\right)$ (Appendix: Theorems). Andrieu and Roberts (2009) present theoretical properties of Beaumont's (2003) findings.

The construction of $\dot{p}\left(\beta_i \left| \{\beta\} \right.\right)$ requires draws from $p\left(\theta \left| Y \right.\right)$. This suggests a two-stage algorithm $\mathcal{A}_2$ that is equivalent to $\mathcal{A}_1$ for the $\{\beta_i\}$ draws. Stage one draws from $p\left(\theta \left| Y \right.\right)$ to construct $\dot{p}\left(\beta_i \left| \{\beta\} \right.\right)$, and stage two draws β_i from $\dot{p}\left(\beta_i \left| \{\theta^r\}, Y \right.\right)$.

For the first stage we run algorithm $\mathcal{A}_1$ (3.5 - 3.6), discard the $\{\beta_i\}$ draws, and keep the θ draws to construct $\dot{p}\left(\beta_i \left| \{\beta\} \right.\right)$ using (3.11). For the second stage, given $\dot{p}\left(\beta_i \left| \{\beta\} \right.\right)$, we draw β_i

$$\beta_i \left| y_i \right. \quad \propto \quad \dot{p}\left(\beta_i \left| \{\beta\} \right.\right) \prod_t p\left(y_{it} \left| \beta_i \right.\right) \quad for\ i = 1, ..., N \tag{3.14}$$

Since drawing from (3.14) only requires data y_i and $\dot{p}\left(\beta_i \left| \{\beta\} \right.\right)$, β_i draws may be implemented in an embarrassingly parallel manner. We propose an independence Metropolis-Hastings algorithm, although any MCMC algorithm will suffice. The particular advantage of an independence algorithm when T is small is made more explicit in the next section.

3.4 Distributed Algorithm $\mathcal{A}_3$

Algorithm $\mathcal{A}_3$ is an embarrassingly parallel implementation of $\mathcal{A}_2$. Stage one constructs an asymptotically unbiased estimator of $\dot{p}\left(\beta_i \left| \{\beta\} \right.\right)$. The cost of parallelization is asymptotic unbiasedness. Stage two implements the second stage of $\mathcal{A}_2$ in parallel. An outline of both stages is in Algorithm $\mathcal{A}_3$ (for non-standard posterior distributions).

3.4.1 Stage One

The first stage of the proposed algorithm ($\mathcal{A}_3$) constructs an estimator for the posterior predictive density of β_i in an embarrassingly parallel manner by running algorithm $\mathcal{A}_1$ (3.5 - 3.6) on mutually exclusive subsets of the data. The full data Y is partitioned into S shards such that all of the data for unit i are in the same shard: $Y_s = \cup_{i \in I_s} \{y_{it}\}_{t=1}^T$, $s = 1, ..., S$, I_s is a vector indicating the units

allocated to shard s. The joint posterior distribution of the model parameters for each shard is

$$p\left(\{\beta_i\}, \theta_s \,|\tau, Y_s\right) \propto p(\theta_s \,|\tau) \prod_i \left[p\left(\beta_i \,|\theta_s\right) \prod_t p\left(y_{it} \,|\beta_i\right) \right] \quad for\ i \in I_s \tag{3.15}$$

Although any standard MCMC algorithm may be used to estimate (3.15) we propose a hybrid Gibbs sampler (3.5-3.6). Algorithm $\mathcal{A}_1$ is run across machines, one shard per machine.

$$\beta_i \,|\theta_s, Y_s \quad \propto \quad p\left(\beta_i \,|\theta_s\right) \prod_t p\left(y_{it} \,|\beta_i\right) \quad for\ i \in I_s \tag{3.16}$$

$$\theta_s \left|\{\beta_i\}_{i \in I_s}, \tau \right. \quad \propto \quad p\left(\theta_s \,|\tau\right) \prod_{i \in I_s} p\left(\beta_i \,|\theta_s\right) \tag{3.17}$$

The $\{\beta_i\}_{i \in I_s}$ draws are discarded and the θ_s draws are used to construct an unbiased estimator for the subposterior predictive distribution of β_i for shard s, which we denote as $\dot{p}\left(\beta_i \,|\{\beta\}_s\right)$

$$\dot{p}\left(\beta_i \,|\{\beta\}_s\right) \quad = \quad \frac{1}{R} \sum_r p(\beta_i \,|\theta_s^r) \tag{3.18}$$

where θ_s^r is the rth draw of θ for shard s, and R is the total number of draws.

We define the estimator of $\dot{p}\left(\beta_i \,|\{\beta\}\right)$ as the mean of the estimators for the shard-level subposterior predictive densities.

$$\ddot{p}\left(\beta_i \,|\{\beta\}\right) \quad = \quad \frac{1}{S} \sum_s \dot{p}\left(\beta_i \,|\{\beta\}_s\right) \tag{3.19}$$

$$\quad = \quad \frac{1}{SR} \sum_s \sum_r p(\beta_i \,|\theta_s^r) \tag{3.20}$$

We denote the number of units per shard as $N_s = N/S$ and show (Appendix: Theorems) that $\ddot{p}\left(\beta_i \,|\{\beta\}\right)$ is an asymptotically unbiased estimator of $\dot{p}\left(\beta_i \,|\{\beta\}\right)$. For finite N, $\ddot{p}\left(\beta_i \,|\{\beta\}\right)$ has variance proportional to $S/N = N_s^{-1}$, whereas $\dot{p}\left(\beta_i \,|\{\beta\}_s\right)$ has variance proportional to $S^2/N = SN_s^{-1}$. As expected, averaging shard-level subposterior predictive densities reduces variance by a factor of S.

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. The proposed algorithm uses the $\{\theta_s^r\}$

draws to construct $\ddot{p}\left(\beta_i \left| \{\beta\} \right.\right)$, a mathematical procedure that does not require computation. The computational complexity of the proposed algorithm is therefore independent of the number of shards.

We may decrease communication overhead between stages one and two by drawing the proposal draws (from $\ddot{p}\left(\beta_i \left| \{\beta\} \right.\right)$) in stage one, for the independence Metropolis-Hastings algorithm in stage two. Since R draws from $\ddot{p}\left(\beta_i \left| \{\beta\} \right.\right)$ is probabilistically equivalent to $\frac{R}{S}$ draws from each of the S estimators $\dot{p}\left(\beta_i \left| \{\beta\}_s \right.\right)$, we can decrease communication costs between machines by returning $\frac{R}{S}$ draws of β_i from each worker machine. The combined R draws from $\ddot{p}\left(\beta_i \left| \{\beta\} \right.\right)$ represent the proposal draws for stage two of the algorithm. Drawing from $\dot{p}\left(\beta_i \left| \{\beta\}_s \right.\right) = \frac{1}{R}\sum_r p(\beta_i \left| \theta_s^r \right.)$ is straightforward: draw a multinomial distributed indicator vector with parameter vector $\left(\frac{1}{R}, ..., \frac{1}{R}\right)$ to determine the active component r, and then draw a d-variate vector from the rth component of $\dot{p}\left(\beta_i \left| \{\beta\}_s \right.\right)$ using parameters θ_s^r, where d is the dimension of the parameter vector β_i.

Compared to a naive implementation of extant methods (Scott et al., 2016; Neiswanger et al., 2014), the communication efficiency gains are two-fold: (i) first stage communication costs from each worker machine to the master machine are reduced from R draws of θ_s to $\frac{R}{S}$ draws of β_i, and (ii) second stage communication costs from the master machine to each worker machine are reduced from R draws of θ_s to R draws of β_i. For example, if β_i is a d-dimensional vector, and $p(\beta_i \left| \theta \right.)$ is the normal distribution, θ represents the mean and covariance parameters for a normal distribution which requires $d + \frac{d(d+1)}{2}$ parameters. Communication cost is reduced from $R\left(d + \frac{d(d+1)}{2}\right)$ to $\frac{R}{S}d$ in stage one, and from $R\left(d + \frac{d(d+1)}{2}\right)$ to Rd in stage two.

3.4.2 Maximum Number of Shards

Since $\ddot{p}\left(\beta_i \left| \{\beta\} \right.\right)$ is an asymptotically unbiased estimator of $\dot{p}\left(\beta_i \left| \{\beta\} \right.\right)$, it is approximate for finite N_s. Although the bias goes away as N_s becomes large, it is of practical importance to bound the approximation error for finite N_s . For the practitioner, the object of interest is the maximum number of shards for partitioning the data in stage one as a function of an error bound. As our error bound, we define the maximum expected squared error $\epsilon_{max}^2 = \sup_\beta \left[\mathbb{E}\left[\left|\ddot{p}\left(\beta_i \left| \{\beta\} \right.\right) - \dot{p}\left(\beta_i \left| \{\beta\} \right.\right)\right|^2\right]\right]$. We show (Appendix: Theorems) that the maximum number of shards for given maximum expected

squared error ϵ^2_{max}, N, data Y, and the number of MCMC iterations (after burn-in) R is

$$S_{max} = \left\lfloor \frac{C_0}{2}\left(NR\epsilon^2_{max} + \sqrt{(NR\epsilon^2_{max})^2 - 4C_0^{-2}}\right)\right\rfloor \tag{3.21}$$

$$\approx \left\lfloor C_0 NR\epsilon^2_{max}\right\rfloor \ for\ S^2 \gg 1 \tag{3.22}$$

where $C_0 = \left\{\sup_\beta \left[\nabla p\left(\beta\,|\theta\right)^T I_\theta^{-1} \nabla p\left(\beta\,|\theta\right)\right]\right\}^{-1}$ may be estimated empirically, and I_θ is the Fisher information matrix at θ. As the amount of information that data Y carries about the θ increases, as suggested by an increasing I_θ, the proposed algorithm is more robust to larger values of S.

3.4.3 Stage One Subsampling Algorithm $\mathcal{A}'_3$

Noting that $\ddot{p}\left(\beta_i\,|\{\beta\}\right)$ converges to $\dot{p}\left(\beta_i\,|\{\beta\}\right)$ (given a maximum expected squared error) for sufficiently large $N_s = N/S$, say N_s^*, there may be no practical benefit to using a value of N_s greater than N_s^* for constructing $\ddot{p}\left(\beta_i\,|\{\beta\}\right)$. If $N_s^* < N_s$, we may reduce computation and communication costs with a modification to the first stage by sampling Y with probability p prior to dividing the data into S shards, so that $N_s^* = pN_s = pN/S$. If $N_s^* << N_s$ we may expect substantial savings in first stage computation and execution time. Sampling in the second stage is not possible since our primary interest is in the β draws for all N cross-sectional units. The proposed algorithm with stage one subsampling (for non-standard posterior distributions) is presented in the Appendix (Algorithm $\mathcal{A}'_3$).

The maximum number of shards with subsampling rate p in the first stage (Appendix: Theorems) is

$$S_{max} = \left\lfloor \frac{C_0}{2}\left(NR\epsilon^2_{max}p^2 + \sqrt{(NR\epsilon^2_{max}p^2)^2 - 4p^2C_0^{-2}}\right)\right\rfloor \tag{3.23}$$

$$\approx \left\lfloor C_0 NR\epsilon^2_{max}p^2\right\rfloor \ for\ S^2 \gg p^2 \tag{3.24}$$

Equivalently, the optimal stage one subsampling rate is

$$p \approx min\left\{1, \sqrt{\frac{S}{C_0 NR\epsilon^2_{max}}}\right\} \tag{3.25}$$

3.4.4 Stage Two

For an embarrassingly parallel implementation of stage two of the proposed algorithm $\mathcal{A}_3$, the full data Y is partitioned into S shards such that all of the data for unit i are in the same shard: $Y_s = \cup_{i \in I_s} \{y_{it}\}_{t=1}^{T}$, $s = 1, ..., S$, I_s is a vector indicating the units allocated to shard s. Since $\ddot{p}(\beta_i | \{\beta\})$ converges to $\dot{p}(\beta_i | \{\beta\})$, we replace $\dot{p}(\beta_i | \{\beta\})$ in (3.14) with $\ddot{p}(\beta_i | \{\beta\})$.

$$\beta_i | y_i \quad \sim \quad \ddot{p}(\beta_i | \{\beta\}) \prod_t p(y_{it} | \beta_i) \;\; for \; i \in I_s \tag{3.26}$$

We propose an independence Metropolis-Hastings algorithm in the second stage in which $\ddot{p}(\beta_i | \{\beta\})$ is used as the proposal distribution. More precisely, the proposal and target densities are $\ddot{p}(\beta_i | \{\beta\})$ and $\ddot{p}(\beta_i | \{\beta\}) \prod_t p(y_{it} | \beta_i)$, respectively. Given draw β_i^{r-1}, the acceptance probability for draw β_i^r is

$$\alpha \quad = \quad min \left\{ 1, \frac{\ddot{p}(\beta_i^r | \{\beta\}) \prod_t p(y_{it} | \beta_i^r)}{\ddot{p}(\beta_i^{r-1} | \{\beta\}) \prod_t p(y_{it} | \beta_i^{r-1})} \frac{\ddot{p}(\beta_i^{r-1} | \{\beta\})}{\ddot{p}(\beta_i^r | \{\beta\})} \right\} \tag{3.27}$$

$$= \quad min \left\{ 1, \frac{\prod_t p(y_{it} | \beta_i^r)}{\prod_t p(y_{it} | \beta_i^{r-1})} \right\} \tag{3.28}$$

a very fast computation.

For not large T, an independence Metropolis-Hastings algorithm is especially attractive: $\ddot{p}(\beta_i | \{\beta\})$, as the prior, has a large influence on the unit-specific posterior densities since the unit likelihood $\prod_t p(y_{it} | \beta_i)$ is relatively flat, and therefore obviates the need for a more computationally intensive random walk algorithm. Under mild conditions, it has the additional advantage of uniform convergence rather than a geometric or worse rate of convergence for a random walk algorithm (Robert and Casella, 2010). However, for large T, each unit's likelihood may be more sharply defined and influenced less by the prior. Proposal draws from $\ddot{p}(\beta_i | \{\beta\})$ are therefore expected to have lower acceptance rates with increasing T.

3.4.5 $p(\beta_i \,|\theta\,)$ and $p\,(y_{it}\,|\beta_i\,)$ are Conjugate

As presented, Algorithm $\mathcal{A}_3$ assumes that $p(\beta_i \,|\theta\,)$ and $p\,(y_{it}\,|\beta_i\,)$ are not conjugate. If they are conjugate, the β_i draws in stage two may be sampled directly from a mixture whose components are a standard distribution.

$$\begin{aligned} \beta_i \,|y_i \quad &\sim \quad \ddot{p}\,(\beta_i \,|\{\beta\}\,) \prod_t p\,(y_{it}\,|\beta_i\,) \;\; for\; i \in I_s \qquad (3.29) \\ &\sim \quad \frac{1}{SR} \sum_s \sum_r p(\beta_i \,|\theta_s^r\,) \prod_t p\,(y_{it}\,|\beta_i\,) \;\; for\; i \in I_s \qquad (3.30) \end{aligned}$$

Drawing from (3.30) is straightforward: draw a multinomial distributed indicator vector with parameter vector $\left(\frac{1}{SR}, ..., \frac{1}{SR}\right)$ to determine the active component $\{s, r\}$, and then draw β_i from the standard distribution $p(\beta_i \,|\theta_s^r\,) \prod_t p\,(y_{it}\,|\beta_i\,)$. Performance of this second stage implementation is independent of the number of observations T per unit.

3.4.6 Common Parameters

The proposed algorithm ($\mathcal{A}_3$) simulates draws from the unit-level posterior densities for all N cross-sectional units. If the researcher is interested in the common parameters, they may be easily simulated using the unit-level draws. Given the rth draw of the unit-level parameters $\{\beta_i^r\}$, the rth draw of the common parameters θ^r may be sampled from

$$\theta^r \,|\{\beta_i^r\}\,, \tau \;\propto p(\theta^r \,|\tau\,) \prod_i p\,(\beta_i^r \,|\theta^r\,) \qquad (3.31)$$

where $p(\theta \,|\tau\,)$ is the prior density for θ. If $p\,(\beta_i \,|\theta\,)$ and $p(\theta \,|\tau\,)$ are conjugate, as is commonly the case, the θ draws may be sampled directly from a standard distribution.

4 Simulation

We 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 an alternative-specific covariate (e.g. price). Each unit's response parameter β_i is drawn from a normal distribution with mean $\mu' = (1, 2, 3, -2)$ and covariance Σ equal to the identity matrix.

We implement the simulation on a 32-core Linux computer with 256GB of memory, each core representing a single machine. As our benchmark model, we draw from (3.5 - 3.6) with a hybrid Gibbs algorithm ($\mathcal{A}_1$), implemented by the *bayesm* routine `rhierMnlMixture` (Rossi, 2015). *bayesm* is mostly written in R except for the computationally intensive loops which are written in C++. For the proposed algorithm ($\mathcal{A}_3$), the first stage is an embarrassingly parallel hybrid Gibbs algorithm (we appropriately modify `rhierMnlMixture`) to draw from (3.16 - 3.17) and to construct $\ddot{p}\left(\beta_i \left| \{\beta\} \right.\right)$ (3.20). The second stage is an embarrassingly parallel independence Metropolis-Hastings algorithm to draw from (3.26). The second stage is implemented in R and C++ (for the computationally intensive loops). Parallelism is implemented using the *parallel* (built-in to R) routine `mclapply`. Due to the memory limitations of our computer, both algorithms return the posterior draws for a random sample of 1,000 cross-sectional units.

4.1 Convergence

We first test our theoretical finding that $\ddot{p}\left(\beta \left| \{\beta\} \right.\right)$ converges to and is an asymptotically unbiased estimator of $\dot{p}\left(\beta \left| \{\beta\} \right.\right)$ for sufficiently large N_s. Figure 1 compares plots of the marginals of $\dot{p}\left(\beta \left| \{\beta\} \right.\right)$ for several components of the β vector, $\dot{p}\left(\beta_k \left| \{\beta\} \right.\right)$, marginals of the S shard-specific estimators $\dot{p}\left(\beta_k \left| \{\beta\}_s \right.\right)$, and the marginals of $\ddot{p}\left(\beta \left| \{\beta\} \right.\right)$. We do this for two components of β: one of the three intercepts β_3, and the coefficient of the alternative-specific covariate β_4. For our simulated data and model, we find that $\ddot{p}\left(\beta_k \left| \{\beta\} \right.\right) \approx \dot{p}\left(\beta_k \left| \{\beta\} \right.\right)$ for $N_s \gtrsim 3,333$. That is, for all practical purposes convergence of our estimator to that of the single machine hybrid Gibbs algorithm is achieved at $N_s^* \simeq 3,333$. We also note that $\ddot{p}\left(\beta_k \left| \{\beta\} \right.\right)$ converges to $\dot{p}\left(\beta_k \left| \{\beta\} \right.\right)$ at a faster rate than $\dot{p}\left(\beta_k \left| \{\beta\}_s \right.\right)$ since most of the S shard marginal densities, $\dot{p}\left(\beta_k \left| \{\beta\}_s \right.\right)$, have not yet converged for $N_s = 3,333$, and many have not converged for $N_s = 33,333$. This finding is consistent with the theoretically larger variance of $\dot{p}\left(\beta \left| \{\beta\}_s \right.\right)$ by a factor of S. Plots of $\frac{\ddot{p}(\beta_k|\{\beta\})}{\dot{p}(\beta_k|\{\beta\})}$ are consistent with

these findings (Appendix figure 1).

We next evaluate the convergence of the unit-level posterior densities from the proposed algorithm ($\mathcal{A}_3$) to those of the single machine hybrid Gibbs algorithm ($\mathcal{A}_1$). The Q-Q plots in figure 2 compare quantiles of the β draws from the single machine hybrid Gibbs algorithm with those of the proposed algorithm for a random sample of five cross-sectional units, for $T = 5$, 15, and 45 observations per unit, when $N_s = 3,333$ units per shard ($N = 100,000$). For convenience in interpretation we plot the 45-degree line. Qualitatively, we find excellent convergence of results for $T = 5$, and some slight degradation for $T = 15$ and $T = 45$ (for example, see unit 1). The reason for this slight degradation in convergence is due to the independence Metropolis-Hastings algorithm in the second stage. As T increases, the unit-level posterior densities become narrower as the unit likelihood is more sharply defined and influenced less by the prior, which in turn reduces the acceptance rate of the independent proposal draws. With fewer accepted draws, the effective sample size decreases and each unit-level empirical posterior density (for the construction of the Q-Q plot) exhibits greater bias and therefore poorer convergence to the single machine hybrid Gibbs algorithm. The effect of a low acceptance rate (as T increases) may be overcome by increasing the number of MCMC iterations in stage two, or with a random walk Metropolis-Hastings algorithm instead of an independence algorithm.

Since correlation is a measure of the linear relationship between two variables, it is suitable for quantifying the linear relationship between the quantiles of the single machine hybrid Gibbs draws ($\mathcal{A}_1$) and those of the proposed algorithm ($\mathcal{A}_3$). A correlation that is very close to one indicates an exact linear relationship (i.e., a straight line) and excellent convergence of the algorithms. Table 1 presents the first, fifth and fiftieth (median) correlation percentiles of unit-specific draw quantiles for a random sample of 1,000 cross-sectional units. For $T = 5$, the proposed algorithm exhibits superior convergence to the single machine hybrid Gibbs algorithm: the median correlation of 0.999 suggests an exact linear relationship, and the first percentile correlation of 0.99 suggest an almost exact linear relationship. As T increases to 15, the proposed algorithm exhibits moderate convergence since the median correlation remains at 0.999, the fifth percentile decreases to 0.99, but the first percentile decreases to about 0.98. At $T = 45$, the proposed algorithm exhibits inferior convergence with a

median correlation of about 0.997 and a first percentile correlation of about 0.91. These results are consistent with the qualitative findings from the Q-Q plots.

Figure 3 and table 2 present convergence results for $N_s = 33,333$ ($N = 1,000,000$). We note the very close agreement between the correlations for $N_s = 3,333$ (table 1) and $N_s = 33,333$ (table 2). The reason why correlations do not increase further for N_s greater than $3,333$ is because $N_s^* \simeq 3,333$: once convergence to the single machine hybrid Gibbs posterior predictive distribution is achieved at $N_s = N_s^*$, any further increase in N_s decreases the squared error between $\dot{p}(\beta \,|\{\beta\})$ and $\ddot{p}(\beta \,|\{\beta\})$ negligibly. Although several units (e.g., units 3 and 5 for $T = 45$) in figure 3 seem to converge rather poorly as compared to other units in figure 1, we surmise that this may be due to the small sample of units (five) chosen for the figures. In particular, for $T = 45$ the number of accepted proposal draws for units in the tails of the posterior predictive density may be small due to the small number of proposal draws in the tail and a narrow likelihood function, resulting in many consecutive identical draws (as seen for units 3 and 5 in figure 3).

4.2 Performance

To quantify the performance of the proposed algorithm ($\mathcal{A}_3$) relative to the single machine hybrid Gibbs algorithm ($\mathcal{A}_1$), table 3 presents 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 simulation draws is the size of an i.i.d. sample with the same variance (or information) 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.

For small T ($T = 5$), the proposed algorithm dominates the single machine hybrid Gibbs algorithm in two respects: (i) distributed processing decreases execution time by an order of magnitude, and (ii) draws from the independence Metropolis-Hastings algorithm are less correlated resulting in a ESS that is about three times larger. The resulting increase in efficiency (ESS per minute) over the single machine hybrid Gibbs algorithm is by a factor of thirty-seven. For large N ($N = 1,000,000$), the proposed algorithm takes two hours and the single machine hybrid Gibbs algorithm runs in 26

hours.

For moderate T ($T = 15$), execution time of the proposed algorithm is still about an order of magnitude faster but ESS is just slightly better than the single machine hybrid Gibbs algorithm. The degradation in ESS (relative to that for small T) is due to the narrowing of the likelihood function, which in turn decreases the acceptance rate and ESS. The resulting efficiency gain is thus about an order of magnitude greater, driven primarily by the distributed processing of the proposed algorithm. For large N, the proposed algorithm takes three hours and the single machine hybrid Gibbs algorithm needs 36 hours.

For large T ($T = 45$), although the proposed algorithm is an order of magnitude faster than the single machine hybrid Gibbs algorithm, ESS is about forty percent of that for the single machine hybrid Gibbs algorithm, due to a further decrease in the acceptance rate (table 4). Efficiency is about 3-4 times higher than the single machine hybrid Gibbs algorithm. For large N, the proposed algorithm takes 6.5 hours and the single machine hybrid Gibbs algorithm runs in 2.5 days.

Table 4 quantifies the effect of increasing T (and the narrowing of unit-level posterior densities) on acceptance rates. The single machine hybrid Gibbs algorithm's adoption of a random walk for the Metropolis-Hastings draws results in slightly increasing acceptance rates with T, whereas the use of an independence Metropolis-Hastings algorithm in the second stage of the proposed algorithm results in significantly lower acceptance rates with T. This finding is due to the adaptability of the random walk algorithm to the shape of each unit's posterior whereas draws from the independence algorithm are from a common fixed proposal distribution that is independent of any unit's posterior density. With increasing T, we expect that a random walk algorithm instead of an independence algorithm for stage two of the proposed algorithm will result in an increase in acceptance rates and convergence. Alternatively, a simpler but less computationally efficient modification may be to increase the number of MCMC iterations in the second stage so that the number of accepted draws is sufficiently large for convergence and inference purposes.

4.3 Scalability with S

The scalability of the proposed algorithm ($\mathcal{A}_3$) with the number of shards S is established by the relationship (3.21) between the maximum number of shards and the maximum expected squared error between $\ddot{p}\left(\beta\,|\{\beta\}\right)$ and $\dot{p}\left(\beta\,|\{\beta\}\right)$. For $S^2 \gg 1$, $S_{max} \approx \left\lfloor C_0 N R \epsilon^2_{max} \right\rfloor$. The constant C_0 characterizes the proposed algorithm's scalability and is dependent on the data Y and model. We estimate C_0 for the simulation dataset (Appendix table 5) by running the hybrid Gibbs sampler and stage one of the proposed algorithm with $N = 10,000$, $S = 3$ shards, and $R = 16,000$ (after burn-in), and approximate the resulting maximum squared error between $\dot{p}\left(\beta\,|\{\beta\}\right)$ and $\ddot{p}\left(\beta\,|\{\beta\}\right)$: $C_0 = 2.278 \times 10^{-4}$. It takes about twenty minutes to run the simulations. To test the robustness of our estimate, we use (3.21) to solve for the maximum expected squared error ϵ^2_{max} when $N = 1,000,000$, $S = S_{max} = 30$, and $R = 16,000$ (after burn-in), and compare it to the actual maximum squared error. We impose the constraint that $S = S_{max} = 30$ because of a limitation of our computing system. We find $\epsilon^2_{max} = 8.24 \times 10^{-6}$ and $\epsilon^2_{actual} = 1.04 \times 10^{-5}$, a difference of about 20%.

4.4 Scalability with N

To evaluate the scalability of the proposed algorithm ($\mathcal{A}_3$) for N up to 100 million units, we run $\mathcal{A}_3$ on Comet, a large-scale cluster computing system with a parallel distributed file system at the University of California San Diego Supercomputer Center (table 6). Comet's compute nodes run Linux and are equipped with 24 cores and 128GB of memory. To avoid the inefficiencies inherent in the *parallel* (built-in to R) routine `mclapply`, we implement parallelization with Linux shell scripts, Perl scripts, and C++ code, and optimize the I/O intensive parts of the R code. We keep track of the total time to run each stage including communication overhead. For each run, $T = 5$ observations, $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in, and we keep every 10^{th} draw for each of the N units. We set the number of shards S so that $N_s = 33,333$ or larger.

For $N = 1$ million units and $T = 5$, the proposed algorithm runs about twice as fast on Comet compared to the 32-core Linux computer: 61 minutes on Comet, compared to 117 (table 3) on the

32-core Linux computer. Communication overhead is negligible. This doubling in performance on Comet is due to the decrease in communication costs attributed to Comet's parallel distributed file system and the removal of the inefficiencies related to the `mclapply` routine and I/O intensive R code on the 32-core computer. 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, we need 3,000 shards to maintain $N_s = 33,333$. Due to a limitation on Comet we are constrained to only 1,728 shards, causing execution times to increase to 162 minutes, still a relatively modest amount of time. Communication overhead increases to 21 minutes due to the even larger number of file transfers. We conclude that the proposed algorithm scales very well to $N = 100$ million units.

4.5 Subsampling in Stage One

Since $\ddot{p}\left(\beta\,|\{\beta\}\right)$ converges to $\dot{p}\left(\beta\,|\{\beta\}\right)$ at $N_s \simeq 3,333$ (see figure 1), we evaluate the impact of subsampling Y at rate $p = 10\%$ prior to dividing the data into S shards for the first stage (algorithm $\mathcal{A}_3'$). That is, for $N = 1$ million units and $S = 30$ shards, $N_s = pN/S = 3,333$ rather than $N_s = N/S = 33,333$. We find no difference in convergence between $N_s = 3,333$ and $N_s = 33,333$ in the first stage (see figure 2 and table 1 in the Appendix, as compared to figure 3 and table 2).

Subsampling in the first stage improves the performance of the proposed algorithm. For small T ($T = 5$), execution time is reduced from 117 minutes to 27 minutes, resulting in a further increase in efficiency to over two orders of magnitude greater than the single machine hybrid Gibbs algorithm (Appendix table 2). For moderate T ($T = 15$), a further reduction in execution time to 1 hour (from 3 hours) boosts efficiency to about thirty-five times greater than the single machine hybrid Gibbs algorithm. For large T ($T = 45$), efficiency increases to about seven times higher than the single machine hybrid Gibbs algorithm due to a decrease in execution time from 6.5 hours to 3 hours. Stage two acceptance rates are similar with and without subsampling (table 4 and Appendix table

3).

The scalability of the proposed algorithm for N up to 100 million units also increases with stage one subsampling (Appendix table 4). For $N = 1$ million units and $T = 5$, subsampling reduces execution time from 61 minutes to 13 (communication overhead remains negligible). For $N = 10$ million, execution time decreases from 76 minutes to 19 with subsampling (communication overhead is relatively constant at 3 minutes). For $N = 100$ million, execution time decreases from 162 minutes to 78 (communication overhead is relatively constant at 19 minutes).

5 Application: Donor Response

We further illustrate the proposed algorithm ($\mathcal{A}_3$) by modeling donor response to solicitation using data from a nonprofit charitable organization. In charitable fund-raising, the question of who and how often should be solicited is critical to the success of the organization. Any data-based solicitation strategy will require a model of donor response with individual level parameters. As the number of actual and/or potential donors can be very large, the need for more efficient methods is clear in this context.

The data from a leading US nonprofit organization (Malthouse, 2009) contains the donation and solicitation histories for 1,097,671 donors, 3,020,479 donations, and 28,417,598 solicitations for donations over a fifteen-year period (1992-2006). Data for each donor is collected beginning from the date of her first donation, resulting in an approximately linear increase in the total number of donors over time. After removing histories with incomplete data, we have 1,088,269 cross-sectional units. Donors are heterogenous with each making 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 number of donations per donor after their initial donation are 0, 0, and 2, respectively. That is, 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 we use an intercept, the number of days since her last donation (recency), and the number of past donations (frequency). We do not find that the average of past donation amounts influences donor response. We log transform the data and estimate a hierarchical binomial logit model (with a normal prior) of solicitation response with our proposed algorithm and the single machine hybrid Gibbs algorithm.

For the results of the first stage, figure 4 compares plots of $\dot{p}\left(\beta_k \left| \{\beta\} \right.\right)$ constructed by the single machine hybrid Gibbs algorithm ($\mathcal{A}_1$), the S shard-specific marginals $\dot{p}\left(\beta_k \left| \{\beta\}_s \right.\right)$ from the proposed algorithm ($\mathcal{A}_3$), and the marginals $\ddot{p}\left(\beta_k \left| \{\beta\} \right.\right)$ of the proposed algorithm for $S = 30$ shards ($N_s \approx 36,000$). We find excellent convergence of the marginal posterior predictive densities. Although plots of $\frac{\ddot{p}(\beta_k|\{\beta\})}{\dot{p}\left(\beta_k \left| \{\beta\}_s \right.\right)}$ (Appendix figure 3) show that β_1 (intercept) and β_3 (frequency) do not converge as well as β_2 (recency), especially in the tails, second stage convergence is excellent. 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. Figure 5 shows the second stage convergence of the unit-level posterior densities from the proposed algorithm to those of the single machine hybrid Gibbs algorithm for a random sample of twenty-four units. Table 7 presents the first, fifth and fiftieth (median) correlation percentiles of unit-specific draw quantiles for a random sample of 1,000 cross-sectional units. We find excellent qualitative and quantitative unit-level posterior convergence.

The single machine hybrid Gibbs algorithm ($\mathcal{A}_1$) requires about 3 days and 5 hours to run compared to about 6 hours for the proposed algorithm ($\mathcal{A}_3$). Acceptance rates are 23% and 54% for the single machine hybrid Gibbs and the proposed algorithms, respectively. Effective sample sizes (ESS) are 3,178 and 15,389, respectively (out of 40,000 MCMC iterations including 5,000 for burn-in), and ESS per minute are 0.69 and 40.15, respectively. The proposed algorithm is more efficient than the single machine hybrid Gibbs algorithm by about a factor of 58 for this application, significantly better than the 13-fold efficiency gain with simulated data for $T = 15$ (table 3). The supposed reason for this apparent paradox may be due to the simulated dataset's high variability compared to the more realistic variability of the donation dataset. The donation dataset's smaller

Fisher information about β_i (due to its lower variability) induces a flatter likelihood function for β_i, even though there are a median of 21 observations per unit for the donation dataset and only 15 for the 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 about β_i may not be representative of real datasets, which suggests that the efficiency gains of the proposed algorithm are expected to be markedly higher than those cited using the simulation dataset in this article.

C_0 is estimated for the donation dataset (Appendix table 5) with $N = 10,000$ and $S = 3$ shards: $C_0 = 7.980 \times 10^{-7}$ (it takes about 1 hour to run the simulations). For the complete dataset ($N = 1,088,269$) and $S = S_{max} = 30$, the maximum expected squared error and the actual maximum squared errors are $\epsilon^2_{max} = 9.88 \times 10^{-4}$ and $\epsilon^2_{actual} = 8.97 \times 10^{-4}$, a difference of about ten percent. When we compare estimates of C_0 for the simulation ($C_0 = 2.278 \times 10^{-4}$) and donation ($C_0 = 7.980 \times 10^{-7}$) datasets, inverse measures of their respective Fisher information about θ, we find that the simulation data carries about three orders of magnitude more information about the common parameters θ than does the donation data. The simulation dataset is thus more scalable with S than the donation dataset (3.21).

We conclude that low Fisher information (about θ and β_i) datasets are more efficient in terms of ESS per unit of computing, but less scalable in terms of S. We therefore expect that for low Fisher information datasets, scalability is more likely to be limited by S_{max}. For high Fisher information datasets, scalability is more likely to be limited by the number of available computers. In the latter case, execution time may be improved by subsampling in the first stage (algorithm $\mathcal{A}'_3$) to accommodate the limited number of available computers, while keeping within a given error tolerance (3.24).

5.1 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 subsampling[2]

[2]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

proposal. Some might suggest that if we sample the units, we can obtain satisfactory estimates of the common parameters (that is, the random coefficient distribution of individual parameters) at much lower computational cost. These subsample based estimates might be used to estimate a population-wide response to a donor solicitation or a market share in the case of multinomial logits applied to product demand. Here we explain that some caution must be exercised in the subsampling approach in the sense that the resulting inferences from the subsample of units can be very different from the inference based on the entire universe of units. Even large subsamples can provide misleading inferences.

We consider a subsample with a moderate value of N ($N = 10,000$) which can easily run on a single processor MCMC implementation. We compare inferences about the common parameters for this subsample with the full sample of big N ($N \approx 1,100,000$). In our hierarchical binary logit model of donor response, we assume that the population of units has a multivariate normal distribution of individual logit coefficients, $\beta_i \sim N(\mu, \Sigma)$. In the notation of the paper, the common parameter θ consists of a vector μ and the unique elements of Σ. We plot marginal posterior densities of the common parameter draws for the normal prior specified in our model for moderate ($N = 10,000$) and big N ($N = 1,088,269$), a difference by factor of about 100. The marginals of each element of μ are in figure 6, those for the diagonal elements of Σ in figure 7, and those for the off-diagonal elements of Σ in figure 8. Tables 8 and 9 provide corresponding summary and test statistics. We note that the standard deviations for all elements of $\{\mu, \Sigma\}$ decrease by a factor of about 10, as expected when increasing the sample size by a factor of 100. More interestingly, there is a clear separation of the posterior means for μ and Σ between moderate and big N (multivariate tests for equality of means are significant). 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, we consider a prediction task. Using the same donation dataset we estimate a hierarchical binomial logit model of solicitation response to predict the probability of response for 1,000 donors

posterior predictive density of β.

in a holdout period. We find that a model with an intercept term and three covariates performs well for prediction: time since last donation, number of donations in the current year, and total donation amount in the current year. We estimate the model with an increasing number of donors, from 1,000 to 1 million donors (data in the holdout period is not used for estimation). Figure 9 plots the mean prediction error and prediction accuracy as a function of the number of donors for model estimation. 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. 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.

6 Conclusions

We 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 ~~both~~ the unit-level ~~and common~~ parameters. As compared to other distributed approaches, 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 leaner 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 results in an overall efficiency gain of several orders of magnitude over the single machine hybrid Gibbs algorithm. To boost performance further, a modification to the proposed algorithm introduces subsampling 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 requires 2 hours,

and the proposed algorithm with stage one subsampling completes processing in 30 minutes. For larger T, the algorithm still dominates the standard 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. In real applications efficiency gains are expected to be markedly higher.

The proposed algorithm may be implemented on a multicore computer or a cluster of computers. We have demonstrated its scalability with simulated and real panels of one million units on a multicore computer, a pedestrian environment. Scalability is even greater on a large-scale cluster computing system. In our example with 100,000,000 units, only a few hours of computation time is required to simulate unit-level parameter posteriors.

Algorithm $\mathcal{A}_3$ Stage one of the proposed algorithm (for non-standard posteriors)

Stage One: Draw from $\beta \sim \ddot{p}\left(\beta \left| \{\beta\} \right.\right)$

Input: $Y = \cup_i \{y_{it}\}_{t=1}^T$

Output: $\{\beta^r\}_{r=1}^R$

1. Divide Y into S independent shards
   (a) $I_s = \{i : z_i = s\}$, $s = 1, ..., S$, where $p\left(z_i = s\right) = \frac{1}{S}$, and $|I_s| = \frac{N}{S}$
   (b) $Y_s = \cup_{i \in I_s} \{y_{it}\}_{t=1}^T$, $s = 1, ..., S$
2. Run S parallel MCMC simulations ($s = 1, ..., S$)
   (a) Set θ_s^0, β_i^0 $for\ all\ i \in I_s$
   (b) **for** $r = 1$ to R
      i. Draw $\beta_i^r \left| \theta_s^{r-1}, Y_s \right. \propto p\left(\beta_i^r \left| \theta_s^{r-1} \right.\right) \prod_t p\left(y_{it} \left| \beta_i^r \right.\right)$ $\quad for\ all\ i \in I_s$ (Metropolis-Hastings draw)
      ii. Draw $\theta_s^r \left| \{\beta_i^r\}_{i \in I_s} \right. \sim p(\theta_s^r \left| \tau \right.) \prod_{i \in I_s} p\left(\beta_i^r \left| \theta_s^r \right.\right)$ (conjugate prior)
   (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 $\beta_s^r \left| \theta_s^{z^r} \right. \sim p(\beta \left| \theta_s^{z^r} \right.)$
3. Collect the β draws and shuffle
   (a) $\beta = \cup_{s=1}^S \{\beta_s^r\}_{r=1}^{r=R/S}$
   (b) $\beta = permute\left(\beta\right)$

Algorithm $\mathcal{A}_3$ (cont'd) Stage two of the proposed algorithm (for non-standard posteriors)

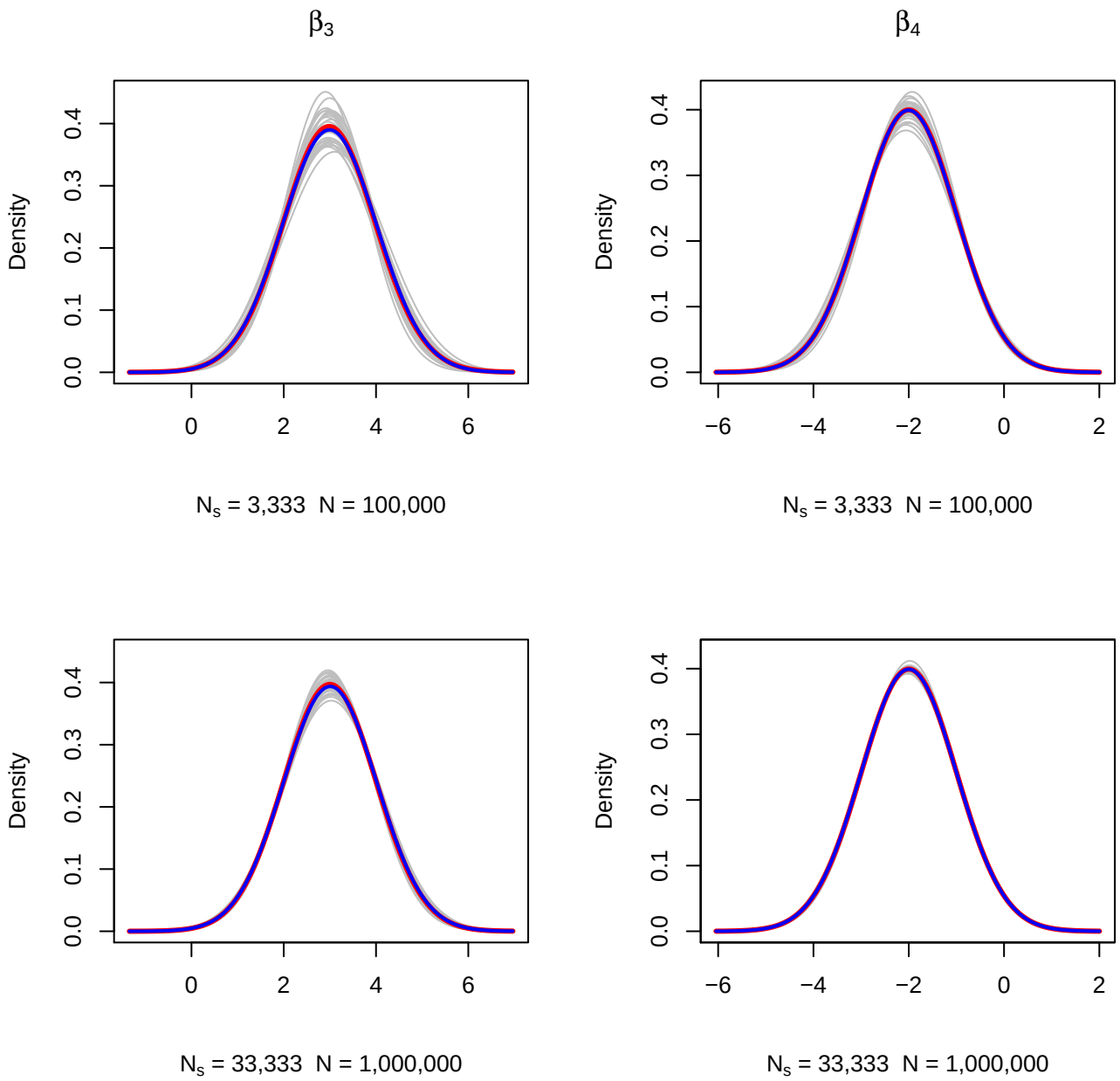
Stage Two: Draw β_i for $i = 1, ..., N$

Input: Y, $\{\beta^r\}_{r=1}^R$

Output: $\left\{\{\beta_i^r\}_{r=1}^R\right\}_{i=1}^N$

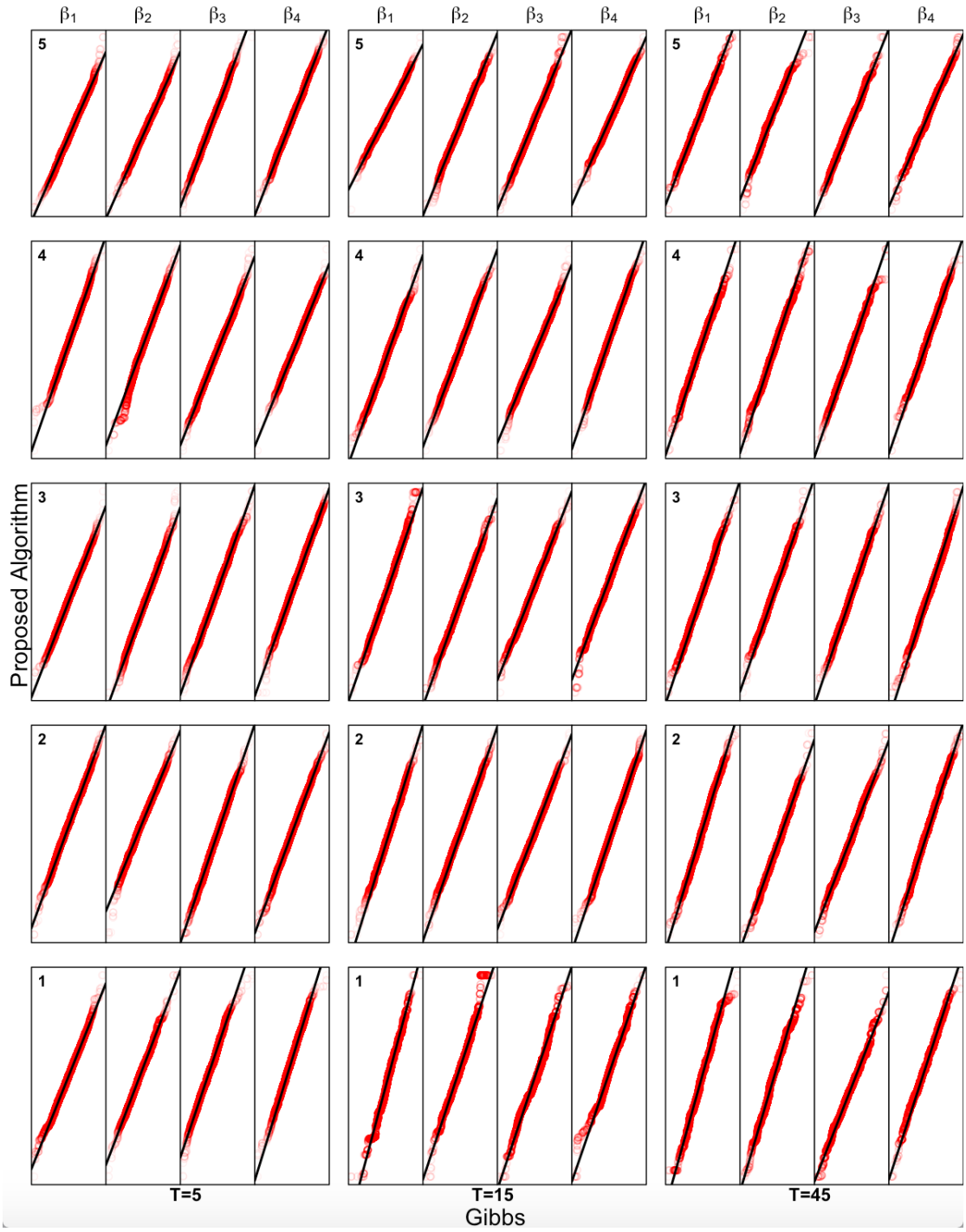
1. Divide Y into S independent shards
   (a) $I_s = \{i : z_i = s\}$, $s = 1, ..., S$, where $p(z_i = s) = \frac{1}{S}$, and $|I_s| = \frac{N}{S}$
   (b) $Y_s = \cup_{i \in I_s} \{y_{it}\}_{t=1}^T$, $s = 1, ..., S$
2. Run S parallel independence Metropolis-Hastings simulations ($s = 1, ..., S$).
   (a)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

Figure 1: Convergence of marginals of posterior predictive density estimators



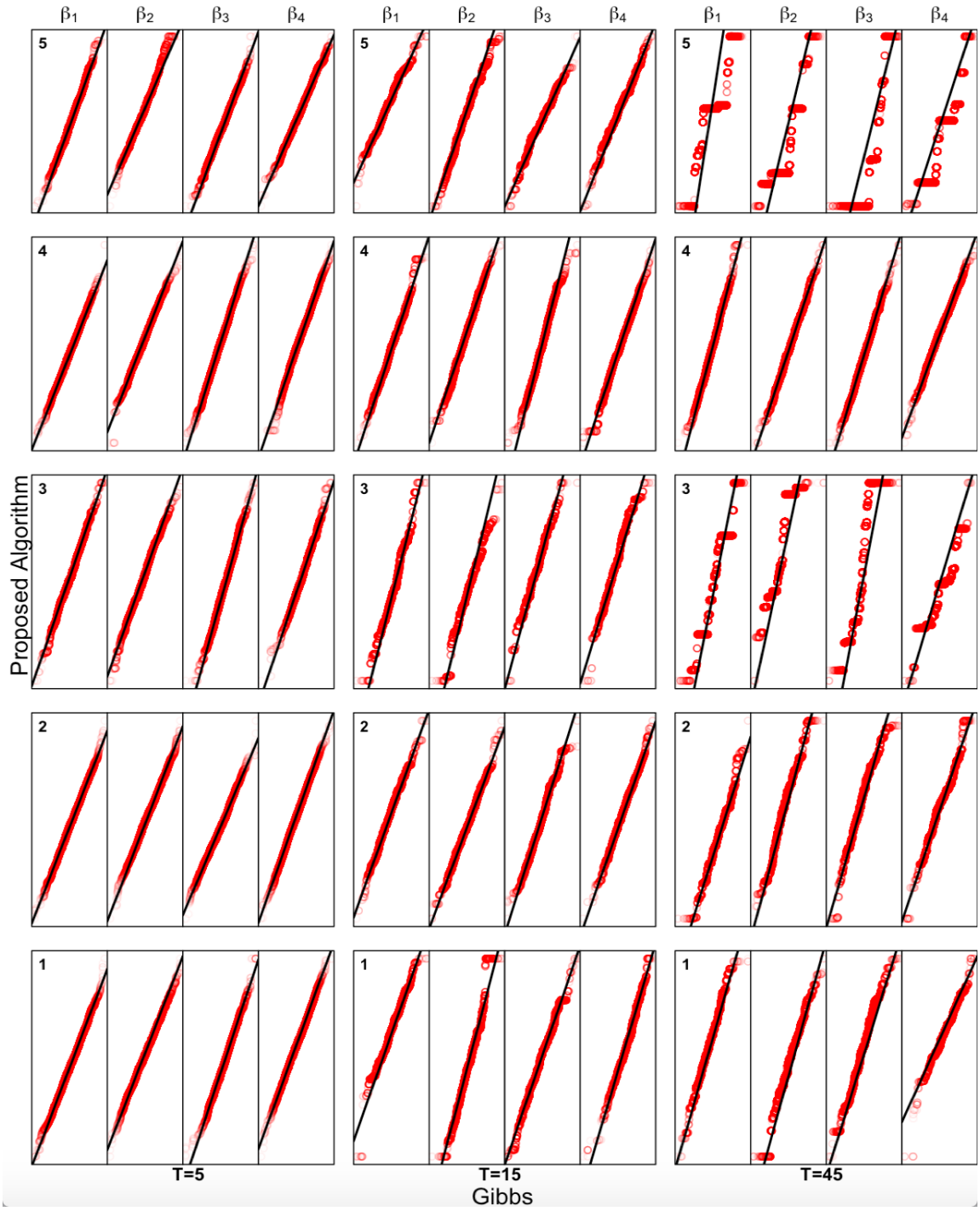
$T = 5$ observations per unit, $S = 30$ shards, and $R = 10,000$ MCMC iterations including 2,000 iterations for burn-in ($\dot{p}(\beta|\{\beta\}_s)$ in grey, $\ddot{p}(\beta|\{\beta\})$ in red, $\dot{p}(\beta|\{\beta\})$ in blue)

Figure 2: Convergence of unit-level posteriors: Q-Q plots ($N_s = 3,333$)



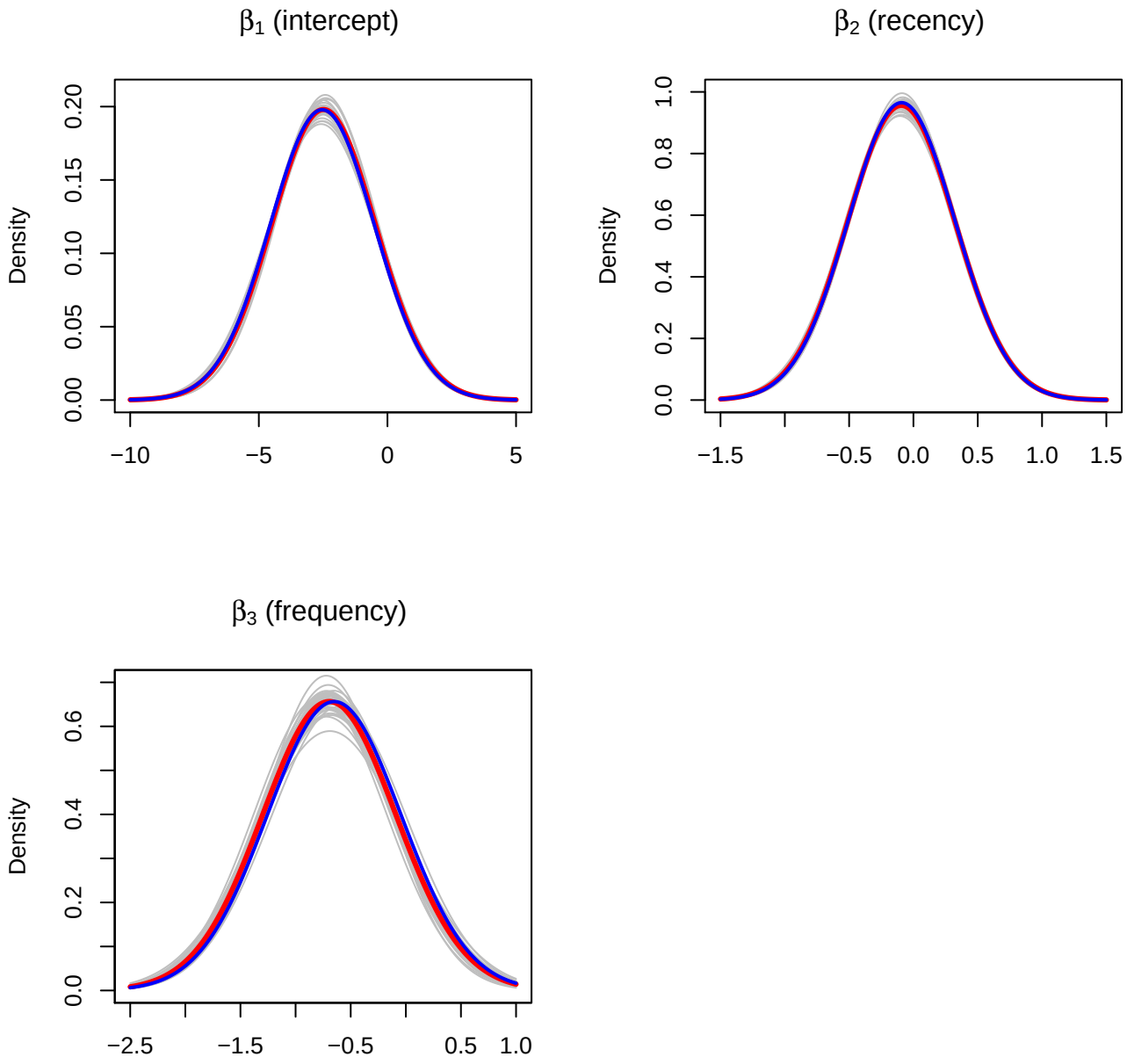
Q-Q plots of unit-specific draws from the proposed algorithm and the single machine hybrid Gibbs algorithm, for a random sample of five units and $T = 5$, 15, and 45 observations per unit. $N_s = 3,333$ units per shard, $N = 100,000$ units, $S = 30$ shards, and $R = 20,000$ MCMC iterations including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

Figure 3: Convergence of unit-level posteriors: Q-Q plots ($N_s = 33,333$)



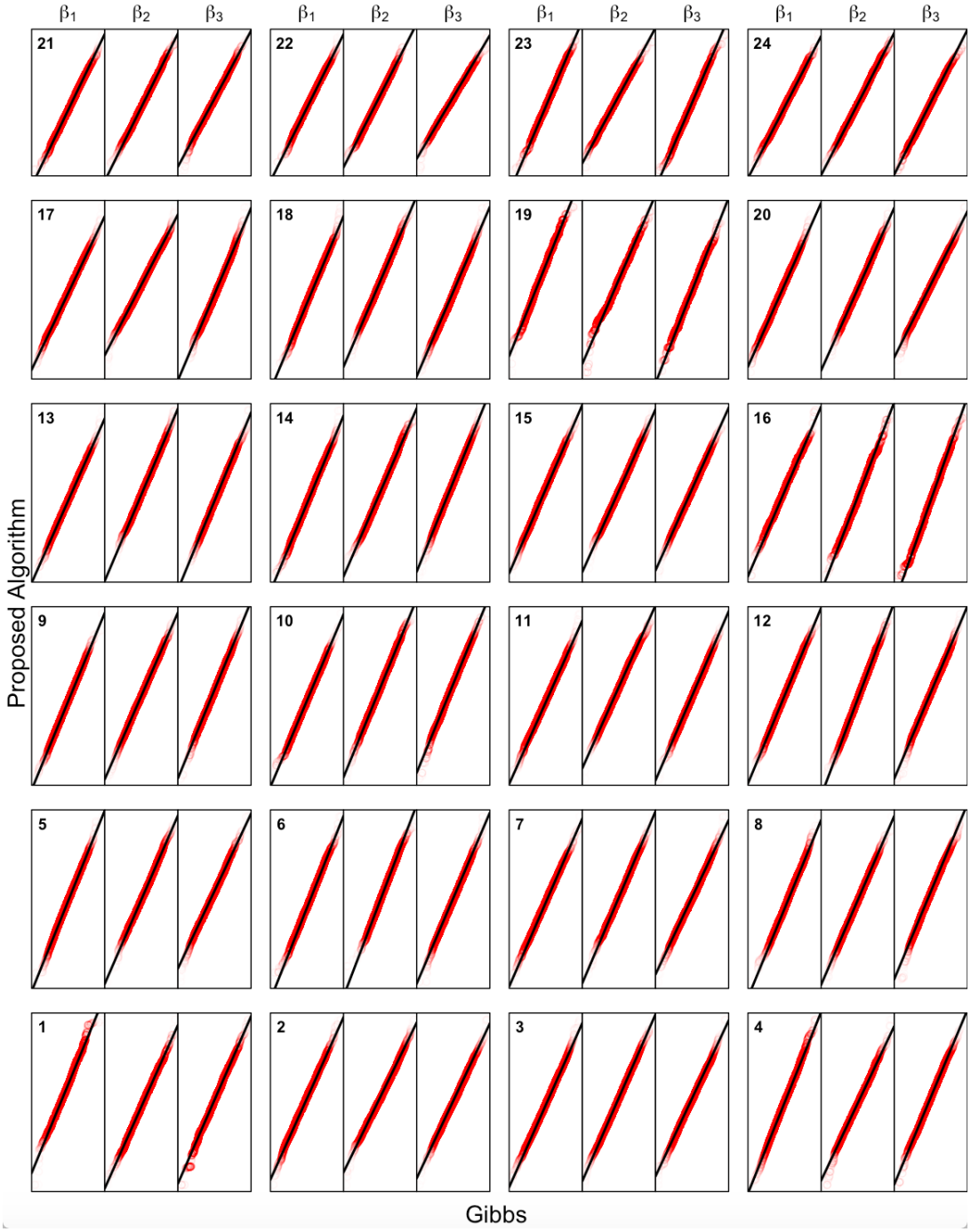
Q-Q plots of unit-specific draws from the proposed algorithm and the single machine hybrid Gibbs algorithm, for a random sample of five units and $T = 5$, 15, and 45 observations per unit. $N_s = 33,333$ units per shard, $N = 1,000,000$ units, $S = 30$ shards, and $R = 20,000$ MCMC iterations including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

Figure 4: Donor response: Convergence of marginals of posterior predictive density estimators



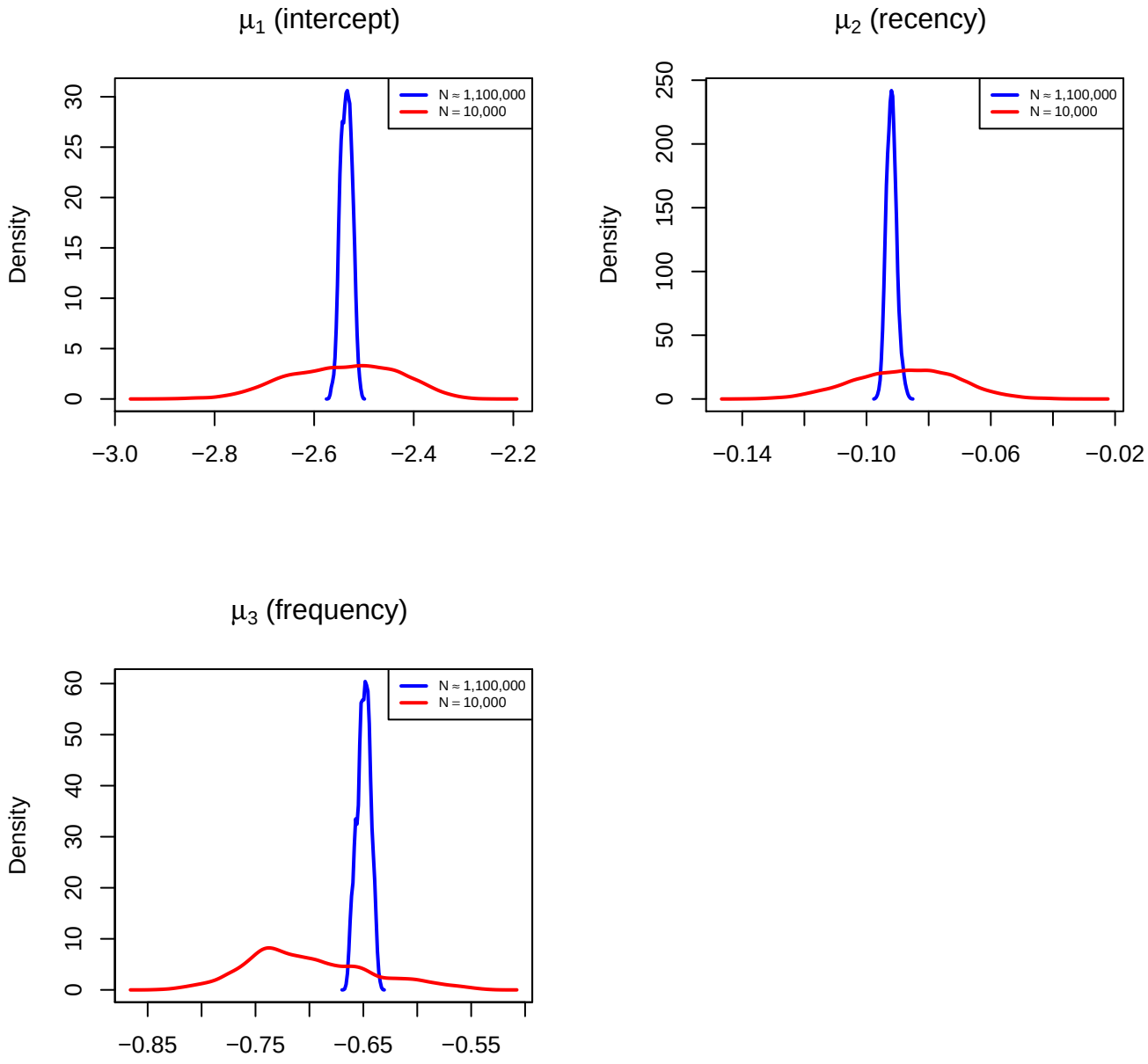
$S = 30$ shards, $N_s \approx 36,000$ units per shard, and $R = 40,000$ MCMC iterations including 5,000 iterations for burn-in ($\dot{p}\left(\beta \,|\{\beta\}_s\right)$ in grey, $\ddot{p}\left(\beta \,|\{\beta\}\right)$ in red, $\dot{p}\left(\beta \,|\{\beta\}\right)$ in blue)

Figure 5: Donor response: QQ plots showing convergence of unit-level posteriors



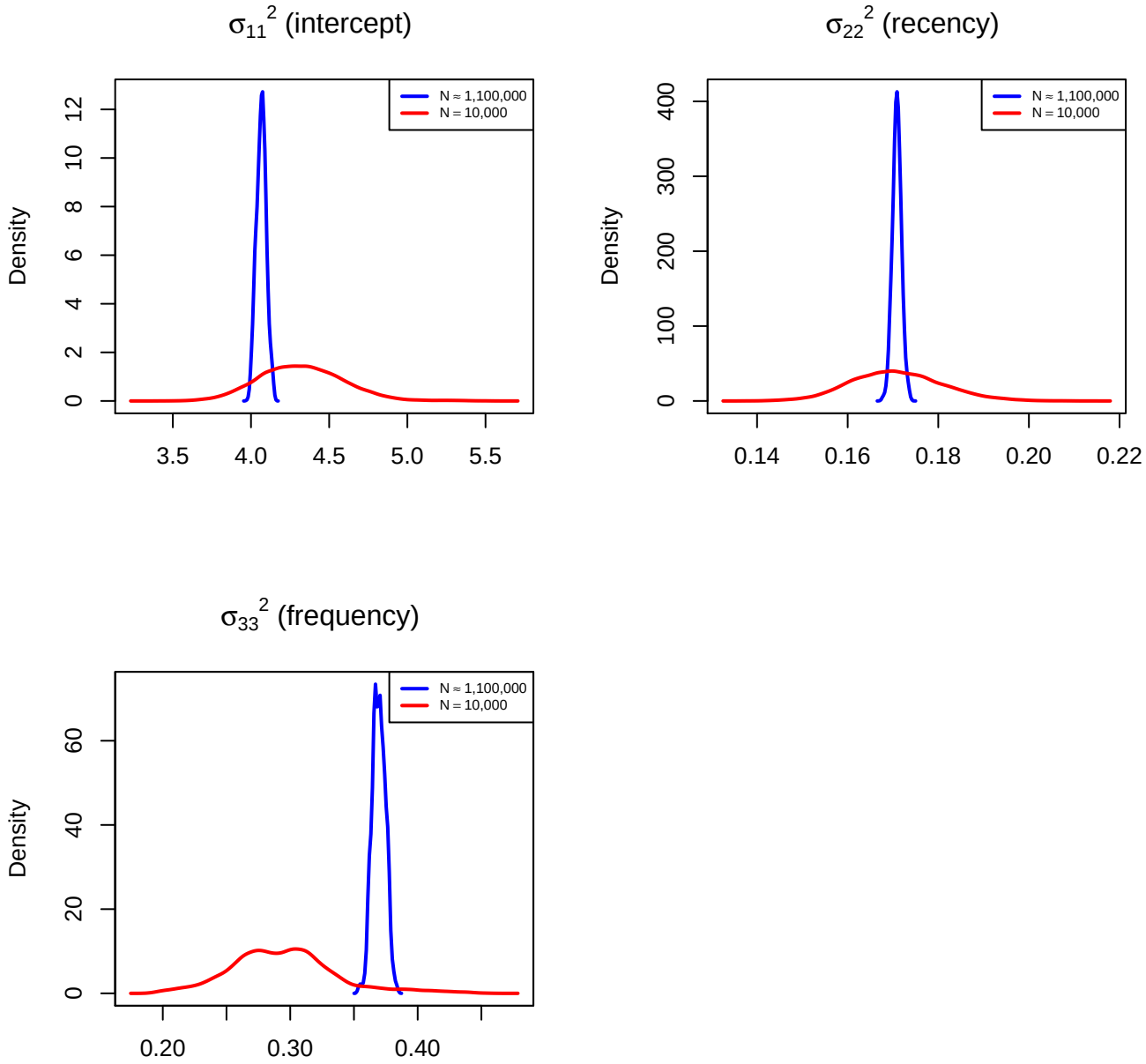
Q-Q plots of unit-specific draws from the proposed algorithm and the single machine hybrid Gibbs algorithm, for a random sample of twenty-four units. $N_s \approx 36,000$ units per shard, $N = 1,088,269$ units, $S = 30$ shards, and $R = 40,000$ MCMC iterations including 5,000 iterations for burn-in

Figure 6: Donor response: Marginal posterior densities of μ (for moderate and big N)



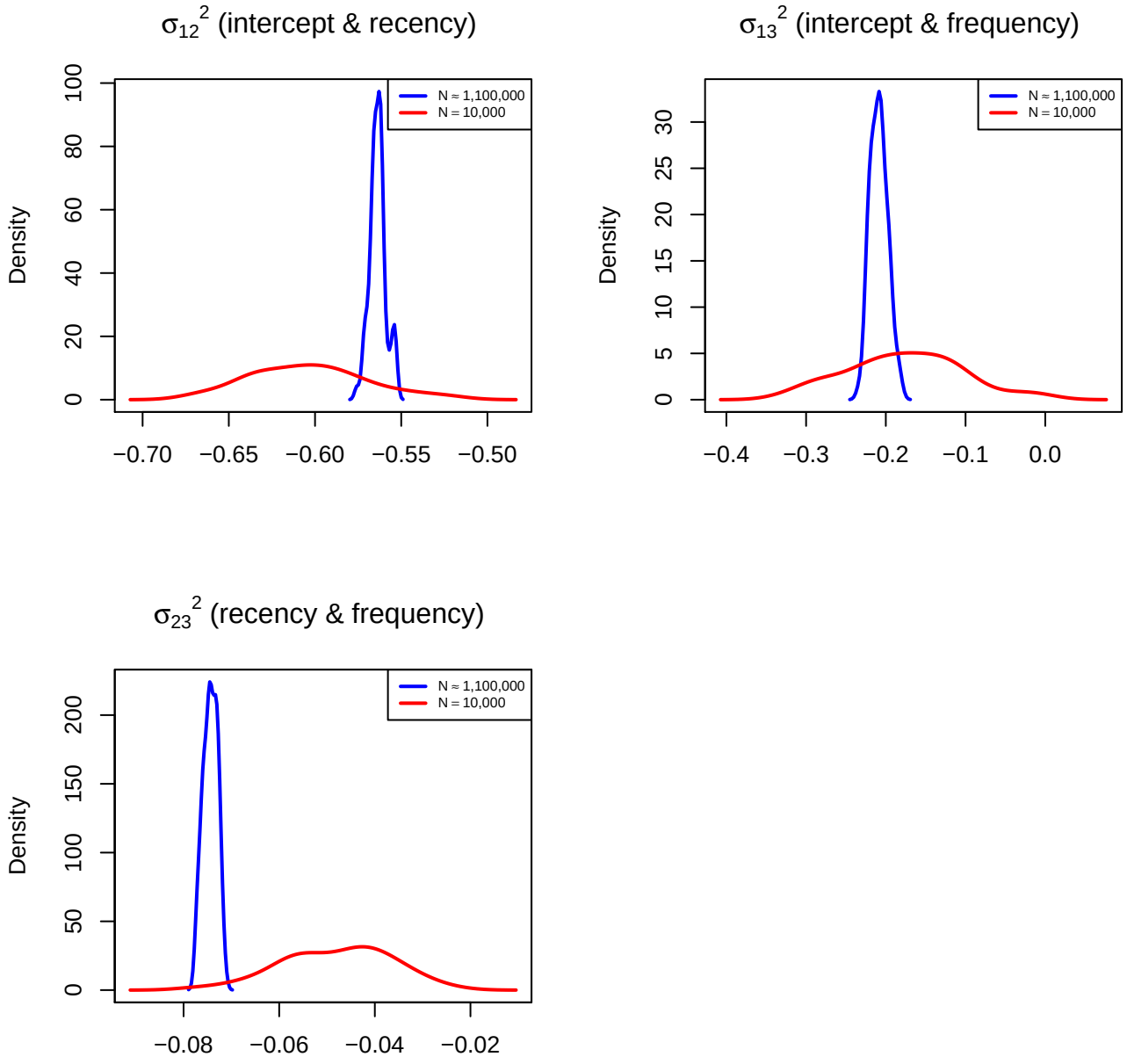
$R = 40,000$ MCMC iterations including 5,000 iterations for burn-in

Figure 7: Donor response: Marginal posterior densities of diagonal elements of Σ (for moderate and big N)



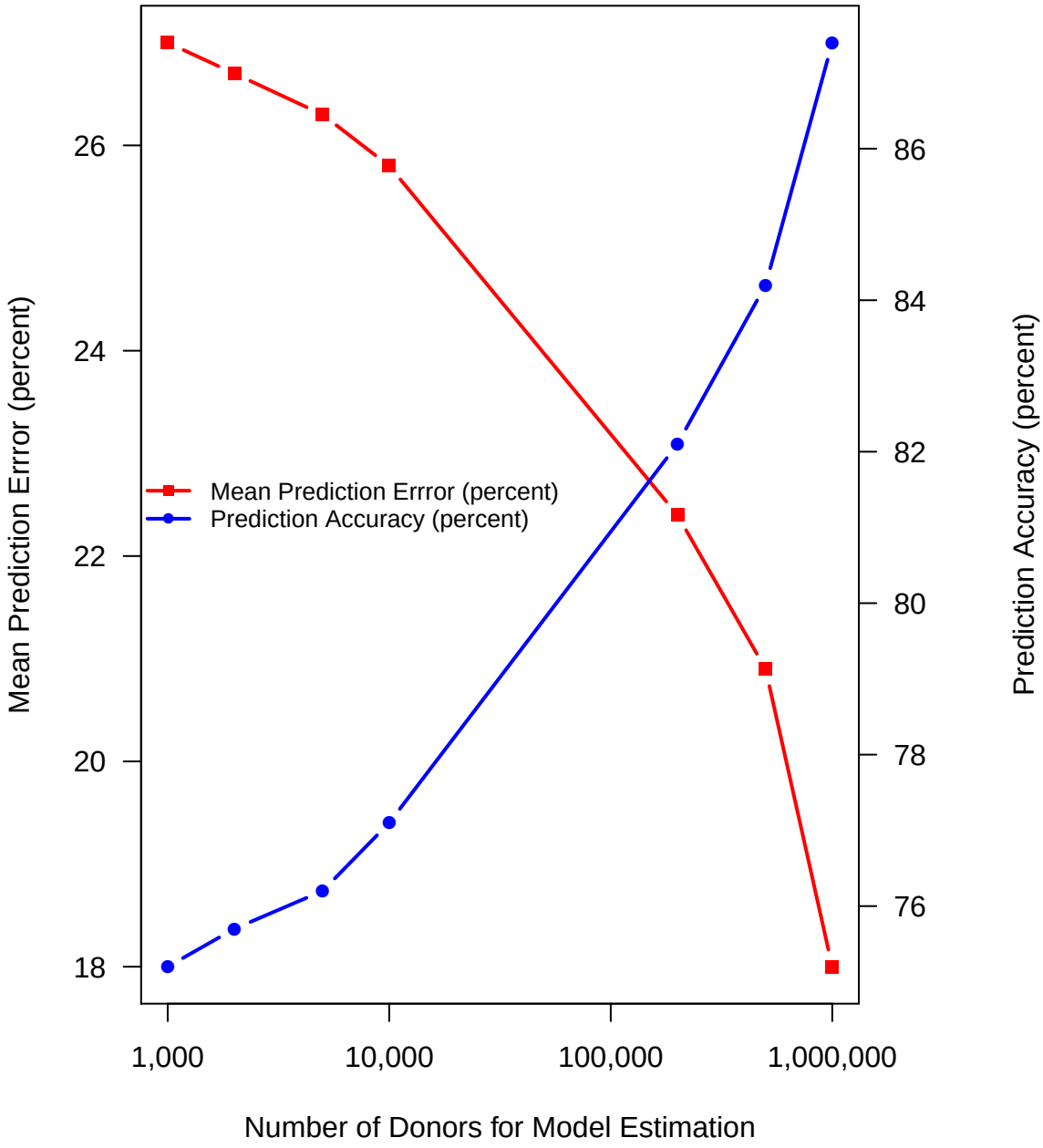
$R = 40,000$ MCMC iterations including 5,000 iterations for burn-in

Figure 8: Donor response: Marginal posterior densities of off-diagonal elements of Σ (moderate and big N)



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

Figure 9: 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.

Table 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	0.994	0.998	0.999	0.991	0.997	0.999	0.994	0.998	0.999	0.992	0.998	0.999
15	0.982	0.993	0.999	0.977	0.993	0.999	0.976	0.993	0.999	0.978	0.994	0.999
45	0.912	0.976	0.997	0.926	0.974	0.997	0.909	0.976	0.997	0.936	0.979	0.998

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

Table 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	0.993	0.998	0.999	0.995	0.997	0.999	0.995	0.998	0.999	0.994	0.998	0.999
15	0.980	0.994	0.999	0.978	0.993	0.999	0.979	0.993	0.999	0.982	0.995	0.999
45	0.916	0.973	0.997	0.919	0.974	0.997	0.918	0.970	0.997	0.900	0.973	0.997

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

Table 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	0.7	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	0.5	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	0.3	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

Table 4: Acceptance rates for proposed algorithm

T	Hybrid Gibbs (percent)	Proposed algorithm (percent)
5	20.2	36.4
15	21.5	16.5
45	22.9	5.0

Performance metrics are for a random sample of 1,000 units. $N_s = 33,333$ units per shard, $N = 1,000,000$ units, $S = 30$ shards, $R = 20,000$ MCMC iterations including a burn-in of $4,000$ iterations.

Table 5: Scalability with S: Estimation of C_0

Dataset	C_0 Estimation[1] with Small N		Squared Error[2] with Big N	
	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, we first construct $\dot{p}(\beta|\{\beta\})$ and $\ddot{p}(\beta|\{\beta\})$ for $N = 10,000$ units, $S = 3$ shards, $R = 16,000$ MCMC iterations after burn-in $(4,000)$. We approximate the maximum expected squared error as the actual maximum squared error $\epsilon^2 = \sup_{\beta_k, k\in\{1,\ldots,d\}} \left[|\ddot{p}(\beta_k|\{\beta\}) - \dot{p}(\beta_k|\{\beta\})|^2\right]$ and approximate $C_0 \approx \left(\frac{S^2+1}{SNR\epsilon^2}\right)$. For the donation dataset we 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. We solve for the maximum expected squared error ϵ^2_{max} for $S_{max} = 30$ because we are limited to a maximum of 30 cores in our computing environment: $\epsilon^2_{max} \approx \left(\frac{S^2_{max}+1}{S_{max}NR}\right) C_0^{-1}$. 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 $\ddot{p}(\beta|\{\beta\})$ and using $\epsilon^2_{actual} = \sup_{\beta_k, k\in\{1,\ldots,d\}} \left[|\ddot{p}(\beta_k|\{\beta\}) - \dot{p}(\beta_k|\{\beta\})|^2\right]$.

Table 6: 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 Comet, a large-scale cluster computing system at the University of California San Diego Supercomputer Center that uses a Lustre parallel distributed file system. $T = 5$ observations, $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in, 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. Comet limits the number of cores that a single application may use at one time to 1,728

Table 7: Donor response: Convergence of unit-level posterior densities, Q-Q correlations

β_1			β_2			β_3		
1%ile	5%ile	Median	1%ile	5%ile	Median	1%ile	5%ile	Median
0.996	0.999	1.000	0.995	0.999	1.000	0.995	0.999	1.000

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. $N = 1,088,269$ units, $S = 30$ shards, and $R = 40,000$ MCMC iterations including 5,000 iterations for burn-in

Table 8: Donor response: Comparing posterior means of μ draws (moderate and big N)

sample statistic	$N = 10,000$	$N \approx 1,100,000$	test statistic	p-value
mean	$\begin{bmatrix} -2.550 \\ -0.0859 \\ -0.7011 \end{bmatrix}$	$\begin{bmatrix} -2.535 \\ -0.0920 \\ -0.649 \end{bmatrix}$	43.92	4.866×10^{-16}
std. dev.	$\begin{bmatrix} 0.0959 \\ 0.0153 \\ 0.0578 \end{bmatrix}$	$\begin{bmatrix} 0.0118 \\ 0.0017 \\ 0.0067 \end{bmatrix}$		

$R = 40,000$ MCMC iterations including 5,000 iterations for burn-in. To remove any correlation between consecutive draws, every 500^{th} draw is kept for Krishnamoorthy and Yu's (2004) modified Nel and van der Merwe multivariate test for the equality of means.

Table 9: Donor response: Comparing posterior means of Σ draws (moderate and big N)

sample statistic	$N = 10,000$	$N \approx 1,100,000$	test statistic	p-value
mean	$\begin{bmatrix} 4.323 & -0.6036 & -0.1767 \\ -0.6036 & 0.1716 & -0.0479 \\ -0.1767 & -0.0479 & 0.2934 \end{bmatrix}$	$\begin{bmatrix} 4.069 & -0.5636 & -0.2087 \\ -0.5636 & 0.1709 & -0.0744 \\ -0.2087 & -0.0744 & 0.3694 \end{bmatrix}$	136.8	1.346×10^{-35}
std. dev.	$\begin{bmatrix} 0.2244 & 0.0337 & 0.0722 \\ 0.0337 & 0.0087 & 0.0114 \\ 0.0722 & 0.0114 & 0.0363 \end{bmatrix}$	$\begin{bmatrix} 0.0320 & 0.0047 & 0.0108 \\ 0.0047 & 0.0010 & 0.0015 \\ 0.0108 & 0.0015 & 0.0050 \end{bmatrix}$		

$R = 40,000$ MCMC iterations including 5,000 iterations for burn-in. To remove any correlation between consecutive draws, every 500^{th} draw is kept for Krishnamoorthy and Yu's (2004) modified Nel and van der Merwe multivariate test for the equality of means.

7 References

Andrieu, Christophe and Gareth O. Roberts. 2009. The Pseudo-Marginal Approach for Efficient Monte Carlo Computations. *The Annals of Statistics*, **37**(2), 697–725

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 J. 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

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, 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

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

8 Appendix: Theorems

In the following proofs, $\dot{p}\left(\beta\,|\{\beta\}\right)$, $\dot{p}\left(\beta\,|\{\beta\}_s\right)$ and $\ddot{p}\left(\beta\,|\{\beta\}\right)$ are estimators of posterior predictive densities constructed from convergent MCMC chains. $Y = \{Y_s\}_{s=1}^{S}$ is the full data, Y_s is the data in shard s, N is the number of units of data in Y, $N_s = \frac{N}{S}$ is the number of units of data in Y_s, and S is the number of shards (greater than 1). We omit the unit-dependent subscript of β for notational simplicity, unless it is required for clarity.

Theorem. *Posterior Predictive Density*

Claim. $\mathbb{E}_\theta\left[p\left(\beta\,|\theta\right)\right]$ is the posterior predictive density of β

Proof. The posterior predictive density of β is the density of β for a unit whose data y has not yet been observed, given $\{\beta_j\}$ (Gelman et al., 2014). We denote $\{\beta_j\}$ as $\{\beta\}$.

$$
\begin{aligned}
p\left(\beta\,|\{\beta\}\right) &= \int p\left(\beta,\theta\,|\{\beta\}\right) d\theta && (8.1)\\
&= \int p\left(\beta\,|\{\beta\},\theta\right) p\left(\theta\,|\{\beta\}\right) d\theta && (8.2)\\
&= \int p\left(\beta\,|\theta\right) p\left(\theta\,|Y\right) d\theta && (8.3)\\
&= \mathbb{E}_\theta\left[p\left(\beta\,|\theta\right)\right] && (8.4)
\end{aligned}
$$

From (8.2) to (8.3), (i) $p\left(\beta\,|\{\beta\},\theta\right) = p\left(\beta\,|\theta\right)$ follows from the assumed conditional independence of β and $\{\beta\}$ given θ, and (ii) $p\left(\theta\,|\{\beta\}\right) = p\left(\theta\,|Y\right)$ follows from the fact that $\{\beta\}$ contains all of the information about θ that is in Y. The latter may also be seen from the directed acyclic graph of the model (3.1 - 3.3): $y \rightarrow \beta \rightarrow \theta$. □

Theorem. *Unbiased Estimator of the Posterior Predictive Density*

Claim. $\dot{p}\left(\beta\,|\{\beta\}\right)$ is an unbiased estimator of $\mathbb{E}_\theta\left[p\left(\beta\,|\theta\right)\right]$

Proof. We show that $\mathbb{E}\left[\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right] = \mathbb{E}_{\theta}\left[p\left(\beta\left|\theta\right.\right)\right]$

$$\begin{aligned}
\mathbb{E}\left[\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right] &= \mathbb{E}\left[\frac{1}{R}\sum_{r} p(\beta\left|\theta^{r}\right.)\right] \\
&= \frac{1}{R}\sum_{r}\mathbb{E}_{\theta^{r}}\left[p(\beta\left|\theta^{r}\right.)\right] \\
&= \frac{1}{R}\sum_{r}\mathbb{E}_{\theta}\left[p(\beta\left|\theta\right.)\right] \\
&= \mathbb{E}_{\theta}\left[p(\beta\left|\theta\right.)\right]
\end{aligned}$$

□

Theorem. *$\dot{p}\left(\beta\left|Y\right.\right)$ and $p\left(\beta\left|Y\right.\right)$ share a Common Marginal Distribution*

Claim. $\dot{p}\left(\beta\left|Y\right.\right) = \mathbb{E}_{\{\theta^{r}\}}\left[\dot{p}\left(\beta\left|\{\theta^{r}\},Y\right.\right)\right] = p\left(\beta\left|Y\right.\right)$

Proof. The proof is similar to that by Beaumont (2003). We first show that $\dot{p}\left(\beta\left|\{\theta^{r}\},Y\right.\right)$ is an unbiased estimator of $p\left(\beta\left|Y\right.\right)$

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

where $\int p\left(\theta^{j}\left|Y\right.\right) d\theta^{j} = 1$.

We derive the posterior marginal distribution $\dot{p}(\beta|Y)$ by integrating out $\{\theta^r\}$ from the joint distribution induced by $\dot{p}(\beta|\{\theta^r\},Y)$

$$\begin{aligned}\dot{p}(\beta|Y) &= \int \dot{p}(\beta|\{\theta^r\},Y)\,p(\{\theta^r\}|Y)\,d\{\theta^r\} \\ &= \mathbb{E}_{\{\theta^r\}}[\dot{p}(\beta|\{\theta^r\},Y)] \\ &= p(\beta|Y)\end{aligned}$$

□

Theorem. *Limit Distributions*

Claim. The limit distributions of $\dot{p}(\beta|\{\beta\}_s)$ and $\ddot{p}(\beta|\{\beta\})$, for $\beta \in \mathbb{R}^d$, are:

1. $\frac{\sqrt{NR}}{S}\left(\dot{p}(\beta|\{\beta\}_s) - \dot{p}(\beta|\{\beta\})\right) \to^P N\left(0, \nabla p(\beta|\theta)^T I_\theta^{-1} \nabla p(\beta|\theta)\right)$ for $S^2 \gg 1$
2. $\sqrt{\frac{NR}{S}}\left(\ddot{p}(\beta|\{\beta\}) - \dot{p}(\beta|\{\beta\})\right) \to^P N\left(0, \nabla p(\beta|\theta)^T I_\theta^{-1} \nabla p(\beta|\theta)\right)$ for $S^2 \gg 1$

Proof. **1.** limit distribution of $\frac{\sqrt{NR}}{S}\left(\dot{p}(\beta|\{\beta\}_s) - \dot{p}(\beta|\{\beta\})\right)$

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

For purposes of deriving limit distributions, we take a frequentist view in that data Y is a random sample from a distribution for some fixed, nonrandom, unknown parameter. We 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; van der Vaart, 1998). In particular, we 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 $\mathcal{A}_1$ 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 van der Vaart (1998) Theorem 2.7. For notational simplicity, we dispense with the N subscript for θ^r.

We derive the distribution of $\sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}\right.\right)-p\left(\beta\left|\theta\right.\right)\right)$ by applying the multivariate delta method to $\sqrt{N}\left(\theta^{r}-\theta\right)\rightarrow^{P}N\left(0,I_{\theta}^{-1}\right)$ and using the transformation $\dot{p}\left(\beta\left|\{\beta\}\right.\right)=\frac{1}{R}\sum_{r}p\left(\beta\left|\theta^{r}\right.\right)$. $\beta\in\mathbb{R}^{d}$ where d is the dimension of β.

$$\begin{aligned}
\sqrt{N}\left(\theta^{r}-\theta\right) &\rightarrow^{P} N\left(0,I_{\theta}^{-1}\right) \\
\sqrt{N}\left(p\left(\beta\left|\theta^{r}\right.\right)-p\left(\beta\left|\theta\right.\right)\right) &\rightarrow^{P} N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^{T}I_{\theta}^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right) \text{ multivariate delta method} \\
\sqrt{N}\left(\sum_{r}p\left(\beta\left|\theta^{r}\right.\right)-\sum_{r}p\left(\beta\left|\theta\right.\right)\right) &\rightarrow^{P} N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^{T}RI_{\theta}^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right) \text{ sum of R random variables} \\
\sqrt{N}\left(\frac{1}{R}\sum_{r}p\left(\beta\left|\theta^{r}\right.\right)-p\left(\beta\left|\theta\right.\right)\right) &\rightarrow^{P} N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^{T}\left(RI_{\theta}\right)^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right) \text{ divide by R} \\
\sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}\right.\right)-p\left(\beta\left|\theta\right.\right)\right) &\rightarrow^{P} N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^{T}\left(RI_{\theta}\right)^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right)
\end{aligned}$$

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

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

$$\sqrt{N_{s}}\left(\dot{p}\left(\beta\left|\{\beta\}_{s}\right.\right)-p\left(\beta\left|\theta\right.\right)\right)\rightarrow^{P}N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^{T}S\left(RI_{\theta}\right)^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right)$$

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

Solve for $\sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}_{s}\right.\right)-p\left(\beta\left|\theta\right.\right)\right)$ by multiplying $\sqrt{N_{s}}\left(\dot{p}\left(\beta\left|\{\beta\}_{s}\right.\right)-p\left(\beta\left|\theta\right.\right)\right)$ by $\sqrt{S}$

$$\begin{aligned}
\sqrt{N_{s}}\left(\dot{p}\left(\beta\left|\{\beta\}_{s}\right.\right)-p\left(\beta\left|\theta\right.\right)\right) &\rightarrow^{P} N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^{T}S\left(RI_{\theta}\right)^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right) \\
\sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}_{s}\right.\right)-p\left(\beta\left|\theta\right.\right)\right) &\rightarrow^{P} N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^{T}S^{2}\left(RI_{\theta}\right)^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right) \text{ multiply by } \sqrt{S}
\end{aligned}$$

1.4. limit distribution of $\frac{\sqrt{NR}}{S}\left(\dot{p}\left(\beta\left|\{\beta\}_{s}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right)$ for $S>1$, $\beta\in\mathbb{R}^{d}$

First derive the limit distribution of $\sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}_{s}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right)$ by subtracting $\sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}\right.\right)-p\left(\beta\left|\theta\right.\right)\right)$

from $\sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}_s\right.\right)-p\left(\beta\left|\theta\right.\right)\right)$

$$\begin{aligned}\sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}_s\right.\right)-p\left(\beta\left|\theta\right.\right)\right)-\sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}\right.\right)-p\left(\beta\left|\theta\right.\right)\right) &\rightarrow^P N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^T\left[S^2\left(RI_\theta\right)^{-1}+\left(RI_\theta\right)^{-1}\right]\right.\\ &\left.\times\nabla p\left(\beta\left|\theta\right.\right)\right)\\ \sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}_s\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right) &\rightarrow N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^T\left(\frac{S^2+1}{R}\right)I_\theta^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right)\end{aligned}$$

Therefore,

$$\begin{aligned}\frac{\sqrt{NR}}{S}\left(\dot{p}\left(\beta\left|\{\beta\}_s\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right) &\rightarrow^P N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^T\left(\frac{S^2+1}{S^2}\right)I_\theta^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right)\text{ multiply by }\frac{\sqrt{R}}{S}\\ \frac{\sqrt{NR}}{S}\left(\dot{p}\left(\beta\left|\{\beta\}_s\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right) &\rightarrow^P N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^T I_\theta^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right)\ S^2\gg 1\end{aligned}$$

2. limit distribution of $\sqrt{\frac{NR}{S}}\left(\ddot{p}\left(\beta\left|\{\beta\}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right)$, for $\beta\in\mathbb{R}^d$

First solve for $\sqrt{N}\left(\ddot{p}\left(\beta\left|\{\beta\}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right)$ by using the transformation $\widehat{p}\left(\beta\left|Y\right.\right)=\frac{1}{S}\sum_s p\left(\dot{p}\left(\beta\left|\{\beta\}_s\right.\right)\right)$

$$\begin{aligned}\sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}_s\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right) &\rightarrow^P N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^T\left(\frac{S^2+1}{R}\right)I_\theta^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right)\text{ from }\mathbf{1.4}\text{ above}\\ \sqrt{N}\left(\frac{1}{S}\sum_s\left(\dot{p}\left(\beta\left|\{\beta\}_s\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right)\right) &\rightarrow^P N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^T\left(\frac{S^2+1}{SR}\right)I_\theta^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right)\text{ mean of S rand. vars.}\\ \sqrt{N}\left(\ddot{p}\left(\beta\left|\{\beta\}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right) &\rightarrow^P N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^T\left(\frac{S^2+1}{SR}\right)I_\theta^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right)\end{aligned}$$

Therefore

$$\begin{aligned}\sqrt{\frac{NR}{S}}\left(\ddot{p}\left(\beta\left|\{\beta\}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right) &\rightarrow^P N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^T\left(\frac{S^2+1}{S^2}\right)I_\theta^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right)\text{ multiply by }\frac{\sqrt{R}}{S}\\ \sqrt{\frac{NR}{S}}\left(\ddot{p}\left(\beta\left|\{\beta\}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right) &\rightarrow^P N\left(0,\nabla p\left(\beta\left|\theta\right.\right)^T I_\theta^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right)\ S^2\gg 1\end{aligned}$$

□

Theorem. *Expected Squared Error*

Claim. The expected squared errors between $\dot{p}\left(\beta\left|\{\beta\}_s\right.\right)$ and $\dot{p}\left(\beta\left|\{\beta\}\right.\right)$, and between $\ddot{p}\left(\beta\left|\{\beta\}\right.\right)$ and

$\dot{p}\left(\beta\left|\{\beta\}\right.\right)$, for $\beta \in \mathbb{R}^d$, are:

1. $\mathbb{E}\left[\left|\dot{p}\left(\beta\left|\{\beta\}_s\right.\right) - \dot{p}\left(\beta\left|\{\beta\}\right.\right)\right|^2\right] \approx \left(\frac{S^2}{NR}\right) \nabla p\left(\beta\left|\theta\right.\right)^T I_\theta^{-1} \nabla p\left(\beta\left|\theta\right.\right)$ for $S^2 \gg 1$
2. $\mathbb{E}\left[\left|\ddot{p}\left(\beta\left|\{\beta\}\right.\right) - \dot{p}\left(\beta\left|\{\beta\}\right.\right)\right|^2\right] \approx \left(\frac{S}{NR}\right) \nabla p\left(\beta\left|\theta\right.\right)^T I_\theta^{-1} \nabla p\left(\beta\left|\theta\right.\right)$ for $S^2 \gg 1$

Proof. Since $\sqrt{N}\left(\dot{p}\left(\beta\left|\{\beta\}_s\right.\right) - \dot{p}\left(\beta\left|\{\beta\}\right.\right)\right) \rightarrow^P N\left(0, \nabla p\left(\beta\left|\theta\right.\right)^T \left(\frac{S^2+1}{R}\right) I_\theta^{-1} \nabla p\left(\beta\left|\theta\right.\right)\right)$, assume that for large finite N

$$\dot{p}\left(\beta\left|\{\beta\}_s\right.\right) - \dot{p}\left(\beta\left|\{\beta\}\right.\right) \sim N\left(\frac{B_s}{N}, \nabla p\left(\beta\left|\theta\right.\right)^T \left(\frac{S^2+1}{NR}\right) I_\theta^{-1} \nabla p\left(\beta\left|\theta\right.\right)\right)$$

where B_s is some non-zero bias. Therefore $\dot{p}\left(\beta\left|\{\beta\}_s\right.\right) - \dot{p}\left(\beta\left|\{\beta\}\right.\right)$ is normally distributed with mean $\frac{B_s}{N}$ and variance $\nabla p\left(\beta\left|\theta\right.\right)^T \left(\frac{S^2+1}{NR}\right) I_\theta^{-1} \nabla p\left(\beta\left|\theta\right.\right)$ for $\beta \in \mathbb{R}^d$, and

$$\begin{aligned}
\mathbb{E}\left[\left|\dot{p}\left(\beta\left|\{\beta\}_s\right.\right) - \dot{p}\left(\beta\left|\{\beta\}\right.\right)\right|^2\right] &= Var\left(\dot{p}\left(\beta\left|\{\beta\}_s\right.\right) - \dot{p}\left(\beta\left|\{\beta\}\right.\right)\right) + \mathbb{E}\left[\dot{p}\left(\beta\left|\{\beta\}_s\right.\right) - \dot{p}\left(\beta\left|\{\beta\}\right.\right)\right]^2 \\
&= \nabla p\left(\beta\left|\theta\right.\right)^T \left(\frac{S^2+1}{NR}\right) I_\theta^{-1} \nabla p\left(\beta\left|\theta\right.\right) + \left(\frac{B_s}{N}\right)^2 \\
&\approx \nabla p\left(\beta\left|\theta\right.\right)^T \left(\frac{S^2+1}{NR}\right) I_\theta^{-1} \nabla p\left(\beta\left|\theta\right.\right) \qquad \text{large N} \\
&\approx \left(\frac{S^2}{NRp^2}\right) \nabla p\left(\beta\left|\theta\right.\right)^T I_\theta^{-1} \nabla p\left(\beta\left|\theta\right.\right) \qquad S^2 \gg 1
\end{aligned}$$

Similarly, since $\sqrt{N}\left(\ddot{p}\left(\beta\left|\{\beta\}\right.\right) - \dot{p}\left(\beta\left|\{\beta\}\right.\right)\right) \rightarrow^P N\left(0, \nabla p\left(\beta\left|\theta\right.\right)^T \left(\frac{S^2+1}{SR}\right) I_\theta^{-1} \nabla p\left(\beta\left|\theta\right.\right)\right)$

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

where B is some non-zero bias. Therefore

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

□

Theorem. *Maximum Number of Shards*

Claim. The maximum number of shards S_{max}, and an empirical estimate for C_0 are, respectively,

1. $S_{max} = \left\lfloor \frac{C_0}{2} \left(NR\epsilon_{max}^2 + \sqrt{(NR\epsilon_{max}^2)^2 - 4C_0^{-2}} \right) \right\rfloor \approx \lfloor C_0 NR\epsilon_{max}^2 \rfloor$ for $S^2 \gg 1$, where $C_0 = \left\{ \sup_\beta \left[\nabla p\left(\beta \left| \theta \right.\right)^T I_\theta^{-1} \nabla p\left(\beta \left| \theta \right.\right) \right] \right\}^{-1}$ and ϵ_{max}^2 is the maximum expected squared error
2. $C_0 \approx \left(\frac{S^2+1}{SNR} \right) \left\{ \sup_{\beta_k, k \in \{1,\ldots,d\}} \left[\left| \ddot{p}\left(\beta_k \left| \{\beta\} \right.\right) - \dot{p}\left(\beta_k \left| \{\beta\} \right.\right) \right|^2 \right] \right\}^{-1}$

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

$$\begin{aligned} \epsilon_{max}^2 &= \sup_\beta \left[\epsilon^2\left(\beta\right) \right] \\ &= \sup_\beta \left[\left(\frac{S^2+1}{SNR} \right) \nabla p\left(\beta \left| \theta \right.\right)^T I_\theta^{-1} \nabla p\left(\beta \left| \theta \right.\right) \right] \\ &= \left(\frac{S^2+1}{SNR} \right) \sup_\beta \left[\nabla p\left(\beta \left| \theta \right.\right)^T I_\theta^{-1} \nabla p\left(\beta \left| \theta \right.\right) \right] \end{aligned}$$

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

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

$\nabla p\left(\beta \left| \theta \right.\right)^T I_\theta^{-1} \nabla p\left(\beta \left| \theta \right.\right)$ must be computed at θ for $\beta \in \mathbb{R}^d$. It is more convenient to empirically estimate $C_0 = \left\{ \sup_\beta \left[\nabla p\left(\beta \left| \theta \right.\right)^T I_\theta^{-1} \nabla p\left(\beta \left| \theta \right.\right) \right] \right\}^{-1}$ for some small but sufficiently large N, so that

N/S is large enough that the Bernstein-von Mises theorem applies.

$$\begin{aligned}
\epsilon_{max}^2 &= \sup_{\beta}\left[\epsilon^2\left(\beta\right)\right] \\
\left(\frac{S^2+1}{SNR}\right)\sup_{\beta}\left[\nabla p\left(\beta\left|\theta\right.\right)^T I_{\theta}^{-1}\nabla p\left(\beta\left|\theta\right.\right)\right] &= \sup_{\beta}\left[\mathbb{E}\left[\left|\ddot{p}\left(\beta\left|\{\beta\}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right|^2\right]\right] \\
\left(\frac{S^2+1}{SNR}\right)C_0^{-1} &\approx \sup_{\beta}\left[\frac{1}{M}\sum_m\left|\ddot{p}_m\left(\beta\left|\{\beta\}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right|^2\right] \\
C_0 &\approx \left(\frac{S^2+1}{SNR}\right)\left\{\sup_{\beta}\left[\frac{1}{M}\sum_m\left|\ddot{p}_m\left(\beta\left|\{\beta\}\right.\right)-p\left(\beta\left|Y\right.\right)\right|^2\right]\right\}^{-1} \\
&\approx \left(\frac{S^2+1}{SNR}\right)\left\{\sup_{\beta}\left[\left|\ddot{p}\left(\beta\left|\{\beta\}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right|^2\right]\right\}^{-1}
\end{aligned}$$

where $\ddot{p}_m\left(\beta\left|\{\beta\}\right.\right)$ is the posterior predictive density estimator for the m^{th} random partitioning of data Y into S shards. For computational convenience, we let $M=1$. C_0 may be approximated by computing $\left(\frac{S^2+1}{SNR}\right)\left\{\sup_{\beta}\left[\left|\widehat{p}\left(\beta\left|Y\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right|^2\right]\right\}^{-1}$.

Since $\beta\in\mathbb{R}^d$, for $d>1$ it may be computationally demanding to estimate $\sup_{\beta}\left[\left|\ddot{p}\left(\beta\left|\{\beta\}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right|^2\right]$. Let $\beta=\left(\beta_1,...,\beta_d\right)^T$ so that $\dot{p}\left(\beta_k\left|\{\beta\}\right.\right)$ and $\ddot{p}\left(\beta_k\left|\{\beta\}\right.\right)$ denote the estimator for marginal posterior predictive density and its estimator for β element $k\in\{1,...,d\}$. We adopt the approximation $\sup_{\beta}\left[\left|\ddot{p}\left(\beta_k\left|\{\beta\}\right.\right)-\dot{p}\left(\beta\left|\{\beta\}\right.\right)\right|^2\right]\approx C\sup_{\beta_k,k\in\{1,...,d\}}\left[\left|\ddot{p}\left(\beta_k\left|\{\beta\}\right.\right)-\dot{p}\left(\beta_k\left|\{\beta\}\right.\right)\right|^2\right]$ where C is a proportionality constant that may be absorbed by C_0. Therefore

$$C_0\approx\left(\frac{S^2+1}{SNR}\right)\left\{\sup_{\beta_k,k\in\{1,...,d\}}\left[\left|\ddot{p}\left(\beta_k\left|\{\beta\}\right.\right)-\dot{p}\left(\beta_k\left|\{\beta\}\right.\right)\right|^2\right]\right\}^{-1}$$

□

Theorem. *Asymptotic Unbiasedness of* $\ddot{p}\left(\beta\left|\{\beta\}\right.\right)$

Claim. $lim_{N\to\infty}\mathbb{E}\left[\ddot{p}\left(\beta\left|\{\beta\}\right.\right)\right]=\dot{p}\left(\beta\left|\{\beta\}\right.\right)$, for $\beta\in\mathbb{R}^d$

Proof. Let θ_N^r denote the r^{th} draw of θ from the posterior density $p\left(\theta\left|Y\right.\right)$, 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 we may choose any prior density $p\left(\theta\right)$ that appropriately restricts

the amount of probability in the tails of posterior $p(\theta|Y)$ Since $\theta_N^r \to^P \theta$ (**Theorem.** *Limit Distributions*, 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 $\beta \in \mathbb{R}^d$, $\dot{p}(\beta|\{\beta\}) \to^{L_1} p(\beta|\theta)$ by the same reasoning as in the Limit Distributions theorem in this article (replace convergence in probability with L_1-convergence).

Similarly, by the above reasoning, $\theta_{N_s}^r \to^{L_1} \theta$ and $\dot{p}(\beta|\{\beta\}_s) \to^{L_1} p(\beta|\theta)$. Again, following the same reasoning as in the Limit Distributions theorem in this article (replace convergence in probability with L_1-convergence), it follows that $\ddot{p}(\beta|\{\beta\}) \to^{L_1} \dot{p}(\beta|\{\beta\})$ and therefore that $lim_{N\to\infty}\mathbb{E}[\ddot{p}(\beta|\{\beta\})] = \dot{p}(\beta|\{\beta\})$. □

Theorem. *Maximum Number of Shards with Stage One Subsampling*

Claim. The maximum number of shards S_{max} with stage one sampling, and an empirical estimate for C_0 are, respectively,

1. $S_{max} = \left\lfloor \frac{C_0}{2}\left(NR\epsilon_{max}^2 p^2 + \sqrt{(NR\epsilon_{max}^2 p^2)^2 - 4p^2 C_0^{-2}}\right)\right\rfloor \approx \left\lfloor C_0 NR\epsilon_{max}^2 p^2 \right\rfloor$ for $S^2 \gg p^2$, where $C_0 = \left\{\sup_\beta \left[\nabla p(\beta|\theta)^T I_\theta^{-1} \nabla p(\beta|\theta)\right]\right\}^{-1}$ and ϵ_{max}^2 is the maximum expected squared error
2. $C_0 \approx \left(\frac{S^2+p^2}{SNR}\right)\left\{\sup_{\beta_k, k\in\{1,\ldots,d\}}\left[|\ddot{p}(\beta_k|\{\beta\}) - \dot{p}(\beta_k|\{\beta\})|^2\right]\right\}^{-1}$

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

$$\begin{aligned}
\sqrt{NRp^2/S}\,(\ddot{p}(\beta|\{\beta\}) - \dot{p}(\beta|\{\beta\})) &\to^P N\left(0, \nabla p(\beta|\theta)^T \left(\frac{S^2+p^2}{S^2}\right) I_\theta^{-1} \nabla p(\beta|\theta)\right) \\
\sqrt{NRp^2/S}\,(\ddot{p}(\beta|\{\beta\}) - \dot{p}(\beta|\{\beta\})) &\to^P N\left(0, \nabla p(\beta|\theta)^T I_\theta^{-1} \nabla p(\beta|\theta)\right) \qquad S^2 \gg p^2
\end{aligned}$$

Follow **Theorem.** *Expected Squared Error* to show that

$$\begin{aligned}
\mathbb{E}\left[|\ddot{p}(\beta|\{\beta\}) - \dot{p}(\beta|\{\beta\})|^2\right] &= \nabla p(\beta|\theta)^T \left(\frac{S^2+p^2}{SNRp^2}\right) I_\theta^{-1} \nabla p(\beta|\theta) \\
&\approx \left(\frac{S}{NRp^2}\right) \nabla p(\beta|\theta)^T I_\theta^{-1} \nabla p(\beta|\theta) \qquad S^2 \gg p^2
\end{aligned}$$

Follow **Theorem.** *Maximum Number of Shards* to show that

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

and C_0 may be approximated by

$$C_0 \approx \left(\frac{S^2+p^2}{SNR}\right) \left\{ \sup_{\beta_k, k \in \{1,\ldots,d\}} \left[\left| \ddot{p}\left(\beta_k \left| \{\beta\} \right.\right) - \dot{p}\left(\beta_k \left| \{\beta\} \right.\right) \right|^2 \right] \right\}^{-1}$$

□

9 Appendix: Proposed Algorithm with Stage One Subsampling

Algorithm $\mathcal{A}_3'$ Stage one of the proposed algorithm with subsampling (for non-standard posteriors)

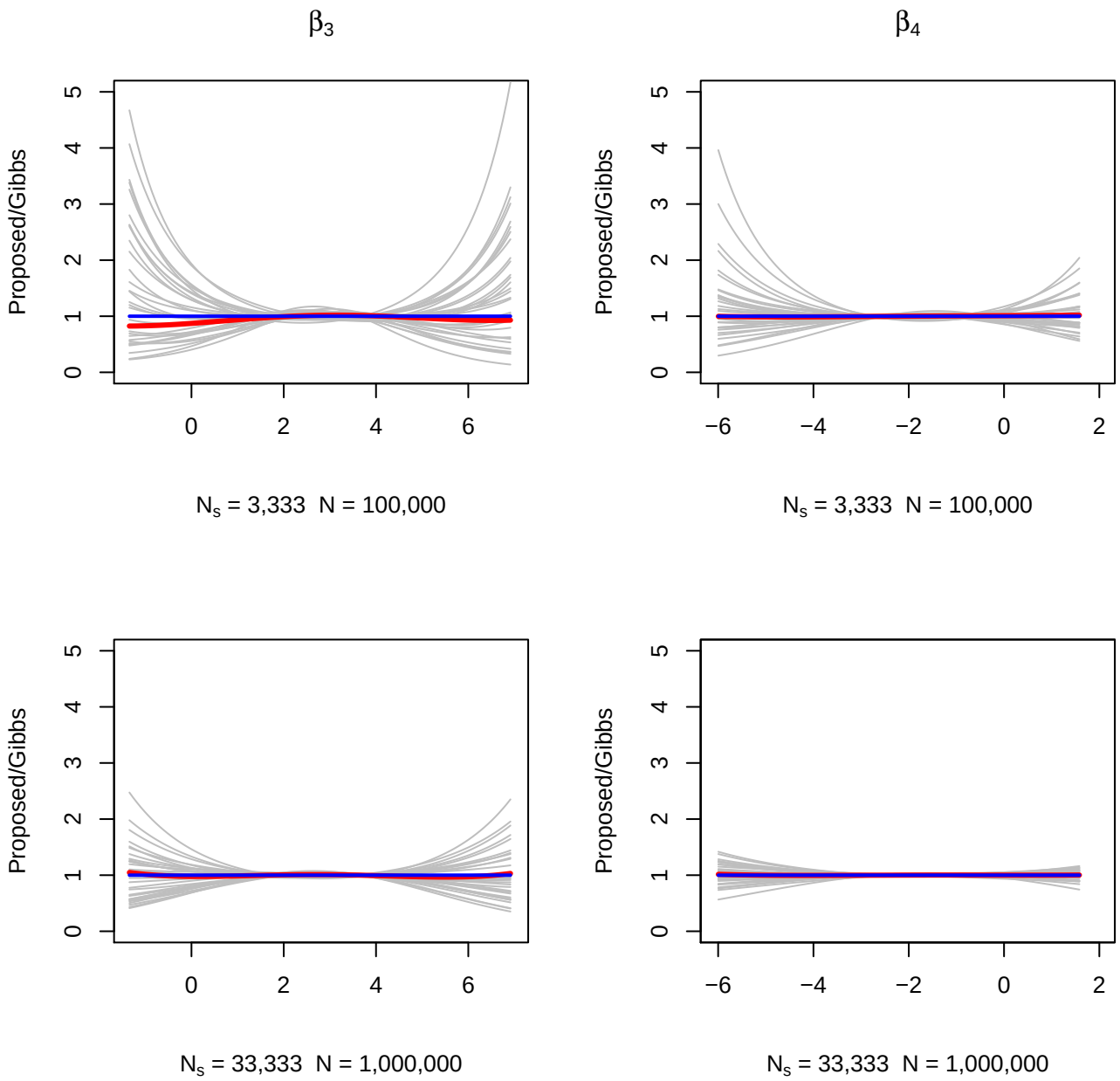
Stage One: Draw from $\beta \sim \ddot{p}\left(\beta \left| \{\beta\} \right.\right)$

Input: $Y = \cup_i \{y_{it}\}_{t=1}^T$

Output: $\{\beta^r\}_{r=1}^R$

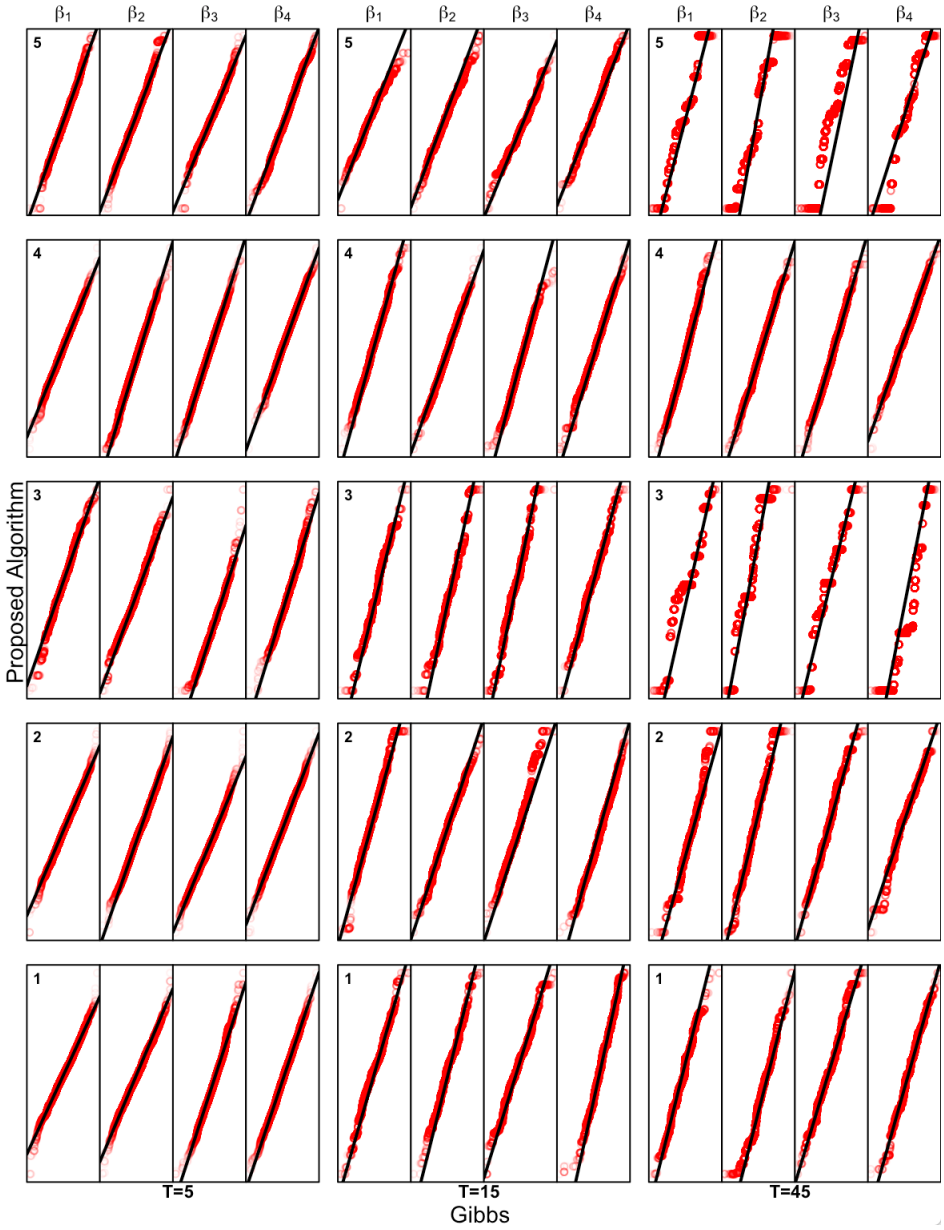
1. Sample Y with probability $p \in (0,1)$
   (a) Randomly assign each cross-sectional unit in the stage one sample
      i. $I_i \sim Bernoulli\left(p\right)$, $i = 1, ..., N$
   (b) Define $I^p = \{i : I_i = 1\}$, $N^p = \sum_i I_i$,
   (c) $Y^p = \cup_{i \in I^p} \{y_{it}\}_{t=1}^T$
2. Divide Y^p into S independent shards
   (a) $I_s^p = \{i \in I^p : z_i = s\}$, $s = 1, ..., S$, where $p\left(z_i = s\right) = \frac{1}{S}$, and $|I_s^p| = \frac{Np}{S}$
   (b) $Y_s^p = \cup_{i \in I_s^p} \{y_{it}\}_{t=1}^T$, $s = 1, ..., S$
3. Run S parallel MCMC simulations ($s = 1, ..., S$)
   (a) Set θ_s^0, β_i^0 $for\ all\ i \in I_s^p$
   (b) **for** $r = 1$ to R
      i. Draw $\beta_i^r \left| \theta_s^{r-1}, Y_s^p \right. \propto p\left(\beta_i^r \left| \theta_s^{r-1} \right.\right) \prod_t p\left(y_{it} \left| \beta_i^r \right.\right)$ $for\ i \in I_s^p$ (Metropolis-Hastings draw)
      ii. Draw $\theta_s^r \left| \{\beta_i^r\}_{i \in I_s^p} \right. \sim p(\theta_s^r \left| \tau \right.) \prod_{i \in I_s^p} p\left(\beta_i^r \left| \theta_s^r \right.\right)$ (standard draw using a conjugate prior)
   (c) **for** $r = 1$ to R/S
      i. Draw $z^r \sim Multinomial\left(n = 1, p = \left\{\frac{1}{R}, ..., \frac{1}{R}\right\}\right)$
      ii. Draw $\beta_s^r \left| \theta_s^{z^r} \right. \sim p(\beta \left| \theta_s^{z^r} \right.)$
4. Collect the β draws and shuffle
   (a) $\beta = \cup_{s=1}^S \{\beta_s^r\}_{r=1}^{r=R/S}$
   (b) $\beta = permute\left(\beta\right)$

Figure 1: Convergence of marginals of posterior predictive density estimators



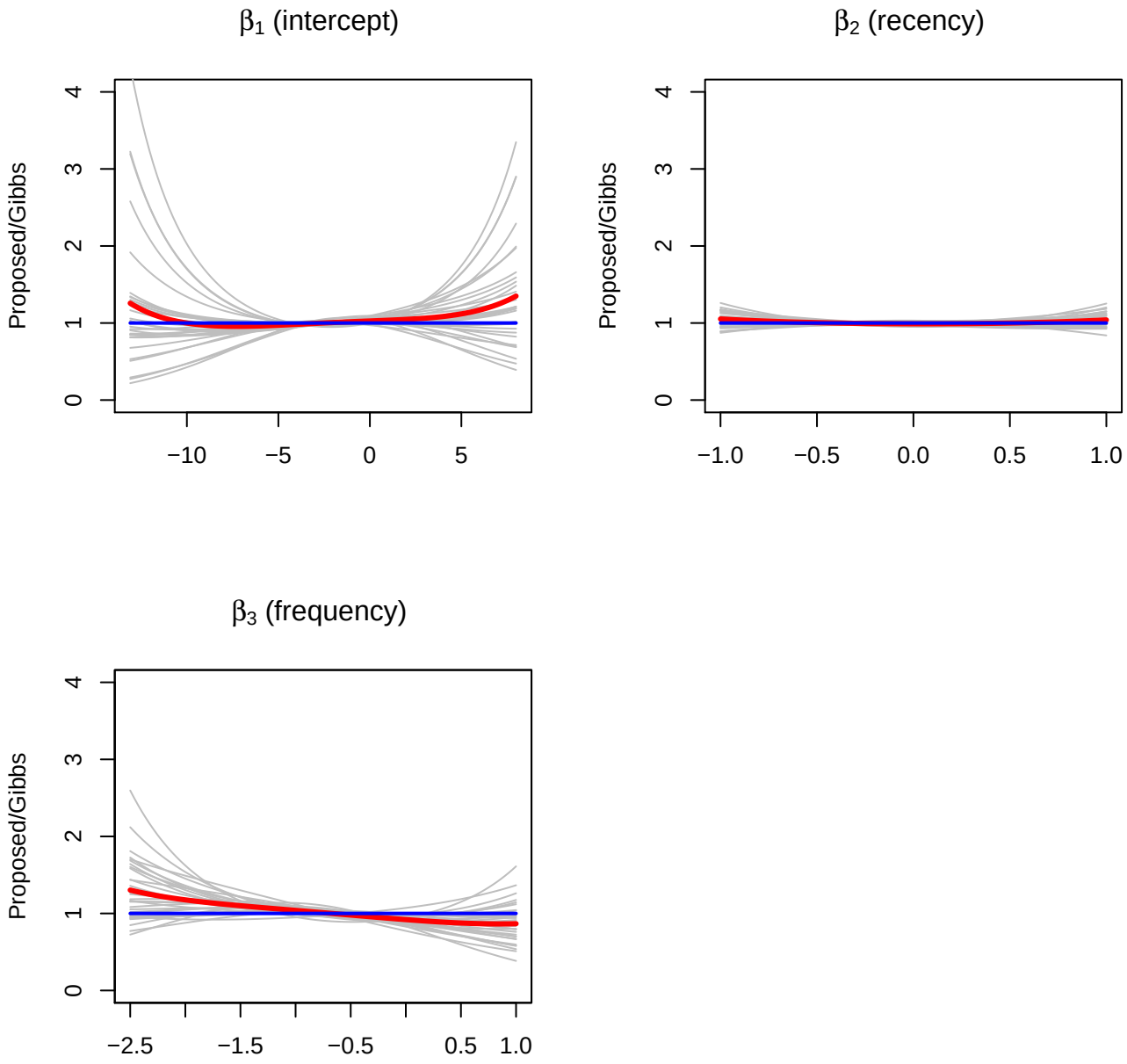
$T = 5$ observations per unit, $S = 30$ shards, and $R = 10,000$ MCMC iterations including 2,000 iterations for burn-in ($\frac{\dot{p}(\beta|\{\beta\}_s)}{\dot{p}(\beta|\{\beta\})}$ in grey, $\frac{\ddot{p}(\beta|\{\beta\})}{\dot{p}(\beta|\{\beta\})}$ in red, $\frac{\dot{p}(\beta|\{\beta\})}{\dot{p}(\beta|\{\beta\})}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

Figure 2: 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

Figure 3: 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 ($\frac{\dot{p}(\beta|\{\beta\}_s)}{\dot{p}(\beta|\{\beta\})}$ in grey, $\frac{\ddot{p}(\beta|\{\beta\})}{\dot{p}(\beta|\{\beta\})}$ in red, $\frac{\dot{p}(\beta|\{\beta\})}{\dot{p}(\beta|\{\beta\})} = 1$ in blue)

Table 1: 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	0.992	0.996	0.999	0.994	0.997	0.999	0.990	0.996	0.999	0.992	0.9978	0.999
15	0.971	0.993	0.999	0.972	0.992	0.999	0.974	0.993	0.999	0.980	0.995	0.999
45	0.908	0.970	0.997	0.913	0.968	0.997	0.916	0.971	0.997	0.887	0.971	0.998

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

Table 2: 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

Table 3: Acceptance rates of the proposed algorithm (with stage one subsampling)

T	Hybrid Gibbs (percent)	Proposed Algorithm with Stage One Subsampling (percent)
5	20.2	36.8
15	21.5	16.6
45	22.9	5.0

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

Table 4: 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	61 (0)
			Subsampling*	13 (0)
10 million	300	33,333	Proposed	76 (2)
			Subsampling*	19 (3)
100 million	1,728[2]	57,870	Proposed	162 (21)
			Subsampling*	78 (19)

Scalability testing is implemented on Comet, a large-scale cluster computing system at the University of California San Diego Supercomputer Center that uses a Lustre parallel distributed file system. $T = 5$ observations, $R = 20,000$ MCMC iterations including 4,000 iterations for burn-in, keeping every 10^{th} draw. For the proposed algorithm with stage one subsampling $p = 10\%$.

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. Comet limits the number of cores that a single application may use at one time to 1,728