Bayesian Logistic Regression with the Local Bouncy Particle Sampler for COVID-19

by

Jie Wang

B.Sc., Simon Fraser University, 2015

Project Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
Department of Statistics & Actuarial Science
Faculty of Science

SIMON FRASER UNIVERSITY
Summer 2020

Approval

Name:	**Jie Wang**
Degree:	**Master of Science (Statistics)**
Title:	**Bayesian Logistic Regression with the Local Bouncy Particle Sampler for COVID-19**
Examining Committee:	**Chair:** X. Joan Hu Professor
	Liangliang Wang Senior Supervisor Associate Professor
	Jiguo Cao Supervisor Professor
	Lloyd T. Elliott Internal Examiner Assistant Professor
Date Defended:	**August 24, 2020**

Abstract

A novel coronavirus, called SARS-CoV-2, has caused the outbreak of the pandemic of COVID-19. The global economy, people's health and life have been facing a tremendous threat in COVID-19. This project is to determine some important factors in COVID-19 severity based on 137 Tianjin patients who have been exposed to COVID-19 since January 5, 2020. We fit a logistic regression model and estimate the parameters using standard Markov chain Monte Carlo (MCMC) methods. Due to the weaknesses and limitations of the standard MCMC methods, we then perform model estimation in one special example of a Piecewise Deterministic Markov Process, named the Bouncy Particle Sampler (BPS). This method is also known as a rejection-free and irreversible MCMC, and can draw samples from our target distribution efficiently. One type of the BPS algorithm, the Local Bouncy Particle Sampler (LBPS), has advantages in computational efficiency. We apply the standard MCMC method and the LBPS to our dataset. We conclude that age and Wuhan-related exposures (i.e. people who have lived or traveled from Wuhan) are two important factors in a COVID-19 severity test.

Keywords: COVID-19; Logistic regression model; Markov chain Monte Carlo; Bouncy particle sampler; Local bouncy particle sampler.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my senior supervisor Dr. Liangliang Wang for the continuous support of my study and research in the past 2 years. I will always remember her patience throughout my graduate study. Our weekly reading group meetings and individual meetings were always the time I learned the most. Without her encouragement, I could not have completed this project.

I would also like to thank my thesis committees: Dr. X. Joan Hu, Dr. Jiguo Cao, and Dr. Lloyd T. Elliott for spending their time on participating and providing valuable comments in my defense. Moreover, I would like to thank all the professors and staff who taught and helped me during my master's study: Dr. Richard Lockhart, Dr. Derek Bingham, Dr. Rachel Altman, Dr. Jinko Graham, Dr. Tom Loughin, Dr. Jiguo Cao, Ian Bercovitz, Charlene Bradbury, and Jay-TempYoung. Besides, I would like to thank Dr. Caroline Calijin et al for organizing the EpiCoronaHack workshop and helping us to understand the data.

Furthermore, I would like to thank all my fellow graduate students in the Statistics Department. Special thanks go to Matthew Berkowitz for proofreading my course projects and thesis; also Dr. Yuping Yang, Haoxuan Zhou, and Boyi Hu, for cheering me up whenever I felt depressed. I am also grateful to have the opportunity to work with Dr. Liangliang Wang, Dr. Jiguo Cao, Dr. Farouk S. Nathoo, Eugene Opoku, and Sidi Wu on the paper "Spectral Dynamic Causal Modelling of Resting-State fMRI: An Exploratory Study Relating Effective Brain Connectivity in the Default Mode Network to Genetics".

Last but not least, I would like to thank my parents for unconditional love, understanding, support, and encouragement.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

A novel coronavirus, called SARS-CoV-2, has caused the outbreak of the pandemic of COVID-19. Patients with COVID-19 have suffered strokes and also Respiratory Tract Infection (RTI) symptoms like fever, sore throat, running nose, and fatigue. Also, some COVID-19 reports [13] have mentioned patients that have ben infected with pneumonia, Severe Acute Respiratory Syndrome (SARS), renal failure, or even death in severe cases. In a recent paper, Wu and Jennifer [32] have analyzed the epidemiological characteristics of the COVID-19 outbreak in China, suggesting that `age` is one important factor in COVID-19 severity. They also mention that most COVID-19 cases were diagnosed in Wuhan and that the majority of exposures were related to Wuhan. Hence, `Wuhan-related exposures` is another important factor in COVID-19 severity.

Logistics regression models have a specific interpretation to analyze some potential factors in COVID-19 severity. We tackle the estimation via posterior sampling in Piece-wise Deterministic Markov Process (PDMP). Compared to traditional Markov chain Monte Carlo (MCMC) algorithms, PDMP is rejection-free. This means that there is no need to waste proposal samples. In addition, PDMP belongs to one irreversible Markov chain, while traditional MCMC approaches are reversible with the detailed balance condition [25]. Notably, some theoretical work and numerical simulations, including [4], [27] and [28], have shown that the irreversible Markov chain (i.e., algorithms in the PDMP family) is more efficient than traditional MCMC algorithms in terms of mixing rate and asymptotic variance. So far, there are three different algorithms in the PDMP family: the Bouncy Particle Sampler (BPS), such as [12], the Zig-Zag sampler [8] and [10], and the Boomerang sampler [9]. Wu and Robert [31] have summarized 3 class of dynamics for the PDMP algorithms: the deterministic dynamic, the event occurrence, and the transition dynamic (i.e. the velocity update).

The BPS and Zig-Zag sampler have the same pattern of deterministic dynamic which means they generate a piece-wise deterministic Markov chain. In contrast, the Boomerang sampler was developed by constructing the trajectory along with a piece-wise elliptical path. Moreover, the BPS and Boomerang sampler have indistinguishable velocity updates, while the Zig-Zag sampler updates velocity by some fixed coordinate. Furthermore, all algorithms share the same event occurrence that is generated from an inhomogeneous Poisson Process. To overcome the reducibility problem, both the BPS and Boomerang sampler add one homogeneous Poisson Process (i.e., exponential distribution with positive rate parameter) in generating the event occurrence, whereas that is not required in the Zig-Zag sampler since it has been proved to be ergodic under very mild conditions by [11]. Besides, the Generalized Bouncy Particle Sampler (GBPS) was first proposed by Wu and Robert [30], and can act as a bridge between the Zig-Zag sampler and the BPS. That means the velocity update in the GBPS is comprised of the one in both the BPS and Zig-Zag sampler. Particularly, the algorithm avoids the tuning process (i.e., rate parameter in exponential distribution) like the Zig-Zag sampler and has shown some advantages in sampling from multimodal distributions. In this project, we mainly focus on parameter estimation in the logistic regression model with the BPS and then Local Bouncy Particle Sampler (LBPS) - which can be considered as a natural extension of the BPS with applications to large datasets.

This project is organized as follows. In Chapter 2, we will start with an overview of the logistics regression model. We then describe the techniques for estimating parameters in frequentist and Bayesian frameworks. Particularly, we will discuss the sampling strategies of the BPS and Local BPS with application to large datasets. Next, numerical simulations are conducted to evaluate the accuracy and computational cost in both frequentist and Bayesian frameworks in Chapter 3. In Chapter 4, we describe applications of our sampling approaches to a real-world dataset from COVID-19. Finally, in Chapter 5, we conclude with a discussion of these methods and our analyses, as well as outline potential future work.

Chapter 2

Methodology

2.1 Overview of Methods

In this chapter, we describe parameter estimation in a logistics regression model from two schools of statistical inference: frequentist and Bayesian. In the frequentist approach, the unknown parameters in regression models can be estimated through maximum likelihood estimation (MLE). In the Bayesian framework, we simulate a random sample from the posterior distribution with different sampling mechanisms with the traditional Metropolis–Hastings(MH) algorithm, as well as a Piece-wise Deterministic Markov Process (PDMP) called the Local Bouncy Particle Sampler (LBPS). Finally, we implement both approaches by using some existing `R` functions or our own code to compute the approximated posterior mean and standard deviation (SD).

2.2 Logistic Regression Model

Consider the response y has a binary outcome, $y \in \{0, 1\}$. Each response is associated with p-dimensional predictors, $z = (z_1, z_2, ..., z_p) \in \mathbb{R}^p$. Vector $x = (x_1, ..., x_p) \in \mathbb{R}^p$ are its corresponding coefficients, which are assumed to be unknown. Let $\boldsymbol{Y}$ be a column vector of length n. For $i = 1, ..., n$, each y_i follows an independent Bernoulli distribution with the parameter π_i (i.e., the probability of y being 1 for any given observation in the i-th population).

Consider a binary GLM model, $\Pr(Y_i = y) = \pi_i^y(1 - \pi_i)^{1-y}$. The natural parameter is the canonical link:

$$\log\left(\frac{\pi_i}{1 - \pi_i}\right) = \text{logit}(\pi_i). \tag{2.1}$$

We can use Equation (2.1) as our link function, so we can write the model in regression form:

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \sum_{j=0}^{p} z_{ij}x_j \quad i = 1, 2, \ldots, n. \tag{2.2}$$

After taking the base number e on both sides of the equation, we have

$$\frac{\pi_i}{1-\pi_i} = \exp\left\{\sum_{j=0}^{p} z_{ij}x_j\right\},$$

for $i = 1, ..., n$.

So,

$$\pi_i = \frac{\exp\left\{\sum_{j=0}^{p} z_{ij}x_j\right\}}{1+\exp\left\{\sum_{j=0}^{p} z_{ij}x_j\right\}}. \tag{2.3}$$

Formally, the logistic regression model has the form:

$$\Pr(y_i = 1|\boldsymbol{z}, \boldsymbol{x}) = \frac{1}{1+\exp\left\{-\sum_{j=0}^{p} z_{ij}x_j\right\}}. \tag{2.4}$$

2.3 Parameter Estimation via MLE

We aim to estimate $p+1$ unknown parameters from Equation (2.2). This can be achieved by maximum likelihood estimation (MLE). Outcome vector y_i's follow independent Bernoulli(π_i), which are fixed and known. Given the outcome vector $\boldsymbol{Y}$, the likelihood function for x can be expressed as

$$\mathcal{L}(\boldsymbol{x}|\boldsymbol{Y}) = \prod_{i=1}^{n} \pi_i^{y_i} \left(1-\pi_i\right)^{1-y_i}. \tag{2.5}$$

The log-likelihood function can be derived by taking the natural logarithm of Equation (2.5),

$$l(\boldsymbol{x}|\boldsymbol{Y}) = \sum_{i=1}^{n} y_i \cdot \log\left(\frac{\pi_i}{1-\pi_i}\right) + \log(1-\pi_i). \tag{2.6}$$

Plugging Equation (2.3) into Equation (2.6), the log-likelihood function is thus simplified into

$$l(\boldsymbol{x}|\boldsymbol{Y}) = \sum_{i=1}^{n}\left[y_i \cdot \sum_{j=0}^{p} z_{ij}x_j - \log\left(1+\exp\left\{\sum_{j=0}^{p} z_{ij}x_j\right\}\right)\right]. \tag{2.7}$$

Therefore, the global maximizer $\hat{\boldsymbol{x}}$ in Equation (2.7) would be the MLE for $\boldsymbol{x}$. In practice, we derive the MLEs via the Fisher scoring method. To be more specific, it replaces the second derivatives in the Newton Raphson method with the Fisher Information matrix. For example, if we want to find the MLE for $\boldsymbol{x}$, this algorithm can be summarized as follows:

Algorithm 1 Fisher Scoring Algorithm

1: Initialize $\boldsymbol{x}^{(0)} = \widetilde{\boldsymbol{x}}$.
2: **for all** n = 1, 2, ..., N **do**
3: (a) Update the estimates:

$$\boldsymbol{x}^{(n)} = \boldsymbol{x}^{(n-1)} - \frac{f\left(\boldsymbol{x}^{(n-1)}\right)}{f'\left(\boldsymbol{x}^{(0)}\right)}. \tag{2.8}$$

4: (b) Compute its error:

$$\epsilon^{(n)} = \boldsymbol{x}^{(n)} - \boldsymbol{x}^{(n-1)}. \tag{2.9}$$

5: **if** $\epsilon^{(n)} = 0$ **then** exit the For loop in Line 2.
6: **end if**
7: **end for**

We first set the initial value to be its method of moments (MOM) estimator $\widetilde{\boldsymbol{x}}$, and compute the Fisher Information matrix evaluated at its MOM that is $f'(\widetilde{\boldsymbol{x}})$. The second step involves updating the estimates obtained from the previous step, and computing its corresponding error $\epsilon^{(n)}$. To acquire its MLE, we will iterate these two steps as many times as it is specified, or until a convergence criterion is met, such as when the error function in Equation (2.9) is equal to 0 or less than a specified threshold.

2.4 Parameter Estimation in Bayesian Inference

In Bayesian framework, we are given the data denoted

$$\texttt{Data} = \{(Z_1, y_1), (Z_2, y_2), ..., (Z_n, y_n)\}$$

as well as its likelihood function, which we presented in Equation (2.5). Our goal is to generate random samples from a posterior distribution, which is the distribution over unknown parameters given the data and prior. In addition, it encodes the uncertainly around the parameters in the model. This can be computed by the posterior distribution of x given the data using "Bayes rule", that is,

$$\pi(\boldsymbol{x}|\texttt{Data}) \propto \pi(\boldsymbol{x}) \cdot \mathcal{L}(\boldsymbol{x}|\texttt{Data}), \tag{2.10}$$

where $\pi(x)$ is the posterior distribution, which describes our subjective beliefs about the unknown parameters $\boldsymbol{x}$. In practice, one commonly used prior is that each x follows an independent standard normal distribution.

Next, we compute the posterior distribution point-wisely up to a normalizing constant. In particular, we are interested in the components that can summarize the posterior distribution, such as posterior mean and variance. And finally, we need to compute the expected value of any arbitrary function $\varphi(x)$ with respect to our target distribution π:

$$E\left(\varphi(x)\right) = \int \varphi(x)\pi(x)\ dx. \tag{2.11}$$

However, it is difficult to solve the integration analytically. Thus, we need to think about solving the integrals by using sampling strategies that generate random samples from the proposal distribution in Markov chain Monte Carlo (MCMC). We can then use the random samples to compute the approximate posterior mean and variance. In the next two sections, we will demonstrate one standard MCMC approach via the Metropolis-Hastings (MH) algorithm and one rejection-free MCMC method via the BPS.

2.4.1 Posterior Inference via Markov Chain Monte Carlo (MCMC)

In this section and Section 2.6, we denote that $X_{1:n}^{(i)}$ be a sample of a Markov chain as it moves from 1 to n, and there are N random samples in total. We also know the target density of each particle only up to a normalizing constant, given by

$$\pi(x_{1:n}) = \frac{\gamma(x_{1:n})}{Z}, \tag{2.12}$$

where $\gamma(x_{1:n})$ is the target density up to a normalizing constant, and Z is unknown. We then average over dependent particles from one Markov chain to evaluate the approximated moments of the posterior distribution, while the Monte Carlo integration approach has been averaging over independent particles to get an estimator (i.e., posterior mean or variance) that is sampled from either one important distribution or the target distribution.

The Metropolis-Hastings (MH) algorithm is commonly used, which was first introduced by Metropolis et al [22]. In 1970, Hastings [18] has developed a more general version of the algorithm by modifying the acceptance ratio. The MH algorithm produces the chain so that it has the target distribution $\pi(x)$ as a stationary distribution where our Markov chain $\{X_t\}_{t=1,..,n}$ will eventually converge to. Let K denote a transition kernel, which describes the random move to a new candidate value in state y given the current position x. In other words, K is a distribution on y given x, denoted by $K(y|x)$. Sometimes, it is called the proposal distribution.

This algorithm performs an accept-or-reject mechanism that is telling us whether a generated particle would be retained or not with the acceptance probability $\alpha(y|x)$. This can be computed by:

$$\alpha(y|x) = \min\left\{1, \frac{\pi(y)K(x|y)}{\pi(x)K(y|x)}\right\}. \tag{2.13}$$

Furthermore, we can choose any arbitrary initial value at state x_0. The proposal distribution can also be selected as long as it meets the requirement that the proposed Markov chain is irreducible. Finally, the algorithm is summarized as follows:

Algorithm 2 Metropolis-Hastings Algorithm

1: Initialize the chain at state x_0.
2: **for all** t = 1, 2, ..., N **do**
3: (a) Generate a new proposal, $x^* \sim K(y|x_{t-1})$.
4: (b) Calculate the acceptance probability, $\alpha(y|x_t)$.
5: (c) Simulate a random variable, $u \sim \text{Uniform}(0, 1)$.
6: **if** $u \leq \alpha(y|x)$ **then**
7: Move to the new state x^* with the above probability.
8: **else**
9: Remain in the current state, $x_t = x_{t-1}$.
10: **end if**
11: **end for**

2.4.2 Posterior Inference via Bouncy Particle Samplers (BPS)

Most MCMC methods that work on discrete-time reversible Markov processes include two major steps: generating a new point based on a proposal distribution from which it is easy to sample, and then performing the accept/reject mechanism according to the acceptance probability to update the transition kernels. In 2018, [12] explored a technique called the BPS that was first proposed by [23]. This technique has been used in physical sciences and data science. It has become more popular in designing new MCMC methods that is particularly efficient in large datasets in recent years. This technique is also known as a nonreversible rejection-free MCMC method which can continuously track an exponentially-distributed travel time, and then propagate to another state. Note that the rate of exponential distribution is dependent on the current state.

To begin with, we sample from a target distribution, such that,

$$\pi(x) = \exp\{-U(x)\}, \tag{2.14}$$

where $U(x)$ is its associate energy function, given by

$$U(x) = -\log(\pi(x)). \tag{2.15}$$

We also assume $U(x)$ is continuously differentiable. Then, the gradient of $U(x)$ evaluated at each x is denoted by $\nabla U(x) = \left(\frac{\partial U(x)}{\partial x_1}, \ldots, \frac{\partial U(x)}{\partial x_p}\right)'$.

The BPS algorithms generate piece-wise linear trajectories through space $\mathbb{R}^p$. For i-th point $x^{(i)} \in \mathbb{R}^p$ in the space, it has two additional elements: a time of travel $\tau_i \in \mathbb{R}^+$ and a velocity $v^{(i)} \in \mathbb{R}^p$. Specifically, the velocity explains the transition path, such as which direction should the point go next, as well as its speed. The time of travel demonstrates the length between two events. In addition, we denote t_i be the total travel time at position i (i.e., $t_i = \sum_{j=1}^{i} \tau_j$ for $i > 1$) and set $t_0 = 0$ for convenience purposes. Furthermore, we denote the total trajectory length (or total time of travel) by T.

The second step consists of all the components (i.e., position, velocity, and total time of travel) are updated by the following scenarios:

(a) Position update: the position $x^{(i)}$ at time $t \in [t_{i-1}, t_i)$ is updated by

$$x^{(i)} \leftarrow x^{(i-1)} + v^{(i-1)}\tau_i,$$

where τ_i is the total travel length between t_{i-1} and t_i, which takes the minimal between τ_{bounce} and τ_{ref}. More specifically, the bounce event occurrence τ_{bounce} can be simulated from the first arrival times of the inhomogeneous Poisson Processes (PP) with rate function $\chi(t)$,

$$\begin{aligned} \chi(t) &= \lambda(x^{(i-1)} + v^{(i-1)}t, v^{(i-1)}) \\ &= \max\left\{0, \left[\nabla U(x^{(i-1)} + v^{(i-1)}t)\right]' v^{(i-1)}\right\}, \end{aligned} \tag{2.16}$$

suggested by [6, 12, 20].

In order to overcome reducibility problems, [12] added one homogeneous PP into the BPS algorithm. That is the refreshment event occurrence τ^{ref} can be generated from an exponential distribution with rate = λ^{ref}, which is a tuning parameter in this algorithm. Moreover, [12] proved that the transition kernel (i.e., velocity updates) of the BPS allows the target distribution to preserve the correct stationary distribution for $\lambda_{\text{ref}} > 0$.

(b) Velocity update: The velocity is reformed depending on which τ is being selected in the "position update" step. When the bounce event occurs (i.e., $\tau = \tau_{\text{bounce}}$), the

velocity is updated by

$$\begin{aligned} v^{(i)} &= R\left(x^{(i)}\right) v^{(i-1)} \\ &= v^{(i-1)} - 2 \cdot \frac{\left[\nabla U\left(x^{(i-1)}\right)\right]' v^{(i-1)}}{||\nabla U(x^{(i-1)})||^2} \nabla U(x^{(i-1)}), \end{aligned} \tag{2.17}$$

where $||\cdot||$ is the Euclidean norm. Otherwise, the velocity is updated from the standard normal distribution.

(c) The total time of travel is updated by adding the updated τ in `(a)`. Note that it will terminate the for loop if the threshold $t_i = \sum_{k=1}^{i} \tau_k \geq T$ is met.

The global BPS algorithm is thus summarized below,

Algorithm 3 Global Bouncy Particle Samplers (BPS) Algorithm

Initialization step:

i. Arbitrarily initialize $\left(x^{(0)}, v^{(0)}\right)$ on $\mathbb{R}^p \times \mathbb{R}^p$.

ii. Let $T = 0$.

for all $i = 1, 2, ..., N$ **do**

τ simulation:

i. Simulate the first arrival time τ_{bounce} according to Equation (2.16).

ii. Simulate

$$\tau_{\text{ref}} \sim \text{Exponential}\left(\text{rate} = \lambda^{\text{ref}}\right).$$

iii. Set

$$\tau_i \leftarrow \min\{\tau_{\text{bounce}}, \tau_{\text{ref}}\}.$$

(a) Position update:

$$\text{Set } x^{(i)} \leftarrow x^{(i-1)} + v^{(i-1)}\tau_i.$$

(b) Velocity update:

if $\tau_i = \tau_{\text{ref}}$ **then**

Simulate $v^{(i)} \sim \mathcal{N}(0_p, \mathbf{I}_p)$.

else

Set $v^{(i)} \leftarrow R\left(x^{(i)}\right) v^{(i-1)}$ in Equation (2.17).

end if

(c)Total travel time update: Set $T \leftarrow T + \tau_i$.

return $\left\{x^{(i)}, v^{(i)}, T\right\}$.

end for

Figure 2.1 illustrates one example of the global BPS sampler to a Gaussian distribution with mean $= (1, 1)'$ and covariance matrix $I_2 = \text{diag}(1, 1)$. The left plot shows the trajectories of the global BPS sampler after two iterations, and the right plot shows the trajectories after 1,000 iterations. In addition, Figure 2.2 displays the trace and density plots of the X-axis and Y-axis. Note that the red line represents the true mean. Based on the plots, we can see that the chain converges to a stationary distribution after 1,000 iterations, and the posterior means are very close to their true values.

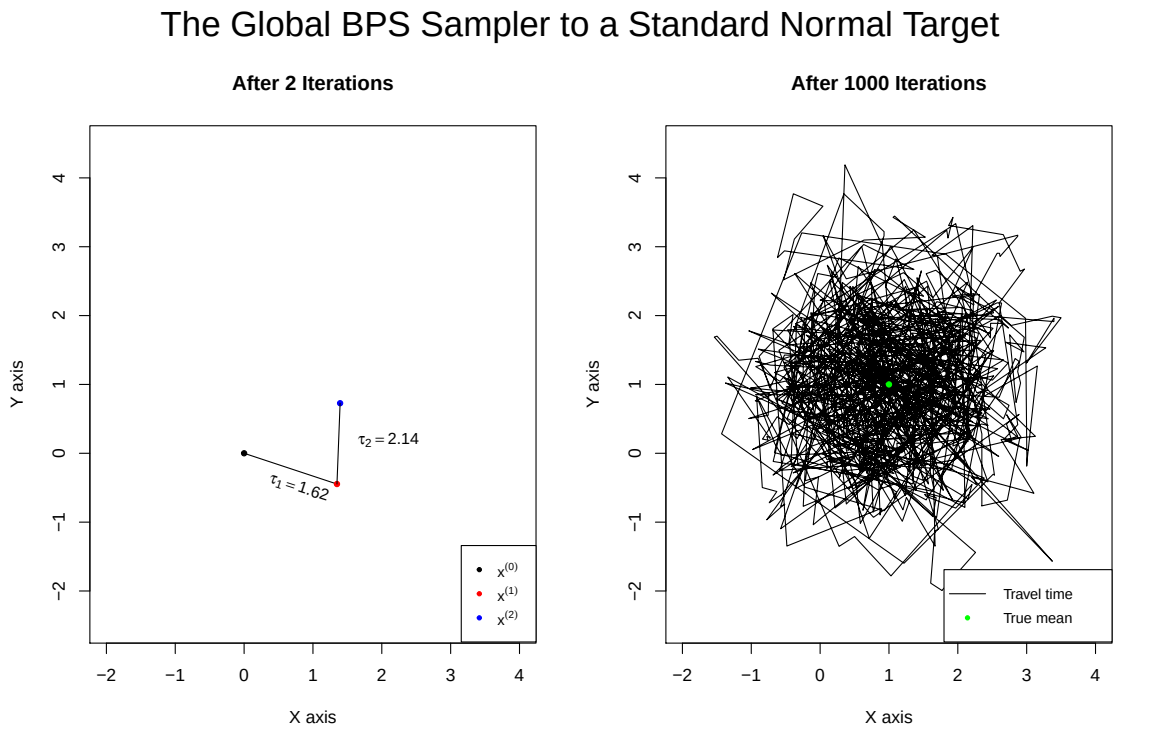

Figure 2.1: The Global BPS Sampler to a Standard Normal Target Distribution. The left plot shows the trajectories of BPS sampler after 2 steps. The right plot shows the trajectories of the PS sampler after 1,000 steps. $x^{(0)} = [0, 0]'$ (the black dot) is the start point with $[0, 0]'$ velocity. $x^{(1)}$ (the red dot) is the first point after velocity update. Likewise, $x^{(2)}$ is the second point after velocity update. All the solid black line segments describe the total travel length after each bounce/refreshment event. The true mean (on the right plot) is pointed by the light green.

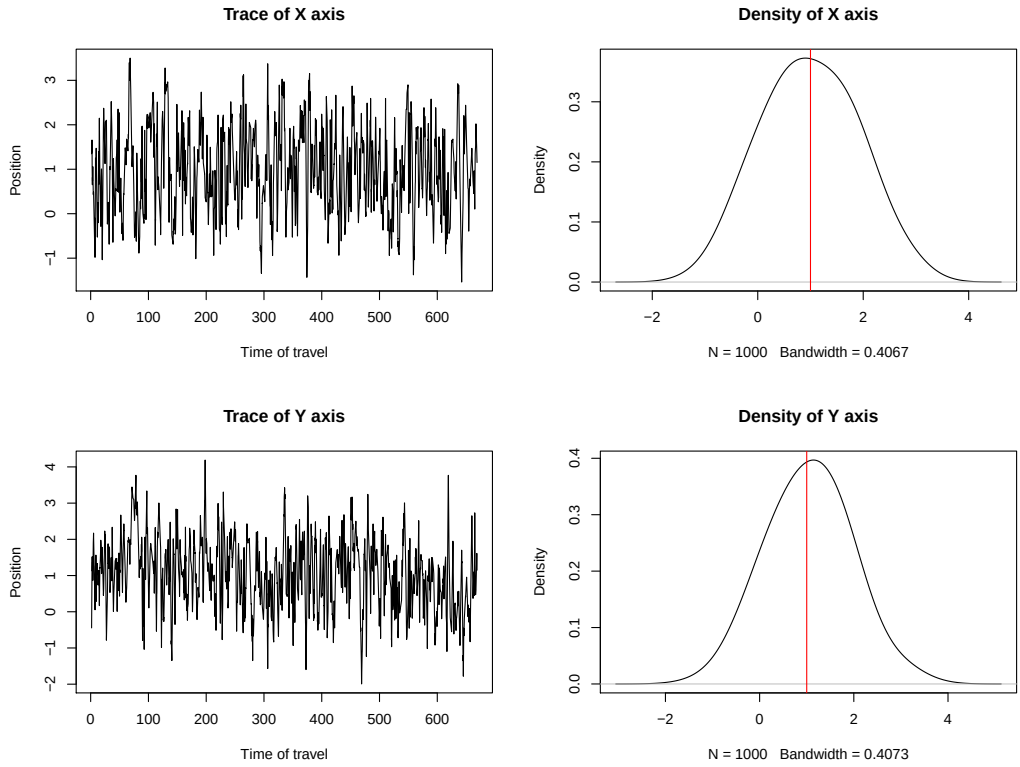

Figure 2.2: Trace and Density plots of Global BPS algorithm for X-axis and Y-axis. Note that the red solid line on the right plot indicates the true mean $= [1, 1]'$ for X-axis and Y-axis.

Ultimately, we need to calculate the expected value of any arbitrary function $\varphi(x)$ with respect to our target distribution π. In this case, we are given a set of $x^{(i)}$ associated with its velocity $v^{(i)}$ for $i = 1, ..., n$ over the interval $[0, T]$. Then,

$$\begin{aligned} E\left(\varphi(x)\right) &\approx \frac{1}{T}\int_0^T \varphi(x(t))\pi(x)\, dt \\ &= \frac{1}{T}\left[\sum_{i=1}^{n-1}\int_0^{\tau_i} \varphi\left(x^{(i-1)} + v^{(i-1)}t\right) dt + \int_0^{t_n - T} \varphi\left(x^{(n-1)} + v^{(n-1)}t\right) dt\right]. \end{aligned} \tag{2.18}$$

When $\varphi(x) = x_j^{(i)}$ for $i = 1, ..., n$, $j = 1, ...p$ then the first moment of $x_j^{(i)}$ is given by,

$$\begin{aligned} E\left(x_j^{(i)}\right) &= \int_0^{\tau_i}\left[x_j^{(i-1)} + v_j^{(i-1)}t\right] dt \\ &= x_j^{(i-1)}\tau_i + \frac{1}{2}v_j^{(i-1)}\tau_i^2, \end{aligned} \tag{2.19}$$

the second moment of $x_j^{(i)}$ is given by,

$$E\left[\left(x_j^{(i)}\right)^2\right] = \int_0^{\tau_i} \left[x_j^{(i-1)} + v_j^{(i-1)}t\right]^2 dt$$
$$= \left[x_j^{(i-1)}\right]^2 \tau_i + x_j^{(i-1)} v_j^{(i-1)} \tau_i^2 + \frac{1}{3}\left[v_j^{(i-1)}\right]^3 \tau_i^3. \tag{2.20}$$

So the standard deviation of $x_j^{(i)}$ can be calculated according to Equation (2.19) and Equation (2.20).

However if Equation(2.18) is intractable, we may need to borrow the idea from linear interpolation; that is, we select some new data points within the range of each line segment. For example, let b be the data points selected from the trajectory. Then an estimator can be constructed as

$$\frac{1}{b}\sum_{l=0}^{b-1} \varphi\left[x\left(\frac{l \cdot T}{b-1}\right)\right]. \tag{2.21}$$

Likewise, the second moment needs to be approximated by numerical quadrature.

After N BPS iterations, we can perform Geweke's diagnostic [17] to evaluate the accuracy of the BPS using the posterior means and SD. Similar to standard MCMC samplers, Geweke's diagnostic is used to determine the burn-in period.

Example 1. Gaussian Distribution

Consider the target distribution is a p-dimensional multivariate Gaussian with 0 means and variance-covariance matrix Σ. The probability density function is thus:

$$\pi(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}x'\Sigma^{-1}x\right\}$$
$$\propto \exp\left\{-\frac{1}{2}x'\Sigma^{-1}x\right\}.$$

The associated energy function is given by $U(x) = \frac{1}{2}x'\Sigma^{-1}x$, and the gradient of the $U(x)$ evaluated at x is $\nabla U(x) = x'\Sigma^{-1}$. We aim to solve the $\tau_\star$ such that

$$\tau_\star = \text{argmin}_{t:t\geq 0} U(x + vt).$$

So, the closed-form for τ_* is

$$\tau_* = (v'v)^{-1}\left[-x'v + \sqrt{(x'v)^2 - 2v'v\log(V)}\right], \tag{2.22}$$

which is given in [12] and [15], where V is the ratio between $\pi(x + v\tau)$ and $\pi(x + v\tau_*)$.

2.4.3 Posterior Inference via Local Bouncy Particle Sampler (LBPS)

The Local Bouncy Particle Sampler (LBPS) is known as one natural extension of the global BPS, which combines the global BPS with sub-sampling strategies. First, this requires the target density to decompose into a representation of the form:

$$\pi(x) = \prod_{f \in F} \pi_f \left(x_f \right), \tag{2.23}$$

where x_f denotes some subset of the variables x and F is an index set called the set of factors. Hence, the energy function can be expressed as

$$U(x) = U^{(0)}(x) + \underbrace{\sum_{i=1}^{n} U^{(i)}(x)}_{(\star)}. \tag{2.24}$$

We also have $\partial U_f(x)/\partial x_k = 0$ for $k \in \{1, 2, ..., p\} \backslash N_f$ that is the variables exclude from factor f. Second, we need to uniformly select one factor f at random, and then uniformly pick one observation within this factor. Using that observation to update the weight component. Specifically, the sub-sampling strategies contain two parts: first, we construct one joint probability mass function, $q_{i,j}(i, j)$ that contains both observations and covariates; second, we sample j from its marginal distribution $q_j(\cdot)$; and finally we select observation i based on the conditional distribution $q_{i|j}(\cdot|j)$. Example 2 demonstrates how to sample particles with the local BPS in logistic regression analysis.

Example 2. Logistics Regression via LBPS

We use the same settings of data in Section 2.2. We further assume the unknown parameters $x \in \mathbb{R}^p$ follow a standard Gaussian prior, denoted by $\psi(x)$. Then the posterior distribution can be derived by,

$$\begin{aligned} \pi(x) &\propto \psi(x) \cdot L(x) \\ &\propto \prod_{j=1}^{p} \exp\left[-\frac{(x_j)^2}{2} \right] \cdot \prod_{i=1}^{n} \frac{\exp\left\{ y_i \cdot \sum_{j=0}^{p} z_{ij} x_j \right\}}{1 + \exp\left\{ \sum_{j=0}^{p} z_{ij} x_j \right\}}. \end{aligned} \tag{2.25}$$

In this case, the energy function in Equation (2.24) can be decomposed into two parts: $U^{(0)}(x)$ which comes from the prior and $U^{(i)}(x)$ (where $i = 1, ..., n$), which comes from the likelihood function. Specifically, $U^{(0)}(x)$ has the following form:

$$U^{(0)}(x) = \sum_{j=1}^{p} \left[\frac{(x_j^2)}{2} \right] + \text{constant}, \tag{2.26}$$

and $(\star)$ in Equation (2.24) is formulated by,

$$\sum_{i=1}^{n} U^{(i)}(x) = \sum_{i=1}^{n} \left[\log\left(1 + \exp\left\{\sum_{j=0}^{p} z_{ij}x_j\right\}\right) - y_i\left(\sum_{j=0}^{p} z_{ij}x_j\right)\right]. \tag{2.27}$$

So the gradient of Equation (2.26) is

$$\nabla U^{(0)}(x) = \frac{\partial U^{(0)}(x)}{\partial x_j} = x_j \leq \|x\|, \tag{2.28}$$

and the gradient of Equation (2.27) for each component i can be derived as:

$$\begin{aligned}
\nabla U^{(i)}(x) = \frac{\partial U^{(i)}(x)}{\partial x_j} &= \sum_{i=1}^{n}\left[\frac{z_{ij}\cdot\exp\left\{\sum_{j=0}^{p} z_{ij}x_j\right\}}{1+\exp\left\{\sum_{j=0}^{p} z_{ij}x_j\right\}} - y_i z_{ij}\right] \\
&= \sum_{i=1}^{n}\left[z_{ij}\cdot\underbrace{\text{logit}\left(\sum_{j=0}^{p} z_{ij}x_j\right)}_{(\star\star)} - y_i z_{ij}\right].
\end{aligned} \tag{2.29}$$

We notice that $0 < (\star\star)$ above < 1 for $\sum_{j=0}^{p} z_{ij}x_j \in \mathbb{R}$.

Moreover, simulation of τ can be performed according to the following scenarios:

- Simulation of $\tau^{(0)}$ can be approached using Example 1 above;
- Simulation of $\tau^{(i)}$ for $i = 1, ..., n$ can be applied to the thinning algorithm in PP [20].

Now we need to find the upper bound for the rate function described in (2.16) on its domain. The derivation procedure is shown below:

$$\begin{aligned}
\chi(t) &\leq \sum_{j=1}^{p} \chi^{(i)}(t) \\
&= \max\left\{0, \left[\nabla U^{(i)}(x+vt)\right]' v\right\} \\
&= \max\left\{0, \left[\sum_{i=1}^{n}\left[z_{ij}\cdot\text{logit}\left(\sum_{j=0}^{p} z_{ij}(x_j+v_j t)\right) - y_i z_{ij}\right]' v_i\right]\right\} \\
&= \max\left\{0, \left[\sum_{i=1}^{n}\left[\text{logit}\left(\sum_{j=0}^{p} z_{ij}(x_j+v_j t)\right) - y_i\right]' z'_{ij} v_i\right]\right\}.
\end{aligned} \tag{2.30}$$

When $y_i = 0$, Equation (2.30) becomes

$$\begin{aligned}\chi^{(i)}(t) &= \max\left\{0, \left[\sum_{i=1}^{n}\left[\text{logit}\left(\sum_{j=0}^{p} z_{ij}(x_j + v_j t)\right) - y_i\right]' z'_{ij} v_i\right]\right\} \\ &\leq \sum_{j=1}^{p} z'_{ij}\left[v_i \cdot \mathbb{I}\left\{v_i > 0\right\}\right] = \overline{\chi}^{(i)}, \end{aligned} \tag{2.31}$$

where $\mathbb{I}\left\{v_i > 0\right\}$ is an indicator function which equals to 1 when $v_i > 0$ for $i = 1, .., n$ and 0 otherwise, $\overline{\chi}^{(i)}$ is the upper bounds based on the i-th datapoint, and z_{ij} is assumed to be non-negative. When $y_i = 1$, likewise, Equation (2.30) becomes

$$\chi^{(i)}(t) \leq \sum_{j=1}^{p} z'_{ij}\left[|v_i| \cdot \mathbb{I}\left\{v_i < 0\right\}\right] = \overline{\chi}^{(i)}. \tag{2.32}$$

However it is possible to have negative covariates; that is, $z_{ij} < 0$. Thus, we can further expand Equation (2.31) into

$$\begin{aligned}\chi^{(i)}(t) &= \sum_{i=1}^{n} \max\left\{0, \left[\text{logit}\left(\sum_{j=0}^{p} z_{ij}(x_j + v_j t)\right) - y_i\right]' z'_{ij} v_i\right\} \\ &\quad + \min\left\{-\left[\text{logit}\left(\sum_{j=0}^{p} z_{ij}(x_j + v_j t)\right) - y_i\right]' z'_{ij} v_i, 0\right\} \\ &\leq \sum_{j=1}^{p} z'_{ij}\left[v_i \cdot \mathbb{I}\left\{v_i > 0\right\}\right] = \overline{\chi}^{(i)}. \end{aligned} \tag{2.33}$$

Therefore, the upper bound of $\sum_{i=1}^{n} \chi^{(i)}(t)$ is

$$\overline{\chi} = \sum_{j=1}^{p} |v_j| \sum_{i=1}^{n} \mathbf{1}\left[v_j(-1)^{y_i} \geq 0\right] z_{i,j}, \tag{2.34}$$

which is a constant and only depends on the velocity.

In order to exploit the sub-sampling, we can manipulate the joint probability mass function which contains both data points and covariates. That is,

$$q_{i,j}(i,j) = \frac{|v_j| \mathbf{1}\left[v_j(-1)^{y_i} \geq 0\right] z_{i,j}}{\overline{\chi}}. \tag{2.35}$$

The velocity component is updated by sub-samplings and an adaptive thinning algorithm. To do so, we first sample

$$j \sim q_j(\cdot) = \frac{\sum_{i=1}^{n} |v_j| \, \mathbf{1}\left[v_j(-1)^{y_i} \geq 0\right] z_{i,j}}{\overline{\chi}},$$

and then we select

$$i \sim q_{i|j}(\cdot|j). \tag{2.36}$$

In other words, we uniformly select the observation i at random according to Equation (2.35) based on the j. According to the thinning algorithm, the velocity component will only be updated when the ratio $\frac{\overline{\chi}^{(i)}}{\overline{\chi}} < V$, where i is given by Equation (2.36), $V \sim \text{Uniform}(0, 1)$ and the velocity updated can be referred to Equation (2.17). Alternatively, we may consider to apply the naive sub-sampling method. That is

$$\overline{\chi}^{(i)} = \max_i |z_{ij}|.$$

Finally, the local BPS algorithm is summarized below.

2.5 Implementation

We implemented the methods for parameter estimation in `R`. We used the `glm` function to fit the logistic regression model to the COVID-19 Tianjin dataset.

In the Bayesian framework, there are some available packages for the MH method, such as `MCMCpack::MCMClogit` [21]. For the global BPS algorithm, various functions from the `RZigZag` [5] package (i.e. `BPSGaussian`, `BPSIIDGaussian`, and `BPSStudentT`) are able to implement the global BPS sampler for a Gaussian target distribution or Student `T` distribution. The ESS can be computed by the `R` functions `RZigZag::effectiveSize` or `mcmcse::ess` [14]. The Geweke's diagnostic can be implemented via `coda::geweke.diag` [24]. Moreover, Galbraith [15] wrote his own Python script to use the global BPS as well as the LBPS for various settings.

However, we cannot find any existing `R` functions to fulfill the logistic regression model with the LBPS. Thus, we write our own `R` script to implement the LBPS algorithm, which is available in Appendix C.

Algorithm 4 Local Bouncy Particle Samplers (LBPS) Algorithm

1: Initialization step:

2: (a) Arbitrarily select $\left(x^{(0)}, v^{(0)}\right)$ on $\mathbb{R}^p \times \mathbb{R}^p$.

3: (b) Set global travel time $T = 0$.

4: (c) Set local time upper-bounds

$$\overline{T} \leftarrow \Delta.$$

5: (d) Calculate local-in-time upper bound on the prior factor

$$\overline{\chi}_{\text{prior}} = \frac{1}{\sigma^2} \max\left\{0, \left(x^{(0)} + \Delta v^{(0)}\right)' v^{(0)}\right\}.$$

6: **for all** $i = 1, 2, ..., n$ **do**

7: (a) Calculate local-in-time upper bound on the data factors $\overline{\chi}$ in Equation (2.34),

8: (b) Simulate τ such that

$$\tau \sim \text{Expoential}\left(\text{rate} = \overline{\chi}_{\text{prior}} + \overline{\chi} + \lambda^{\text{ref}}\right).$$

9: **if** $T + \tau > \overline{T}$ **then**

10: i. Position update:

$$\text{Set } x^{(i)} \leftarrow x^{(i-1)} + v^{(i-1)}(\overline{T} - T).$$

11: ii. Velocity update:

$$\text{Set } v^{(i)} \leftarrow v^{(i-1)}.$$

12: iii. Local-in-time upper bound on the prior factor update:

$$\overline{\chi}_{\text{prior}} = \frac{1}{\sigma^2} \max\left\{0, \left(x^{(i)} + \Delta v^{(i)}\right)' v^{(i)}\right\}.$$

13: iv. Travel time update:

$$T \leftarrow \overline{T},$$
$$\overline{T} \leftarrow \overline{T} + \Delta.$$

14: **else**

15: i. Position update:

$$x^{(i)} \leftarrow x^{(i-1)} + v^{(i-1)}\tau.$$

16: ii. Velocity update: generate a random integer k from

$$\text{Discrete}\left(\frac{\bar{\chi}}{\bar{\chi}_{\text{prior}} + \bar{\chi} + \lambda^{\text{ref}}}, \frac{\lambda^{\text{ref}}}{\bar{\chi}_{\text{prior}} + \bar{\chi} + \lambda^{\text{ref}}}, \frac{\bar{\chi}_{\text{prior}}}{\bar{\chi}_{\text{prior}} + \bar{\chi} + \lambda^{\text{ref}}}\right).$$

if k=1 **then**

i. Sample j according to $q_j(\cdot)$,

ii. Generate i based on the conditional distribution $q_{i|j}(\cdot|j)$,

iii. Generate $V \sim \text{Uniform}(0,1)$.

if $V < \dfrac{\max\left\{0, \left[\nabla^{[i]}(x^{(i)})\right]' v^{(i-1)}\right\}}{\overline{\chi}^{[i]}}$ **then**

$v^{(i)}$ is updated by Equation (2.17).

else

$$\text{Set } v^{(i)} \leftarrow v^{(i-1)}$$

.

end if

end if

if k=2 **then**

$$\text{Simulate } v^{(i)} \sim N(0_p, \boldsymbol{I}_p).$$

end if

if k=3 **then**

i. Let b be the position of τ_* referring to Equation (2.22),

ii. Generate $V \sim \text{Uniform}(0,1)$.

if $V < \dfrac{\max\left\{0, \left[\nabla^{[i]}(x^{(b)})\right]' v^{(i-1)}\right\}}{\overline{\chi}_{\text{prior}}}$ **then**

$v^{(i)}$ is updated by (2.17).

else

$$\text{Set } v^{(i)} \leftarrow v^{(i-1)}.$$

end if

end if

iii. Local-in-time upper bound on the prior factor update:

$$\overline{\chi}_{\text{prior}} = \frac{1}{\sigma^2} \max\left\{0, \left(x^{(i)} + \Delta v^{(i)}\right)' v^{(i)}\right\}.$$

iv. Travel time update:

$$\text{Set } T \leftarrow T + \tau.$$

end if

return $\left\{x^{(i)}, v^{(i)}, T\right\}$.

end for

Chapter 3

Numerical Simulation

In this chapter, we design 2 simulation studies to evaluate the model performance in terms of the computation time and accuracy:

- In the first study, we consider two settings where $p = 2$ and $n \in \{30, 100\}$. We tune the λ^{ref} for the LBPS algorithm, compare it to the BPS with the naive subsampling, and compare the estimation accuracy among the MH, LBPS as well as MLE;
- In the second study, we first consider the setting where $p = 5$, and various values of n, where $n \in \{30, 100, 200, 500\}$. We further discover the setting where we encounter large and sparse datasets. In both experiments, we fix the computation budget, and compare the estimation accuracy between the MH and LBPS.

3.1 Simulation Design

3.1.1 Experiment Design I

Experiment I

We consider a setting where $p = 2$ and $n = 100$. The synthetic data were generated by sampling the covariates: X_1 and X_2 are independent and identically distributed (iid) Uniform(0.1, 1). The responses y_i are sampled from Equation (2.4) with true coefficients. That is,

$$\boldsymbol{x} = [x_0, x_1, x_2]' = [0.5, 1, 1]'.$$

Firstly, we need to tune the rate parameter of the homogeneous PP to stable efficiency and accuracy in the LBPS. The tuning procedure for each variable is as follows:

- We consider the values of $\lambda^{\text{ref}} = (0.01, 0.1, 0.5, 1, 10)$.
- Each LBPS algorithm was executed by using $N = 10,000$ (the number of iterations for the LBPS).
- The initial position $x^{(0)}$ was set to $[-1, 2, 2]$ with velocity vector $[1, -1, -1]$.
- We further assume the posterior distribution given in Equation (2.25) to be a standard normal distribution and set $\Delta = 0.5$.
- We then repeat this process M times. In this case, $M = 20$.
- Finally, we calculate the in-sample mean square error (sMSE) for each value of λ^{ref}. The optimal tuning parameter will be the one with the minimum average sMSE.

Secondly, we compare the LBPS method with the optimal λ^{ref} versus the BPS with naive sub-sampling in terms of the square root sMSE and ESS.

Finally, we compare parameter estimations by the LBPS to those produced by the MH as well as MLE. MLEs were generated by the Fisher scoring algorithm as described in Algorithm 1. We then use $N = 10,000$ (the number of MH/LBPS iterations for the MCMC sampler). For the MH algorithm, we also add a burn-in 1000 iterations to ensure that the MCMC estimates are free of early failures. For both Bayesian approaches, we assume one multivariate prior on $\boldsymbol{x}$ with 0 means and variance-covariance $= \sigma^2 \mathbb{I}_3$. The value of σ^2 is set to be 0.04 for the MH and 1 for the LBPS. Moreover, the initial values for x_0, x_1, x_2 are set to be their corresponding MLE values in the MH algorithm. In the LBPS, the initial position, velocity, and the value of Δ are set the same as we applied in Experiment 1. In addition to that, we used the optimal value of λ^{ref}. Similar to the MH algorithm, we treat the first 10% of samples as burn-in. Overall, we repeat each approach (i.e., MLE, MH, and tuned LBPS (LBPS with optimal value of λ^{ref}) 100 times and calculate the coverage probabilities (CP) of 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 95% confidence/credible interval.

Experiment II

We repeat the same procedures as Simulation Design I except the sample size of the data. In this experiment, we sample X_1 and X_2 with $n = 30$.

3.1.2 Simulation Design II

Experiment I

We consider settings where $p = 5$ and $n = \{30, 100, 200, 500\}$. The synthetic data were generated by sampling the covariates from iid Uniform$(0.1, 1)$. The response y_i are sample from Equation (2.4) with true coefficients $[0.5, 1, 1, 1, 1, 1]'$.

For the MH algorithm, we start with their corresponding MLE values. In addition, we assume one multivariate prior on $\boldsymbol{x}$ with 0 means and variance-covariance $= \sigma^2\mathbb{I}_6$. The value of σ^2 is set to be 0.04. We also add a burn-in 10 % sample to ensure that the MCMC estimates are free of early failures.

For the LBPS algorithm, we start with $x^{(0)} = [-1, 2, 2, 2, 2, 2]$ with the velocity vector

$$v^{(0)} = [1, -1, -1, -1, -1, -1],$$

and set $\lambda^{\text{ref}} = 0.5$. Other variables such as σ^2, Δ were set to be same as Simulation I.

For each $n \in \{30, 100, 200, 500\}$, we run the MH and LBPS algorithms 20 times given by the fixed computation cost (i.e. same iterations for both algorithms), and then compare the estimation accuracy like the last procedure in Simulation I.

Experiment II

We consider a setting where $p = 5$ and $n = 500$. The synthetic data were generated by sampling from the mixture distribution [26]:

$$\nu_\alpha(\mathrm{d}x) = (1 - \alpha)\delta_0(\mathrm{d}x) + \alpha\rho(x)\mathrm{d}x, \tag{3.1}$$

where

- $\delta_0(\mathrm{d}x)$ is a point mass at 0.
- $\alpha \in [0, 1]$ controls the level of sparsity. For example, $\alpha = 0.1$ means 10 % of the data are non-zero.
- ρ can be chosen from any smooth distribution. In this case, we choose $\rho =$ Uniform $(0.1, 1)$.

The responses y_i are sampled from Equation (2.4) with true coefficients. That is,

$$\boldsymbol{x} = [x_0, x_1, x_2]' = [0.5, 1, 1, 1, 1, 1]'.$$

The initial values and other variables (i.e. σ^2, λ^{ref}) for the MH and the LBPS were set to be the same as Simulation III.

In this experiment, we consider 3 types of sparse dataset: `serve` sparse ($\alpha = 0.1$), `intermediate` sparse ($\alpha = 0.5$), and `light` sparse ($\alpha = 0.9$). For each setting, we repeat the MH and LBPS algorithms 20 times given by the fixed computation cost, and then compare the estimation accuracy.

3.2 Simulation Results

3.2.1 Experiment Design I

Experiment I

According to the relative square root sMSE boxplot below (Figure 3.1 and Figure 3.2), we find $\lambda^{\text{ref}} = 0.5$ performs the best in terms of sMSE and ESS for each variable.

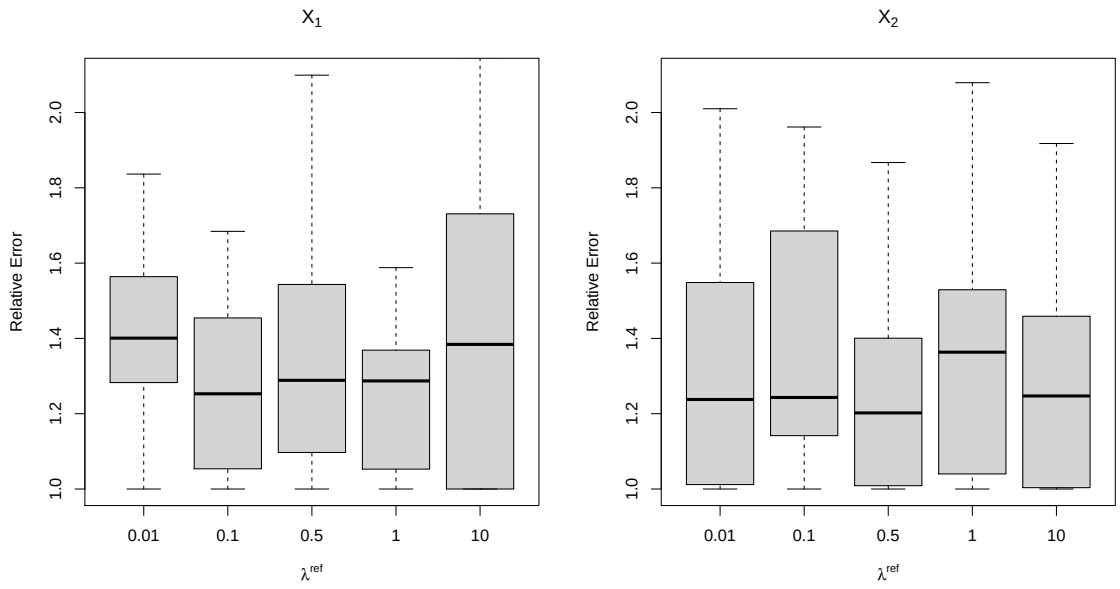

Figure 3.1: Boxplot of relative square-root sMSE for the LBPS with different values of $\lambda^{\text{ref}} = (0.01, 0.1, 0.5, 1, 10)$ resulting from the synthetic data in Simulation Design I. For each run, all sMSE are divided by the smallest sMSE produced; values of 1 means that the LBPS with the specified λ^{ref} produced the lowest sMSE.

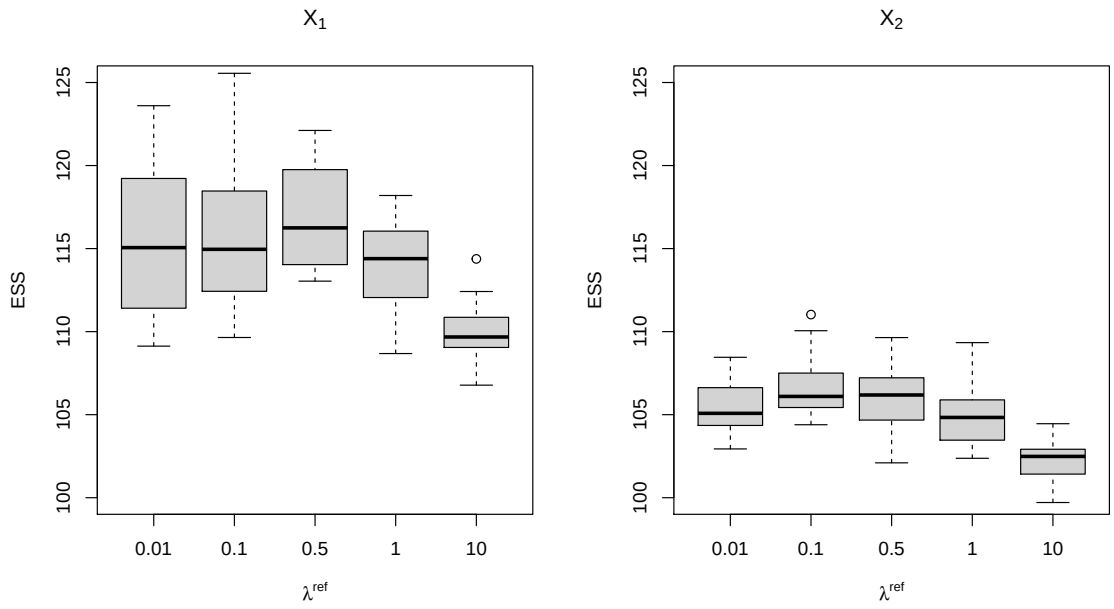

Figure 3.2: Boxplot of ESS for the LBPS with different values of $\lambda^{\text{ref}} = (0.01, 0.1, 0.5, 1, 10)$ resulting from the synthetic data in Simulation Design I.

We further compare relative square-root sMSE between the LBPS with $\lambda^{\text{ref}} = 0.5$ and the BPS with naive sub-sampling in terms of sMSE and ESS. According to Figure 3.3, the LBPS with $\lambda^{\text{ref}} = 0.5$ outperforms the other method with the lower variance and sMSE. Figure 3.4 displays the ESS of X_1 and X_2 for both methods, the LBPS with $\lambda^{\text{ref}} = 0.5$ has a larger ESS than that for the BPS with sub-sampling. Therefore, the LBPS with a tuned rate parameter outperforms in this study.

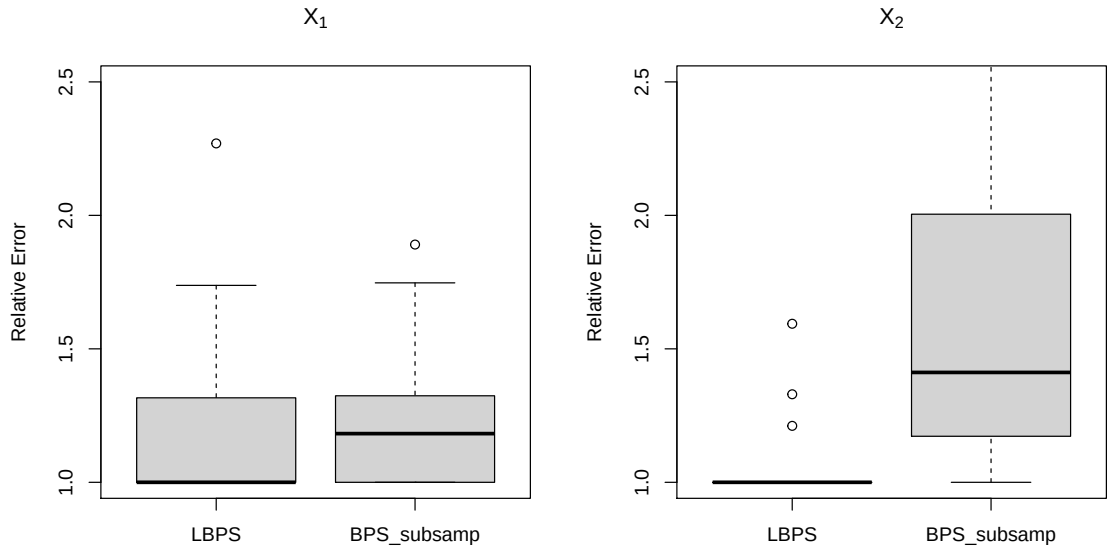

Figure 3.3: Comparison of relative square-root sMSE and the LBPS with $\lambda^{\text{ref}} = 0.5$ and the BPS with naive sub-sampling resulting from the synthetic data in Simulation Design I. For each run, each sMSE is divided by the smallest sMSE produced; values of 1 means that the LBPS with that λ^{ref} produced the lowest sMSE.

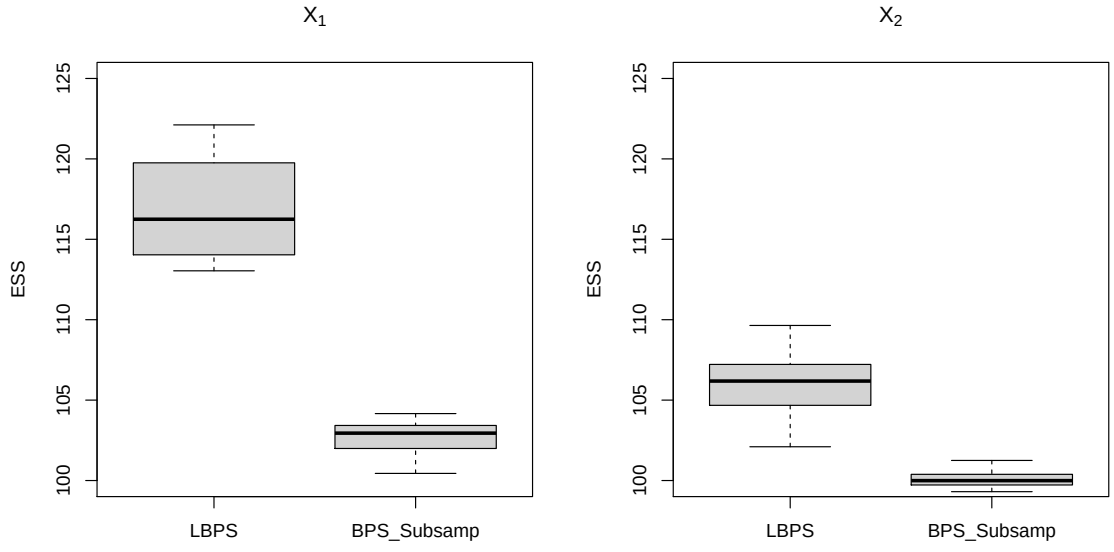

Figure 3.4: Comparison between the LBPS with $\lambda^{\text{ref}} = 0.5$ and the BPS with naive sub-sampling in terms of ESS resulting from the synthetic data in Simulation Design I.

Figure A.1 and Figure B.1 display the trace plots (the values generated from the Markov chain versus the iteration number we specified) and the distribution of x_1, x_2 posterior that were drawn from each MCMC iteration of the MH and the LBPS with optimal value of λ^{ref} suggested in the Experiment 1. The acceptance rate for the proposed parameters is 0.36 for the MH samplers. All the p-values from the Geweke's diagnostic are greater than 0.05. Based on the trace plots, they all show random scatter around one mean value, suggesting that the chain mixed well and the sample of 10,000 values is adequate to produce accurate parameter approximations of the posterior distribution.

Figure 3.5 illustrates the coverage probability (CP) of the a set of confidence intervals (CI) ranging from 10% to 95% for MLE as well as the same set of credible intervals (CI) for both the MH algorithm and tuned the LBPS. The diagonal line (the black dashed line) is one reference line, indicating that the theoretical quantile is equal to the true quantile. The point estimates will become more accurate when the CP is closer to the reference line. Based on the plots, we observe that the point estimates among all methods perform similarly.

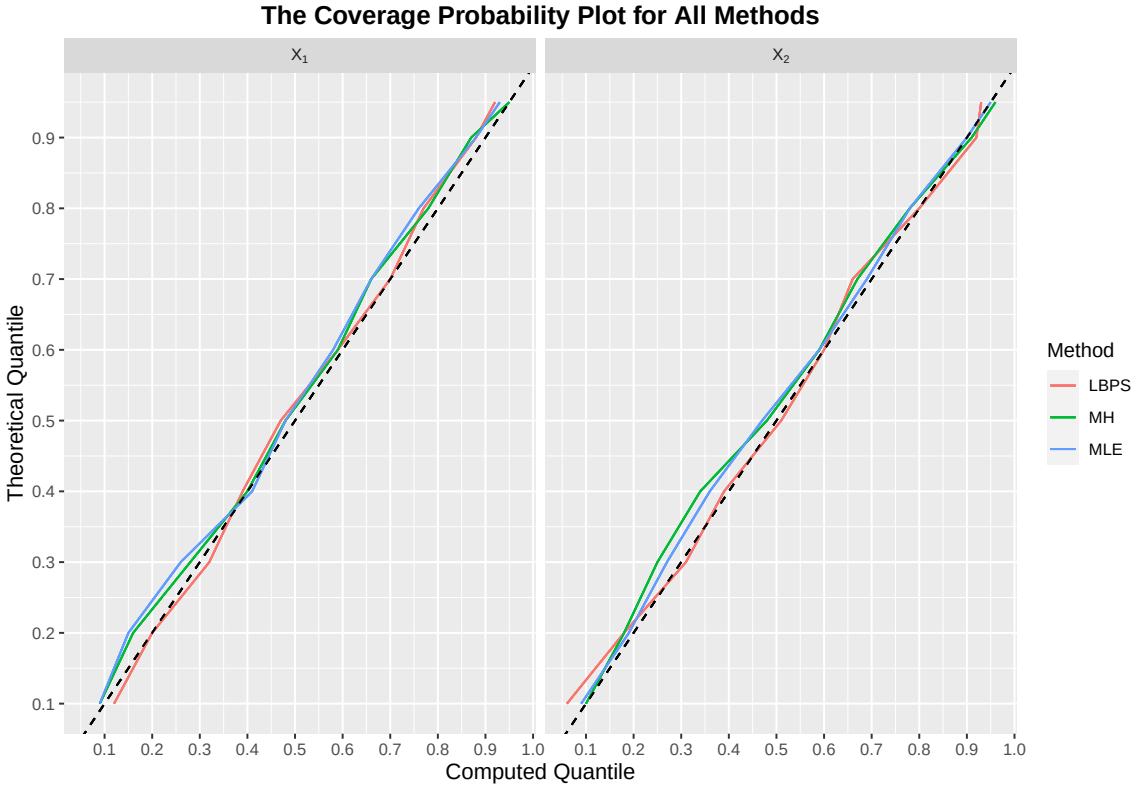

Figure 3.5: The CP of CIs for MLE as well as CIs for both the MH algorithm and the LBPS resulting from the synthetic data in Simulation Design I. The black dashed line is used as a benchmark indicating where theoretical quantile is equal to the computed quantile.

Experiment II

Based on the relative square root sMSE and ESS boxplot below (Figure 3.6, we find none of the λ^{ref} values outperform the others in terms of sMSE and ESS for each variable. From the top two boxplots, the LBPS with $\lambda^{\text{ref}} = 10$ is suggested since it has the lowest average error, while it has the largest variation in X_1. In addition, the LBPS with $\lambda^{\text{ref}} = 0.01$ has the largest ESS from the bottom two boxplots.

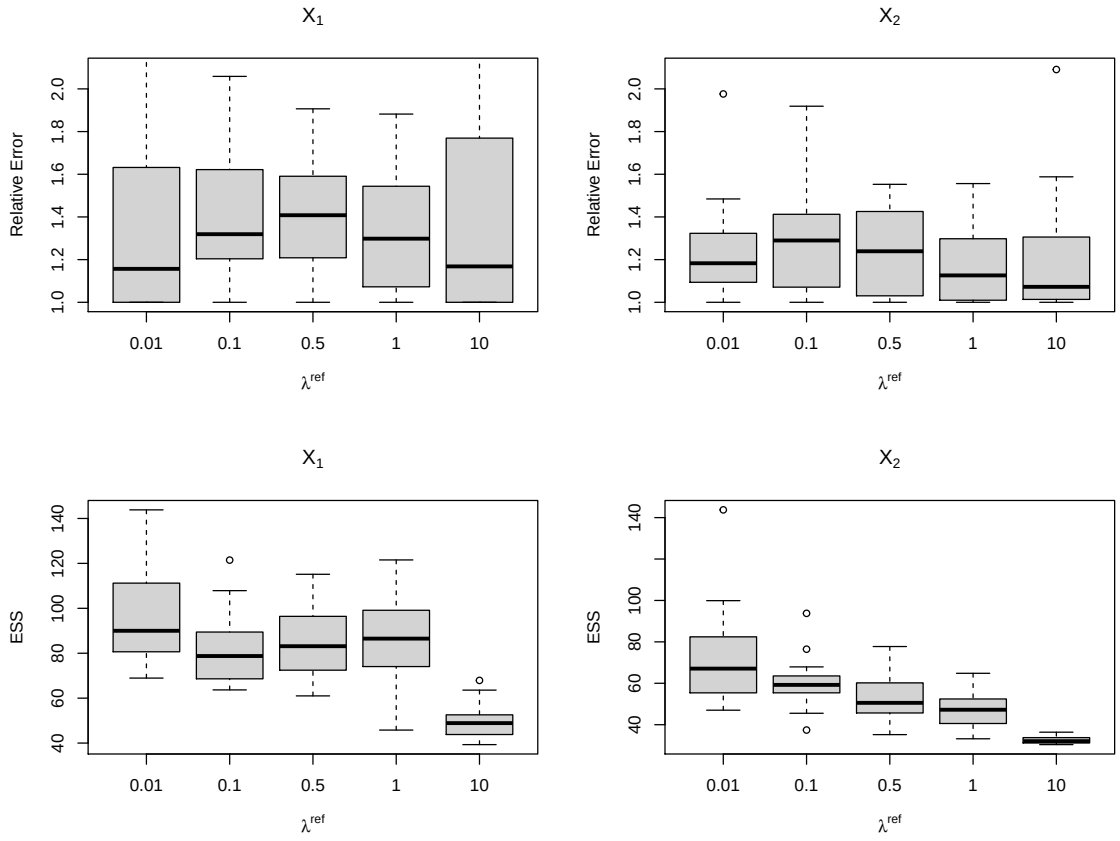

Figure 3.6: Boxplot of relative square-root sMSE and ESS for the LBPS with different values of $\lambda^{\text{ref}} = (0.01, 0.1, 0.5, 1, 10)$ resulting from the synthetic data in Simulation Design II The top two plots are boxplots of the relative square-root sMSE with $\lambda^{\text{ref}} = (0.01, 0.1, 0.5, 1, 10)$. For each run, each sMSE is divided by the smallest sMSE produced; values of 1 means that the LBPS with that λ^{ref} produced the lowest sMSE. The bottom two plots illustrate the ESS.

Next, we manually checked the trace and density plots of the X_1 and X_2 posteriors that were drawn from each MCMC iteration of the MH algorithm and the LBPS with the optimal value of λ^{ref}. Both plots showed random scatter around one mean value, suggesting that the chain mixed well and the sample of 10,000 values was adequate to produce accurate parameter approximations of the posterior distribution. The acceptance rate for the proposed

parameters is 0.43 for the MH samplers. All the p-values from the Geweke's diagnostic are greater than 0.05.

Finally, Figure 3.7 illustrates the CP of a set of CIs ranging from 10% to 95% for MLE as well as the same set of CIs for both the MH algorithm and a tuned LBPS resulting from the smaller dataset. Based on the plots, we observe that the point estimates produced via the LBPS perform worse than that for MLE and the MH algorithm, while the latter two methods perform similarly.

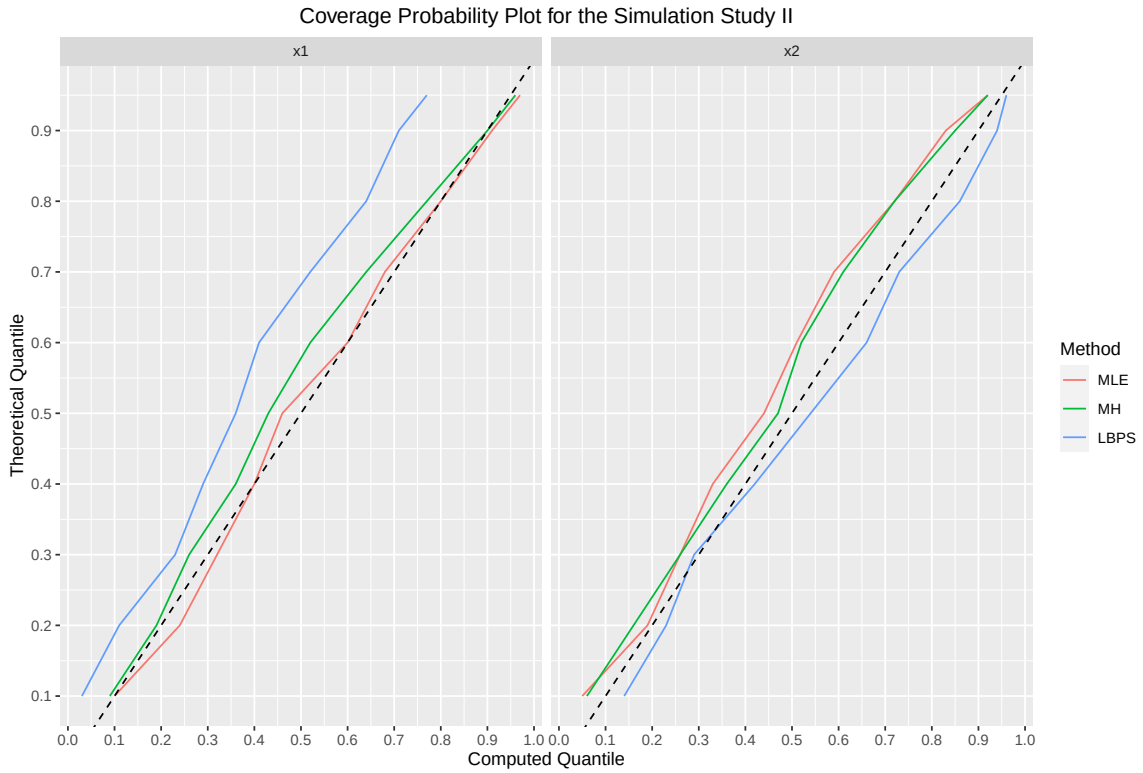

Figure 3.7: The CP of CIs for MLE and CIs for both MH and the LBPS resulting from the synthetic data in Simulation Design II. The black dashed line is used as a benchmark indicating where theoretical quantile is equal to the computed quantile.

3.2.2 Simulation Design II

Experiment Design I

Figure 3.8 illustrates the CP of a set of CIs ranging from 10% to 95% for the MH and LBPS algorithms given by the fixed computation cost. The synthetic datasets were generated by different values of n. Compare to both algorithms, the LBPS performs more efficiently for $n = 200$, while it performs less efficiently for $n = 500$. Both algorithms have similar performance for $n = 100$, and they all perform poorly for $n = 30$, suggesting that $n = 30$ is

to small too acquire accurate estimates. Therefore, it is difficult to conclude which algorithm works efficiently in the given circumstances. Instead, we discover the model efficiency in large sparse data.

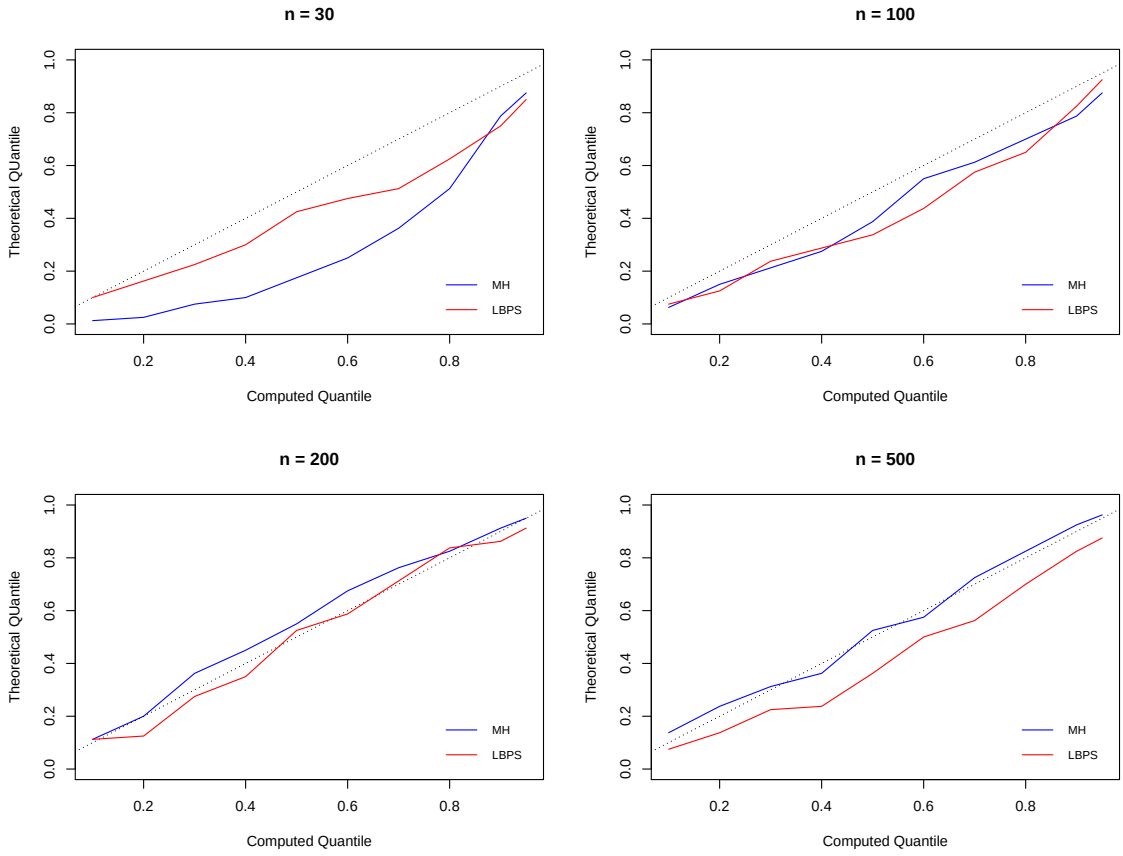

Figure 3.8: The CP of CIs of both the MH (red line) and LBPS (blue line) resulting from the synthetic data in Experiment I of Simulation Design II. The black dashed line is used as a benchmark indicating where theoretical quantile is equal to the computed quantile.

Experiment Design II

Figure 3.9 illustrates the CP of a set of CIs ranging from 10% to 95% for the MH and LBPS algorithms given by the fixed computation cost. The synthetic datasets were generated by different values of α. The value of α controls the level of sparsity. The smaller the α, the sparser the data. As the data become sparser, the efficiency of using the LBPS increases relative to using the MH for the large dataset.

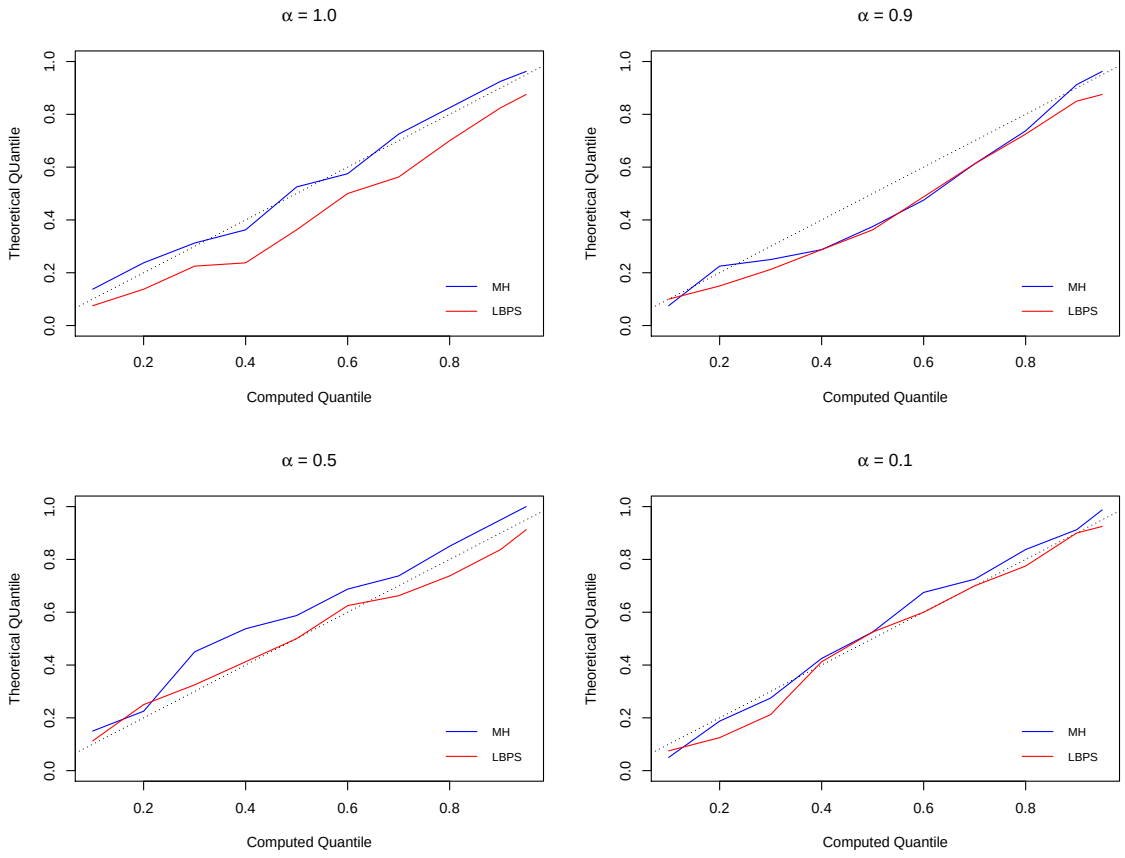

Figure 3.9: The CP of CIs of both the MH (red line) and LBPS (blue line) resulting from the synthetic data in Experiment II of Simulation Design II. The black dashed line is used as a benchmark indicating where theoretical quantile is equal to the computed quantile. The value of α controls the level of sparsity. The smaller the α, the sparser the data.

Chapter 4

Application to COVID-19 Data

4.1 Dataset Overview

The original dataset contains 125 patients who have tested positive for COVID-19 in Tianjin, China over the period of January 5, 2020 to February 17, 2020. The data were available on the Tianjin health commission official website (`http://wsjk.tj.gov.cn/`) and other online resources such as Weibo. It took mathematicians in MAGPIE Research Group several days to collect the data. Finally, the specialist generated an aggregated dataset from each patient. The COVID-19 Tianjin data has been released on the GitHub (`https://github.com/EpiCoronaHack`) at EpiCoronaHack workshop, which was organized by Dr. Caroline Colijn. We then manually added the data points from online resources including China Daily (`http://global.chinadaily.com.cn/`) and other local newspapers like the Beijing News and Tianjin Daily. Up to now, we have 137 patients who have tested positive for COVID-19 in Tianjin, China since January 5, 2020: 11 patients from February 17, 2020 to February 27, 2020; 1 patient from From February 27, 2020 to June 17, 2020; and no exposure after June 17, 2020.

4.2 Variables of Interest

In this study, the responses are whether each patient was either "normal" or "severe" in a severity test. In addition, we consider four predictors: `age`, `sex`, `Wuhan-related exposures`, and `symptom_type` (with three levels). Specifically, `symptom_type` can be classified as follows:

- Level 1: `None`, which means the patient has no symptoms;

- Level 2: `RTI`. The patient who has some respiratory tract infection (RTI) symptoms, such as fever, sore throat, running nose; fatigue, etc.;
- Level 3: `Others`. The patient has suffered from other symptoms, which are excluded from the above.

Let Y_{ijkl} be the response (0 = "normal", 1 = "severe") of the l-th patient of `sex` i (1 = "female", 2 = "male"), `Wuhan-related exposures` j (1 = "no", 2 = "yes"), `symptom_type` k (1 = "none", 2 = "rti", and 3 = "others"). In addition, we define z_{ijkl} as the `age` of the l-th patient of `sex` i, `Wuhan-related exposures` j, and `symptom_type` k.

Moreover, we assume $Y_{ijkl} \sim \text{Bernoulli}(\pi_{ijkl})$ and that the Y_{ijkl}'s are independent. Additionally, we use the logit link. The model is thus:

$$\text{logit}(\pi_{ijkl}) = x_0 + x_1 z_{ijkl} + \alpha_i + \beta_j + \gamma_k, \tag{4.1}$$

where we need to set the baseline constraints to 0; that is, $\alpha_1 = \beta_1 = \gamma_1 \equiv 0$. The next section will describe how we estimate the following model parameters:

$$\boldsymbol{x} = [x_0, x_1, \alpha_2, \beta_2, \gamma_2, \gamma_3]\,.$$

4.3 Parameters Estimation

The MLEs were derived by the Fisher scoring method described in Algorithm 1.

The MH algorithm was executed by using $N = 30,000$ MCMC iterations and a burn-in of 3,000 iterations to ensure that the Markov chain has converged. We assumed one multivariate prior on $\boldsymbol{x}$ with 0 means and variance-covariance $= \sigma^2 \mathbb{I}_6 = \text{diag} = [1, 1, 1, 1, 1, 1]'$, where $\sigma^2 = 0.04$. Moreover, the initial values for $\boldsymbol{x}$ were set to their corresponding MLE values.

The LBPS algorithm was executed by using $N = 30,000$ iterations. We assume the standard normal prior for x_0, x_1, α_2, β_2, γ_2, and γ_3, respectively. Additionally, we set the values for $\lambda^{\text{ref}} = 0.5$, $\Delta = 0.5$. The initial values were set to be $[-4, 3, 1, 2, -1, 0]'$ with $v^{(0)} = [-1.23, -0.42, -0.57, -0.78, -0.56]'$.

4.4 Results

The Fisher scoring method converged after six iterations. In the Bayesian framework, we confirmed that the Markov chain for each model parameter estimate converges to a stationary distribution after the specified MCMC iterations.

Figure A.2 and Figure A.3 illustrate the trace and density plots of x_1, α_2, β_2, γ_2, and γ_3 posteriors that were drawn from each MCMC iteration of the MH algorithm. The acceptance rate for the proposed parameters was 0.24.

Figure B.2 and Figure B.3 display the trace and density plots of x_1, α_2, β_2, γ_2, and γ_3 posteriors that were drawn from each MCMC iteration of the LBPS algorithm. After performing Geweke's diagnostic for each chain, all the p-values from the `Z` tests (Table 4.1) are greater than 0.05. Thus, we may need to treat the first 10% samples as burn-in.

	x_0	x_1	α_2	β_2	γ_2	γ_3
p-value	0.62	0.97	0.17	0.78	0.83	0.18

Table 4.1: P-values resulting from Geweke's diagnostic for each chain. Note that x_0, x_1, α_2, β_2, γ_2, γ_3 are the coefficients for `intercept`, `age`, `male`, `Wuhan-related exposures`, `Sympton_RTI`, and `Sympton_Others`, respectively.

Finally, all parameter estimations for MLE, the MH algorithm and the LBPS are summarized in Table 4.2. Based on the table, all moment estimations are tend to be similar among all methods.

	MLE			**MH**		**LBPS**	
	Mean	SD	P-value	Empirical Mean	Empirical SD	Empirical Mean	Empirical SD
x_0	-3.74	1.22	0.00	-3.76	1.18	-3.65	1.21
x_1	2.57	1.22	**0.04**	2.41	1.18	2.40	1.24
α_2	0.81	0.52	0.11	0.82	0.52	0.84	0.50
β_2	1.54	0.59	**0.01**	1.55	0.60	1.59	0.51
γ_2	-1.44	1.29	0.69	-1.77	1.35	-1.80	1.20
γ_3	0.32	0.80	0.26	0.35	0.80	0.30	0.75

Table 4.2: Summary of parameter estimation resulting from MLE, the posterior parameter estimation resulting from the MH algorithm and the LBPS for COVID-19 Tianjin data. Note that "Std.Error" calculates the standard deviation of each coefficient point estimate in Equation (4.1). Also note that x_0, x_1, α_2, β_2, γ_2, γ_3 are the coefficients for `intercept`, `age`, `male`, `Wuhan-related exposures`, `Sympton_RTI`, and `Sympton_Others`, respectively.

Next, we use a likelihood ratio test (LRT) to examine whether there is an non-zero coefficient in Equation (4.1). In this study, the null hypothesis is that all coefficients are 0, and the alternative hypothesis is that at least one coefficient term is not 0. As a result of this study, the test statistics are: 2.57 (p-value is 0.04) for x_1; 1.54 (p-value is 0.01) for β_2; while the other p-values are greater than 0.05. Hence, there is sufficient evidence to conclude that the effects of `age` and `Wuhan-related exposures` are non-zero in the model. Also, these 95%

CIs in Table 4.3 never overlap 0 in both MH and LBPS. Therefore, we can conclude that `age` and `Wuhan-related exposures` are two important factors in a COVID-19 severity test based on our dataset.

	x_0	x_1	α_2	β_2	γ_2	γ_3
MLE	[-6.35, -1.50]	[0.25, 5.10]	[-0.18, 1.87]	[0.39, 2.72]	[-1.16, 2.05]	[-4.60, 0.93]
MH	[-6.21 -1.58]	[0.13 4.81]	[-0.15 1.89]	[0.38, 2.75]	[-1.15, 2.03]	[-4.62, 0.63]
LBPS	[-6.02, -1.28]	[0.49, 4.88]	[-0.16, 2.13]	[0.19, 3.23]	[-0.90, 1.59]	[-4.26, 0.45]

Table 4.3: Confidence/Credible intervals (CI) resulting from MLE, the MH algorithm and the LBPS for COVID-19 Tianjin data.

When holding other covariates constant (i.e., same `age`, `sex`, and `sympton_type`), we can conclude that the estimated odds of patients who have lived or travelled from Wuhan for indicating a "serve" status are 4.66 times the estimated odds for indicating a "normal" status. Moreover, we have 95% confidence to say that the estimated odds that patients who have lived or travelled from Wuhan indicated "severe" on a severity test ranges from 1.48 to 15.18 times those of indicating "normal" in a severity test. Using the quantiles of the posterior distribution, we have 95% confidence to conclude that those empirical means lie in the CI [1] $[1.46, 15.64]$ for the LBPS, and $[1.21, 25.28]$ for the MH algorithm.

[1] The 95% equal-tailed interval is used to compute 95% CI for the MH and LBPS algorithms.

Chapter 5

Conclusion and Future work

5.1 Discussion and conclusion

In this project, we described the Bouncy Particle Sampler (BPS) and Local Bouncy Particle Sampler (LBPS) and applied the techniques to estimate model parameters in the logistics regression setting. We also compared the model performance among the LBPS and other commonly used methods like maximum likelihood estimation (MLE) and the Metropolis–Hastings (MH) algorithm.

To improve the performance of the LBPS, the value of λ^{ref} does affect the implementation results even though the BPS has been proved in theory to be ergodic for a positive rate parameter. However, as we increase the value of λ^{ref}, ESS increases and the Markov chain more slowly mixes. Furthermore, the LBPS samplers will produce Markov chains that poorly approximate our target distribution for either extremely large or small values of λ^{ref}. Apart from that, we can apply the alias sampling method [19] to greatly reduce computation time. Some partial Python code can be found at the website [1]. Another challenge involves how to simulate τ in the non-homogeneous Poisson process setting efficiently. In general, the simulation of τ is based on the thinning and superposition algorithms in the BPS and LBPS. The upper bound of the intensity function $\chi(t)$ is important since it would directly affect the efficiency. In fact, we would prefer a tighter upper bound.

In simulation studies, computational time and accuracy are two metrics that can directly reflect model performance. The MH samplers perform better in a smaller dataset, while the LBPS is more efficient in a larger and sparser dataset. Moreover, naive sub-sampling in the BPS assigns equal weight on each observation and then randomly selects one observation to run the algorithm. This method is appealing due to its fast computational time, while it sacrifices the accuracy of parameter estimation since this method only accesses one observation rather than the whole dataset. Therefore, the BPS with naive subsampling cannot compete

in parameter estimation since it does not offer the best trade-off between computation time and accuracy. Finally, there is evidence to conclude `age` and `Wuhan-related exposures` are important factors in COVID-19 severity according to 137 Tianjin COVID-19 patients, which confirms the results reported in [32]. However, this data set is not a large data set. In order to verify our conclusion, we need need larger data sets that contain more COVID-19 cases and their information.

5.2 Future work

Firstly, the majority of papers such as [12, 26, 30] have considered that model parameters follow normal distributions, while other popular priors such as Cauchy priors suggested by [16], a generalized double Pareto prior [2], and a Laplace prior (also known as double exponential prior) [2, 29] have not been applied in the BPS and LBPS algorithms.

Secondly, we have shown that the LBPS can draw samples from our target distribution efficiently. To improve this method, we can borrow the control variate ideas suggested by [3, 7]. This method applies on the assumption that the gradient of each component's energy function satisfies the Lipschitz condition. The Lipschitz bounds can be used for a logistics regression model, while the LBPS uses the bounds on the intensity function in a non-homogeneous Poisson process.

Next the value of λ^{ref} can be sensitive in the LBPS and BPS algorithms. So on the one hand, other methods in the PDMP family such as the Zig-Zag sampler and the GBPS might be alternatives without the tuning process. On the other hand, we can try Boomerang samplers with a sound tuning process. For all the specified methods, we need to compare model performance in terms of the efficiency and accuracy with various numerical simulation settings. For example, we can try a more challenging data setting, such as high-dmensional sparse data. In this case, the number of predictors and number of observations are relatively large, and we choose the level of sparsity.

Finally, there are limited `R` packages that implement PDMP methods. `RZigZag` is the only available package, and it can only implement limited PDMP methods. Indeed, we can write our own `R` package with general functions to implement the BPS, LBPS, Zig-Zag sampler, GBPS, and Boomerang since they all have something in common.

Bibliography

[1] Ryan Adams. The alias method: Efficient sampling with many discrete outcomes. Available at `https://lips.cs.princeton.edu/the-alias-method-efficient-sampling-with-many-discrete-outcomes/` (2020/07/10).

[2] Artin Armagan, David B. Dunson, and Jaeyong Lee. Generalized double pareto shrinkage. *Statistica Sinica*, 23(1):119–143, 2013.

[3] Rémi Bardenet, Arnaud Doucet, and Chris Holmes. On markov chain monte carlo methods for tall data. Microtome Publishing, 2017.

[4] Joris Bierkens. Non-reversible metropolis-hastings, 2014.

[5] Joris Bierkens. *RZigZag: Zig-Zag Sampler*, 2019. R package version 0.2.1.

[6] Joris Bierkens, Alexandre Bouchard-Côté, Arnaud Doucet, Andrew B. Duncan, Paul Fearnhead, Thibaut Lienart, Gareth Roberts, and Sebastian J. Vollmer. Piecewise deterministic markov processes for scalable monte carlo on restricted domains. *Statistics & Probability Letters*, 136(C):148–154, 2018.

[7] Joris Bierkens, Paul Fearnhead, and Gareth Roberts. The zig-zag process and super-efficient sampling for bayesian analysis of big data, 2016.

[8] Joris Bierkens, Paul Fearnhead, and Gareth Roberts. The zig-zag process and super-efficient sampling for bayesian analysis of big data. *The Annals of Statistics*, 47(3):1288–1320, 2019.

[9] Joris Bierkens, Sebastiano Grazzi, Kengo Kamatani, and Gareth Roberts. The boomerang sampler. 2020.

[10] Joris Bierkens, Sebastiano Grazzi, Frank van der Meulen, and Moritz Schauer. A piecewise deterministic monte carlo method for diffusion bridges. 2020.

[11] Joris Bierkens, Gareth Roberts, and Pierre-André Zitt. Ergodicity of the zigzag process. 2017.

[12] Alexandre Bouchard-Côté, Sebastian J Vollmer, and Arnaud Doucet. The bouncy particle sampler: A nonreversible rejection-free markov chain monte carlo method. *Journal of the American Statistical Association*, 113(522):855–867, 2018.

[13] Neil Ferguson. Covid-19 reports. Available at `https://www.imperial.ac.uk/mrc-global-infectious-disease-analysis/covid-19/covid-19-reports/` (2005/06/12).

[14] James M. Flegal, John Hughes, Dootika Vats, and Ning Dai. *mcmcse: Monte Carlo Standard Errors for MCMC*. Riverside, CA, Denver, CO, Coventry, UK, and Minneapolis, MN, 2020. R package version 1.4-1.

[15] Nicholas Galbraith. On event-chain monte carlo methods. Department of Statistics, Oxford University, 2016.

[16] Andrew Gelman, Aleks Jakulin, Maria Grazia Pittau, and Yu-Sung Su. A weakly informative default prior distribution for logistic and other regression models. *The annals of applied statistics*, 2(4):1360–1383, 2008.

[17] John F. Geweke. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. Staff Report 148, Federal Reserve Bank of Minneapolis, 1991.

[18] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[19] Richard A Kronmal and Arthur V Peterson. On the alias method for generating random variables from a discrete distribution. *The American Statistician*, 33(4):214–218, 1979.

[20] P. A. W Lewis and G. S. Shedler. Simulation of nonhomogeneous poisson processes by thinning. *Naval Research Logistics Quarterly*, 26(3):403–413, 1979.

[21] Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. MCMCpack: Markov chain monte carlo in R. *Journal of Statistical Software*, 42(9):22, 2011.

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

[23] E.A.J.F. Peters and G. With. Rejection-free monte carlo sampling for general potentials. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 85:026703, 02 2012.

[24] Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. Coda: Convergence diagnosis and output analysis for mcmc. *R News*, 6(1):7–11, 2006.

[25] Christian P Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer New York, New York, second edition, 2004.

[26] Deborshee Sen, Matthias Sachs, Jianfeng Lu, and David B Dunson. Efficient posterior sampling for high-dimensional imbalanced logistic regression. *Biometrika*, 2020.

[27] Yi Sun, Faustino Gomez, and Juergen Schmidhuber. Improving the asymptotic performance of markov chain monte-carlo by inserting vortices. 09 2012.

[28] Konstantin S Turitsyn, Michael Chertkov, and Marija Vucelja. Irreversible monte carlo algorithms for efficient sampling. *Physica. D*, 240(4-5):410–414, 2011.

[29] Peter M Williams. Bayesian regularization and pruning using a laplace prior. *Neural computation*, 7(1):117–143, 1995.

[30] Changye Wu and Christian Robert. Generalized bouncy particle sampler. Springer Verlag (Germany), 2019.

[31] Changye Wu and Christian P Robert. Coordinate sampler: a non-reversible gibbs-like mcmc sampler. *Statistics and computing*, 30(3):721–730, 2019. Springer Science and Business Media LLC.

[32] Zunyou Wu and Jennifer M McGoogan. Characteristics of and important lessons from the coronavirus disease 2019 (covid-19) outbreak in china: Summary of a report of 72 314 cases from the chinese center for disease control and prevention. *JAMA : the journal of the American Medical Association*, 323(13):1239–1242, 2020.

Appendix A

Further Model Assessment in MH

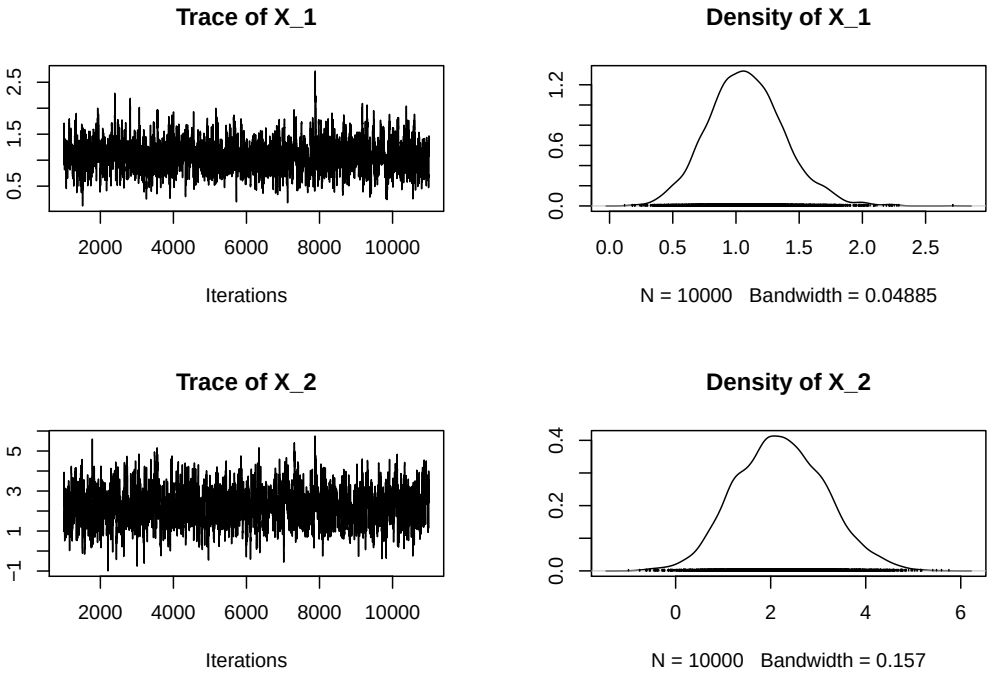

Figure A.1: The trace and density plots for X_1 and X_2 resulting from the synthetic data in Simulation Design I with the MH algorithm.

Figure A.2 and A.3 display the trace plots (the values generated from the Markov chain versus the iteration number we specified) and the distribution of x_1 (`age`), α_2 (`male`), β_2 (`Wuhan-related exposures`), γ_2 (`Symptom_RTI`), and γ_3 (`Sympton_Others`) posterior that draw from the MH algorithm. Based on the trace plots, they all show random scatter around one mean value, suggesting that the chain mixed well and the sample of 10,000 values is adequate to produce accurate parameter approximations of the posterior distribution.

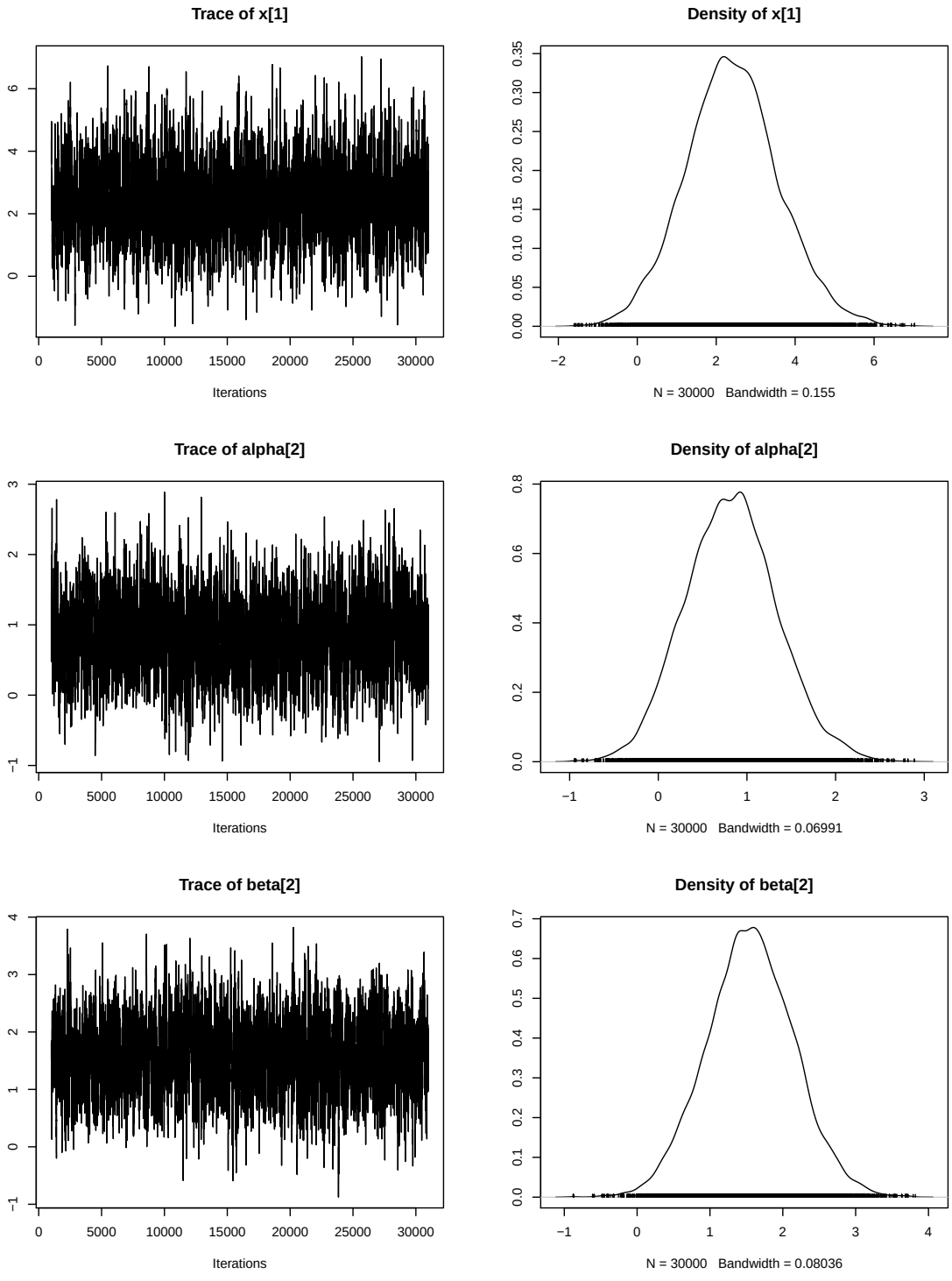

Figure A.2: The trace and density plots of the standard MCMC samplers for x_1, α_2, and β_2 from COVID-19 Tianjin data.

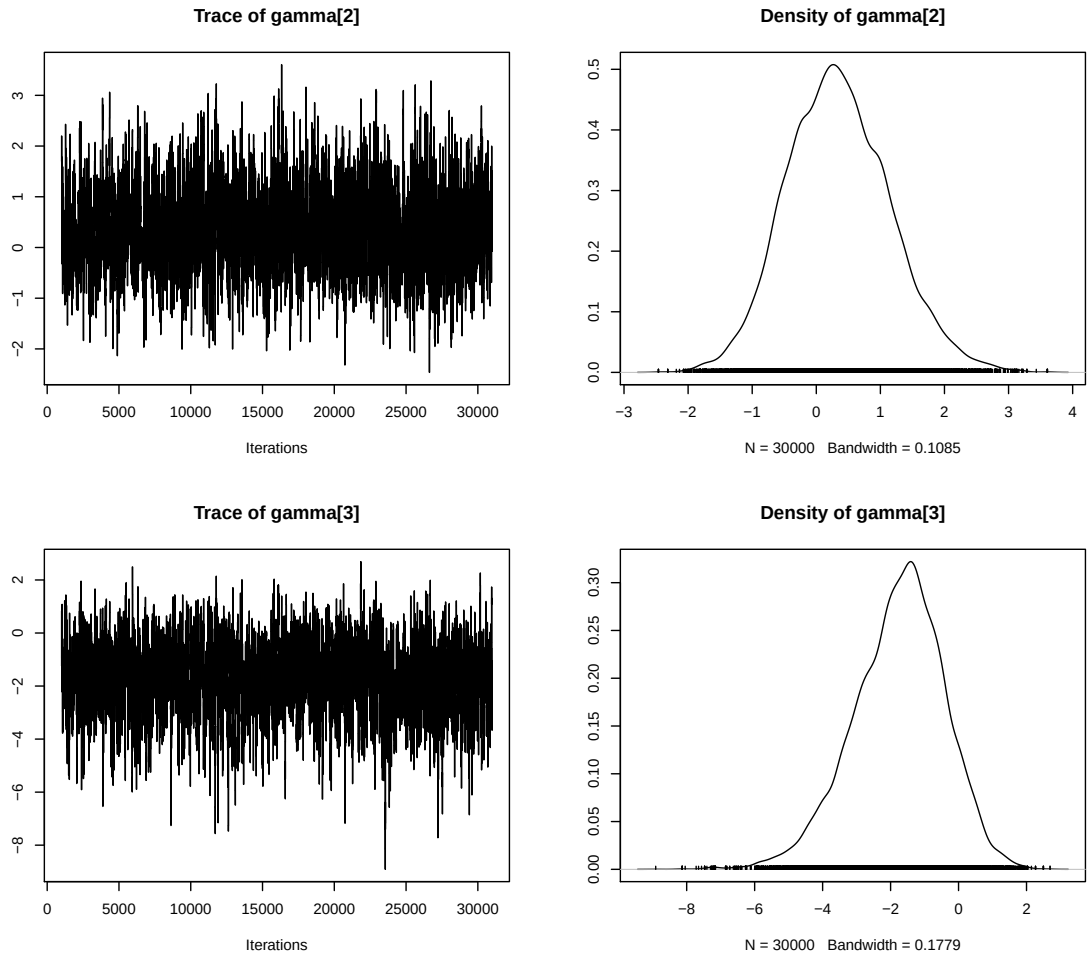

Figure A.3: The trace and density plots of the standard MCMC samplers for γ_2 and γ_3 from COVID-19 Tianjin data.

Appendix B

Further Model Assessment in LBPS

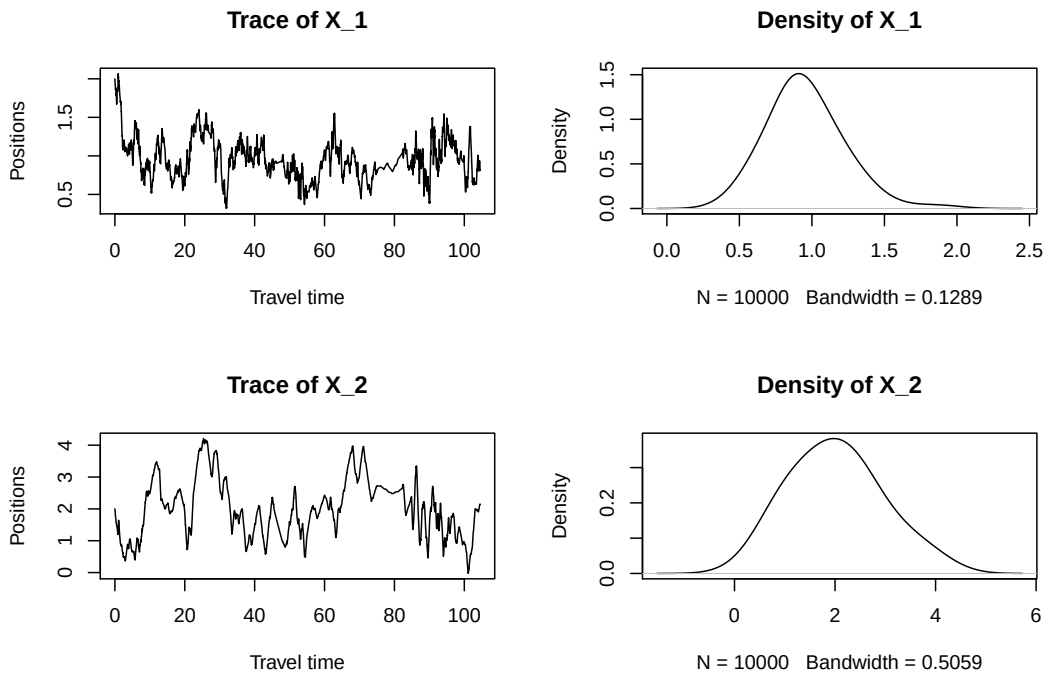

Figure B.1: The trace and density plots for X_1 and X_2 resulting from the synthetic data in Simulation Design I with the LBPS Algorithm.

Figure B.2 and B.3 display the trace plots (the values generated from the Markov chain versus each travel time) and the distribution of x_1 (`age`), α_2 (`male`), β_2 (`Wuhan-related exposures`), γ_2 (`Symptom_RTI`), and γ_3 (`Sympton_Others`) posterior that draw from the LBPS. Based on the trace plots, they all show random scatter around one mean value, suggesting that the chain mixed well and the sample of 30,000 values is adequate to produce accurate parameter approximations of the posterior distribution.

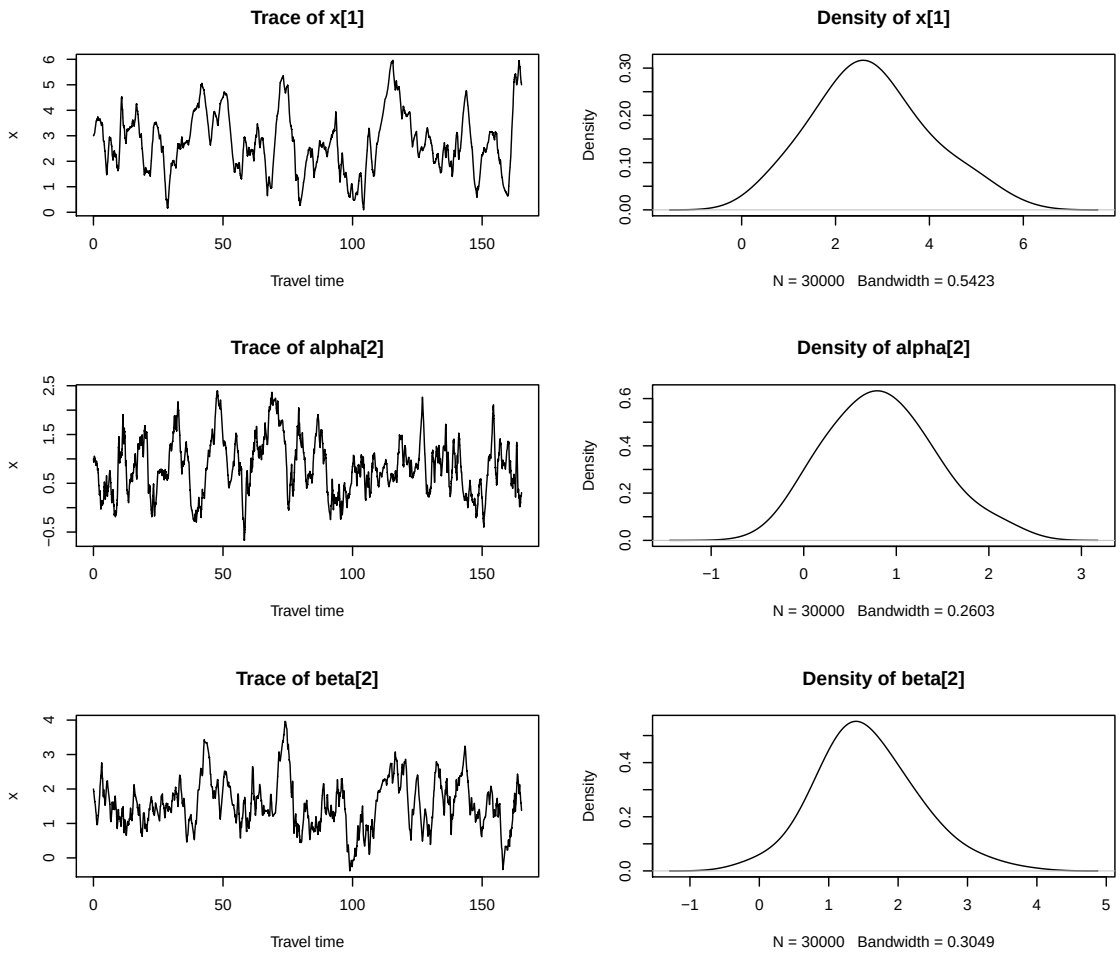

Figure B.2: The trace and density plots of the LBPS for x_1, α_2, and β_2 from COVID-19 Tianjin data.

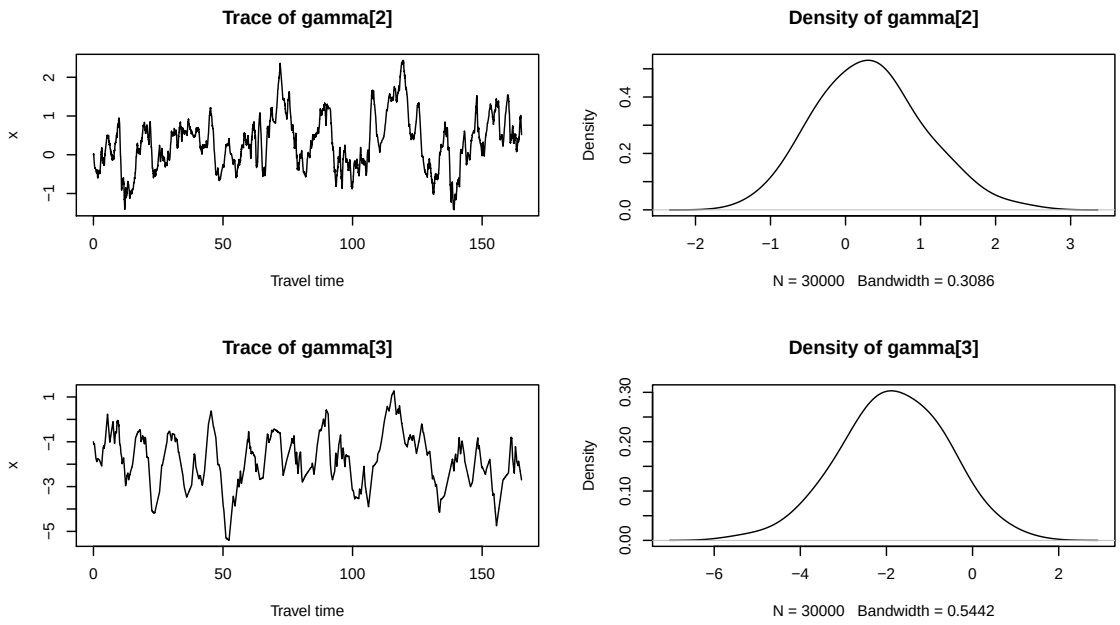

Figure B.3: The trace and density plots of the LBPS for γ_2 and γ_3 from COVID-19 Tianjin Data.

Appendix C

Code

```
# R Version: 4.0.2 (2020-06-22)
# LBPS algorithm

# Functions
expit = function(k) 1 - 1/(1+exp(k))

chi_bar_fun = function(v, i, j, z){
  if (z[i,j]>0){
    result = abs(v[j]) * (v[j] * (-1)^(y[i]) >= 0)  * z[i,j]
  } else{
    result = abs(v[j]) * (v[j] * (-1)^(y[i]) < 0)  * (-z[i,j])
  }
  return(result)
}

Gaussian_PP <- function(z,v){
  z = as.matrix(z)
  a = as.numeric(v %*% t(z))
  b = sum(v^2)
  V = runif(1, 0, 1)
  result = (1/b) * (-a + sqrt(a^2 - 2* b * log(V)))
  return(result)
}

local_BPS_fun = function(N, n, z, Delta, Time, T_bar, sigma, lambda_ref){
  Time_vec = chi_bar_i = chi_bar_j = k = c()
  x_mat =  matrix(0, nrow = N + 1 , ncol = p)
  v_mat = matrix(0, nrow = N + 1 , ncol = p)
  chi_bar_mat = matrix(0, nrow = n, ncol = p)

  # Record the initial value
  x_mat[1, ] = x_0; v_mat[1, ] = v_0; Time_vec[1] = Time

  v = v_0
  x = x_0

  # (a) local-in-time upper bound on the data points
  for (i in 1:N){

   chi_bar_mat = sapply(1:p, function(d)
```

```
      sapply(1:n, function(r) chi_bar_fun(v=v,r,ind,z)))

    chi_bar_i = rowSums(chi_bar_mat)
    chi_bar_j = colSums(chi_bar_mat)
    chi_bar = sum(chi_bar_i)

    # (b) Simulate tau
    tau = rexp(1, chi_bar + Chi_bar_p + lambda_ref)

    # When the global time and time of travel reach the local time upper-
        bound
    if (Time + tau > T_bar){
      ## i. Position update:
      x = x + v*(T_bar - Time)

      ## ii. Velocity update:
      v = v

      ## iii. Local-in-time upper bound on the prior factor update:
      Chi_bar_p = (1/(sigma^2)) * max(0, sum((x + v * Delta)*v))

      ## iv. Travel time update
      Time = T_bar
      T_bar = T_bar + Delta
    }
    else{
      ## i. Position update:
      x = x + v*tau

      ## ii. Velocity update: generate a random integer k from
      k = sample(1:3, size = 1, prob = c(chi_bar, lambda_ref, Chi_bar_p))

      ## k = 1, based on the data points
      if (k == 1){
        ### i. Sample j~ q_j(.)
        j_ind = sample(p, size = 1, prob = chi_bar_j)

        ### ii. Sample i ~ q_i|j(.|j)
        i_ind = sample(n, size = 1, prob = chi_bar_mat[, j_ind])

        ### Thinning algorithm
        gU = (as.numeric(z[i_ind,]) *
                (expit(sum(as.numeric(z[i_ind,]) * x)) - y[i_ind]))
        if (runif(1) < max(0, sum(gU*v)) / chi_bar_i[i_ind]){
          v = v - 2 * sum(gU*v) / sum(gU^2) * gU
        }
        else v = v
      }

      ##  k = 2, based on the velocity refreshment
      if (k == 2) v = rnorm(p)

      ##  k = 3, based on the prior
      if (k == 3){
        tau_sim = c()
        for(ind in 1:n){
          tau_sim[ind] = Gaussian_PP(z = z[ind,],v = v)
        }
```

```
98
99          ### i. b be the position where minimize tau_{*}
100         b = which.min(tau_sim)
101
102         ### Thinning algorithm
103         gU_prior = (as.numeric(z[b,]) * (expit(sum(as.numeric(z[b,]) * x)) -
                y[b]))
104         if (runif(1) < max(0, sum(x*v)) / (Chi_bar_p)){
105           v = v - 2 * sum(gU_prior*v) / sum(gU_prior^2) * gU_prior
106         }
107         else v = v
108       }
109
110       # iii. Local-in-time upper bound on the prior factor update:
111       Chi_bar_p = (1/(sigma^2)) * max(0, sum((x + v * (T_bar - Time - tau))*
              v))
112
113       # iv. Travel time update:
114       Time = Time + tau
115     }
116
117     # Record into matrix
118     Time_vec[i+1] = Time
119     x_mat[i+1, ] = x
120     v_mat[i+1, ] = v
121   }
122   return(list(x = x_mat, v = v_mat, Time = Time_vec))
123 }
124 # ------------------------------------------------------------------------
125 # Simulated Data
126 n = 100
127 x1 = runif(n, min = 0.1, max = 1)
128 x2 = runif(n, min = 0.1, max = 1)
129
130 ## True values
131 true = c(0.5, 1, 1)
132 y_0 = true[1] + true[2]*x1 + true[3]*x2
133
134 ## Inverse-logit function
135 Pr = 1/(1+exp(-y_0))
136 ## The simulate response
137 y = rbinom(n, 1, Pr)
138
139 ## The simulate data
140 data_sim = data.frame(y,x1,x2)
141 z = as.data.frame(model.matrix(~ x1 + x2, data = data_sim))
142 p = dim(z)[2]
143 # ------------------------------------------------------------------------
144 # Initialization
145 fit = glm(y ~ ., family = binomial(link = logit), data = data_sim)
146
147 ## (a)
148 x_0 = coef(fit) # position
149 v_0 = rnorm(p) # velocity
150
151 ## (b)
152 Time = 0  # Global travel Time
153
```

```r
## (c)
Delta = 0.5 # Local upper bounds

## The prior variance
sigma = 1

## Lambda refreshment
lambda_ref = 0.5

## (d) local-in-time upper bound on prior
Chi_bar_p = (1/(sigma^2)) * max(0, sum((x_0 + v_0 * Delta)*v_0))

## Number of MCMC iterations
N = 10000
# ------------------------------------------------------------------------
# LBPS
#   return: x^{(i)}, v^{(i)}, and Time
local_BPS = local_BPS_fun(N = N, n = n,  z = z, Delta = Delta, Time = Time,
            T_bar = Delta, sigma = sigma, lambda_ref = lambda_ref)

# Trace plot and density plot
par(mfrow = c(p,2))
x_lab = c("x0", "x1", "x2")
for (k in 2:p){
  new_tau = approx(x = local_BPS$Time, y =local_BPS$x[,k], n = N)
  # Trace plot
  plot(x = (local_BPS$Time[1:N]), y = local_BPS$x[1:N,k],
       type="l",main = paste0("Trace of ", x_lab[k]),
       ylab = "Position", xlab = "Time of Travel")
  # Density plot
  plot(density(new_tau$y, adjust=2))
  abline(v = true[k], col = "red")
}

# Calculate the posterior mean and SD
first_mom =  second_mom = c()
t = tail(local_BPS$Time[1:N], n = 1)
Lag_time =  diff(local_BPS$Time[1:N], 1)

for (k in 2:p){
  first_mom[k] =
    (1/t) * (sum(Lag_time * local_BPS$x[1:N-1,k])+
               0.5 * sum(Lag_time ^ 2 * local_BPS$v[1:N-1,k]))

  second_mom[k] =
    (1/t) * (sum(Lag_time * (local_BPS$x[1:N-1,k]^2)) +
               sum(Lag_time^2 * (local_BPS$x[1:N-1,k]*local_BPS$v[1:N-1,k]))
                  +
               (1/3) * sum((Lag_time^3) * (local_BPS$v[1:N-1,k]^2)))
}

mean = first_mom
sd = sqrt(second_mom - first_mom^2)
parp_est = data.frame(Mean = mean, SD = sd)
rownames(parp_est) = x_lab
parp_est
```