UC Irvine

UC Irvine Electronic Theses and Dissertations

Title

A New Population-based MCMC Method

Permalink

https://escholarship.org/uc/item/3sc8w2n8

Author

Zhang, Di

Publication Date

2019

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, availalbe at https://creativecommons.org/licenses/by/4.0/

Peer reviewed|Thesis/dissertation

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

UNIVERSITY OF CALIFORNIA,
IRVINE

A New Population-based MCMC Method

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Statistics

by

Di Zhang

Dissertation Committee:
Associate Professor Yaming Yu, Chair
Professor Michele Guindani
Assistant Professor Weining Shen

2019

© 2019 Di Zhang

DEDICATION

To my parents, for their endless support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

I would like to extend my greatest gratitude to my advisor Yaming Yu for his tremendous support throughout my graduate career, as it is impossible for me to reach the finishing line and obtain my Ph.D. degree without his guidance. His attitude towards research and passion for science have always been a source of my career aspiration. I am proud and fortunate to work with Yaming in the past few years, which definitely benefits me for life.

I would also like to thank my dissertation committee members Michele Guindani and Weining Shen for their help and comments on my thesis. Many thanks to Hal Stern and Shuang Zhao for being part of my candidate committee and thanks to my academic advisor Zhaoxia Yu for her suggestions on both academic and professional life during the first two years. Special thanks go to David van Dyk and Vinay Kashyap, as they lead me to the to the wonderful field of astrostatistics and provide guidance along with the astronomical project. I appreciate all the support from members in CHASC group. My sincere gratitude also goes to all the professors and teachers and staff members in Department of Statistics at UC Irvine, for their help and trust in my graduate education.

In addition, I would like to thank all my fellow students and friends who have made my life easier in UC Irvine. Thanks also go to my friends that are not physically in Irvine, for their help and sharing feelings; particularly, among this special group of fellow friends, I would like to express my gratitude to the following few in Mingwei Tang, Zhengyang Wang and Yuanji Xie, as they voluntarily helped on the polish of my thesis.

Last but not least, I would like to thank my family that love and support me throughout my life, without them I cannot even start the very first step, and now we have made it this far.

All the guidance, support and love make it possible for me to reach the end of this journey, and I am forever grateful.

CURRICULUM VITAE

Di Zhang

EDUCATION

Doctor of Philosophy in Statistics **2019**
University of California, Irvine *Irvine, California*

Bachelor of Science in Statistics **2013**
Nanjing University *Nanjing, China*

RESEARCH EXPERIENCE

Graduate Research Assistant **2013–2019**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Teaching Assistant **2013–2019**
University of California, Irvine *Irvine, California*

SOFTWARE

R, Python, Bash Script, LaTeX

ABSTRACT OF THE DISSERTATION

A New Population-based MCMC Method

By

Di Zhang

Doctor of Philosophy in Statistics

University of California, Irvine, 2019

Associate Professor Yaming Yu, Chair

Spectral analysis in high-energy astrophysics typically requires sampling from difficult posterior distributions, *e.g.*, multimodal distributions, in a highly structured model with complex data collection mechanisms. Markov chain Monte Carlo (MCMC) methods have been widely explored in such tasks. However, traditional MCMC methods suffer from slow convergence incurred by the local trap problem. In this thesis, we propose a general population-based MCMC strategy, which is able to speed up convergence significantly. Specifically, we initialize multiple chains from dispersed starting values and perform a two-step jump with our proposed between-chain jump. The novel between-chain jumping proposal tries to move to the neighborhood of the iterate in another chain, which encourages full exploration of the parameter space. As for the numerical illustration, we apply the proposed method to fit thermal models to Capella data. The results indicate that our method is effective in two aspects. First, it can be applied in Bayesian model selection to decide which mixture model is more appropriate; secondly, it can be served as an exploratory step to identify modes. The strength shown from the second aspect can help determine the importance of those temperature components and decide upon the relative proportion among those modes. In addition, we show that our method is also useful in other applications including Bayes factor computation for Bayesian model selection, variance component estimation in mixed effect models, and sensor network localization.

Chapter 1

Outline

We introduce a new population-based MCMC method which is motivated by a problem in astronomy where our goal is to sample from the posterior distribution in a highly structured model without an excessive amount of parameters. The key issues lie in high time cost during each iteration along with concerns on convergence. In addition, the posterior surface is complicated and potentially multimodal as it comes from a mixture model. In order to overcome the above obstacles, we propose the new method which is able to speed up convergence significantly. The following chapters will first introduce the methodology and illustrate how the method works with a few examples; eventually we proceed to the final chapter which summarizes all the work, along with discussions on the pros and cons as well as future work.

In Chapter 2, we provide the background information on MCMC methods led by Metropolis-Hastings, Gibbs sampler and simulated tempering.

In Chapter 3, a new population-based MCMC method is proposed and developed to sample from complicated posterior distributions with possibly multiple modes. First we explain the mechanism of the proposed method, followed by an intuitive example so as to have a better

understanding. Secondly, we discuss more about the theoretical aspects of the method as well as the general suggestions of how to apply the method.

In Chapter 4, we evaluate our method on several applications. To be more specific, we apply the multiple chain method in computing Bayes factors for Bayesian model selection, estimating variance components in mixed effect models, and sensor network localization.

In Chapter 5, we focus on the astronomical data, particularly to serve the study on spectral analysis of X-ray astronomy. The model is highly structured with continuum, emission/absorption lines and the instrument response. It is potentially multimodal with a complicated likelihood function that takes time to evaluate. We apply the multiple chain method to estimate the parameters of interest, and find our method useful in two aspects. First, it can still be applied in Bayesian model selection to decide which mixture model is more appropriate; secondly, it can be served as an exploratory step to identify modes. The latter feature can help determine the importance of those temperature components and decide upon the relative proportion among those modes.

In Chapter 6, we summarize our work and discuss the limitations as well as prospective future work of our proposed method.

Chapter 2

Background

In this chapter, we review a few different existing MCMC methods.

2.1 A Review of MCMC methods

In Bayesian statistical inference (Gelman et al., 1995), we aim to learn from the posterior distribution of parameters of interest. For example, we seek inferences about a parameter, to compute expectations or to make predictions. These types of posterior estimations often involve computations of integrals. It is quite common that the integrals do not have analytical forms thus we often employ a sampling approach.

Markov chain Monte Carlo (MCMC) method is a widely applied sampling approach developed from the Monte Carlo integration. Essentially, it is Monte Carlo integration using Markov chains. Gilks et al. (1996) provides more details. The key idea is to construct a Markov chain that has the desired distribution as its stationary distribution. After we acquire samples from the chain, we can use them to draw inference about a parameter or to compute expectations, etc. Some well known MCMC methods include Metropolis-Hastings

(Hastings 1970) and Gibbs sampler (Geman and Geman, 1987), which will be discussed later.

With advancing of computers more efficient and scalable MCMC algorithms are created and widely applied. The development of computing power makes MCMC methods well known by statisticians and we are able to use them to solve real world problems. Geyer (2011) provides some examples in detail. In Section 2.2 and 2.3, we review two fundamental MCMC algorithms, the Metropolis-Hastings algorithm and Gibbs sampler.

2.2 The Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is a generalization of the Metropolis algorithm. Let us start by describing the Metropolis algorithm proposed by Metropolis et al. (1953).

Suppose we want to sample from a probability distribution with the corresponding unnormalized density p. For the Metropolis algorithm, it initializes a Markov chain at some arbitrary point x_0. At each iteration, the algorithm proposes a candidate point y given the current state x, according to some proposal density $q(\cdot|x)$. Note that the proposal q here is symmetric, *i.e.*, $q(y|x) = q(x|y)$. A typical example would be Gaussian type random walk, in other words, $q(\cdot|x)$ is a Gaussian distribution centered at x. And the term *random walk* here means q satisfies that $q(y|x) = q(|x - y|)$. Next, we calculate the acceptance ratio

$$r(y|x) = \frac{p(y)}{p(x)}$$

and then accept the proposed point y with probability

$$\alpha(y|x) = \min\left(1, \frac{p(y)}{p(x)}\right).$$

The pseudocode for the Metropolis algorithm is shown in Algorithm 1. So after each iteration,

Algorithm 1 Metropolis

Initialize $x_0 \sim \pi(x)$ // $\pi(\cdot)$ is the initial distribution
for $t = 0$ to $T - 1$ **do**
  $y \leftarrow X \sim q(\cdot|x_t)$
  $\alpha \leftarrow \min\left(1, \frac{p(y)}{p(x_t)}\right)$
  $u \leftarrow U \sim U[0, 1]$
  if $u \leq \alpha$ **then**
    $x_{t+1} \leftarrow y$
  else
    $x_{t+1} \leftarrow x_t$
  end if
end for

the current state could be updated with certain probability. To be specific, if $p(y) > p(x_t)$, the chain will move to y for sure, otherwise it will move to y with probability $\frac{p(y)}{p(x_t)}$. In other words, the uphill move always gets accepted while the downhill move is accepted stochastically. This acceptance-rejection mechanism is the general approach for MCMC algorithms. The transition kernel for the Metropolis algorithm is

$$T(x_{t+1}|x_t) = q(x_{t+1}|x_t)\alpha(x_{t+1}|x_t) + I(x_{t+1} = x_t) \int q(y|x_t)(1 - \alpha(y|x_t))dy, \tag{2.1}$$

where $I(\cdot)$ denotes the indicator function and α is the acceptance probability. In order to ensure that the Markov chain converges to the stationary distribution p, we have to show that the Metropolis update is reversible with respect to p, in other words, the detailed balance condition (2.2) is fulfilled:

$$p(x_t)T(x_{t+1}|x_t) = p(x_{t+1})T(x_t|x_{t+1}). \tag{2.2}$$

It is trivial that the equation holds if $x_{t+1} = x_t$, in the cases that either y is rejected or y equals to x_t, which is very rare. If $x_{t+1} \neq x_t$, we drop the second term in Equation 2.1 and

Algorithm 2 Metropolis-Hastings

Initialize $x_0 \sim \pi(x)$ // $\pi(\cdot)$ is the initial distribution
for $t = 0$ to $T - 1$ **do**
  $y \leftarrow X \sim q(\cdot|x_t)$
  $\alpha \leftarrow \min(1, \frac{p(y)q(x_t|y)}{p(x_t)q(y|x_t)})$
  $u \leftarrow U \sim U[0, 1]$
  if $u \leq \alpha$ **then**
    $x_{t+1} \leftarrow y$
  else
    $x_{t+1} \leftarrow x_t$
  end if
end for

it becomes:

$$T(x_{t+1}|x_t) = q(x_{t+1}|x_t)\alpha(x_{t+1}|x_t) = q(x_{t+1}|x_t)\min(1, \frac{p(x_{t+1})}{p(x_t)}). \tag{2.3}$$

Note that $q(\cdot|\cdot)$ is symmetric, Equation 2.2 is then satisfied. Thus, once we move to a state that comes from the target distribution p, all further draws can be safely treated as coming from that distribution.

However, it is not always very efficient to use the symmetric proposals. For example, if the target distribution is defined on $[0, \infty)$, using an asymmetric proposal might be more appropriate. In this situation, it is more desirable to have the proposal distribution that fits the support of the target distribution. In such cases, the Metropolis-Hastings algorithm comes into play and we provide the pseudocode in Algorithm 2 to compare. We no longer have the constraint that the proposal distribution has to be symmetric. Compared with Algorithm 1, Hastings changed the acceptance ratio r to ensure the algorithm would work. In other words, the Metropolis-Hastings algorithm satisfies the detailed balance condition. To see how the above statement is valid, we need to prove Equation 2.2. Recall that Equation 2.1 is true for Metropolis-Hastings and we follow the exactly same steps as in the previous

page. The only difference is that Equation 2.3 now has the form:

$$T(x_{t+1}|x_t) = q(x_{t+1}|x_t)\alpha(x_{t+1}|x_t) = q(x_{t+1}|x_t)\min\left(1, \frac{p(x_{t+1})q(x_t|x_{t+1})}{p(x_t)q(x_{t+1}|x_t)}\right)$$
$$= \min\left(q(x_{t+1}|x_t), \frac{p(x_{t+1})q(x_t|x_{t+1})}{p(x_t)}\right).$$

Substituting $T(x_{t+1}|x_t)$ and $T(x_t|x_{t+1})$ into Equation 2.2 with the above form, we can see that both sides of Equation 2.2 now equal to $\min\left(p(x_t)q(x_{t+1}|x_t), p(x_{t+1})q(x_t|x_{t+1})\right)$. Thus the Metropolis-Hastings transition kernel satisfies detailed balance, and $p(\cdot)$ is the invariant distribution of $T(\cdot|x)$. Once the chain passes through the initial burn-in period that removes the dependence on the starting value, all the subsequent draws will come from $p(\cdot)$.

2.3 Gibbs Sampler

In Section 2.2, we present the classic Metropolis-Hastings algorithm. One issue with this algorithm is that as the number of dimensions gets higher, it becomes harder to find the appropriate proposal distribution. Since the behavior of the target distribution for each dimension varies, the proposal distribution has to match the behavior in all dimensions. For example, in a random walk type of Metropolis algorithm, as the dimension increases, the acceptance rate drops dramatically, which is known as the curse of dimensionality (Robert, 2014). In such a situation, instead of updating all the dimensions at once, we can divide the dimensions into blocks and update sequentially, which is known as the block-wise Metropolis-Hastings. That is to say, if $x \sim p(\cdot)$ is defined over a D-dimensional space, we can split x into k small blocks $\{x_{[1]}, x_{[2]}, ..., x_{[k]}\}$, such as $x = (x_{[1]}, x_{[2]}, ..., x_{[k]})$. Ideally we hope that within the same block, components are correlated and they are as independent as possible between blocks. Then we can update each $x_{[d]}$ at one time, holding all other blocks fixed. This method ignores the correlations of components between blocks and enables specialized sampling

Algorithm 3 Component-Wise Metropolis-Hastings

1: Initialize $x^{(0)} \sim \pi(x)$ // $\pi(\cdot)$ is the initial distribution
2: **for** $t = 0$ to $T - 1$ **do**
3: **for** $d = 1$ to D **do**
4: $y \leftarrow X_d \sim q_d(\cdot|x_d^{(t)}, x_{-d}^{(t)})$
5: $\alpha \leftarrow \min\left(1, \frac{p(y, x_{-d}^{(t)}) q_d(x_d^{(t)}|y, x_{-d}^{(t)})}{p(x_d^{(t)}, x_{-d}^{(t)}) q_d(y|x_d^{(t)}, x_{-d}^{(t)})}\right)$
6: $u \leftarrow U \sim U[0, 1]$
7: **if** $u \leq \alpha$ **then**
8: $x_d^{(t+1)} \leftarrow y$
9: **else**
10: $x_d^{(t+1)} \leftarrow x_d^{(t)}$
11: **end if**
12: **end for**
13: **end for**

strategy for different blocks. If each block contains exactly one component, this becomes a component-wise update procedure. We now only need to work on a univariate case, one component at a time. Now $x = (x_1, x_2, \ldots, x_D)$ and we let $x_{-d} = (x_1, \ldots, x_{d-1}, x_{d+1}, \ldots, x_D)$, *i.e.*, all components except x_d. Then $\{x_d, x_{-d}\}$ contains all components of x and we also let $x_{-d}^{(t)} = (x_1^{(t+1)}, \ldots, x_{d-1}^{(t+1)}, x_{d+1}^{(t)}, \ldots, x_D^{(t)})$, the fixed components when updating x_d at the t-th iteration. The pseudocode for component-wise Metropolis-Hastings is outlined in Algorithm 3. Note that the specialized proposal distribution $q_d(\cdot|\cdot)$ here is a univariate function of x_d. It may depend on the most recent values of any other components of x. While we are updating the d-th dimension, we keep all other dimensions fixed at the current values. The subscript $(t + 1)$ in the definition of $x_{-d}^{(t)}$ reveals that the corresponding components in dimension 1 up to $d - 1$ are newly updated whereas the others are kept at values at their previous iterations. Also note that in line 5 of Algorithm 3, we can further simplify the form of the acceptance rate α because of the component-wise update. Let $p(\cdot|x_{-d})$ stand for the

conditional distribution of x_d under $p(\cdot)$, we have

$$\begin{aligned}
\alpha &= \min\left(1, \frac{p(y, x_{-d}^{(t)}) q_d(x_d^{(t)} | y, x_{-d}^{(t)})}{p(x_d^{(t)}, x_{-d}^{(t)}) q_d(y | x_d^{(t)}, x_{-d}^{(t)})}\right) \\
&= \min\left(1, \frac{p(y | x_{-d}^{(t)}) p(x_{-d}^{(t)}) q_d(x_d^{(t)} | y, x_{-d}^{(t)})}{p(x_d^{(t)} | x_{-d}^{(t)}) p(x_{-d}^{(t)}) q_d(y | x_d^{(t)}, x_{-d}^{(t)})}\right) \qquad (2.4) \\
&= \min\left(1, \frac{p(y | x_{-d}^{(t)}) q_d(x_d^{(t)} | y, x_{-d}^{(t)})}{p(x_d^{(t)} | x_{-d}^{(t)}) q_d(y | x_d^{(t)}, x_{-d}^{(t)})}\right). \qquad (2.5)
\end{aligned}$$

Note that in Equation 2.4 the term $p(x_{-d}^{(t)})$, the marginal distribution of $x_{-d}^{(t)}$, cancels out because it does not change during this update. The simplified form of α (Equation 2.5) maybe more convenient to compute. For instance, it simplifies the computation if the target distribution $p(\cdot)$ is intrinsically defined by the form of conditional distributions, as is often the case in Bayesian hierarchical models. Sometimes the conditional distribution of each component may be known and not hard to sample from, which leads to a special case of the component-wise Metropolis-Hastings algorithm, Gibbs sampler. It was first described by Geman and Geman (1987) and then introduced to the statistical community by Gelfand and Smith (1990). As just stated, the method is applicable when the conditional distribution of each component has an analytical form, conditioning on all other components. This allows us to update each component iteratively. Specifically, we set the conditional distribution as the proposal distribution, *i.e.*, $q_d(\cdot|x) = p(\cdot|x_{-d})$, when updating the d-th component. Putting this back into Equation 2.5, we get

$$\begin{aligned}
\alpha &= \min\left(1, \frac{p(y | x_{-d}^{(t)}) q_d(x_d^{(t)} | y, x_{-d}^{(t)})}{p(x_d^{(t)} | x_{-d}^{(t)}) q_d(y | x_d^{(t)}, x_{-d}^{(t)})}\right). \\
&= \min\left(1, \frac{p(y | x_{-d}^{(t)}) p(x_d^{(t)} | x_{-d}^{(t)})}{p(x_d^{(t)} | x_{-d}^{(t)}) p(y | x_{-d}^{(t)})}\right) \\
&= \min(1, 1) \\
&= 1,
\end{aligned}$$

which shows that the proposals are accepted for sure. By decomposing the target distribution into full conditionals that are tractable, the method attempts to alleviate the problem with high dimensional targets. The pseudocode for Gibbs sampler is outlined below. Again, $p(\cdot|x_{-d})$ denotes the conditional distribution of x_d under $p(\cdot)$ and while we are updating the d-th dimension, we keep all other dimensions fixed at the most recent values.

Algorithm 4 Gibbs Sampler

Initialize $x^{(0)} \sim \pi(x)$ // $\pi(\cdot)$ is the initial distribution
for $t = 0$ to $T - 1$ **do**
  for $d = 1$ to D **do**
    $x_d^{(t+1)} \leftarrow X_d \sim p(\cdot|x_{-d}^{(t)})$
  end for
end for

So far we have briefly introduced the Metropolis-Hastings algorithm and Gibbs sampler. Essentially, Gibbs sampler is a special Metropolis-Hastings algorithm. It is a hybrid type of algorithm, where in each step the method has a Metropolis-Hastings jumping kernel. The algorithm can be very helpful in certain scenarios, e.g., in Bayesian hierarchical models. The conditionals are in closed forms whereas the multidimensional target distribution as a whole is hard to handle. A famous example would be the Ising model, where the joint distribution has 2^n possible states but each conditional is just a coin flipper (Geman and Geman, 1987). The conditional-independence structure can be utilized to simplify the calculation of the full conditionals at each step, especially if these are well-known distributions like Gamma, Poisson or Binomial. However, if components are highly correlated, Gibbs sampler can be difficult to move around efficiently. It is hard to update one component without simultaneously changing others. Metropolis-Hastings is a more general method. For instance, in Bayesian model selection problems, which is introduced in Chapter 3, different candidate models may

have parameter spaces of different dimensions. In such a situation, Gibbs sampler cannot be applied. Like Gibbs sampler, Metropolis-Hastings also suffers from the convergence problem. Section 2.4 describes the case of the Metropolis-Hastings algorithm.

2.4 Convergence of the Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is widely used to construct a Markov chain with some stationary distribution. However, sometimes the algorithm is slow to converge. Two variables that significantly affect convergence are the starting value and the proposal distribution. The choice of the starting value will affect the length of the burn-in period, thus affect the efficiency of the convergence. If the rate of convergence is not excessively slow, one possible solution is to simply run the chain longer. As long as the chain is irreducible, by the Ergodic Theorem it will eventually converge (Roberts et al., 2004). The other variable to specify is the proposal distribution, that will greatly affect the speed of convergence. The choice of the proposal distribution is much more crucial, which requires fine tuning. For example, in random walk type Metropolis, the most popular proposals include the multivariate normal and multivariate t distributions. Even the type of the proposal distribution has been selected, we still need to carefully decide the scale of the proposal distributions. In multivariate normal case, we need to consider the scale of the variance for each parameter. This determines the size of a potential move, also known as the step size. For the random walk type Metropolis algorithm, step size is a crucial parameter. An inappropriate step size will result in a slow mixing Markov chain. The term mixing here is to describe how well the chain explores the posterior sample space. We use Figure 2.1 from Hartig (2011) to illustrate. It represents two random walk type of Metropolis situations and perfectly shows the trade-off between the step size and the acceptance rate. In the upper panel, suppose our proposal distribution is narrow compared to the target distribution. Each step it will only move slightly away

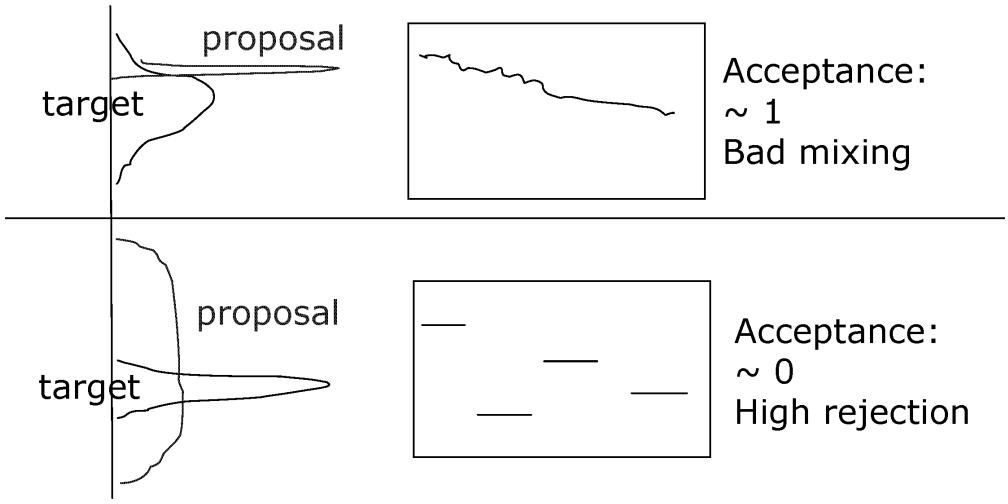

Figure 2.1: Problems in the proposal distribution and the corresponding traceplots (Hartig, 2011).

from the current iteration. Hence the successive iterations are very heavily correlated. Even though the chain has a high acceptance rate, it has difficulty moving around. The traceplot shows tiny vacillation and indicates a bad mixing, as the chain explores the parameter space very slowly. In the lower panel, the proposal distribution is too wide compared to the target distribution, and the proposed moves are highly likely to be rejected because typically they land in low density regions. Hence the acceptance rate is near zero and most of the time the chain stays where it was. This simple example highlights the importance of choosing a good proposal distribution, especially the step size in the random walk type Metropolis proposal. Ideally one would like to choose a step size so that the acceptance rate is not too extreme. In these random walk Metropolis algorithms, Roberts et al. (1997) suggests an asymptotically optimal value of around 0.23 based on multivariate normal targets with multivariate normal proposals. Note that in this unimodal case (Figure 2.1), we can adjust the step size to get a reasonable acceptance rate. It could be an issue even in unimodal cases, and the issue becomes more exasperate when we have multimodal distributions. For multimodal cases, poor convergence can happen if chains are near reducible. For example,

the Metropolis-Hastings algorithm has difficulties exploring the sample space if modes are well separated. The chain is easy to get stuck at the local mode. This is known as the local trap problem due to the inability of the Metropolis-Hastings algorithm to step over valley of low density and move to another mode. The local trap problem needs to be addressed and we are motivated to alleviate the problem and to design more efficient Metropolis-Hastings algorithms.

Our goal is to make the Markov chain to converge faster, and we have discussed possible remedies to improve the convergence. To sum up, if the chain converges slow, we may run it longer; if really slow, we may tune the algorithm to make it converge faster. For instance, we can adjust the scale of the proposal distribution in random walk type Metropolis algorithms. However, if the chain converges super slow, we may have to alter the algorithm in a serious way. For example, we can do some reparameterization or add some auxiliary variable, to encourage jumps between modes. See Kass et al. (1998). In Section 2.5 we discuss several new strategies for improving convergence including reparameterization, slice sampling, Hamiltonian Monte Carlo, simulated annealing, simulated tempering and reversible jump MCMC.

2.5 Strategies for Improving Convergence

2.5.1 Reparameterization

Both Metropolis-Hastings and Gibbs sampler can be sensitive to the choice of the model parameterization. Using a different parameterization may help to decorrelate parameters or the transformed posterior may be easier to sample from. The former, for example, Gelfand et al. (1995) demonstrate the hierarchical centering approach that aims to reduce the correlations between parameters in the joint posterior distribution; and the latter, for example,

to deal with a target distribution that is heavily right-skewed, taking the log of a parameter can help so as to get rid of the fat tails. Such methods are important in the sense that they are both straightforward and effective. Another classic example is illustrated by Gilks and Roberts (1996), where we only need to do a simple linear transformation. The posterior becomes roughly spherical and the Metropolis algorithm works. For more complex reparameterization strategies, one way is to add auxiliary variables, e.g., by putting jumps between different regions of the posterior. We give a detailed introduction in Section 2.5.2.

2.5.2 Auxiliary Variable Methods

Recall that in Gibbs sampler, we manage to sample from a lower dimensional space by updating one parameter at a time. Sometimes, however, we can reverse the idea and sample from a higher dimensional space by introducing auxiliary variables. Auxiliary variables are additional variables that are not in the original model but are introduced to help the chain move more rapidly towards the target distribution. While the original target distribution is hard to sample from, exploring the joint space may be more efficient. In other words, suppose we want to sample from $p(x)$ and we introduce an auxiliary variable y such that $p(x) = \int p(x, y)dy$, where $p(x, y)$ is the joint distribution of (x, y). We can also specify the conditional distribution $p(y|x)$ and then $p(x, y) = p(x)p(y|x)$. If $p(x|y)$ and $p(y|x)$ are easy to sample from, we can construct a Markov chain on (x, y) via e.g., Gibbs sampler, and alternately update x and y. In this way we make it easier to navigate $p(x, y)$, and hence $p(x)$. We discard the auxiliary variable y and only the original variable x is retained in the Monte Carlo samples. Another family of auxiliary variable methods is to construct a joint distribution on (x, y) such that the target distribution $p(x)$ equals the conditional distribution $p(x|y = y^*)$, *i.e.*, on some set of values y^* of y. An example that acts in this manner is simulated tempering and is discussed in Section 2.5.3. Before that, we first discuss two methods, slice sampling and Hamiltonian Monte Carlo, that add auxiliary variables to

facilitate sampling.

Slice Sampling

Slice sampling is a family of auxiliary variable methods that adopts the principle of the acceptance-rejection algorithm. That is, to sample from a distribution, it is equivalent to sample uniformly from the region under that target distribution. Suppose we want to sample from $p(x)$, we introduce an auxiliary variable y that is uniformly distributed on the interval $(0, p(x))$ conditioning on x. Let $U = \{(x, y) : 0 < y < p(x)\}$ and now the joint distribution $p(x, y)$ is uniformly distributed over U:

$$p(x, y) = p(x)p(y|x) \propto \begin{cases} 1, & 0 < y < p(x), \\ 0, & \text{otherwise.} \end{cases}$$

The introduction of auxiliary variable y allows us to sample under the density function plot $p(x)$. We can use Gibbs sampler to get samples of (x, y) jointly and then simply discard y. The pseudocode for slice sampling is outlined below.

Algorithm 5 Slice Sampling

1: Initialize $x_0 \sim \pi(x)$ // $\pi(\cdot)$ is the initial distribution
2: **for** $t = 0$ to $T - 1$ **do**
3: Sample y_{t+1} from $U(0, p(x_t))$
4: Sample x_{t+1} uniformly from the slice $\{x : f(x) > y_{t+1}\}$
5: **end for**

The algorithm updates y and then x alternately from the conditional distributions, where x can be multiple dimensional. At each iteration, the auxiliary variable y defines a horizontal slice of the distribution, and next x is sampled uniformly over the union of intervals that

comprises this slice (Neal, 2003). Compared to the Metropolis methods, slice sampling has the advantage of no parameter tuning. It automatically adapts to the local characteristics of the target distribution so we do not need to tune the proposal distribution or the step size manually. Thus slice sampling is a good choice if we do not have much knowledge of the target distribution. Like Gibbs sampler, slice sampling has no rejections and can suffer from the serial correlation. In contrast to Gibbs sampler, it does not require the conditional distributions to be in analytical from. Slice sampling is useful to sample from multimodal distributions, if modes are not separated by large valleys of low probability, since it is free to move between modes within a slice. However, one problem of slice sampling is that sampling uniformly from the region $\{x : f(x) > y\}$ can be hard or unfeasible. We either need to inverse the target distribution or to find appropriate estimates. Next we briefly introduce Hamiltonian Monte Carlo, which takes advantage of auxiliary variables to generate new proposals.

Hamiltonian Monte Carlo

In high dimensions, the Metropolis-Hastings method and Gibbs sampler are often inefficient due to random walk type proposals. To reduce the random walk behavior that leads to slow mixing, Duane et al. (1987) propose Hamiltonian Monte Carlo, an MCMC method that utilizes gradient information to better explore the target distribution $p(x)$. So far we have seen target distributions only being used to evaluate the density, but they are often differentiable and the gradients can be helpful in guiding the updates of samples. Hamiltonian Monte Carlo introduces an auxiliary variable y, a vector of momentum variables with the same length as x. This allows us to simulate Hamiltonian dynamics as the Metropolis proposal distribution. We have the total energy or Hamiltonian function $H(x, y) = U(x) + K(y)$ and it is related to the joint distribution $p(x, y)$ through the canonical distribution $\frac{1}{Z}e^{\frac{-H(\cdot)}{T}}$ with

$T = 1$:

$$\begin{aligned} p(x, y) &\propto e^{-H(x,y)} \\ &\propto e^{-(U(x)+K(y))} \\ &\propto e^{-U(x)} e^{-K(y)}, \end{aligned}$$

where $U(\cdot)$ denotes the potential energy of location x and $K(\cdot)$ denotes the kinetic energy of momentum y. Since $p(x, y)$ is decomposed into two parts, we see that x and y are independent. Therefore we can use arbitrary distribution for x and one convenient option is a standard normal distribution so as the resulting density is exactly the quadratic kinetic energy with unit mass, *i.e.*, $K(y) = y^T y/2$. For the potential energy function $U(\cdot)$, we usually define it as the negative logarithm of the target distribution $p(\cdot)$, *i.e.*, $U(x) = -\log p(x)$. We implement the leapfrog method to update the pair of (x, y) jointly in the augmented space and the momentum variable y will guide the move of the target variable x. Let $\nabla U(\cdot)$ be the gradient of the potential energy function $U(\cdot)$, L be the number of leapfrog steps and δ be the step size. The pseudocode for Hamiltonian Monte Carlo is outlined in Algorithm 6. After we get samples from the Markov chain we can just simply discard the momentum variable y. The advantage of Hamiltonian Monte Carlo over Metropolis-Hastings is that the draws are less correlated and more efficient. The disadvantage is that tuning L and δ can be very difficult. See Neal et al. (2011). Besides, the method struggles with multimodality. In Section 2.5.3, we introduce a family of MCMC methods that aims to sample from multimodal distributions.

2.5.3 Annealing Method

Annealing is a process to heat the metal and then to cool it down slowly to become hard. In a thermodynamic system, to minimize the free energy, we want to slowly decrease the

Algorithm 6 Hamiltonian Monte Carlo

Initialize $x_0 \sim \pi(x)$ // $\pi(\cdot)$ is the initial distribution
for $t = 0$ to $T - 1$ **do**
  $y_t \leftarrow N(0, I)$
  $x^* \leftarrow x_t, y^* \leftarrow y_t$
  for $l = 1$ to L **do**
    $y^* \leftarrow y^* - \frac{\delta}{2}\nabla U(x^*)$
    $x^* \leftarrow x^* + \delta y^*$
    $y^* \leftarrow y^* - \frac{\delta}{2}\nabla U(x^*)$
  end for // leapfrog method
  $\alpha \leftarrow \min\left(1, e^{H(x_t, y_t) - H(x^*, y^*)}\right)$
  $u \leftarrow U \sim U[0, 1]$
  if $u \leq \alpha$ **then**
    $x_{t+1} \leftarrow x^*$
  else
    $x_{t+1} \leftarrow x_t$
  end if
end for

temperature of the system. When the temperature is high, random kinematic fluctuation is strong and the system can be trapped in local minima. By lowering the temperature slowly, we are able to decrease the defects in the metal and reach the minimum system energy. Inspired by this, the annealing algorithm is an optimization method that mimics such dynamics. It was first proposed by Metropolis et al. (1953) and then further developed by Kirkpatrick et al. (1983). Specifically, we briefly describe simulated annealing and simulated tempering, and leave the topic of parallel tempering in Section 2.6.

Simulated Annealing

Simulated annealing combines the idea of the thermal cooling process into the Metropolis algorithm. Suppose $p(x)$ is denoted in terms of the energy function $E(x)$:

$$p(x) \propto e^{-E(x)}$$

Algorithm 7 Simulated Annealing

Initialize $x_0 \sim \pi(x)$ // $\pi(\cdot)$ is the initial distribution
for $t = 0$ to $N - 1$ **do**
  $T \leftarrow A(\frac{t}{N})$ // $A(\cdot)$ is the annealing schedule
  $y \leftarrow X \sim q(\cdot|x_t)$
  $\alpha \leftarrow \min(1, \frac{p_T(y)}{p_T(x_t)})$
  $u \leftarrow U \sim U[0, 1]$
  if $u \leq \alpha$ **then**
    $x_{t+1} \leftarrow y$
  else
    $x_{t+1} \leftarrow x_t$
  end if
end for

and we define a modified distribution $p_T(x)$ that

$$p_T(x) \propto e^{-\frac{E(x)}{T}}.$$

where the exponent is proportional to the inverse temperature $\frac{1}{T}$. The Metropolis algorithm is then applied on $p_T(x)$ instead. We initialize the temperature parameter T at a high value and gradually decrease it to 1. When $T = 1$, $p_T(x)$ corresponds to the target distribution $p(x)$. When T is high, $p_T(x)$ is flat so it is easy to access anywhere in the support. In other words, the chain can escape from local modes when the temperature is high enough. As the temperature slowly decreases at each iteration, it becomes possible to communicate between high probability regions for the target distribution. In cases where $E(x)$ can be decomposed into two terms, *i.e.*, $E(x) = E_1(x) + E_2(x)$, we have a refined simulated annealing algorithm in which $p_T(x)$ has the form

$$p_T(x) \propto e^{-E_1(x) - \frac{E_2(x)}{T}},$$

where $E_1(x)$ has nice properties such as a convex function of x and the more nasty term $E_2(x)$ is controlled by the temperature T. The pseudocode for the simulated annealing algorithm is outlined in Algorithm 7. By properly handling the temperature parameter T, simulated an-

nealing provides a way to sample from a potentially multimodal distribution $p(x)$. However, we can see the method as described above does not guarantee exactly sampling from the target distribution. Moreover, Brown and Head-Gordon (2003) suggest that the algorithm can still get stuck in local optima even the cooling is carefully scheduled. To remedy this, Marinari and Parisi (1992), Geyer and Thompson (1995) proposed the simulated tempering algorithm that treats T as an augmented random variable.

Simulated Tempering

In simulated tempering, the temperature T is no longer in a decreasing order of cooling down. Instead, it is introduced as an auxiliary random variable and the algorithm will sample x and T jointly. Since the purpose is to sample from $p(x)$, we form a temperature ladder $T_1 > T_2 > \cdots > T_m = 1$ and define $p_i(x)$ as

$$p_i(x) = \frac{1}{Z_i} e^{-\frac{E(x)}{T_i}}, \qquad i = 1, 2, \ldots, m,$$

where Z_i is the normalizing constant and $p_m(x) = p(x)$ is the target distribution. We perform a random walk on the temperature ladder $\{T_i\}_i^m$ and allow the current temperature go to nearest neighbors. Specifically, let $A(j|i)$ be the transition probability from T_i to T_j, we have $A(2|1) = A(m-1|m) = 1$ and $A(i+1|i) = A(i-1|i) = 0.5$ for $1 < i < m$, according to Geyer and Thompson (1995). We denote i_t as the temperature level at the t-th iteration, start with $i_0 = 1$ and alternatively update x and T in a Gibbs sampler fashion. The pseudocode for the simulated tempering algorithm is outlined in Algorithm 8. By enabling pacing up and down between different temperature levels, simulated tempering is successful to locate multiple modes for a multimodal distribution. As the temperature goes up, the chain moves freely to explore the sample space. As it goes down, the chain converges to the mode. The algorithm outperforms the simulated annealing method in terms of the ability to escape from local

Algorithm 8 Simulated Tempering

Initialize $(i_0, x_0) \sim \pi(\cdot)$ // $\pi(\cdot)$ is the initial distribution
for $t = 0$ to $N - 1$ **do**
  $y \leftarrow X \sim q(\cdot|x_t, i_t)$
  $\alpha \leftarrow \min\left(1, \frac{p_{i_t}(y)q(x_t|y,i_t)}{p_{i_t}(x_t)q(y|x_t,i_t)}\right)$
  $u \leftarrow U \sim U[0,1]$
  if $u \leq \alpha$ **then**
    $x_{t+1} \leftarrow y$
  else
    $x_{t+1} \leftarrow x_t$
  end if
  $j \leftarrow A(\cdot|i_t)$ // $A(\cdot|\cdot)$ is the temperature transition operator
  $\beta \leftarrow \min\left(1, \frac{c(j)p_j(x_{t+1})A(i_t|j)}{c(i_t)p_{i_t}(x_{t+1})A(j|i_t)}\right)$ // $c(\cdot)$ is the tunable constant
  $v \leftarrow U \sim U[0,1]$
  if $v \leq \beta$ **then**
    $i_{t+1} \leftarrow j$
  else
    $i_{t+1} \leftarrow i_i$
  end if
end for

optima because of the varying temperature. On the other end, since the system is moving up and down within the temperature ladder and we only need the samples with $T = 1$, simulated tempering may be slow and inefficient (Li et al., 2004). In addition, estimation of the normalizing constants $Z_i's$ can also be computationally expensive. Regarding the computational cost, we introduce an efficient variant, the parallel tempering method, in Section 2.6 after we introduce yet another algorithm called reversible jump MCMC.

2.5.4 Reversible Jump MCMC

In this section we introduce another type of MCMC method that is designed to sample from complicated distributions, reversible jump MCMC. It is proposed by Green (1995) as an extension to the general Metropolis–Hastings algorithm. The method is very useful in cases where the dimension of the sample space may vary from one iteration to the next in

Algorithm 9 Reversible Jump MCMC

Initialize $(i_0, x_0) \sim \pi(\cdot)$ // $\pi(\cdot)$ is the initial distribution
for $t = 0$ to $T - 1$ **do**
  $j \leftarrow R \sim r(\cdot|i_t)$
  $u \leftarrow Q \sim q_{i_t,j}(\cdot)$
  $(x', u') \leftarrow h_{i_t,j}(x_t, u)$
  $\alpha \leftarrow \min\left(1, \frac{p(j,x'|D)r(i_t|j)q_{i_t,j}(x',u')}{p(i_t,x_t|D)r(j|i_t)q_{j,i_t}(x,u)} \mid \det\left(\frac{\partial h_{i,j}(x,u)}{\partial(x,u)}\right) \mid\right)$
  $v \leftarrow U \sim U[0,1]$
  if $v \leq \alpha$ **then**
    $(i_{t+1}, x_{t+1}) \leftarrow (j, x')$
  else
    $(i_{t+1}, x_{t+1}) \leftarrow (i_t, x_t)$
  end if
end for

the Markov chain. For example, in Bayesian model selection, we are comparing multiple models and are often interested in conducting the joint inference. We describe a concrete example in Chapter 4. The method is described as follows. We observe data D and have m candidate models $\{M_i\}_i^m$. Let θ_i and n_i denote the model parameters and the model dimension respectively for the i-th model, *i.e.*, $dim(\theta_i) = n_i$ for $i = 1, 2, \ldots, m$. Note that the models need not have the same dimensions. The joint posterior of (i, θ_i) is given by the product of the data likelihood $l(D|i, \theta_i)$ and the joint prior $p(i, \theta_i)$:

$$p(i, \theta_i|D) \propto p(i, \theta_i)l(D|i, \theta_i) \propto p(i)p(\theta_i|i)l(x|i, \theta_i), \tag{2.6}$$

where $p(i)$ denotes the prior of model i and $p(\theta_i|i)$ denotes the prior of θ_i under M_i. Equation 2.6 is the target distribution of the Markov chain. The reversible jump MCMC method samples over the joint state space $\bigcup_{i=1}^m (\{i\} \times \mathcal{R}^{n_i})$ and the goal is to be able to move among different models. Remember that they may no have the same dimensions, so to successfully perform trans-dimensional moves, we introduce an auxiliary random variable u that is used for dimension matching. This is necessary to construct a reversible chain. So after proposing a new model j given the current model i with probability $r(j|i)$, we generate u from $q_{i,j}(\cdot)$ to match the dimension. Then the current state (x, u) under model i is mapped to the

proposed new state (x', u') under model j through a deterministic bijection function $h_{i,j}(\cdot)$. And this proposal is updated according to the Metropolis-Hastings rule. If we consider the reverse jump from x' to x, u' is generated from $q_{j,i}(\cdot)$ and the mapping from (x', u') to (x, u) is through $h_{j,i}(\cdot)$, the inverse function of $h_{i,j}(\cdot)$. The pseudocode for the reversible jump MCMC algorithm is outlined in Algorithm 9. It is widely used in trans-dimensional sampling but choosing the efficient proposals is very challenging. In Section 2.6 we introduce a family of MCMC method that can be used in complicated problems including trans-dimensional sampling.

2.6 Population-Based MCMC Methods

An alternative approach to alleviate the local-trap problem is known as the population-based MCMC method. The idea is that instead of running a single Markov chain, we run a population of chains in parallel. The corresponding stationary distributions can be different but are related. The feature is that information exchange among parallel chains enables the target chains to learn from sampling history and to adjust the sampling distribution, so as to improve the convergence of the target chains. Examples of this type of method includes adaptive direction sampling, parallel tempering and so on. Following the previous section of annealing methods, we now briefly describe parallel tempering.

Adopting the same settings in Section 2.5.3, we have $p(x)$ as the target distribution over a state space $\mathcal{X}$. Formulated by Geyer (1991), the parallel tempering method simulates m parallel chains at different temperature levels $\{T_i\}_i^m$. The method can also be interpreted as constructing a single Markov chain on the product space $\mathcal{X}^m$, where the target distribution is $\prod_{i=1}^m p_i(x)$. That is, we have a temperature ladder $T_1 > T_2 > \cdots > T_m = 1$ and the method simulates a sequence of the corresponding distributions

$$p_i(x) \propto e^{-\frac{E(x)}{T_i}}, \qquad i = 1, 2, \ldots, m,$$

Algorithm 10 Parallel Tempering

Initialize $x_1^0, ...x_m^0 \sim \pi(\cdot)$ // $\pi(\cdot)$ is the initial distribution
for $t = 0$ to $N - 1$ **do**
  for $i = 1$ to m **do**
    $x \leftarrow X \sim q(\cdot|x_i^t)$
    $\alpha \leftarrow \min\left(1, \frac{p_i(y)q(x_i^t|y)}{p_i(x_i^t)q(y|x_i^t)}\right)$
    $u \leftarrow U \sim U[0,1]$
    if $u \leq \alpha$ **then**
      $x_i^{t+1} \leftarrow x$
    else
      $x_i^{t+1} \leftarrow x_i^t$
    end if
  end for // within-chain update
  for some $i's$ **do**
    $j \leftarrow S \sim A(\cdot|i)$
    $\beta \leftarrow \min\left(1, e^{(H(x_i^{t+1}) - H(x_j^{t+1}))(\frac{1}{T_i} - \frac{1}{T_j})}\right)$
    $v \leftarrow U \sim U[0,1]$
    if $v \leq \beta$ **then**
      $x_i^{t+1} \rightleftharpoons x_j^{t+1}$
    end if
  end for // state swapping update
end for

where $p_m(x) = p(x)$ is the target distribution. Interactions are made by swapping states between adjacent chains to improve mixing. Let $A(j|i)$ be the probability that chain j is proposed to exchange state with chain i, we have $A(2|1) = A(m-1|m) = 1$ and $A(i+1|i) = A(i-1|i) = 0.5$ for $1 < i < m$. Similar to simulated tempering, the swap is usually proposed between neighbors as their target distributions are less different thus the proposal is less likely to be rejected. The Metropolis-Hastings rule is applied to the state swapping operation to ensure the chains converge to the desired distributions. The pseudocode for the parallel tempering algorithm is outlined in Algorithm 10. By allowing state swaps, the target chain is able to visit states explored by those high temperature chains. The parallel tempering method is a modification to the standard Metropolis-Hastings algorithm and proves to be very powerful in sampling complicated distribution. Compared with simulated annealing, parallel tempering is an exact method. Compared with simulated tempering,

parallel tempering does not require estimates of the normalization constants. However, the method struggles if modes are completely separate in the target distribution. They cannot communicate with each other even raising the temperature. To overcome the problem, we propose a novel population-based MCMC method in Chapter 3. In addition, we present more aspects of the family of the population-based MCMC methods.

Chapter 3

A New Multiple Chain Method

As we have introduced and reviewed the existing MCMC methods in the previous chapter, we now describe our proposed population-based MCMC method, the multiple chain method.

3.1 Method Introduction

We propose a novel population-based MCMC method for sampling from complicated multimodal distributions, which is particularly challenging when modes are completely separate, *i.e.*, modes cannot communicate with each other. Based on the Metropolis type algorithm, we construct a multiple chain method by proposing a novel between-chain jump to encourage full exploration of the parameter space. To be specific, after we initialize multiple chains from dispersed starting values, we perform a two-step jump with the usual within-chain jump and our proposed between-chain jump. The intuition behind our method is that, by running multiple chains exploring different modes and enabling between-chain jumps, each chain jumps between modes efficiently and explores the entire parameter space. As a result, we are able to sample from complicated multimodal distributions with completely separate

Algorithm 11 The Multiple Chain Method

Input: Current state $x^t = (x_1^t, \ldots, x_m^t)$, within-chain jumping kernel q, between-chain jumping kernel g_0

Output: Next state $x^{t+1} = (x_1^{t+1}, \ldots, x_m^{t+1})$

for $i = 1$ to m **do**
  $x \leftarrow X \sim q(\cdot|x_i^t)$
  $\alpha \leftarrow \min\left(1, \frac{p(x)}{p(x_i^t)}\right)$
  $u \leftarrow U \sim U[0,1]$
  if $u \leq \alpha$ **then**
    $x_i^t \leftarrow x$
  end if // within-chain jump
  $j \leftarrow S \sim \{1, 2, ..., m\} \backslash \{i\}$
  $y \leftarrow Y \sim g_0(y|x_j^t)$
  $\beta \leftarrow \min\left(1, \frac{p(y)}{p(x_i^t)} \frac{g_i(x_i^t)}{g_i(y)}\right)$
  $v \leftarrow U \sim U[0,1]$
  if $v \leq \beta$ **then**
    $x_i^{t+1} \leftarrow y$
  else
    $x_i^{t+1} \leftarrow x_i^t$
  end if // between-chain jump
end for
return $x_1^{t+1}, \ldots, x_m^{t+1}$

modes. Our method has two highlighted features: first, we introduce a new proposal function in between-chain jump, *i.e.*, the average of densities of the proposed point evaluated at all other chains; secondly, our method allows trans-dimensional moves, which is beneficial since samples may not have the same dimensions, as further illustrated by the Bayes factor example in Chapter 4.

As an illustration, consider a multimodal distribution with the corresponding unnormalized density $p(\cdot)$, whose modes are completely separate. Our method is able to sample from $p(\cdot)$. Basically, we run m Markov chains in parallel for N iterations. Note that our method does not aim at finding new modes. Therefore, we initialize the starting state $x^0 = (x_1^0, \ldots, x_m^0)$ based upon the principle that there is at least one chain near each mode, and update the state iteratively with predefined within-chain jumping kernel q and between-chain jumping kernel g_0. Algorithm 11 illustrates one iteration of our proposed method. In detail, at each

iteration, we propose a two-step jump from the current state: a within-chain jump followed by a between-chain jump. As shown by lines $2-7$ of Algorithm 11, in the within-chain jump, a candidate point is proposed from the neighborhood of the current state, which helps the chain explore the local mode. On the contrary, in the following between-chain jump described by lines $8-16$, a candidate point is proposed from the neighborhood of another chain's current state, which allows the chain to jump between modes and capture the global structure. Specifically, when updating the i-th chain, we generate a proposal of evaluating the average density at all chains except the current chain:

$$g_i(y) = \frac{1}{m-1} \sum_{k \neq i} g_0(y|x_k^t). \tag{3.1}$$

As our method is a Metropolis-Hastings algorithm, we need to work out the Metropolis-Hastings rule. Since we are running m parallel chains and they have the same target density $p(\cdot)$, the target distribution here is actually a product of the target density for individual chains, i.e., $\prod_{i=1}^{m} p(x_i) = p(x_i) \prod_{j \neq i} p(x_j)$. Note that we keep all other iterates fixed at their current values when updating x_i, $\prod_{j \neq i} p(x_j)$ thus cancels out in the acceptance ratio evaluation at line 10 of Algorithm 11, leaving only one term $\frac{p(y)}{p(x_i)}$. The proposal distribution $g(\cdot)$, as we just mentioned above, is to keep other iterates unchanged while the current chain moves to a random sample from the neighbors of other iterates. Therefore, the form of such proposal distribution is a mixture of densities of $g_0(\cdot)$ as is shown in Equation 3.1. The example of $g_0(\cdot)$ here can be a univariate normal random walk type kernel, or much more general if necessary. Such specific proposal and the Metropolis-Hastings algorithm guarantee the validity of our algorithm with the resulting Metropolis-Hastings acceptance ratio in $\frac{p(y)}{p(x_i)} \frac{g_i(x_i)}{g_i(y)}$.

Note that this type of between-chain jumping transition kernel is closely related to but also different from adaptive MCMC (Athreya and Atuncar, 1998; Roberts and Rosenthal, 2009). For adaptive MCMC, we can construct a kernel density estimator based on past samples and

propose new samples from this estimator. To be concrete, we specify a kernel, place it on every past sample and obtain a mixture distribution as the proposal. For example, we run 1,000 iterations and obtain a number of samples. Since the sampling distribution of these samples are supposed to be close to the posterior distribution, we attempt to approximate the posterior from these samples, *e.g.*, to construct a kernel smoother based on the samples. In our method, we also use a kernel density estimator as the proposal. However, our method differs from adaptive MCMC in the fact that the proposal distribution is constructed from all current iterates in other chains. Instead of building the kernel density based on past samples (horizontally), we form the mixture density with all current iterates in other chains (vertically). Suppose we run 20 chains, the proposal we have is a mixture of normals centered at those 19 iterates; then we sample from this proposal for the future draws. In other words, we randomly pick one different iterate from the other 19 chains in the current sample, move it around a little bit, and make such an iterate the new proposal for the next iteration. We will present an illustrative example to show how our method works in the next section.

3.2 An Illustrative Example

In this toy example, we aim to sample from a normal mixture of two components. Suppose the target distribution is a mixture of two univariate normals. One is centered at 0 and the other is centered at 100. Both variances are equal to 1 and the mixing proportion is 0.7 for the left mode. That is, we have the target distribution as

$$\lambda N(\mu_1, \sigma_1^2) + (1 - \lambda) N(\mu_2, \sigma_2^2),$$

where $\lambda = 0.7, \mu_1 = 0, \mu_2 = 100$ and $\sigma_1^2 = \sigma_2^2 = 1$. The density plot is shown in Figure 3.1.

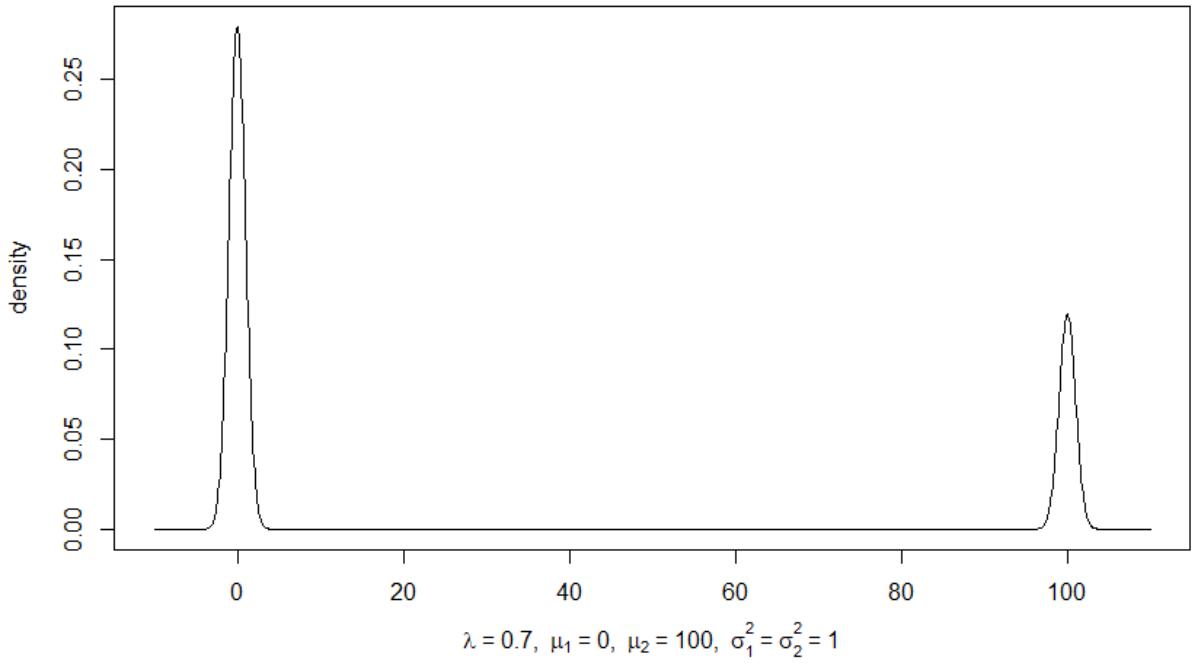

Figure 3.1: Density plot of the normal mixture: $\lambda N(\mu_1, \sigma_1^2) + (1 - \lambda)N(\mu_2, \sigma_2^2)$.

We apply the usual Metropolis method and the multiple chain method with 20 parallel chains. Parameters such as the step size are tuned so that both methods have reasonable and comparable acceptance rates (25%). Figure 3.2 shows the dotplots of simulated draws comparing Metropolis and our method. We plot 1500 draws after burn-in the first 25%, of three replications. In the left panel, we see that Metropolis is stuck in one mode. The top chain only explores the normal centered at 0 (left mode) while the other two chains are stuck at the right mode. This is due to its inability of stepping over valleys of low density and jumping to another mode. Our method, however, is able to visit both modes frequently and keep the right proportion. Note that in the right panel, all three chains jump between and within modes. By checking the values of samples, we can see that 70% of the time samples stay in the left mode, which matches the correct mixing proportion. This toy example demonstrates that the multiple chain method is able to sample from multimodal

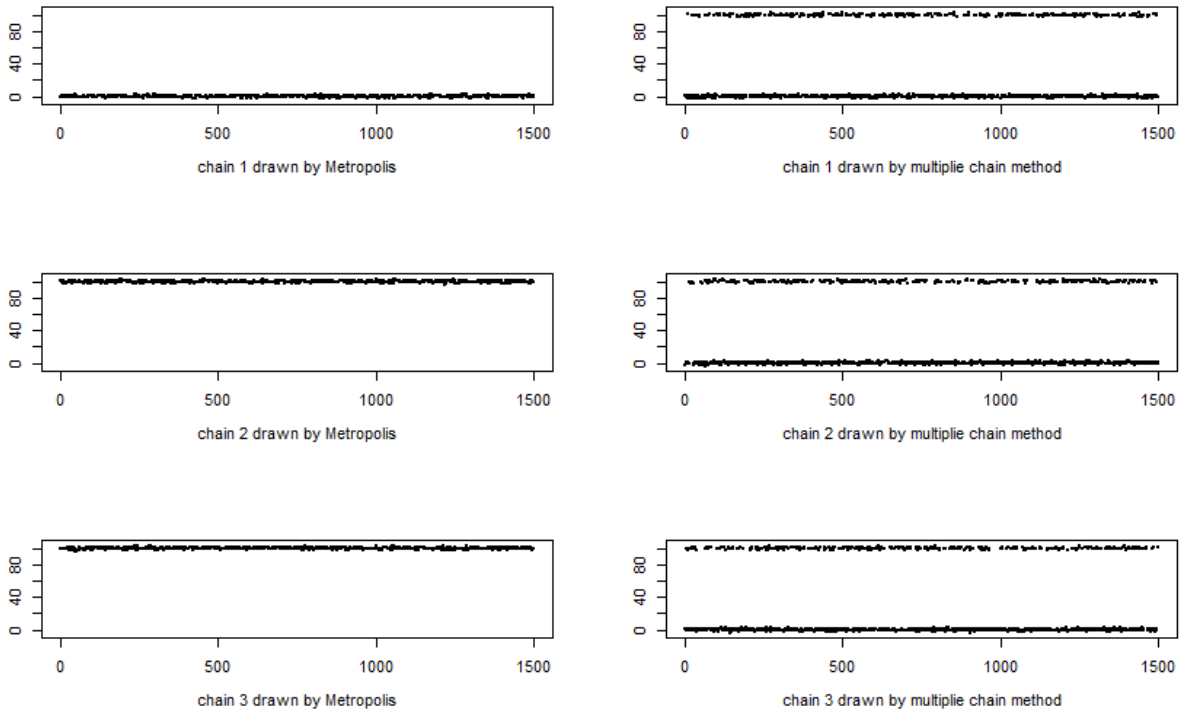

Figure 3.2: Dotplots of draws comparing Metropolis (left) and the multiple chain method (right).

distributions. Next, we study the theoretical properties of the proposed multiple chain method before giving more numerical examples.

3.3 Theoretical Aspects

3.3.1 Reducibility

In this subsection we highlight a particular feature of the multiple chain method in certain situations. Let $f(x)$ be the target distribution that we sample from. It is multimodal with completely separate modes. We assume that in both within-chain and between-chain jumps, the Metropolis-Hastings algorithm proposes local jumps in the sense that if an iterate is near

one mode, it cannot ever jump to the vicinity of another mode. Note that a Markov chain is reducible if it has zero probability to move from one state to some other particular state in a finite number of steps. To sample from $f(x)$ by a single chain of Metropolis-Hastings where the jumping kernel proposes local moves, we may experience reducibility problem as we start from one mode and hence we cannot visit any other modes because the jump is local. The multiple chain method also has the reducibility problem but it is slightly different from what we have in the single chain method. So for our method, if we start all chains from one mode, there is zero chance of visiting any other completely separate mode, as all chains would stay in the vicinity of this mode through within-chain jumps or between-chain jumps. In other words, if the vicinity of some mode does not contain any starting values, we will never visit that mode. Then if we start at least one chain near each mode, it will respect the constraint that we cannot have the situation where one mode is completely without any chains in the simulation. Suppose we have some chains starting from one mode and some chains from another, there are always going to be some chains staying in each of the modes. We run in total n parallel chains so the desired target distribution would be $\prod_{i=1}^{n} f(x_i)$, where x_i is the variable for the i-th chain. These chains explore different modes and jump to each other from time to time. However, they cannot all exit one mode and land in other modes. In other words, once we start from the state with at least one chain near each mode, we have zero chance of getting to states of which some mode is not visited, which means that our algorithm has the reducibility issue that it will never get rid of any modes. Suppose we run a length of N iterations for each chain and let n_j^t be the number of chains near the j-th mode at t-th iteration and $n_j = \{n_j^t : t = 1, 2, \ldots, N\}$, we have $\sum_j n_j^t = n$ for $t = 1, 2, \ldots, N$ and $n_j^t \geq 1$ for $\forall j, t$. So if we start at least one chain near each mode, n_j's will never drop to zero for all subsequent iterations. Thus the actual target distribution for the chains is no longer the multimodal target multiplied by itself many times. It is that target restricted to at least one chain near each mode: $\prod_{i=1}^{n} f(x_i) I\{n_j \geq 1 \text{ for } \forall j\}$, because of the reducibility problem. The reason that we have such reducibility problem is

in the design of our algorithm: the nature of the Metropolis-Hastings rule and the idea of jumping to neighborhood. Let us see this through the normal mixture example in Section 3.2.

We have a random variable X with target distribution $f(x)$ as

$$\lambda N(\mu_1, \sigma_1^2) + (1-\lambda)N(\mu_2, \sigma_2^2) = \sum_{i=1}^{2} \lambda_i f_i(x), \tag{3.2}$$

where $\sum_{i=1}^{2} \lambda_i = 1$ and $f_i = N(\mu_i, \sigma_i^2)$. Now we introduce a latent random variable Z to indicate which normal component X is drawn from. To be specific, we have

$$Z \sim Bernoulli(\lambda), \tag{3.3}$$

$$X|Z = 1 \sim f_1(x) \; and \; X|Z = 0 \sim f_2(x). \tag{3.4}$$

That is to say, to sample from $f(x)$, we could first decide which normal component X comes from and then sample from that component accordingly. If we apply this in the multiple chain method, we can record the component indicator Z for each sample in the Markov Chain. As in Section 3.2, it is done implicitly in the post analysis since the component assignment can be easily determined by the sample's value. We actually use this information of component indicators (70% samples come from the first component) to verify the correctness of the multiple chain method. Now let us take a further look of this proportion. Note that we have 20 chains run in parallel for 1500 iterations. If we look at each iteration, we have 20 samples which contain 20 component indicators $Z_i's, i = 1, 2, \ldots, 20$, where $Z_i \sim Bernoulli(\lambda = 0.7)$. Let

$$T = Z_1 + Z_2 + \cdots + Z_{20}, \tag{3.5}$$

we would have $T \sim B(20, 0.7)$ by the definition of binomial distribution if $Z_i's$ are i.i.d (independent and identically distributed). From the posterior draws, however, we see that

this is not true. Although we have a large value of $\lambda = 0.7$, T does not take the value of 20 (see Table 3.1).

T	7	8	9	10	11	12	13	14	15	16	17	18	19
Frequency	1	3	24	41	85	166	229	280	292	208	51	110	10

Table 3.1: Frequency table of T for 1500 iterations

Because of the symmetry of λ, T cannot be 0 either. It turns out that there is always at least one chain in one mode and at least another chain in another mode. In other words, T subjects to the constraint that we cannot have all 20 samples in one mode or all 20 in another. We have to have at least one chain in one mode, and at most 19 in another. So in equilibrium, the sum of component indicators $Z_i's$ at each iteration, T, goes to a truncated binomial, where the boundaries (0 and 20) are excluded. And the target distribution for the chains, will be the product of these target densities subject to this constraint. Thinking about all 20 draws at each iteration as one sample, it is a reducible Markov chain because we cannot go from 1 to 0, or from 19 to 20. This reducibility problem, or this truncation is due to the nature of the proposal distribution in the multiple chain method. It has specific rules for doing jumps that tries to jump to the neighborhood of other chains. Let us take a closer look at the theoretical reasons for this truncation.

Consider a bimodal distribution where modes are completely separate. We start n parallel chains with a few chains in each mode. Since our method updates the chains sequentially, we need to reach the state where 1 sample is in one mode and $n-1$ in another at some time before having all the samples in one mode. Without loss of generality, suppose for the current iteration we have n chains with chain 1 in the first mode and the remaining chains (chain 2 to chain n) in the other. Now we update x_1 from the first chain that is currently in mode 1, as is shown in Figure 3.3. We will see that by looking at the Metropolis-Hastings ratio, the chance of jumping out of mode 1 is zero.

According to the multiple chain method, we propose y from the neighborhood of one other

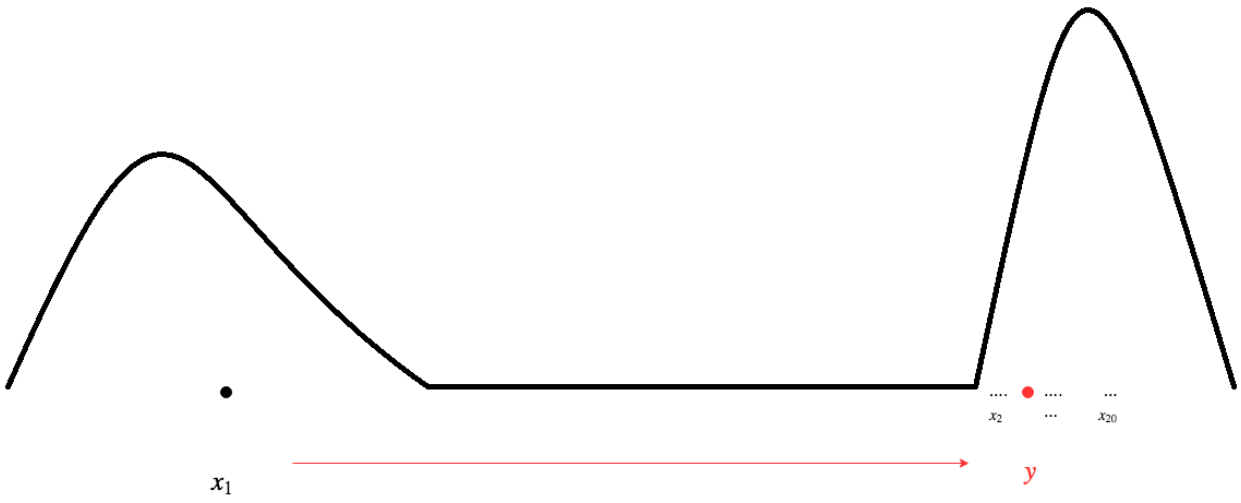

Figure 3.3: Proposed jump to y in between-chain jump for x_1

chain. Denoted by the red dot in Figure 3.3, the proposed point y will be in mode 2 as any of these $n-1$ other chains will be staying in mode 2 after proposing a local move. Then we have the current point x_1 in mode 1 and the proposed point y in mode 2, while two modes are completely separate. Recall that the proposal density is the mixture of normals centered at all other $n-1$ iterates, let us look at the acceptance ratio of Metropolis-Hastings in Algorithm 11:

$$\beta = \min\left(1, \frac{p(y)}{p(x_1)} \times \frac{g_1(x_1)}{g_1(y)}\right), \tag{3.6}$$

where $g_1(y) = \frac{1}{n-1}\sum_{k\neq 1} g_0(y|x_k)$ denotes the average density evaluated at all chains except chain 1. In Equation 3.6 there are two ratios $\frac{p(y)}{p(x_1)}$ and $\frac{g_1(x_1)}{g_1(y)}$ and we focus on the ratio for the proposal, $\frac{g_1(x_1)}{g_1(y)}$. On the numerator, $g_1(x_1) = \frac{1}{n-1}\sum_{k=2}^{n} g_0(x_1|x_k)$ denotes the proposal density of which the original point x_1 being evaluated at others. Since two modes are completely separate, the probability of proposing to go to x_1 in mode 1 from the mixture of $n-1$ normals is just zero (or very close to zero). In other words, $g_1(x_1) \approx 0$. Then the

probability of acceptance β is almost zero. Because of that, we cannot move the current point x_1 from mode 1. It has to stay there because of the zero chance of jumping. It is fine that we propose, as the proposal density $g_1(y)$ is positive for the proposed point y. But that is one side. On the other side, let us calculate the probability that we end up being in the current state x_1 given this proposal. This backward proposal has a density close to zero. We cannot propose back and get accepted. Hence the chain will never jump out of the mode. If otherwise we have irreducible moves within each mode, we can go in mode 1 by itself and go in mode 2 by itself. But we cannot communicate between mode 1 and mode 2. Then the only restriction we have is we cannot have all chains in mode 1 or all chains in mode 2. To sample a multimodal distribution with completely separate modes, we will start at at least one chain in each mode. That is why we state in Algorithm 11 that we start at dispersed starting values. Eventually the chains will converge, not to the correct target distribution, but to the product of these target densities under the constraint that at least one chain in each mode. That is the reducibility of the multiple chain method in the case when we have completely separate modes. If we miss a mode, we will not find it. The algorithm is not designed to find new modes while it aims to portion the mass between modes that we assign the chains to, in particular we have the situation where ordinary Metropolis does not really jump out of the mode. Nevertheless, the evaluation of the bias due to the reducibility is of interest, as discussed in Section 3.3.2.

3.3.2 Estimates of the Proportion in Truncated Binomial

In Section 3.3.1 we highlight the reducibility issue of the multiple chain method in the case of completely separate modes. In such a situation, the target distribution for the chains is actually not the correct one, but the correct one restricted to having at least one chain in each mode. As a result, in bimodal case if we denote the mode indicator as $Z_i's$ that specifies which mode the i-th chain is in, and compute the sum of these indicators as T that

indicates how many chains are in each mode, we then have the distribution of T through the iterations with its target a truncated binomial. Suppose that the posterior probability of one mode is λ and we have n parallel chains, this distribution does not converge to $B(n, \lambda)$ but $TB(n, \lambda)$, truncated at 0 and n, as it is impossible to have zero chains at either mode. More generally, if we get multiple modes that are completely separate, the chains will converge to a multinomial distribution with truncations of these particular pattern. This is an ongoing topic for further exploration but in this subsection we focus on the problem of dealing with estimating λ given the truncated binomial in the bimodal case. In other words, how do we estimate the proportion of a truncated binomial?

We adopt the naive estimator, sample proportion. Namely, we estimate the binomial proportion by simply counting the frequency of posterior draws that are from mode 1:

$$\hat{\lambda_1} = \frac{\#of\{Z = 1\}}{N_{rep}},$$

where N_{rep} is the total number of draws summing up all chains. In this case, suppose we have N iterations in total, N_{rep} is then equal to nN as we have n parallel chains. This naive estimator ignores the fact of truncation, so it is a biased estimator. Also note that we do not have independent samples because of the Markov chain. There exists serial correlation and thus we may lose some efficiency. However we can show that the sample proportion is still a decent estimator.

For a truncated binomial distribution, Patil (1962) shows that there is no uniformly minimum-variance unbiased estimator (UMVUE). As based on the sufficient and complete statistic $S = \sum_{j=1}^{N} T_j$ (assuming independent samples) where T_j is defined in Equation 3.5, there is no unbiased estimator. In this problem, we suggest three candidate estimators: (1) sample proportion (naive estimator); (2) maximum likelihood estimator (MLE); (3) asymptotically UMVUE proposed by Cacoullos and Charalambides (1975). Specifically, we will compare between the first two estimators. The third estimator, which is the minimum variance un-

biased estimator, has a complicated form and is hard to compute. There exists a possibility that the asymptotically UMVUE can be served as an alternative to the naive estimator but we will not discuss here.

For our problem, we have n parallel chains so we get truncated binomial samples truncated at 0 and n, with some unknown parameter λ to estimate. Assuming independence, we can formulate the maximum likelihood equation. This likelihood equation can be further reduced to:

$$\bar{T} = \frac{n\lambda - n\lambda^n}{1 - (1-\lambda)^n - \lambda^n}, \tag{3.7}$$

where $\bar{T}$ on the left-hand side denotes the sample mean. Say we observe draws from 1, 2 up to $n-1$ many times and the simple average of those is the sample mean $\bar{T}$. In other words, $\bar{T}$ is the sum of the sample proportions over n parallel chains: $\bar{T} = n\hat{\lambda_1}$. The MLE $\lambda = \hat{\lambda_2}$ satisfies Equation 3.7 and it does not have a closed form solution for large n. However, it is helpful to know how good the MLE is and if we compare between two estimators (naive estimator v.s. MLE). And the question is under what circumstance is the MLE a better estimator compared to the naive one, and vice verse. For this purpose, we need to be able to compute the MLE. For small n, we can simplify Equation 3.7 to:

$$\bar{T} = \begin{cases} 1, & n = 2; \\ \lambda + 1, & n = 3; \\ \frac{2(1+\lambda+\lambda^2)}{2-\lambda+\lambda^2}, & n = 4. \end{cases} \tag{3.8}$$

If we have $n = 2$ parallel chains, each mode has exactly one chain and the chain would stay in that mode forever. T_i is constant 1 and thus $\bar{T} = 1$. There is no MLE and the naive estimator $\hat{\lambda_1} = \bar{T}/n = 1/2$. The multiple chain method fails since there is no between-chain jumping. When $n = 3$, the MLE $\hat{\lambda_2} = \bar{T} - 1 \in [0, 1]$ as T_i is between 1 and 2. The naive estimator $\hat{\lambda_1} = \bar{T}/n = 1/3 + \lambda/3 \in (1/3, \infty)$. Again, the multiple chain method does

not provide any meaningful results as we only run three chains. When $n = 4$, Equation 3.7 becomes a quadratic equation of λ and we have the unique solution (the other one is discarded considering the range of λ):

$$\hat{\lambda_2} = \begin{cases} 1/2, & \bar{T} = 2; \\ \frac{\bar{T}+2-\sqrt{-7\bar{T}^2+28\bar{T}-12}}{2(\bar{T}-2)}, & \bar{T} \neq 2. \end{cases} \tag{3.9}$$

For $n = 5$, we can still have the closed form solution but we will just come up with the numerical approximations for simplicity. For $n > 5$, it is a high degree polynomial equation and we cannot derive in analytic form. Instead, we turn to numerical methods to get $\hat{\lambda_2}$. We run simulations for different n and λ values and compute the sample mean $\bar{T}$. Then we attempt to solve Equation 3.7. Note that the right-hand side of Equation 3.7 is the population mean of the truncated binomial distribution. To see this, we have the probability mass function (PMF) of the truncated binomial random variable T as:

$$P(T = k) = \frac{\binom{n}{k}\lambda^k(1-\lambda)^{n-k}}{1-\lambda^n-(1-\lambda)^n}, \quad k = 1, 2, ...n-1.$$

Then,

$$\begin{aligned} ET &= \sum_{k=1}^{n-1} \frac{k\binom{n}{k}\lambda^k(1-\lambda)^{n-k}}{1-\lambda^n-(1-\lambda)^n} \\ &= \frac{1}{1-\lambda^n-(1-\lambda)^n}\left(\sum_{k=0}^{k} k\binom{n}{k}\lambda^k(1-\lambda)^{n-k} - 0 - n\lambda^n\right) \\ &= \frac{n\lambda - n\lambda^n}{1-\lambda^n-(1-\lambda)^n}. \end{aligned}$$

So in fact, Equation 3.7 equates the sample mean to the population mean. In other words, this likelihood equation also gives the method of moments (MOM) estimator. If we denote the right-hand side of Equation 3.7 as $f(\lambda)$, $f(\lambda)$ is actually a monotone increasing function of λ. Given the sample mean $\bar{T}$, there is only one λ that satisfies this equation since $f(\lambda)$ is

monotone. Imagine if we do not have truncation, population mean goes up as the Bernoulli mean λ goes up. Now with truncation, we conclude the same is true: the population proportion λ goes up, the theoretical mean goes up. The proof is given by Patil (1962) where we treat the truncated binomial as a generalized power series with parameter $\theta = \lambda/(1-\lambda)$. It is proved that $f(\cdot)$ is increasing in θ. As $\theta = \lambda/(1-\lambda)$ is increasing in λ, $f(\cdot)$ is increasing in λ too by the chain rule. By L'Hospital's rule, we have $\lim_{\lambda\to 0} f(\lambda) = \lim_{\lambda\to 0} \frac{n-n^2\lambda^{n-1}}{n(1-\lambda)^{n-1}-n\lambda^{n-1}} = \frac{n}{n} = 1$ and $\lim_{\lambda\to 1} f(\lambda) = n-1$. Thus $f(\lambda)$ is monotone increasing from 1 to $n-1$. It has correct boundaries which confirms with the range of $\bar{T}$. So as long as $\bar{T}$ does not take the value 1 or $n-1$, we can do bisection to solve for λ at the very worst. If $\bar{T} = 1$, we set the value of the MLE $\hat{\lambda_2}$ as 0 and if $\bar{T} = n-1$, we have $\hat{\lambda_2} = 1$. Bisection can be slow, in which case we can switch to other numerical methods such as Newton-Raphson method. It is faster but is less stable compared with bisection, in the sense that it may not converge. We actually apply both methods and they give the same solution. If we plug in the solution back to $f(\lambda)$, it returns the correct $\bar{T}$ so numerical methods do solve Equation 3.7. Then we can safely compare the MLE obtained by bisection to the naive estimator that we adopt.

As mentioned earlier, we run simulations under different n and λ values. Specifically, we consider truncated binomial with $n = 3, 4, 5, 10, 200$ parallel chains. Truncation is at both ends (0 and $n-1$) and the probability of success p is $p = 0.02, 0.05, 0.10, 0.20, 0.30, 0.40, 0.50$. For each experiment setting, We run 100 replications of simulating 1000 independent truncated binomial samples. Each time we compute the sample proportion (naive estimator) and the MLE. Then we have 100 estimators so we can get the average (treated as the expected value of the estimator) as well as the variance. Bias is then computed as the expected value of the estimator minus the true value. Summary results of comparing the bias/standard deviation (sd) of two estimators are displayed in Table 3.2. For example, the third row represents the experiment results of truncated binomial of $p = 0.02$ with $n = 3$ parallel chains. Because of the symmetry of truncation, we consider different p values up to 0.5. As for example, $p = 0.9$ would mimic the behavior of $p = 0.1$.

Experiment Settings		Naive Estimator		MLE	
		Bias	**Sd**	**Bias**	**Sd**
$n = 3$	$p = 0.02$	0.320053	0.0015278	0.000160	0.004583
	$p = 0.05$	0.300263	0.002504	0.000790	0.007513
	$p = 0.10$	0.266703	0.003284	0.000110	0.009853
	$p = 0.20$	0.200797	0.004381	0.002390	0.013143
	$p = 0.30$	0.133807	0.004593	0.001420	0.013778
	$p = 0.40$	0.066997	0.005407	0.000990	0.016220
	$p = 0.50$	-0.000087	0.005302	-0.000260	0.015907

Experiment Settings		Naive Estimator		MLE	
		Bias	**Sd**	**Bias**	**Sd**
$n = 4$	$p = 0.02$	0.237923	0.001492	0.000756	0.003843
	$p = 0.05$	0.219720	0.002238	0.000455	0.005513
	$p = 0.10$	0.190293	0.003254	-0.000688	0.007461
	$p = 0.20$	0.137463	0.004476	0.000990	0.009123
	$p = 0.30$	0.088283	0.005219	-0.000006	0.009787
	$p = 0.40$	0.043960	0.005583	0.001369	0.009935
	$p = 0.50$	-0.000348	0.005954	-0.000608	0.010422
$n = 5$	$p = 0.02$	0.188164	0.001512	-0.000008	0.003625
	$p = 0.05$	0.170980	0.002291	-0.000125	0.005196
	$p = 0.10$	0.144034	0.002763	-0.000322	0.005694
	$p = 0.20$	0.097086	0.004303	-0.000142	0.007511
	$p = 0.30$	0.058374	0.005294	-0.000586	0.008079
	$p = 0.40$	0.027532	0.006154	0.000209	0.008636
	$p = 0.50$	0.000646	0.005599	0.000881	0.007636
$n = 10$	$p = 0.02$	0.089218	0.001045	-0.000247	0.002162
	$p = 0.05$	0.074381	0.001571	-0.000436	0.002938
	$p = 0.10$	0.053422	0.002348	-0.000200	0.003787
	$p = 0.20$	0.024084	0.003490	0.000016	0.004455
	$p = 0.30$	0.008965	0.0042791	0.000264	0.004750
	$p = 0.40$	0.002512	0.0045012	0.000143	0.004670
	$p = 0.50$	-0.000060	0.004752	-0.000061	0.004838

Experiment Settings		Naive Estimator		MLE	
		Bias	**Sd**	**Bias**	**Sd**
$n = 20$	$p = 0.02$	0.040236	0.000734	0.000116	0.001352
	$p = 0.05$	0.027890	0.001289	-0.000088	0.002010
	$p = 0.10$	0.013843	0.001789	-0.000003	0.002271
	$p = 0.20$	0.002123	0.002861	-0.000225	0.003005
	$p = 0.30$	0.000190	0.003459	-0.000051	0.003480
	$p = 0.40$	0.000312	0.003209	0.000297	0.003210
	$p = 0.50$	-0.000324	0.003467	-0.000324	0.003467
$n = 50$	$p = 0.02$	0.011510	0.000557	0.000081	0.000852
	$p = 0.05$	0.004198	0.000904	0.000031	0.001069
	$p = 0.10$	0.000476	0.001318	-0.000044	0.001350
	$p = 0.20$	0.000031	0.001676	0.000028	0.001677
	$p = 0.30$	0.000144	0.002089	0.000144	0.002089
	$p = 0.40$	-0.000074	0.002306	-0.000074	0.002306
	$p = 0.50$	-0.000232	0.002126	-0.000232	0.002126
$n = 100$	$p = 0.02$	0.003063	0.000363	0.000003	0.000458
	$p = 0.05$	0.000292	0.000613	-0.000007	0.000629
	$p = 0.10$	-0.000115	0.000817	-0.000118	0.000817
	$p = 0.20$	0.000083	0.001316	0.000083	0.001316
	$p = 0.30$	-0.000054	0.001257	-0.000054	0.001257
	$p = 0.40$	0.000076	0.001530	0.000077	0.001530
	$p = 0.50$	0.000229	0.001630	0.000229	0.001629

Experiment Settings		Naive Estimator		MLE	
		Bias	**Sd**	**Bias**	**Sd**
$n = 200$	$p = 0.02$	0.000387	0.000285	0.000030	0.000302
	$p = 0.05$	0.000013	0.000422	0.000012	0.000422
	$p = 0.10$	-0.000007	0.000675	-0.000007	0.000675
	$p = 0.20$	0.000032	0.000896	0.000032	0.000896
	$p = 0.30$	-0.000017	0.001013	-0.000017	0.001013
	$p = 0.40$	-0.000026	0.001075	-0.000026	0.001075
	$p = 0.50$	-0.000216	0.001076	-0.000216	0.001075

Table 3.2: Summary results comparing the sample proportion (naive estimator) with the MLE.

From Table 3.2 we can see that the MLE estimates the true p fairly well. That is not surprising as MLE suggests that we can estimate the truth if we have infinite samples. Now we have 1000 samples and we do see that the MLE is a consistent estimator. However to compute the MLE, note that when $n = 3$ or 4, we have the exact solution by solving the quadratic equation; when $n \geq 5$, we have a high degree polynomial equation and thus apply the numerical method. The solution can be found using bisection, and we just need to fix up the cases regarding endpoints ($\bar{T} = 0$ or $n - 1$). With all these efforts, is it worth it to compute the MLE? Now we can answer this question by examining the other half of Table 3.2. We see that when n is huge and p is not too small or too large, the sample proportion has negligible bias and is very close to the MLE. Both estimators are approximately unbiased and that is what we expect. For example, when we have $n = 200$ parallel chains and the true p is around 0.5, two estimators are comparable (almost the same), even for the standard deviation. That still holds true when $n = 200$ and $p = 0.05$. In such a case, $n = 200$ is fairly

large, and the only cases we are excluding in the truncation is 0 and 200. And $p = 0.05$ is still far away from the boundary because 1/200 is only 1/10 of that. The sample proportion (naive estimator) is consistent by the central limit theorem (CLT), when p is close to 0.5. However, we do not always have the luxury of running 200 parallel chains. Then there is some benefit using the MLE when p is far away from 0.5. For example, if we run 20 parallel chains and the true p is 0.02, one chain will get stuck in one mode with the rest 19 at the other mode. The naive estimator is 0.05, hence overestimating the truth. The naive estimator does not have a desired performance. If we have even less parallel chains, the naive estimator is way off as we can see from Table 3.2, for example, when n equals 3 or 4. That suggests us either not to use the naive estimator or we have to increase the number of parallel chains n when p is too small or too large, to avoid the situations where the naive estimator is going to be biased. Or, we can try to avoid the extreme p. Remember that in most situations, we are sampling the posterior draws so we can adjust the prior so that the posterior proportion would be half-half ($p = 0.5$). The conclusion is intuitive as we know the naive estimator can only estimate $p \geq \frac{1}{n}$ when we have n parallel chains. This is supported by the experiments. Then we may wonder how it looks if we do not have independent samples, which is true for the MCMC samples. By the law of large numbers (LLN), we expect the conclusion still holds. To confirm this, we do experiments following the same settings except that we are sampling correlated truncated binomial draws. Specifically, we generate correlated normal random variables from $AR(1)$ process, get correlated $U(0, 1)$ samples by evaluating the normal density and then sample correlated truncated binomial draws from the inversion of the truncated binomial CDF. These samples are correlated, as to imitate the MCMC draws, and their correlation is slightly lower than the $AR(1)$ autocorrelation ρ that we specify. For the results, things are roughly the same as Table 3.2 as long as the autocorrelation is not too heavy, for example $\rho = 0.4$. However we do see a difference for high autocorrelation situations. As an example, we set the autocorrelation $\rho = 0.75$, and present the summary results of comparing the bias/standard deviation (sd) of two estimators

in Table 3.3.

Experiment Settings		**Naive Estimator**		**MLE**	
		Bias	**Sd**	**Bias**	**Sd**
$n = 3$	$p = 0.02$	0.341780	0.005742	0.065340	0.017226
	$p = 0.05$	0.328983	0.006398	0.086950	0.019193
	$p = 0.10$	0.299547	0.009309	0.098640	0.027927
	$p = 0.20$	0.231137	0.011239	0.093410	0.033716
	$p = 0.30$	0.153937	0.011968	0.061810	0.035905
	$p = 0.40$	0.079177	0.012748	0.037530	0.038245
	$p = 0.50$	-0.000843	0.012275	-0.002530	0.036826
$n = 4$	$p = 0.02$	0.259612	0.006173	0.054253	0.014571
	$p = 0.05$	0.252060	0.008792	0.075666	0.019533
	$p = 0.10$	0.227400	0.010541	0.080055	0.021896
	$p = 0.20$	0.173405	0.010882	0.071703	0.020852
	$p = 0.30$	0.115685	0.014634	0.050387	0.026639
	$p = 0.40$	0.057730	0.013390	0.025722	0.023698
	$p = 0.50$	-0.000075	0.015919	-0.000130	0.027888

Experiment Settings		Naive Estimator		MLE	
		Bias	**Sd**	**Bias**	**Sd**
$n = 5$	$p = 0.02$	0.208222	0.005440	0.045925	0.011894
	$p = 0.05$	0.196784	0.007557	0.055139	0.015369
	$p = 0.10$	0.178148	0.010167	0.065744	0.018749
	$p = 0.20$	0.132588	0.014720	0.058775	0.023696
	$p = 0.30$	0.088526	0.015676	0.044226	0.022971
	$p = 0.40$	0.042396	0.017108	0.020848	0.023779
	$p = 0.50$	-0.001734	0.018304	-0.002368	0.025000
$n = 10$	$p = 0.02$	0.103994	0.003443	0.028786	0.006430
	$p = 0.05$	0.094252	0.005450	0.034633	0.009188
	$p = 0.10$	0.077127	0.008402	0.035946	0.012286
	$p = 0.20$	0.048947	0.012368	0.030757	0.014936
	$p = 0.30$	0.023062	0.015207	0.015683	0.016676
	$p = 0.40$	0.011247	0.015958	0.009145	0.016510
	$p = 0.50$	0.000529	0.017779	0.000540	0.018107
$n = 20$	$p = 0.02$	0.049654	0.002586	0.016584	0.004321
	$p = 0.05$	0.039900	0.004718	0.017744	0.006799
	$p = 0.10$	0.027279	0.006649	0.016536	0.008046
	$p = 0.20$	0.012728	0.010791	0.010816	0.011229
	$p = 0.30$	0.006517	0.012397	0.006304	0.012467
	$p = 0.40$	0.001952	0.013457	0.001937	0.013464
	$p = 0.50$	-0.002237	0.014667	-0.002237	0.014668

Experiment Settings		Naive Estimator		MLE	
		Bias	**Sd**	**Bias**	**Sd**
$n = 50$	$p = 0.02$	0.016755	0.001686	0.007734	0.002364
	$p = 0.05$	0.009613	0.003744	0.006284	0.004303
	$p = 0.10$	0.005017	0.004900	0.004588	0.004997
	$p = 0.20$	0.003113	0.007856	0.003110	0.007857
	$p = 0.30$	0.001512	0.008674	0.001512	0.008674
	$p = 0.40$	0.001642	0.008114	0.001642	0.008114
	$p = 0.50$	-0.001155	0.008575	-0.001155	0.008575
$n = 100$	$p = 0.02$	0.005971	0.001587	0.003561	0.001913
	$p = 0.05$	0.002856	0.002645	0.002613	0.002699
	$p = 0.10$	0.001524	0.003618	0.001521	0.003619
	$p = 0.20$	0.001758	0.005159	0.001758	0.005159
	$p = 0.30$	0.001912	0.006111	0.001912	0.006111
	$p = 0.40$	0.000714	0.006148	0.000714	0.006148
	$p = 0.50$	-0.000922	0.006653	-0.000922	0.006653
$n = 200$	$p = 0.02$	0.001770	0.001109	0.001483	0.001162
	$p = 0.05$	0.000835	0.001958	0.000833	0.001959
	$p = 0.10$	0.001133	0.002691	0.001133	0.002691
	$p = 0.20$	0.001381	0.003289	0.001381	0.003289
	$p = 0.30$	0.000050	0.003705	0.000050	0.003705
	$p = 0.40$	-0.000012	0.004279	-0.000012	0.004279
	$p = 0.50$	-0.000329	0.004299	-0.000329	0.004299

Table 3.3: Summary results comparing the sample proportion (naive estimator) with the MLE from heavily dependent samples ($\rho = 0.75$).

From Table 3.3 we see that most of the time the two estimators are accurate. However, unlike the consistency in Table 3.2, the MLE is biased in some situations. For example, when n is 3 and the true p is 0.02, the bias of the MLE is 0.06. Although it is better than that of the naive estimator, the MLE is still off the mark. So if we have few chains and they are highly correlated, the MLE may not do well as we know the MLE best works under the independence assumption. From these two experiments, we recommend the following when applying the multiple chain method. If chains are going really slowly which suggests high correlation, probably it is not worth going through the MLE calculation because that is wrong anyway. If chains converge really fast, we know the autocorrelation dies down and it might be worthwhile to try to get the MLE. The key issue is that bias often arises when p is too small or too large, which is the same through the experience with other parallel chain methods (Han and Carlin, 2001). We can fix the problem, however, by adjusting the prior proportions so the posterior proportions are about to equal. In that way, if we cannot have any fancier estimators like the MLE, the naive estimator is still doing well. We do not need to get the MLE or to worry about autocorrelations. All these work is trying to understand the behavior of our algorithm when we apply to things such as Bayes factor calculation as discussed in Section 4.1. When there are two modes that absolutely have nothing in common meaning that they are completely separate, we cannot move all chains from one mode to the other mode. This feature is very intuitive but we think it is also very special. It is amazing to see that, even though the two modes are completely separate, we can still get the decent estimator as long as the posterior proportion provided is not too close to 0 or 1. In Chapter 4 we discuss this in more detail.

3.4 Some General Guidelines

In this section we will discuss how to apply the multiple chain method.

3.4.1 When to Use the Multiple Chain Method?

The purpose of the multiple chain method is to enable between-chain jumps and to speed up convergence. And we require a between-chain jump proposal, which is essentially random walk type within each chain. The price we pay is that, at each iteration, we have extra computation on the density evaluation for this between-chain jumping kernel compared to ordinary Metropolis (single chain method). The reward we gain is the more efficient draw. As a result, if the density evaluation of the target distribution is costly, our method is desirable in terms of the computational cost. Such a situation is not rare as we will see in the astronomical example in Chapter 5. The evaluation of the target density requires complicated model settings, which is done in a specific astronomical software, as opposed to the simple normal mixture density evaluation in the between-chain jumps. However, if the target density has many modes and it is easy to evaluate the target density, our method will not work well. In general, our method works especially well if the evaluation of the target density takes time and the convergence of usual MCMC method is slow, which can be sped up by the multiple chain.

Compared with parallel tempering (Geyer, 1991), our method and parallel tempering are both population-based methods. If we are interested in finding the mode, parallel tempering is helpful to find the global maximum. But regarding sampling efficiency, other population-based methods like parallel tempering produce only one chain (corresponding to $T = 1$) with correct draws. Instead, in our method, all chains would converge to the same target distribution and have desired draws. So our method has a larger effective sample size. In addition, parallel tempering requires a careful fine tuning of the temperature ladder. In our method, we only require choosing the local move proposal in between-chain jumps, which is often the multivariate normal density and relatively easy to tune. However, this does not work well in high dimensions because of the nature of the proposal and the acceptance-rejection method. Compared with product space search (Carlin and Chib, 1995) and reversible jump MCMC

(Green, 1995) type method, all these three methods can handle trans-dimensional sampling (see Section 4.1.3). In terms of parameter tuning, each method requires a different type of tuning. Carlin & Chib's method requires specifying the sudo prior that affects the sampling efficiency. Reversible jump MCMC requires substantive effort designing dimension matching function between two spaces and proposing explicitly the jump to another space. Our method, as just mentioned, requires between-chain jumping kernel that will also affect the sampling efficiency. We need to find the appropriate proposal, e.g. multivariate normal, and tune the step size. The tuning effort each method requires is problem dependent. In the Bayes factor example which is discussed in Section 4.1.3, our proposed multiple chain method requires fewer tuning parameters and will likely be more efficient. We discuss the general strategy of tuning between-chain jumping kernel in Section 3.4.2.

3.4.2 Tuning of the Proposal Distribution

In this subsection we focus on the between-chain jump proposals since recommendations for the within-chain jump proposals are just the same as those in the ordinary Metropolis cases. As we suggest in Section 3.1, the choice of the between-chain step transition kernel is not limited to the normal random walk type. It can be much more general as long as we have a decent between-chain jump acceptance rate. But even for the simple normal random walk transition kernel, we have to carefully tune the step size (standard deviation of the normal proposal) to make the algorithm work. The general recommendations for the between-chain jump step size tuning, is to find a reasonable range of acceptance rate and then within that range to select the large step size. We use the following example to illustrate. We consider the mixture of two univariate normal example but slightly change the distribution to have one component more dispersed. And we run simulations under different step sizes, collect the samples and assess the sample efficiency. Specifically, we consider the target distribution

as

$$\lambda N(\mu_1, \sigma_1^2) + (1 - \lambda)N(\mu_2, \sigma_2^2),$$

where $\lambda = 0.7, \mu_1 = 0, \mu_2 = 1000$, $\sigma_1^2 = 1$ and $\sigma_2^2 = 10$. The density plot is shown in Figure 3.4.

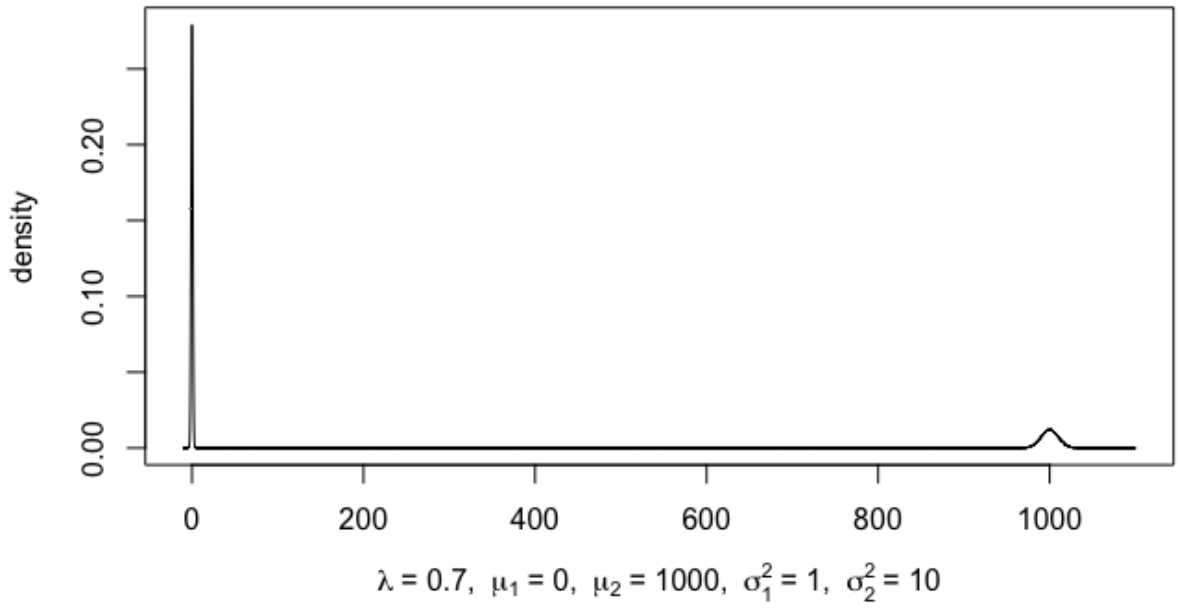

Figure 3.4: Density plot of the normal mixture: $\lambda N(\mu_1, \sigma_1^2) + (1 - \lambda)N(\mu_2, \sigma_2^2)$.

Given the target distribution that we aim to sample from, we apply the multiple chain method with $n = 20$ parallel chains, each running 50,000 iterations. For within-chain jumps, we tune the step size so that we have roughly half draws accepted in within-chain jumps. For between-chain jumps, we let the step size $s = 10, 1, 0.1, 0.01$, and the resulting between-chain jump acceptance rate is $21\%, 57\%, 28\%, 5\%$ accordingly. As discussed in Section 3.3.1, the parameter of interest is the mixing proportion λ and we estimate it by the sample proportion $\hat{\lambda} = \frac{T}{n}$ at each iteration, where T is defined in Equation 3.5. So $\hat{\lambda}$ is the proportion of draws

for mode 1 in parallel chains per iteration, and thus form a time series as we move the iterations along. We now have the time series of Monte Carlo draws and we can check their dependency via the autocorrelation plot. The traceplots of $\hat{\lambda}$ with the corresponding autocorrelation plots under four different parameter settings are displayed in Figure 3.5.

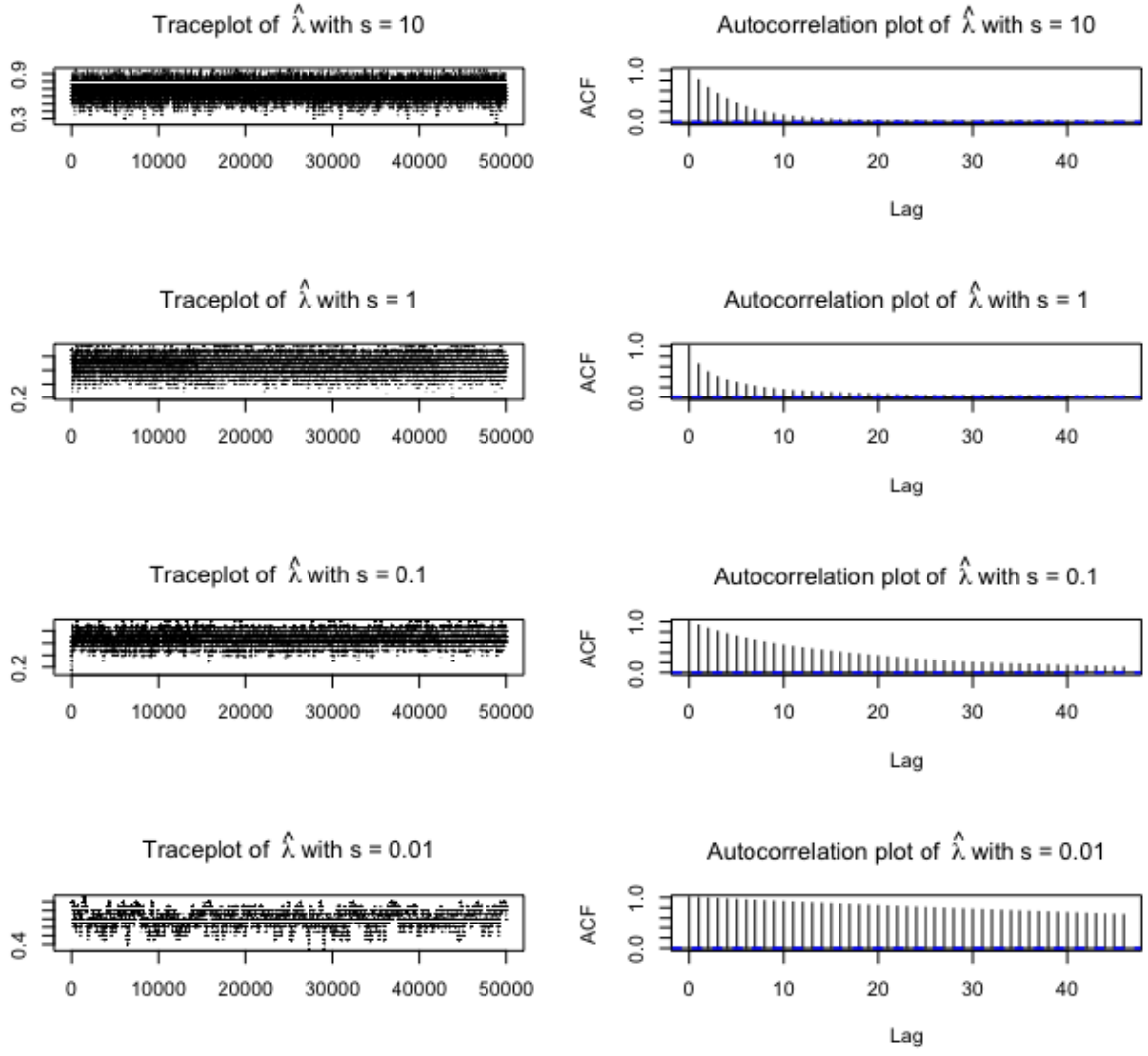

Figure 3.5: Traceplots (left) and autocorrelation plots (right) of $\hat{\lambda}$ comparing different between-chain jump step size from $s = 10, 1, 0.1$ to 0.01, with the corresponding between-chain jump acceptance rate $21\%, 57\%, 28\%$ and 5%.

From Figure 3.5, we see that the chain at the bottom ($s = 0.01$) has the highest autocorrelation while the second one ($s = 1$) has the lowest one. It is not surprising to see this happen

as we know if the acceptance rate is too low (just as 5%), the chain will not jump very often and thus it will not get the portion of mass between two modes very efficiently. Whereas for any other ones, as long as the acceptance rate is decently higher, it seems to suggest the results are fine. That confirms our intuition that the higher acceptance rate is, the faster autocorrelation dies down, and thus the less correlated draws are. More to notice is that even though the first chain has lower acceptance rate (21%) than the third one (28%), it has much smaller autocorrelation. That is because the step size of the first chain ($s = 10$) is way larger than that of the third one ($s = 0.1$). In a successful between-chain jump, we can still end up in the same mode. In such a within mode jump situation, larger step size will help to obtain less correlated draws. So the general recommendation is to find a reasonable range of decent between-chain jump acceptance rate, and then to have a large step size within this range.

To assess convergence, we also conduct Gelman and Rubin convergence diagnostic (Gelman and Rubin, 1992) by running five replications under each parameter settings. Specifically, for each parameter setting, we run five independent replications with twenty multiple chains for each replication. Within each replication we allow between-chain jumps and between different replications there are no interactions. Then we can apply Gelman and Rubin statistic by computing the summaries for each replication. That would be computational intensive, however, and selection of the static monitored is nontrivial in high dimensional cases as we have many possible statics. Figure 3.6 shows the dotplots of point estimates of the potential scale reduction factor (PSRF) changing through every 5000 (left panel) and every 100 (right panel) iterations comparing four different between-chain jump step size from $s = 10, 1, 0.1$ to 0.01. In the left panel of Figure 3.6, we see that all four chains do converge since PSRF < 1.05. Among the four lines from the right panel of Figure 3.6, we see that the red line (corresponding to $s = 1$) has relatively lowest PSRF as they gradually die down, indicating that it converges the fastest. It suggests that the higher acceptance rate is, the more effective draws we have. We also find that the chains with $s = 10$ and $s = 1$ have converged

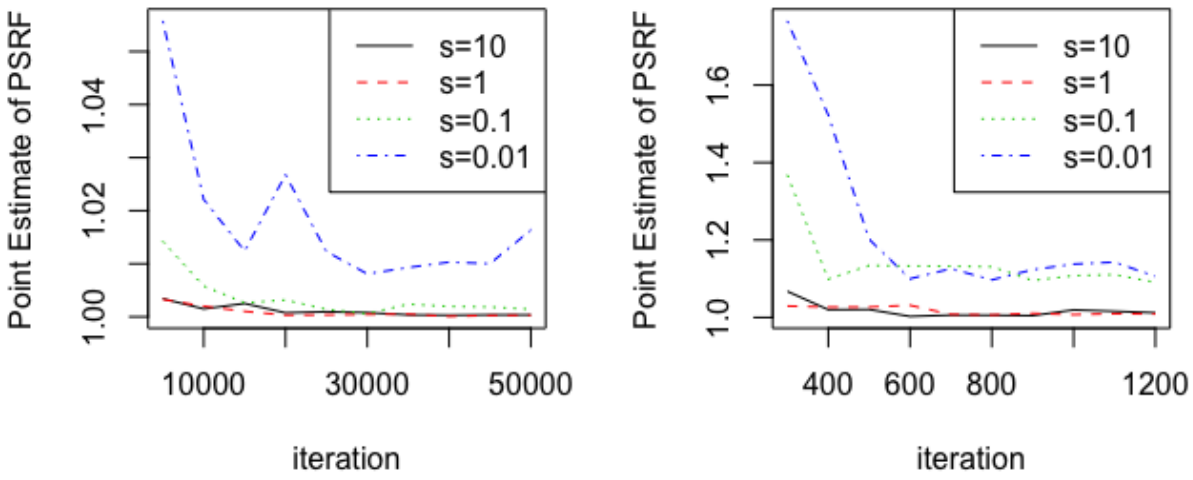

Figure 3.6: Dotplots of point estimates of the potential scale reduction factor (PSRF) decreasing through every 5000 (left panel) and every 100 (right panel) simulated draws comparing different between-chain jump step size from $s = 10, 1, 0.1$ to 0.01, with the corresponding between-chain jump acceptance rate $21\%, 57\%, 28\%$ and 5%.

after 1200 iterations, while the chains with $s = 0.1$ and $s = 0.01$ have not, which agrees with the findings from Figure 3.5. In between-chain jumps, a success can lead to either a between-mode jump or a within-mode jump. In the latter case, *i.e.*, two chains are in the same mode, step size does matter as a larger step size will result in less correlated draws in neighborhood jumps. This is also the reason that in Figure 3.6, two chains ($s = 10$ and $s = 1$) are comparable in terms of convergence despite the huge difference in the acceptance rate. This toy example suggests that in the between-chain jumps, we shall first pick up a reasonable range of acceptance rate and then make the step size as large as possible. One possible explanation is that by doing this, the chain can either jump between modes more frequently or jump as far away as possible in the same mode, so we can get more efficient samples drawn from different modes. Next we discuss an issue that arises from the between-chain jumping proposals.

3.4.3 Issue in High Dimensions

One issue of the multiple chain method is that it does not work well in high dimensional cases because of the independent Hastings proposal we are using. In other words, it is an issue with between-chain jumps that we have. Currently we are using a mixture of normals (or some other proposal distribution) centered at other iterates. It suffers from the curse of dimensionality because, just like kernel density estimation does not work in high dimensions, the mixture of normals does not approximate the target distribution very well in high dimensional cases.

In between-chain jumping step, the transition kernel is to propose a full dimensional jump, usually by multivariate normal random walk type proposal. If the dimension is too high, it is very unlikely that the proposed point can be accepted. As we can see from Table 4.5, the between-chain jumping acceptance rate is only 8×10^{-5}. In that case, some may wonder why we cannot just break up the dimensions in a Gibbs fashion and propose one dimension at a time as in the within-chain jumping step. Here we use a simple example to illustrate. The issue with Gibbs is that there is no way to jump back and this proposal simply does not even move.

Suppose we are working on a two dimensional posterior distribution in $[0, 1]^2$. We run six parallel chains and the current iterates are shown in Figure 4.5, *i.e.*, $x_1, \ldots, x_6$. Now we want to update x_1, and we break the between-chain jumping down to two subsequent moves, *i.e.*, first, along x-axis and secondly along y-axis. For the first move, we randomly pick another iterate to generate the proposed point along x-axis. Then immediately we see there arises a problem that the jump is not reversible. x_1 is not in any neighborhood of other iterates along x-axis. In other words, we can propose but we cannot propose back, so the acceptance rate is always zero; thus a Gibbs fashion trick will not work in our method. As our method can only have full dimensional proposals in between-chain jumping, thus cannot overcome the curse of dimensionality. In future work we will explore to address this problem. We illustrate

more about the multiple chain method via concrete examples in the following chapter.

Chapter 4

Applications

In this chapter, we present both synthetic and real data examples where the multiple chain method can be applied, such as computing Bayes factors for Bayesian model selection, estimating variance components in mixed effect models, and sensor network localization. We present and analyze these three applications in Section 4.1, 4.2, and 4.3, respectively.

4.1 Computing Bayes Factors for Bayesian Model Selection

4.1.1 Introduction of Bayes Factors

Given data Y, we want to select the best model among candidates. Suppose the observed data Y are generated by M_i in $\{M_i : i \in I\}$, where $\{M_i : i \in I\}$ is the finite set of competing models. We have a corresponding distinct parameter vector θ_i of dimension n_i, and a prior model probability $Pr(M_i)$. The posterior probability $Pr(M_i|Y)$ is widely used, such as determining the posterior distribution of some shared parameter by model averaging (Carlin

and Louis, 2000). However, this quantity is not very useful in model selection, since the prior probability $Pr(M_i)$ may vary. The posterior probability $Pr(M_i|Y)$ changes whenever we change the prior probability $Pr(M_i)$. Instead, we perform pairwise comparison based on the Bayes factor (Kass and Raftery, 1995). Suppose there are two competing models, the decision is made by checking the Bayes factor B_{21} (Han and Carlin, 2001), which is the ratio of marginal likelihood of data under two models:

$$B_{21} = \frac{Pr(Y|M_2)}{Pr(Y|M_1)}.$$

It shows evidence provided by data in favor of model 2 (M_2) against model 1 (M_1). We have the data likelihood $Pr(Y|M_1)$ and $Pr(Y|M_2)$, which represent the probability that data Y are produced under the assumption of M_1 and M_2, respectively. Given a prior probability $Pr(M_1)$, we form a posterior probability $Pr(M_1|Y)$. And as a result, we have $Pr(M_2) = 1 - Pr(M_1)$ and $Pr(M_2|Y) = 1 - Pr(M_1|Y)$. From Bayes Theorem, we know that

$$Pr(M_i|Y) = \frac{Pr(M_i)Pr(Y|M_i)}{\sum_i Pr(M_i)Pr(Y|M_i)}, \quad i = 1, 2.$$

Therefore, we derive

$$\frac{Pr(M_2|Y)}{Pr(M_1|Y)} = \frac{Pr(M_2)}{Pr(M_1)} \frac{Pr(Y|M_2)}{Pr(Y|M_1)} = \frac{Pr(M_2)}{Pr(M_1)} \times B_{21},$$

which indicates that the posterior odds equals the prior odds times the Bayes factor. Finally, we obtain

$$B_{21} = \frac{Pr(M_2|Y)}{Pr(M_1|Y)} \Big/ \frac{Pr(M_2)}{Pr(M_1)}, \tag{4.1}$$

which is the ratio of the posterior odds to the prior odds in favor of model 2.

4.1.2 Methods for Computing Bayes Factor

In this subsection we describe the approaches of how to compute the Bayes factor. To be concrete, we take the example where we have data Y and two candidate models:

- Model 1 (M_1):

$$Y_i|\mu \sim N(\mu, \sigma^2), \quad i = 1, 2, ...n,$$

where $\mu|\mu_0, \sigma_0^2 \sim N(\mu_0, \sigma_0^2)$.

- Model 2 (M_2):

$$Y_i|\alpha, \beta, X_i \sim N(\alpha + \beta X_i, \sigma^2), \quad i = 1, 2, ...n,$$

where $(\alpha, \beta)^T|\alpha_0, \beta_0, \Sigma \sim N_2((\alpha_0, \beta_0)^T, \Sigma)$ and $\Sigma = \begin{bmatrix} \sigma_0^2 & 0 \\ 0 & \sigma_1^2 \end{bmatrix}$.

Note that $X_i's$ are known covariates and σ^2 is known. Both models assume normal data, but differ in the mean structure: one has parameter μ and the other has a linear structure with known covariates. Thus, model 1 is nested in model 2 if $\alpha_0 = \mu_0$. Our target is to compute the Bayes factor $B_{21} = \frac{Pr(Y|M_2)}{Pr(Y|M_1)}$.

A Standard Approach

A standard approach is to compute the numerator and denominator separately by integrating out the model parameters (Kass and Raftery, 1995). Each represents the integral of the probability density of data given all parameters under model 1 (or model 2):

$$Pr(Y|M_i) = \int Pr(Y|\theta_i, M_i)\pi(\theta_i|M_i)d\theta_i, \tag{4.2}$$

where θ_i is the collection of parameters under M_i, $\pi(\theta_i|M_i)$ is the prior density of θ_i, and $Pr(Y|\theta_i, M_i)$ is the probability density of Y given θ_i. If $i = 2$, Equation 4.2 becomes the integral of the probability density of data given all parameters under model 2. The task is to integrate over the parameters $\theta_2 = (\alpha, \beta)$, which requires the choice of priors for α and β. However, noninformative priors cannot be used, as we do not know if the integral is finite or not. The Bayes factor computation requires averaging the probability of the data over the prior of parameters for each model and calculating that ratio. Hence, the quantity depends on the prior specified on each model. As the Bayes factor is sensitive to the prior, we have to avoid the improper ones. In this example, after specifying the hyper parameters and generating the data, we can solve Equation 4.2 analytically because of conjugacy. The integral is a normal-normal situation and it turns out that:

- Model 1 (M_1):

$$Y|M_1 \sim N(\vec{1}\alpha_0,\, \sigma^2 I + \sigma_0^2 \vec{1}\vec{1}^T).$$

- Model 2 (M_2):

$$Y|M_2 \sim N((\vec{1}, \vec{X_0})(\alpha_0, \beta_0)^T,\, \sigma^2 I + (\vec{1}, \vec{X_0})\Sigma(\vec{1}, \vec{X_0})^T).$$

Then the computation of $B_{21} = \frac{Pr(Y|M_2)}{Pr(Y|M_1)}$ is straightforward.

Multiple Chain Method for Computing Bayes Factors

To apply the multiple chain method on computing Bayes factors, we combine the two models into a disjoint union of two models. To be specific, we introduce a model indicator I, provide the joint posterior distribution, and then directly sample from the distribution. In other

words, we sample over the joint space created by the model indicators and the parameters for each model. Corresponding to model M_i, we have the data likelihood $Pr(Y|\theta_i, M_i)$, a prior $\pi(\theta_i|M_i)$ for the parameters, and a prior model probability $Pr(M_i)$ that sums up to 1. If $I \in \{1, 2\}$, we have a union of two models. One has parameter space 1 and the other has parameter space 2. As a result, the density is over the union of two spaces expressed as:

$$Pr(I, \theta_1, \theta_2|Y) \propto \prod_{i=1}^{2} [Pr(Y|\theta_i, M_i)\pi(\theta_i|M_i)Pr(M_i)]^{\mathbb{1}_{\{I=i\}}}, \tag{4.3}$$

where $Pr(M_1)$ is the prior probability that data Y comes from M_1, and $Pr(M_2) = 1 - Pr(M_1)$. We introduce this I to specify data with its corresponding model. If $I = 1$, Equation 4.3 refers to the posterior density under M_1. Then we can sample from $Pr(I, \theta_1, \theta_2|Y)$ by applying the multiple chain method. Note that trans-dimensional jumps are made between M_1 and M_2, since $\dim(\theta_2) = \dim(\theta_1) + 1$. Afterwards, we compute the posterior probability $Pr(M_i|Y)$ by simply counting the frequency of samples that come from M_i:

$$Pr(M_i|Y) = \frac{\#of\{I = i\}}{N_{rep}}, \quad i = 1, 2,$$

where N_{rep} is the total number of iterations summing up all chains. That is, we count how many time model i is true in the posterior samples, and divide the counts by the total number of samples to obtain the posterior probability of model i. From Equation 4.1, after having the posterior odds and the prior odds, we can compute B_{21}. In this normal example, our method agrees with the theoretical derivation. In Section 4.1.3 and 4.1.4, we present two real data examples to illustrate the capability of the multiple chain method for computing Bayes factors.

4.1.3 Pine Data Example

In this subsection, we consider the pine data from Williams (1959), an example discussed in Carlin and Chib (1995). There are $n = 42$ specimens of radiata pine. For each specimen, we measure the maximum compressive strength parallel to the grain y_i, the specimen's density x_i, and its density adjusted for resin content z_i. Part of the data are displayed in Table 4.1.

Specimen	strength y_i	density x_i	adjusted z_i
1	3040	29.2	25.4
2	2470	24.7	22.2
...			
41	3030	33.2	29.4
42	3030	28.2	28.2

Table 4.1: Part of the data on 42 specimens of radiata pine

We consider two competing models:

- Model 1: $y_i \sim N(\alpha + \beta(x_i - \bar{x}), \tau_1^2)$
- Model 2: $y_i \sim N(\gamma + \delta(z_i - \bar{z}), \tau_2^2)$

Both models assume that y_i is normal and the mean has a linear structure: one has the covariate x_i and the other has the adjusted one z_i. Unlike the normal example in Section 4.1.2, variance components (τ_1^2, τ_2^2) are unknown here. Figure 4.1 in Spiegelhalter et al. (1996) displays the model structure as well as the prior setting. The left panel in Figure 4.1 shows model 1. We have model parameters α, β for the mean and τ_1 for the standard deviation. Model 2 in the right panel is symmetric except for the covariate z_i. After centering the covariates x_i and z_i at their means, we put normal priors on $(\alpha, \beta)^T$ and $(\gamma, \delta)^T$, as shown in Figure 4.1. For τ_1^2 and τ_2^2, we place the same inverse gamma priors with both mean and

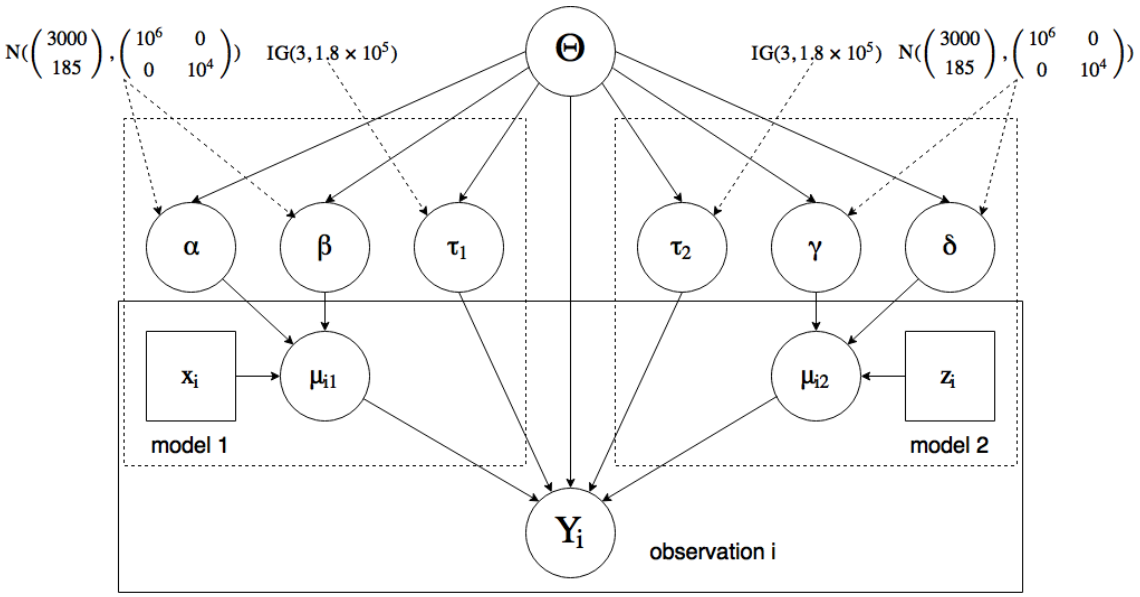

Figure 4.1: Graphical model for pines example showing the two models being simultaneously handled within a unified framework (Spiegelhalter et al., 1996, p. 48).

standard deviation equal to 300^2. All these priors on $(\alpha, \beta, \gamma, \delta, \tau_1^2, \tau_2^2)$ are roughly centered on their appropriate least squares parameter estimate. Therefore, we make the priors to be extremely vague, but still keep them proper.

In this example, both models are simple linear regression models. We have $\theta_1 = (\alpha, \beta, \tau_1^2)$, $\theta_2 = (\gamma, \delta, \tau_2^2)$ and want to solve Equation 4.2. However, the integral does not have an analytical solution, because the priors are not conjugate. Nevertheless, it is easy to integrate out the regression parameters from each model, leaving a one-dimensional integration with respect to the variance parameter. Essentially, we reduce Equation 4.2 to a one-dimensional problem that can be solved by Riemann integral (Green and O'Hagan, 1998). The integral can be carried out by numerical integration, *e.g.*, on 1,000 equally spaced ordinates. Even 100 points are also fine due to the nice smooth behavior of the integrands. We believe that the numerical calculation is exact so that we treat it as the benchmark. Note that we are comparing models based on the Bayes factor, which does not change when we change the prior on different models, since it is the posterior odds divided by the prior odds. The posterior probability, however, changes whenever we change the prior. Thus, we can tune the prior so

Methods	$\hat{P}$(M=2|y)	SD	$\hat{B_{21}}$	95% CI for $\hat{B_{21}}$
CC	.70806	.001721	4848.3	(4769.2, 4927.4)
RJ-G	.70906	.002394	4871.8	(4761.0, 4982.6)
Multiple chain method	.70847	.001750	4857.9	(4777.2, 4938.5)
Target	.70865			4862

Table 4.2: Summary results of different methods

that in the posterior samples, the chain ideally visits each model roughly half of the time. A summary of the results is displayed in Table 4.2. Han and Carlin (2001) discusses several methods including product space search (Carlin and Chib, 1995) and reversible jump MCMC (Green, 1995). And we carry out the multiple chain method following the same setting up. See Section 3.4.1 for theoretical comparisons between these methods. Table 4.2 reports the estimated posterior probability for model 2, a batched standard deviation estimate for this probability (using 2,500 batches of 100 consecutive iterations), the Bayes factor in favor of model 2 and an approximate 95% CI for the the Bayes factor. The huge Bayes factor from the table clearly indicates that model 2 is preferred. For our method, the point estimate is 4858 and the corresponding 95% CI covers the target result, which shows that the multiple chain method is able to compute the Bayes factor accurately. Next, we discuss the galaxy data example to compute the Bayes factor.

4.1.4 Galaxy Data Example

In this subsection, we consider another galaxy data example in Carlin and Chib (1995). The data contain 83 observations of velocity (in 10^3 km/s) of distant galaxies diverging from the Milky Way. After ignoring one missing observation by Roeder (1990), there are 82 data points sampled from six well-separated conic sections of the corona borealis region (Stephens, 2000). A full list of the data in ascending order (Carlin and Chib, 1995) is shown in Table 4.3.

9172	9350	9483	9558	9775	10227	10406	16084
16170	18419	18552	18600	18927	19052	19070	19330
19343	19349	19440	19473	19529	19541	19547	19663
19846	19856	19863	19914	19918	19973	19989	20166
20175	20179	20196	20215	20221	20415	20629	20795
20821	20846	20875	20986	21137	21492	21701	21814
21921	21960	22185	22209	22242	22249	22314	22374
22495	22746	22747	22888	22917	23206	23241	23263
23484	23538	23542	23666	23706	23711	24129	24285
24289	24366	24717	24990	25633	26960	26955	32065
32789	34279						

Table 4.3: Velocities (in 10^3 km/s) for galaxies in the corona borealis region

The interest lies in the multimodality of the velocities, which may support the astronomical theory concerning the clustering of galaxies. In other words, the data may come from a potentially multimodal distribution. To check this, we plot the histogram of the data overlaid with the density estimation of a mixture of six normals in Figure 4.2. Based on the data, we want to decide whether they come from a multimodal distribution, which can indicate the presence of super clusters of galaxies (Roeder, 1990). Here, the problem of interest is to decide the number of mixture components. Given data y, we aim to compare a three-component normal mixture with a mixture having four components. Let ϕ denote the normal pdf and q_{jk} denote the corresponding mixing probabilities for k-th component in the mixture model j, we have

- Model 1:

$$f(y_i|\mu_1, \sigma_1^2, q_1) = \sum_{k=1}^{3} q_{1k}\phi(y_i|\mu_{1k}, \sigma_1^2),$$

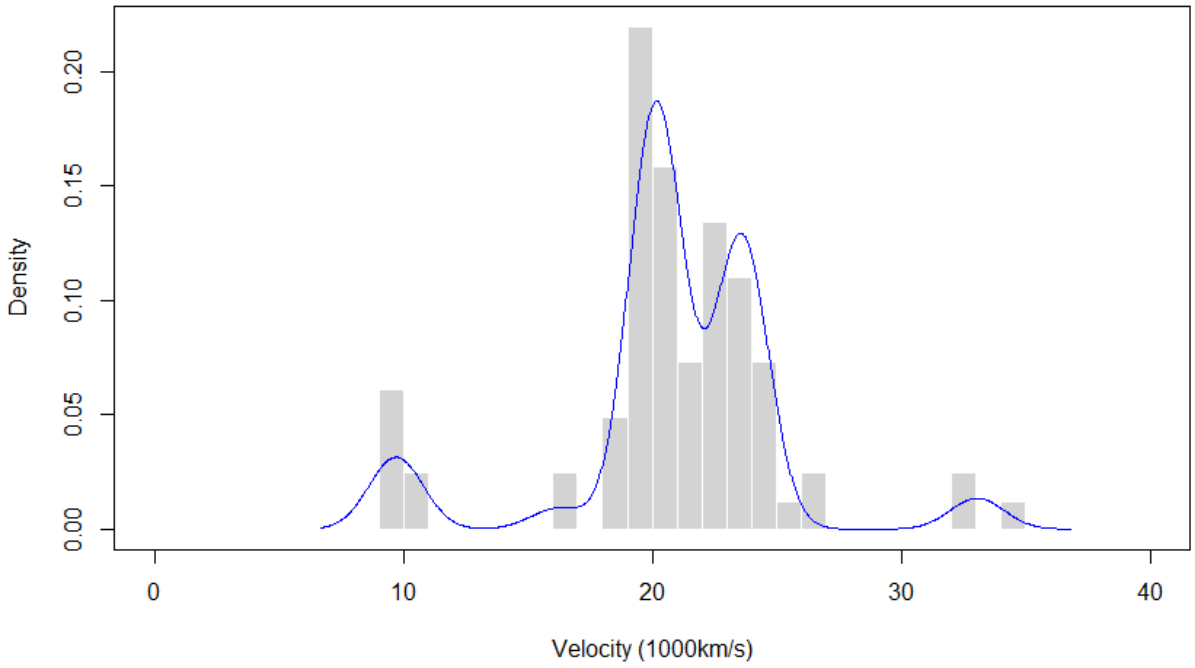

Figure 4.2: Histogram of the galaxy data overlaid with the density estimation of a normal mixture of six components.

- Model 2:

$$f(y_i|\mu_2, \sigma_2^2, q_2) = \sum_{k=1}^{4} q_{2k}\phi(y_i|\mu_{2k}, \sigma_2^2),$$

where $\mu_1 = (\mu_{11}, \mu_{12}, \mu_{13})$, $q_1 = (q_{11}, q_{12}, q_{13})$, and similarly for μ_2 and q_2. Model 1 is a normal mixture of three components while model 2 has four normal components. To formalize the problem, we specify the prior settings in Table 4.4.

Parameter	Prior		Parameter	Prior	
Model 1	Mean	Standard deviation	Model 2	Mean	Standard deviation
μ_{11}	9.000	5.000	μ_{21}	9.000	5.000
μ_{12}	18.000	5.000	μ_{22}	18.000	5.000
μ_{13}	30.000	5.000	μ_{23}	22.000	5.000
			μ_{24}	30.000	5.000
σ_1^2	20.000	20.000	σ_2^2	15.000	15.000
q_{11}	0.333	0.236	q_{21}	0.136	0.100
q_{12}	0.333	0.236	q_{22}	0.364	0.139
q_{13}	0.333	0.236	q_{23}	0.364	0.139
			q_{24}	0.136	0.100

Table 4.4: A summary of the prior settings for two competing models

We have weak priors of normal, inverse gamma, and Dirichlet for μ_{jk}, q_j and σ_j^2, respectively ($j = 1, 2$). In general, the prior means and standard deviations listed in Table 4.4 are to form the rather vague priors. We then compute the Bayes factor following the same strategy in the pine data example. Carlin and Chib (1995) states that the two models are comparable, with the point estimate of B_{21} to be 0.572. Our method, however, shows that the data are greatly in favor of model 2, as $B_{21} > 15$. All parameter estimates agree except for the model indicator, with such discrepancy arising from the Chib's implementation. Additional evidence is shown in Neal's reanalysis (Neal, 1999). For example, Neal points out that Chib "relabels the mixture components after each gibbs sampling step so that it is an increasing order, leading to an underestimate of the marginal likelihood". In Section 4.2, we present an example to illustrate the ability of the multiple chain method for estimating variance components in mixed effect models.

4.2 Estimating Variance Components in Mixed Effect Models

In this section, we discuss the problem of variance component estimation in mixed effect models (Gelman et al., 1995). We know that variance components include the individual variance and the group-level variance. While estimating the individual variance is not difficult as long as we have enough samples, the group-level variance is more challenging to estimate (Cheng et al., 2014). The widely used Gibbs sampler has the following drawbacks. Due to the dependency between groups of coefficients and their variance components in the

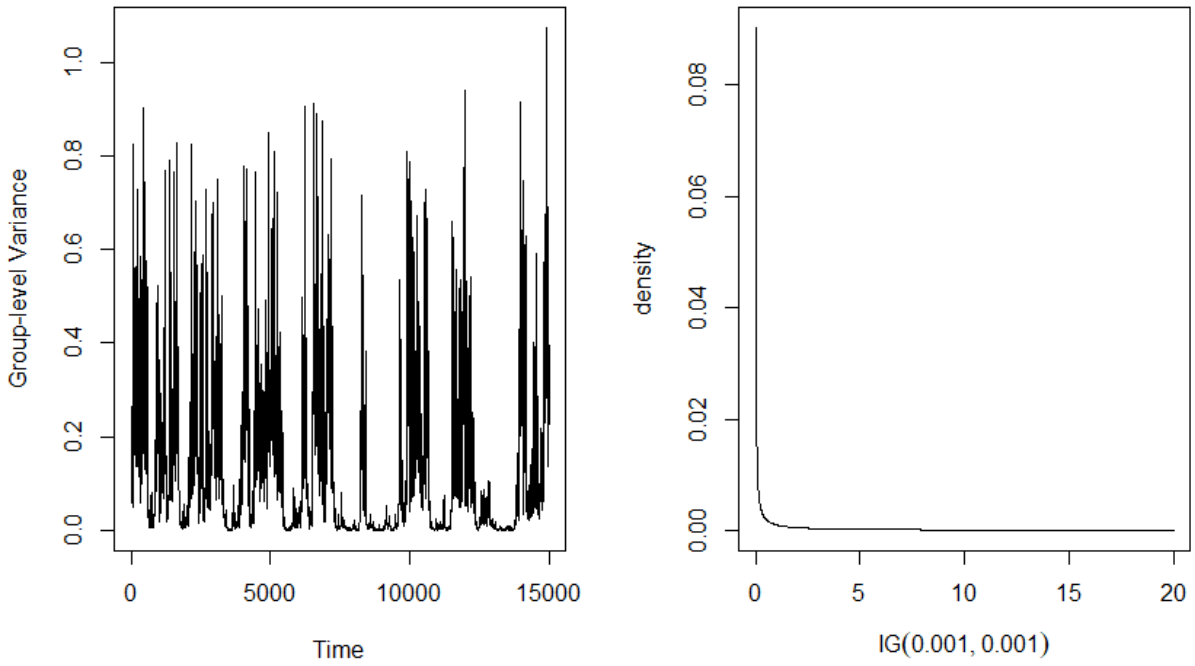

Figure 4.3: Gibbs sampler in estimating group-level variance in conjugate hierarchical models

posterior, Gibbs sampler can get stuck when the variance parameter happens to be estimated near zero (Gelman et al., 2008). We can see from the left panel of Figure 4.3 that the chain gets stuck at zero when the true value equals 0.1. Moreover, the conjugate prior $IG(\epsilon, \epsilon)$ is

too informative in cases where the group-level variance is estimated to be near zero, since the posterior distribution is very sensitive to the choice of ϵ. For example, $IG(0.001, 0.001)$, as shown in the right panel of Figure 4.3, is sharply peaked near zero. Therefore, the inferences of posterior distributions are heavily constrained by this prior. On the contrary, compared with the commonly used Gibbs sampler, the multiple chain method is flexible about the prior choice, allowing a better exploration of the parameter space.

We take the example of a conjugate hierarchical model to illustrate this. Consider J independent experiments, with experiment j having mean parameter θ_j from n_j i.i.d. normal data points, y_{ij}. Each data point has an unknown error variance σ^2. In other words, we consider a one-way ANOVA model:

$$y_{ij}|(\theta_j, \sigma^2) \sim N(\theta_j, \sigma^2), \quad i = 1, 2, ...n_j$$
$$\theta_j \sim N(\mu, \tau^2), \quad j = 1, 2, ...J$$

where $\mu \sim N(\mu_0, \sigma_0^2)$, $\sigma^2 \sim IG(a_1, b_1)$ and $\tau^2 \sim IG(c_1, d_1)$. Note that we assume σ^2 to be the same across all groups. The problem of interest is to estimate variance components (σ^2, τ^2). While Gibbs sampler updates all the parameters iteratively, our method directly samples from the marginal distribution of (σ^2, τ^2). The idea is to reduce the dimensionality of parameters, in order to have a reasonable acceptance rate in MCMC sampling. Thus, we need to integrate out all other parameters. Following Gelman et al. (1995), we integrate out the mean parameters analytically as shown below.

First, we write down the joint posterior of all parameters as

$$
\begin{aligned}
p(\theta, \mu, \sigma^2, \tau^2|y) &\propto p(\mu, \sigma^2, \tau^2)p(\theta|\mu, \tau^2)p(y|\theta, \sigma^2) \\
&\propto p(\mu, \sigma^2, \tau^2) \prod_{j=1}^{J} N(\theta_j|\mu, \tau^2) \prod_{j=1}^{J} \prod_{i=1}^{n_j} N(y_{ij}|\theta_j, \sigma^2) \\
&\propto p(\mu, \sigma^2, \tau^2) \prod_{j=1}^{J} N(\theta_j|\mu, \tau^2) \prod_{j=1}^{J} \prod_{i=1}^{n_j} \frac{1}{\sigma} \exp\{-\frac{1}{2\sigma^2}(y_{ij} - \theta_j)^2\} \\
&\propto p(\mu, \sigma^2, \tau^2) \prod_{j=1}^{J} N(\theta_j|\mu, \tau^2) \prod_{j=1}^{J} (\sigma^2)^{-\frac{n_j}{2}} \exp\{-\frac{1}{2\sigma^2}\{n_j(\bar{y_{.j}} - \theta_j)^2 \\
&\qquad + \sum_{i=1}^{n_j}(\bar{y_{.j}} - y_{ij})^2\}\} \\
&\propto p(\mu, \sigma^2, \tau^2) \prod_{j=1}^{J} N(\theta_j|\mu, \tau^2)(\sigma^2)^{-\frac{N}{2}} \exp\{-\frac{1}{2\sigma^2} \sum_{j=1}^{J}\{n_j(\bar{y_{.j}} - \theta_j)^2 \\
&\qquad + \sum_{i=1}^{n_j}(\bar{y_{.j}} - y_{ij})^2\}\}, \qquad (4.4)
\end{aligned}
$$

where $\theta = (\theta_1, \theta_2, \ldots, \theta_J)$, $\bar{y_{.j}} = \frac{1}{n_j} \sum_{i=1}^{n_j} y_{ij}$ and $N = \sum_{j=1}^{J} n_j$.
Note that for $j = 1, 2, \ldots, J$, we have

$$p(\theta_j|\mu, \sigma^2, \tau^2, y) \propto N(\theta_j|\mu, \tau^2) \times N(\theta_j|\bar{y_{.j}}, \sigma_j^2),$$

where $\sigma_j^2 = \frac{\sigma^2}{n_j}$.
As a result, the conditional posterior of θ_j follows a normal distribution:

$$\theta_j|\mu, \sigma^2, \tau^2, y \sim N(\hat{\theta_j}, V_j), \qquad j = 1, 2, \ldots, J$$

where $\hat{\theta}_j = \frac{\frac{1}{\sigma_j^2}\bar{y_{.j}} + \frac{1}{\tau^2}\mu}{\frac{1}{\sigma_j^2} + \frac{1}{\tau^2}}$ and $V_j = \frac{1}{\frac{1}{\sigma_j^2} + \frac{1}{\tau^2}}$.
Then we get the joint posterior of hyperparameters from (4.4) after integrating out θ:

$$p(\mu, \sigma^2, \tau^2|y) \propto p(\mu, \sigma^2, \tau^2)(\sigma^2)^{-\frac{N}{2}} \exp\{-\frac{1}{2\sigma^2}\sum_{j=1}^{J}(n_j - 1)S_j^2\} \prod_{j=1}^{J} N(\bar{y_{.j}}|\mu, \sigma_j^2 + \tau^2) \prod_{j=1}^{J} \sigma_j, \tag{4.5}$$

where $S_j^2 = \frac{1}{n_j - 1}\sum_{i=1}^{n_j}(\bar{y_{.j}} - y_{ij})^2$.
To get $p(\sigma^2, \tau^2|y)$, we have to know $p(\mu|\sigma^2, \tau^2, y)$ and apply the conditional probability formula:

$$p(\sigma^2, \tau^2|y) = \frac{p(\mu, \sigma^2, \tau^2|y)}{p(\mu|\sigma^2, \tau^2, y)} \tag{4.6}$$

Similar to θ_j, we have

$$p(\mu|\sigma^2, \tau^2, y) \propto N(\mu|\mu_0, \sigma_0^2) \times \prod_{j=1}^{J} N(\mu|\bar{y_{.j}}, \sigma_j^2 + \tau^2),$$

i.e.,

$$\mu|\sigma^2, \tau^2, y \sim N(\hat{\mu}, V_\mu), \tag{4.7}$$

where $\hat{\mu} = \frac{\sum_{j=1}^{J} \frac{1}{\sigma_j^2 + \tau^2}\bar{y_{.j}} + \frac{1}{\sigma_0^2}\mu_0}{\sum_{j=1}^{J} \frac{1}{\sigma_j^2 + \tau^2} + \frac{1}{\sigma_0^2}}$ and $V_\mu = \frac{1}{\sum_{j=1}^{J} \frac{1}{\sigma_j^2 + \tau^2} + \frac{1}{\sigma_0^2}}$.
By substituting the numerator and denominator of (4.6) with (4.5) and (4.7), respectively,

we have

$$p(\sigma^2,\tau^2|y) = \frac{p(\mu,\sigma^2,\tau^2|y)}{p(\mu|\sigma^2,\tau^2,y)} \propto \frac{p(\mu,\sigma^2,\tau^2)(\sigma^2)^{-\frac{N}{2}}\exp\{-\frac{1}{2\sigma^2}\sum_{j=1}^{J}(n_j-1)S_j^2\}\prod_{j=1}^{J}N(\bar{y_{.j}}|\mu,\sigma_j^2+\tau^2)\prod_{j=1}^{J}\sigma_j}{N(\mu|\hat{\mu},V_\mu)}. \tag{4.8}$$

Since (4.8) holds for all $\mu's$, we set $\mu=\hat{\mu}$ and obtain

$$\begin{aligned} p(\sigma^2,\tau^2|y) &\propto \frac{p(\hat{\mu},\sigma^2,\tau^2)(\sigma^2)^{-\frac{N}{2}}\exp\{-\frac{1}{2\sigma^2}\sum_{j=1}^{J}(n_j-1)S_j^2\}\prod_{j=1}^{J}N(\bar{y_{.j}}|\hat{\mu},\sigma_j^2+\tau^2)\prod_{j=1}^{J}\sigma_j}{N(\hat{\mu}|\hat{\mu},V_\mu)} \\ &\propto p(\sigma^2,\tau^2)(\sigma^2)^{-\frac{N}{2}}(V_\mu)^{\frac{1}{2}} \\ &\times \exp\{-\frac{1}{2\sigma^2}\sum_{j=1}^{J}(n_j-1)S_j^2\}\prod_{j=1}^{J}\{(\frac{\sigma_j^2}{\sigma_j^2+\tau^2})^{\frac{1}{2}}\exp[-\frac{(\bar{y_{.j}}-\hat{\mu})^2}{2(\sigma_j^2+\tau^2)}]\} \end{aligned} \tag{4.9}$$

By Equation 4.9, we successfully reduce the dimensionality of parameters to just two in σ^2 and τ^2. We have reasonable acceptance rates in the multiple chain method, as the proposal function is just a product of two univariate normals. With the desired density, we can simply sample from it using the multiple chain method. By setting $\tau^2 = 0.01$ and $J = 100$, we generate the data and apply three methods including Gibbs sampler, the usual Metropolis, and the multiple chain method. Figure 4.4 shows autocorrelation plots (left) and scatterplots of draws overlaid by contours of the exact posterior (right), which compare Gibbs sampler (top), the usual Metropolis (middle) and the multiple chain method showing one single chain (bottom). Clearly, the multiple chain method has better efficiency where samples are much less correlated while the other two methods suffer from very high autocorrelation. This example demonstrates the limitations of both Gibbs sampler and the usual Metropolis, and indicates that our multiple chain method is superior to them. In Section 4.3, we apply the multiple chain method in sensor network localization.

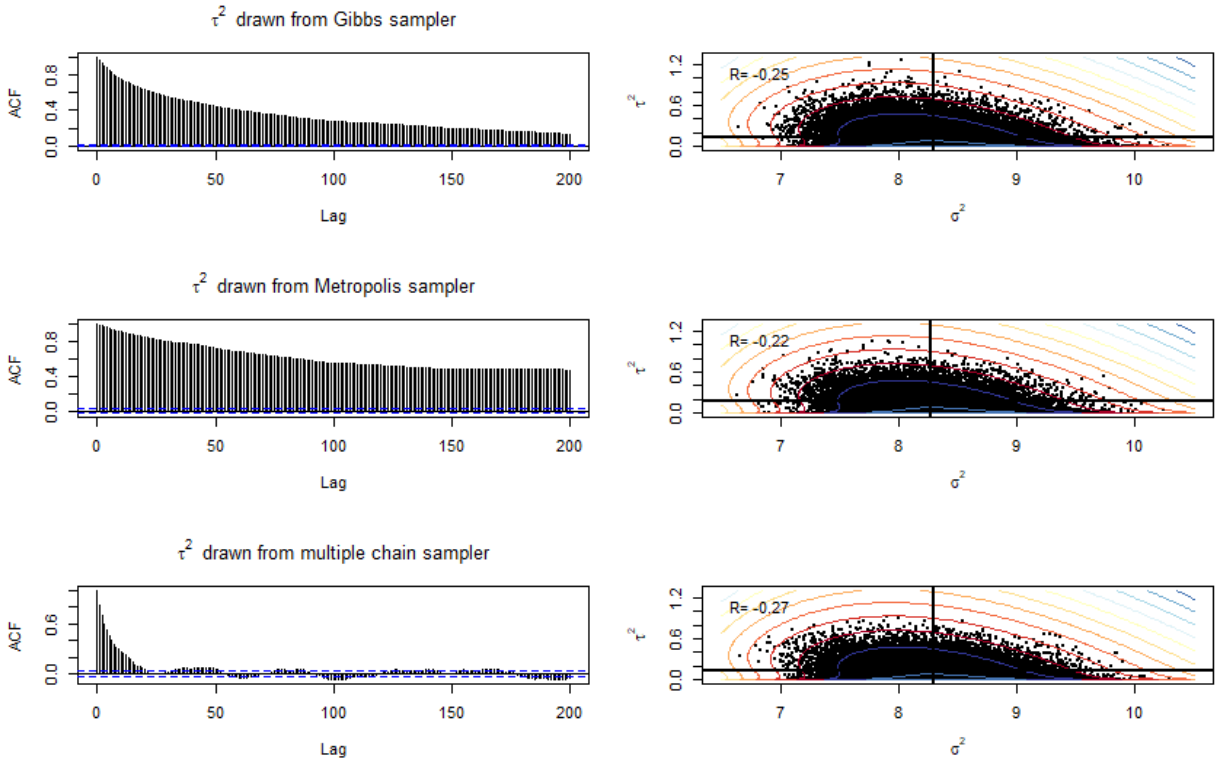

Figure 4.4: Autocorrelation plots (left) and scatterplots of draws overlaid by contours of the exact posterior (right) comparing single chain drawn from Gibbs sampler (top), the usual Metropolis (middle) and the multiple chain method (bottom).

4.3 Sensor Network Localization

In this section, we discuss the sensor network localization problem formed by Ihler et al. (2005). The interest lies in self-localization of unknown sensors in the wireless network, given noisy distance information. This problem has been discussed in MCMC area and the goal is to estimate the sensor locations based on the posterior distribution, which is known to be mulitimodal and complicated-shaped (Ahn et al., 2013; Lan et al., 2014; Tak et al., 2018). To be specific, assume there are N sensors scattered in a 2-D space, denoted as $x_k = (x_{k1}, x_{k2})^T$, $k = 1, 2, \ldots, N$. For any sensor x_i, x_j $(j \neq i)$ is observed with a distance-dependent probability

$$p_{ij} = \exp(-\frac{\|x_i - x_j\|^2}{2R^2}),$$

where $\|\cdot\|$ denotes the Euclidean norm. And the observed distance is distorted by a Gaussian noise $v_{ij} \sim N(0, \sigma^2)$. In other words, we introduce an observation indicator z_{ij} to suggest whether the distance between x_i and x_j $(j \neq i)$ is observed:

$$z_{ij} \sim \text{Bernoulli}\big(\exp\big(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\big)\big).$$

If the distance y_{ij} between x_i and x_j is observed, we have $z_{ij} = 1$ and the observed distance as

$$y_{ij}|z_{ij} = 1 \sim N(\|x_i - x_j\|,\ \sigma^2).$$

For simplicity, we assume that if x_i observes x_j, x_j also observes x_i and the observed distances are the same, *i.e.*, $z_{ij} = z_{ji}$ and $y_{ij} = y_{ji}$ (Ihler et al., 2005). In this work, we follow the experiment settings in Tak et al. (2018) and set $N = 6$, $R = 0.3$ and $\sigma = 0.02$. To avoid ambiguities of translation, rotation, and negation (Ihler et al., 2005), we put two of them, *i.e.*, x_5 and x_6, as base sensors with known locations. See Figure 4.5 from Tak et al. (2018) for the displayed sensors that are simulated. The parameters of interest are the locations of these unknown sensors, *i.e.*, $\{(x_{k1}, x_{k2}),\ k = 1, 2, 3, 4\}$, and the data we have is a set of observed distances $\{y_{ij}\}$. In this case, the locations of the 4 unknown sensors form a multimodal distribution of 8 dimensions. Given data $\{z : z_{ij},\ j > i\}$ and $\{y : y_{ij},\ j > i\}$, the data likelihood is

$$\begin{aligned} L(z, y|x_1, x_2, x_3, x_4) \propto \prod_{j>i} [&\exp(-\frac{(y_{ij} - \|x_i - x_j\|)^2}{2 \times 0.02^2}) \\ &\times \exp(-\frac{\|x_i - x_j\|^2}{2 \times 0.3^2})^{z_{ij}} \times (1 - \exp(-\frac{\|x_i - x_j\|^2}{2 \times 0.3^2}))^{1-z_{ij}}]. \end{aligned}$$

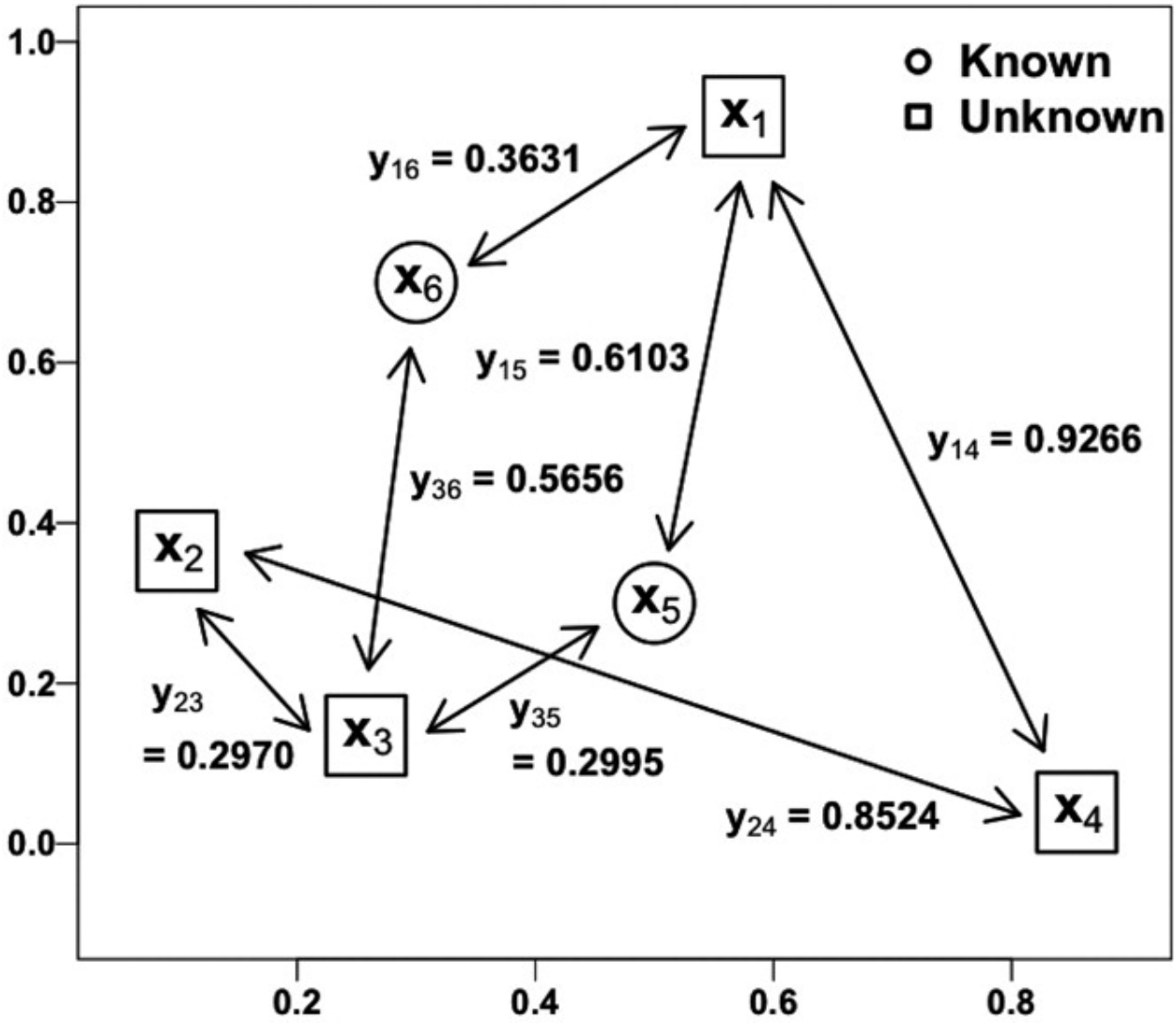

Figure 4.5: The simulated distances y_{ij} among the six stationary sensor locations, $x_1, x_2, \ldots x_6$, are displayed if observed. The observation indicator z_{ij} is one if y_{ij} is specified and is zero otherwise. The locations of the sensors are $x_1 = (0.57, 0.91)$, $x_2 = (0.10, 0.37)$, $x_3 = (0.26, 0.14)$, $x_4 = (0.85, 0.04)$, $x_5 = (0.50, 0.30)$, and $x_6 = (0.30, 0.70)$, where the first four locations, x_1, x_2, x_3, and x_4, are assumed to be unknown (Tak et al., 2018).

Following Tak et al. (2018), we assume a bivariate Gaussian prior

$$\pi_0 \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix}\right)$$

for each $x_i, i = 1, 2, 3, 4$, and the resulting posterior distribution is

$$\pi(x_1, x_2, x_3, x_4|z, y) \propto L(z, y|x_1, x_2, x_3, x_4) \times \exp(-\frac{\sum_{i=1}^{4} x_i^T x_i}{2 \times 10^2}). \tag{4.10}$$

Tak et al. (2018) apply Gibbs sampler on this target (4.10) by sampling each conditional bi-

Kernel	Length of a chain	N_π^X	Details of N_π^X	Acceptance rate
Metropolis	1,980,000	$N_\pi^M = 4$	1 for each of $x_1, \ldots, x_4$	0.00056 for x_1
				0.00162 for x_2
				0.00052 for x_3
				0.00132 for x_4
Multiple chain	79,200	$N_\pi^{MC} = 10$	1 for each of $x_1, \ldots, x_4$	0.00057 for x_1
				0.00147 for x_2
				0.00050 for x_3
				0.00119 for x_4
			6 for between-chain jumping	0.00008 for (x_1, x_2, x_3, x_4)
RAM	220,000	$N_\pi^{RAM} = 36$	9.27 for x_1	0.00326 for x_1
			8.58 for x_3	0.00791 for x_2
			9.03 for x_4	0.00328 for x_3
			8.93 for x_2	0.00722 for x_4
Tempered transition	330,000	$N_\pi^{TT} = 24$	6 for each of $x_1, \ldots, x_4$	0.00348 for x_1
				0.00947 for x_2
				0.00330 for x_3
				0.00839 for x_4

Table 4.5: Summaries of these four methods for the length of a chain (20,000 burn-in period included), the average number of posterior density evaluations per iteration N_π^X as well as its breakdown information, and the acceptance rate.

variate distribution of $\pi_i(x_i|x_j, j \neq i, z, y)$ for $i = 1, 2, 3, 4$ iteratively. Three kernels discussed include Metropolis, repelling-attracting Metropolis (RAM) (Tak et al., 2018) and tempered transition (TT) (Neal, 1996). See Tak et al. (2018) for the detailed implementation of the first three methods. Based on these available results, we present our multiple chain method (MC) and compare performances among all these four kernels. In our method, we run 10 parallel chains from random starting values in $[0, 1]^2$. The within-chain jumps update current iterates by Gibbs sampler, whose procedure is exactly the same as that in the Metropolis

method in Tak et al. (2018). The between-chain jumps update current iterates by proposing candidates of full 8 dimensions together. This is because we cannot reduce it to a lower dimension (see Section 3.4.3) and the purpose of this part is to make between-mode jumps. Specifically, we use a multivariate normal random walk as the proposal distribution. Under these settings, Metropolis can be compared as the baseline method, since our method would be exactly the same if we disable the between-chain jumps. Following Tak et al. (2018), we set the same amount of computational time for each method to make a fair comparison. We denote N_π^X as the average number of times of evaluations of π (4.10) per iteration, and it is evaluated for each method (X = M, MC, TT or RAM). The length of each chain is then computed accordingly. For the between-chain jumping step in the multiple chain method (MC), we evaluate the full posterior density of π once (with caching) and the transition kernel several times (a mixture of bivariate normals). Thus, N_π is approximated based on computational time. The approximation is conservative so that the multiple chain method would make at least as good performance, as shown below. For all methods, we discard the first 20,000 iterations as burn-in period to remove the dependence of chains on their starting locations. A summary of the configurations as well as the acceptance rates of these four methods is shown in Table 4.5. Note that, as we have 10 parallel chains in the multiple chain method, the length of a chain is divided by 10 to have the same number of evaluations of π. From Table 4.5, we can see that RAM and TT improve the acceptance rate compared to Metropolis, while the acceptance rate in MC between-chain jumping is very low. On one hand, this low acceptance rate (8×10^{-5}) is due to the 8-dimensional jumping kernel. On the other hand, even though there are only 62 ($8 \times 10^{-5} \times 772,000$) successful between-chain jumps, we can see that MC indeed helps the chain mix much better compared to Metropolis. Figure 4.6 shows the scatterplot of posterior draws from above four methods. Each row represents the two dimensional coordinates of one sensor location. And for each column samples are drawn by one of the four methods. Note that the dashed lines indicate the true coordinates of eac unknown sensors. We see that the multiple chain method, RAM and TT have

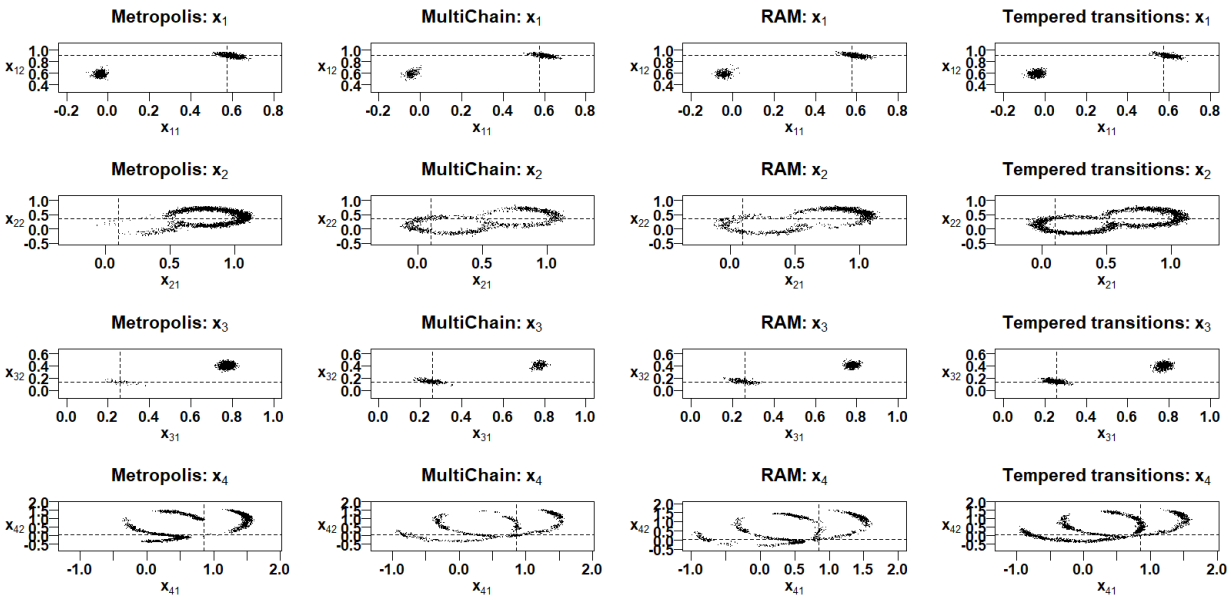

Figure 4.6: Scatterplots of posterior draws of each sensor location (rows) from four methods (column), where true coordinates are denoted by dashed lines.

more dispersed samples than that of Metropolis, especially for x_2 and x_4. And the posterior samples of Metropolis, RAM, and TT, are denser than that of the multiple chain method because of either the larger sample size or the higher acceptance rate. Figure 4.7 compares

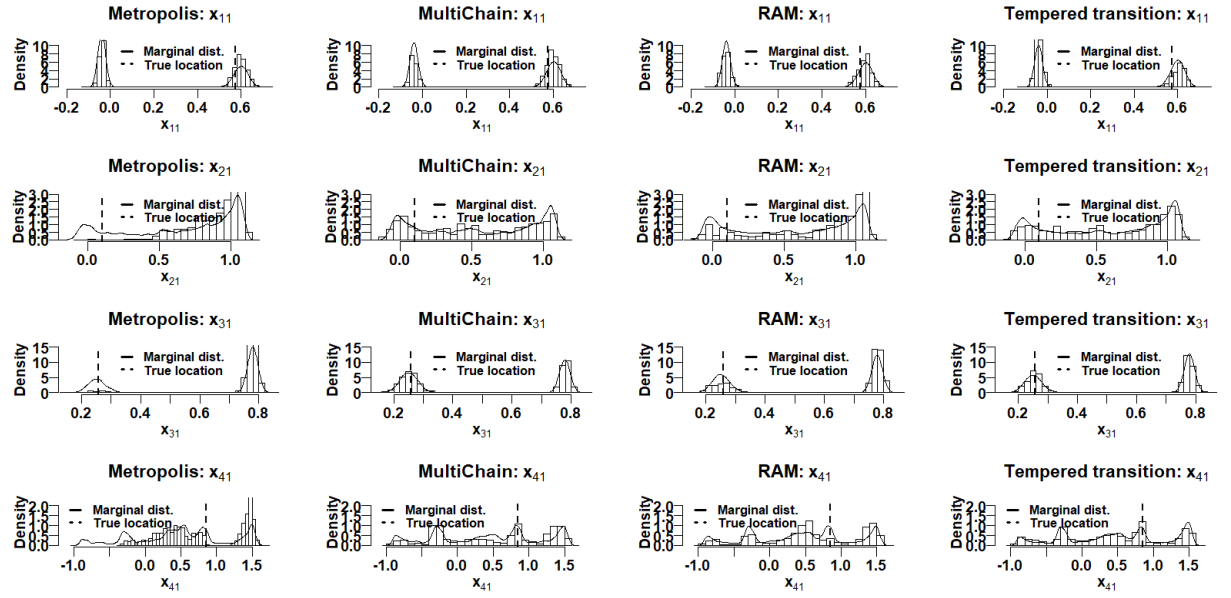

Figure 4.7: Histograms of posterior draws of each first coordinate (rows) from four methods (column), overlaid by the marginal posterior density based on 20 million draws from each method, where true coordinates are denoted by vertical dashed lines.

the relative sizes of modes for the first coordinate of each unknown location (rows) draws from four methods (columns). In each histogram, we superimpose the marginal posterior density based on another twenty million posterior draws from each method. We see that the shapes of the density plots from four methods are almost identical. Therefore, they can be safely treated as the true posterior density. Then we are able to assess the convergence by comparing the histogram (sample) to the density plot (truth). Note that the true locations of sensors are indicated by the vertical dashed lines. The multiple chain method outperforms Metropolis in all four distributions, because of the between-chain jumps that help chains better mix among different modes. RAM and TT still perform relatively well, compared to the multiple chain method. However, these two methods are not stable. Draws obtained by RAM and TT are not always consistent with the marginal posterior truth, indicating that the methods do not fully converge. The Metropolis method draws underrepresent the marginal posterior truth most of the time, as 1980,000 draws might not be large enough for the method to converge. In Chapter 5, we apply the multiple chain method on the astronomical data to estimate the parameters of interest.

Chapter 5

Capella Data

In this chapter, we apply the proposed multiple chain method on the astronomical data to estimate the parameters of interest, where the model is structured without an excessive amount of parameters while its likelihood function is complicated and takes time to evaluate.

5.1 Backgroud

Recent advances in astronomy promote us to perform in-depth studies on high-energy astrophysics. The development of astronomical instrumentation brings the massive amount of high quality data. For example, Figure 5.1 provides four images taken by the Chandra X-ray Observatory (CXO). The top-left panel shows a supernova remnant in the Milky Way (Credit: NASA/CXC/U.Texas/S.Post et al.); the top-right panel shows a supermassive black hole at the center of the Milky Way (Credit: NASA/SAO/CXC/M.Markevitch et al.); the bottom-left panel shows a cluster of galaxies in X-rays (Credit: NASA/CXC/Univ. of Wisconsin/Y.Bai, et al.); the bottom-right panel shows the jet of a quasar (credit: NASA/CXC/A.Siemiginowska(CfA)/J.Bechtold(U.Arizona)). Such image data contain valu-

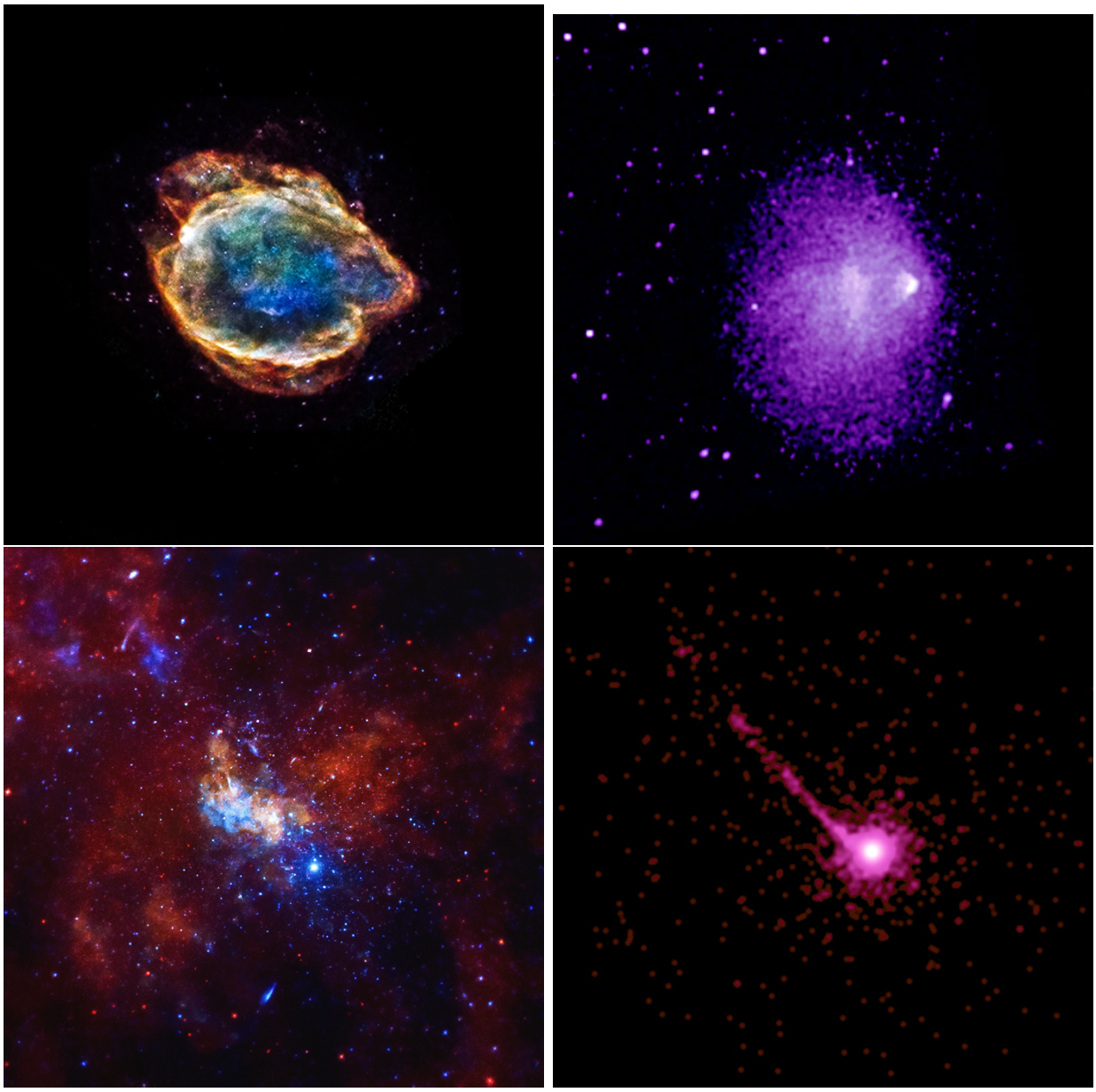

Figure 5.1: Images taken by the Chandra X-ray Observatory. Source: NASA/CXC/SAO

able information about spectral, temporal, and spatial characteristics of the astronomical sources. In practice, high energy electromagnetic radiation also includes Gamma-ray and extreme Ultraviolet, in addition to X-ray.

In this work, we focus on the spectral analysis of X-ray astronomy. We extract spectral data from images, which display the frequency distribution of observed counts as a function of

wavelength, as shown in Figure 5.4. The shape and structure of the frequency distribution can be used to infer the astronomical source's components as well as its density, temperature, etc. To be specific, arriving photons are collected by detectors and then the emission of these photons with energies is represented by a spectrum, which denotes the energy flow per wavelength. This spectrum is characterized by a continuum, several emission lines and absorption features, where the continuum describes spectrum's general shape and emission lines represent extra photon emissions in a narrow energy band (Van Dyk et al., 2001; van Dyk et al., 2004). The continuum may indicate the source's temperature, and emission and absorption lines imply the relative abundances of elements. Therefore, it is important to accurately model the spectra. Such photon counts are modeled with a Poisson process whose Poisson parameter is a function of energy composed of the continuum and several emission and absorption lines. To be concrete, photon counts are recorded into 4096 energy channels for CXO. Then we perform energy binning and obtain I bins.

With perfect instrumentation, the expected counts in bin i, $i \in I$, will be the sum of a continuum and several emission lines, multiplied by an absorption factor accounting for the stochastic censoring of photons. First, the continuum serves as the baseline spectrum and may have one or more continuum components. The continuum with J components can be modeled by the sum of a number of generalized linear models (GLMs) $f_j(\theta_j^C, E)$, where E is the photon energy, C represents the continuum and θ_j^C is the collection of parameters in the continuum component j. Examples of $f_j(\theta_j^C, E)$ are power law:

$$\alpha E^{-\beta},$$

and blackbody emission:

$$\alpha E^2/(e^{\beta E} - 1),$$

where $\theta_j^C = \{\alpha, \beta\}$ in the above two cases. As a result, the Poisson model intensity from the continuum is

$$f(\theta^C, E) = \sum_{j=1}^{J} f_j(\theta_j^C, E),$$

where $\theta^C = \{\theta_j^C, j \in J\}$ is the collection of parameters for the set of continuum components. Assuming the binning is fine enough, the expected number of counts in bin i from the continuum is roughly equal to $\delta_i f(\theta^C, E_i)$, where δ_i and E_i are the width and mean energy of bin i, respectively. Secondly, the emission lines can be modeled by a mixture of normals:

$$\sum_{k=1}^{K} \theta_{k,\lambda}^L p_i(\theta_{k,\mu}^L, \theta_{k,\sigma}^L),$$

where L refers to the emission lines and we assume there are K emission lines in total. Here, $\theta_{k,\lambda}^L$ is the expected photon count from emission line k, $p_i(\theta_{k,\mu}^L, \theta_{k,\sigma}^L)$ is the probability that a photon from emission line k with center $\theta_{k,\mu}^L$ and spread $\theta_{k,\sigma}^L$ falls in bin i. Finally, the sum of the continuum and emission lines is multiplied by an absorption factor representing the probability that a photon in bin i is not absorbed:

$$u(\theta^A, E_i),$$

where A refers to the absorption features and θ^A is the collection of parameters for the absorption features. Therefore, the Poisson model intensity in bin i is

$$\lambda_i(\theta^C, \theta^L, \theta^A) = \left(\delta_i f(\theta^C, E_i) + \sum_{k=1}^{K} \theta_{k,\lambda}^L p_i(\theta_{k,\mu}^L, \theta_{k,\sigma}^L) \right) u(\theta^A, E_i),$$

where $\theta^L = \{\theta_{k,\lambda}^L, \theta_{k,\mu}^L, \theta_{k,\sigma}^L, k \in K\}$ is the collection of parameters for the emission lines. However, photon counts are distorted due to instrumental constraints and background contamination in practice. On one hand, instrumental imperfectness is accounted for by effective

area curves and energy redistribution matrices. An effective area curve models the detector's efficiency of successfully recording photon counts as a function of energy, where we use d_i to represent the probability that an incoming photon corresponding to energy bin i is recorded. An energy redistribution matrix describes instrument response that a photon can be recorded into a range of energy bins due to blurring of the photon energy, where we employ M_{hi} to denote the probability that a photon corresponding to bin i is recorded in bin h. On the other hand, X-ray count data are usually contaminated with background counts. To tackle this problem, we model background counts as a Poisson model and add the background intensity to the source intensity. To summarize, we can model the observed counts in bin h as a Poisson variable with intensity

$$\xi_h(\theta) = \sum_{i=1}^{I} M_{hi}\lambda_i(\theta^C, \theta^L, \theta^A)d_i + \lambda_h^B(\theta^B),$$

where B refers to the background contamination, θ^B is the collection of parameters for the background intensity, $\lambda_h^B(\theta^B)$ is the Poisson intensity in bin h and $\theta = (\theta^C, \theta^L, \theta^A, \theta^B)$. This mixture model is likely to be multimodal while its dimension is not very high as the model is highly structured (van Dyk et al., 2004). In addition, data are stored in a four-way table of photon counts while spectral analysis aims at modeling the one-way energy margin. The models are often in a tabular form and we have to look up values in the table. Therefore, the task becomes more complicated as we are not dealing with a smooth and analytic expression. In other words, it leads to a very time-consuming likelihood evaluation. Our method could be a plausible solution to the above challenges along with the provided information, as the model is structured without too many parameters while its likelihood function is complicated and takes time to evaluate.

In this work, data are recorded by CXO, a space-based telescope launched by NASA. It is located in high-earth orbit and designed to detect X-ray emission from hot and chaotic regions of the Universe. CXO has two detectors: One is the advanced camera for imaging

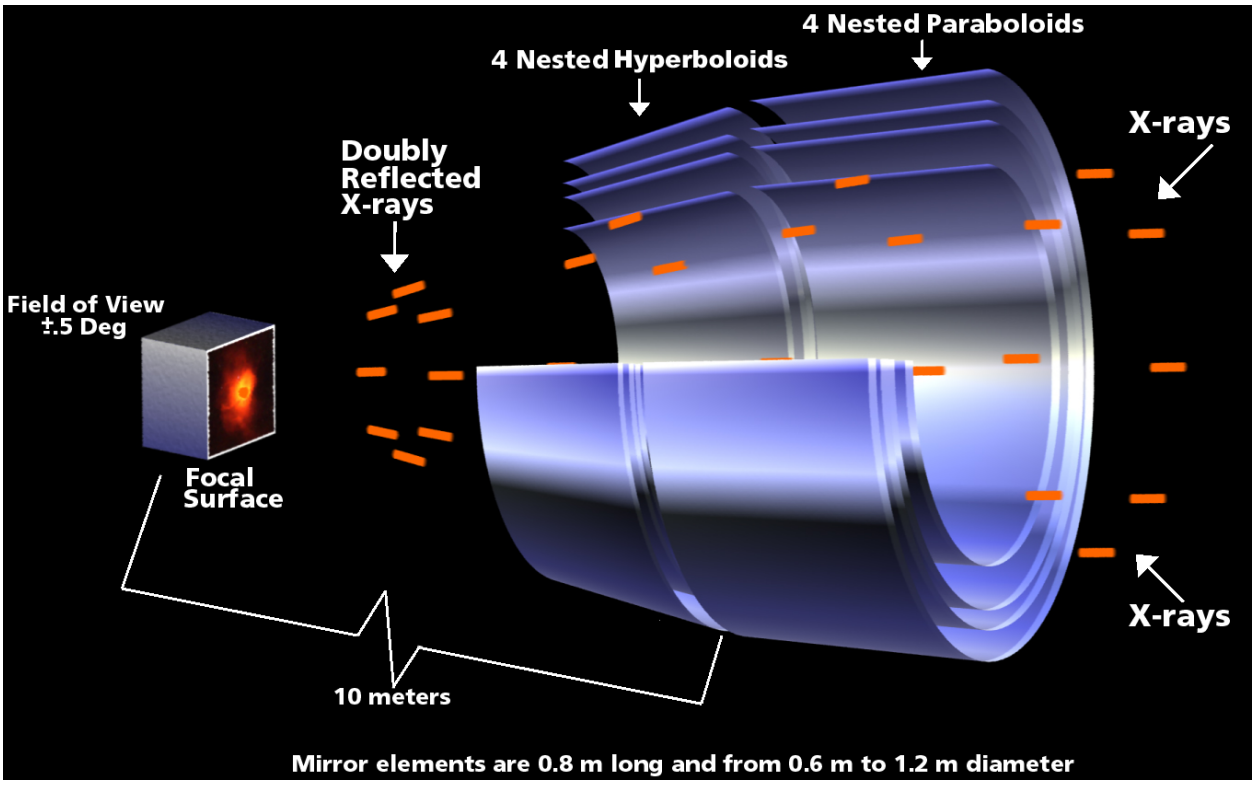

Figure 5.2: Schematic of grazing incidence, X-Ray mirror. Source: NASA/CXC/D.Berry

and spectroscopy (ACIS), which is categorized in charged coupled devices (CCD); the other one is the high resolution camera (HRC), which comprises two micro-channel plate imaging detectors and is used for high resolution imaging. In addition, CXO has two gratings for high resolution spectroscopy: One is the high energy transmission grating spectrometer (HETGS); the other one is the low energy transmission grating spectrometer (LETGS). The mirror of CXO, also referred to as the high resolution mirror assembly (HRMA), consists of four pairs of nested mirrors containing both paraboloids and hyperboloids. NASA/CXC (2018) Figure 5.2 illustrates the design and functioning of the mirrors of CXO. When the X-rays are released by the source and go through the space, they are scattered on the mirror, reflected and detected by the telescope. Then data are collected for each arriving photon, recording its position, energy, and arrival time. Afterwards, the detector is able to generate images that can be used to understand the extremely hot and high-energy regions of the Universe. The data used in this work come from Capella, the strongest non-solar coronal source accessible

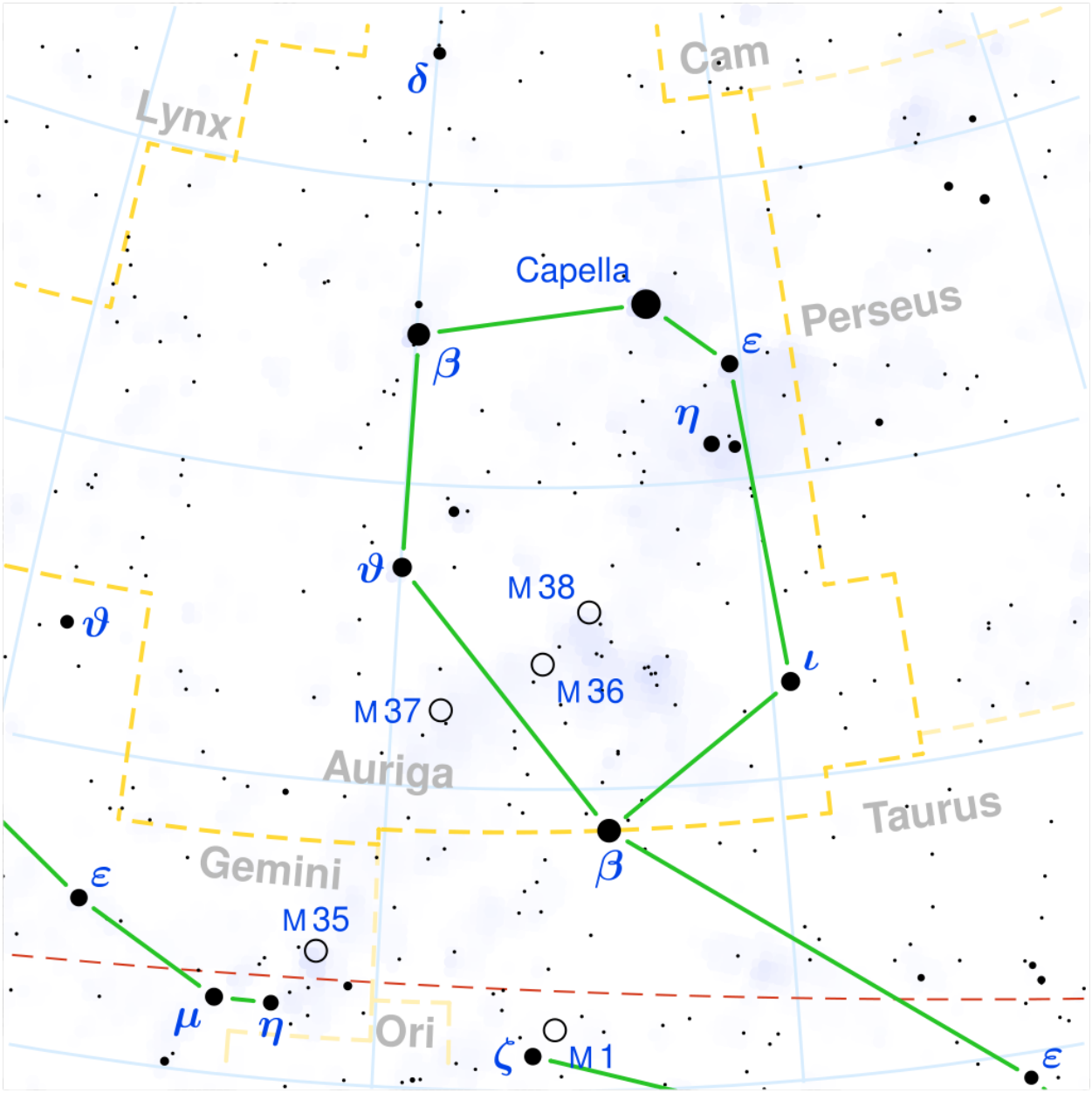

Figure 5.3: Auriga constellation map by Torsten Bronger (2003).

to X-ray telescopes. It is the third-brightest in the northern celestial hemisphere, located in the constellation of Auriga (Figure 5.3). The star is very stable, whose overall luminosity has been steady for many years without discernible flaring activity. As a result, it has been a common calibration target for X-ray instruments such as CXO (Kashyap and Posson-Brown, 2007). To summarize, we have the following information:

- Source: Capella
  - Strongest non-solar coronal source accessible to X-ray telescopes
  - Very stable and the overall luminosity has been steady for many years with no discernible flaring activity
- Instrument: Chandra ACIS-S
  - Obtained a set of contiguous observations of Capella during July 2016 (ObsID: 18358-18364)

Here, contiguous observations refer to those of more than $2 - 3 \times 10^3$ seconds duration, as they are typically split into smaller observing intervals. Note that the number of such intervals is arbitrary and difficult to predict. The dataset with ObsID 18358 is depicted using the Chandra Interactive Analysis of Observations software package (CIAO, Fruscione et al., 2006) version 4.8, shown in Figure 5.4. Here, the x-axis and the y-axis represent the wavelength and photon counts with 1-σ (68.3%) confidence interval, respectively. Note that the two panels are spectrum plots with different orders from the same grating. They are intrinsically equivalent to each other with all respects, except for small differences in the Ancillary Response Files (ARFs).

The purpose of studying the above data is to find global thermal fits to high resolution spectra. However, Capella has a continuum of temperature components while we only model a discrete number of such components. Nevertheless, with the high quality data obtained by Chandra, we are capable of exploring whether our explanatory illustrative method can provide any insight.

5.2 Models

To model the data, we adopt a multiplicative model that has two model components:

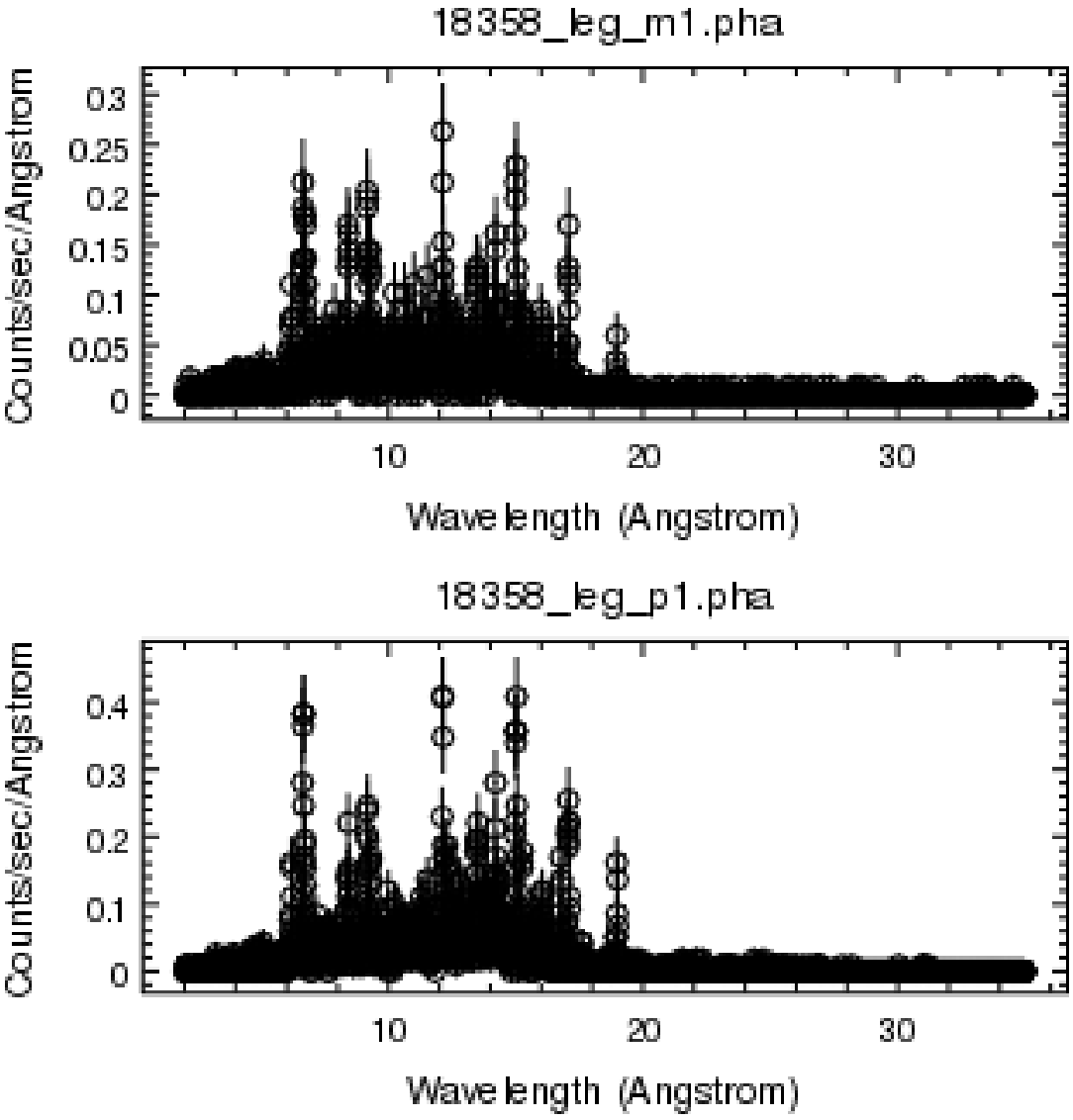

Figure 5.4: Spectra of ObsID: 18358, with different orders from the same grating

- *xsphabs*: photoelectric absorption
- *xsvapec*: thermal plasma model with variable abundances

We are unable to show the explicit forms of these two models. As we have discussed in Section 5.1, these models do not have an mathematical expression but are in a tabular form instead; therefore, evaluations of the data likelihood are time-consuming. The model

Number	Name	Description
1	N_H	The equivalent hydrogen column (in units of 10^{22} $atmos/cm^2$)

Table 5.1: *xsphabs* Parameters (Source: Sherpa help page, CXC/SHERPA/ AHELP)

Number	Name	Description
1	T	The temperature of the plasma, in keV
2-14	(element)	Abundances for He, C, N, O, Ne, Mg, Al, Si, S, Ar, Ca, Fe, Ni in solar units with default value 1.0
15	R	The redshift of the plasma
16	N	The normalization of the model

Table 5.2: *xsvapec* Parameters (Source: Sherpa help page, CXC/SHERPA/ AHELP)

parameters of *xsphabs* and *xsvapec* are shown in Table 5.1 and Table 5.2, respectively. The parameter of interest is the temperature T in *xsvapec* model. The first model is a power law absorption one and the parameter lies within the exponent; and the second one is a thermal plasma model, with the thermal plasmas as ionized gases. The second model describes the optically thin line and continuum emission from collisional excited plasma, with the thin line represents the transitions between quantized energy levels in an ion, where thin here means that the probability of a photon being intercepted after emission by another ion nearby is negligible. Continuum here means that the transitions occur when free electrons are absorbed by an ion or when electrons scatter from each other. The excited plasma stands for the ionized gas with upper energy levels of the ions populated through inelastic collisions. In this multiplicative model, the parameter of interest is the temperature T in *xsvapec* model, and we expect there to be a continuum of temperature components, with the most prominent one probably at 0.54 keV. The fit function in Sherpa (Freeman et al., 2001) (Sherpa fit($\cdot$)), which is the modeling and fitting application used by astronomers, fails to recover this, so we try MCMC method. We first consider a model with just one temperature component (1-T) model: $xsphabs.abs1 * xsvapec.kT1$, and then two temperature components

(2-T): $xsphabs.abs1 * (xsvapec.kT1 + xsvapec.kT2)$.

5.2.1 One Temperature Component

In this subsection, we discuss the 1-T model: $xsphabs.abs1 * xsvapec.kT1$, *i.e.*, one absorption model $abs1$ and one plasma model $kT1$. For this discussion, we work on the data with obsID 18358. As is common practice for astronomers, we set the abundance of He to 1 and other element abundances to 0.6 so that they are treated as known quantities. By thawing Fe[1], we have four parameters in total: N_H, T, A_{Fe}, and N, where A_{Fe} is the abundance for Fe. To get posterior draws by the MCMC method, we apply both the single chain Metropolis method and the multiple chain method. For the single chain method, we run 20 parallel chains for 5,000 iterations and draw the traceplot of temperature parameter T. The acceptance rate is 17% on average. Among these 20 chains, some slowly move towards 0.72, which is the best fit found by Sherpa fit($\cdot$), while others get stuck at local maxima. The local trap problem might be overcome by running longer. However, in our experiment, those chains are far from convergence for a length of 5,000 iterations. For the multiple chain method, we run 20 parallel chains from the same starting values for 2,000 iterations. Traceplots of T comparing the single chain method (left) and the multiple chain method (right) are shown in Figure 5.5. For the multiple chain method, within-chain jump acceptance rate is 19% and between-chain jump acceptance rate is 18%. These two rates are close, since after only 10 iterations, all chains except one go to the same place. As a result, the between-chain jump just performs as the within-chain (mode) jump among these 19 chains, whereas one other chain (yellow) staying somewhere else does not have any communication (successfully between-chain jump) with the major cluster (red), which is indicated by the traceplot (Figure 5.5). This 19 (red) v.s. 1 (yellow) situation has been discussed in Section 3.3.1 and the value indicated by the yellow chain is not meaningful. Note that, in this case, it is probably not a mode, as the

[1]Thaw model parameters means they can vary during a fit.

rough proportion of the yellow cluster is negligible ($< 5\%$). In addition, the zero interaction between these two clusters indicates that the density of red cluster is much higher than that of yellow one. It is worth noting that all chains in the red cluster reach 0.72 in about 1,000 iterations. Therefore, it is clear that the multiple chain method helps converging. In

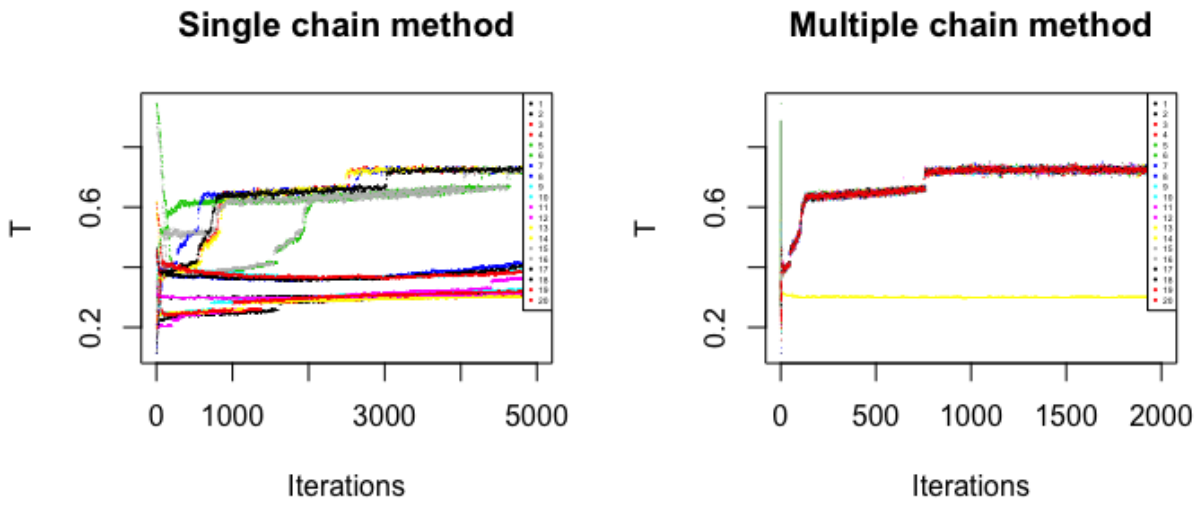

Figure 5.5: Traceplots of T comparing the single chain method (left) and the multiple chain method (right).

summary, our proposed multiple chain method is able to find only one mode with $T \approx 0.72$ keV, while the single chain method is struggling locating any mode. Although the multiple chain method may not be able to discover new modes if they are not nearby, it can definitely tell the rough proportion of each mode and which mode is more important. This 1-T model serves for illustrative purpose, which explains both the astronomical models and the usage of the MCMC methods.

5.2.2 Two Temperature Components

We present a more realistic model with two temperature components in this subsection. To be specific, a two temperature components (2-T) model is defined as $xsphabs.abs1 *$

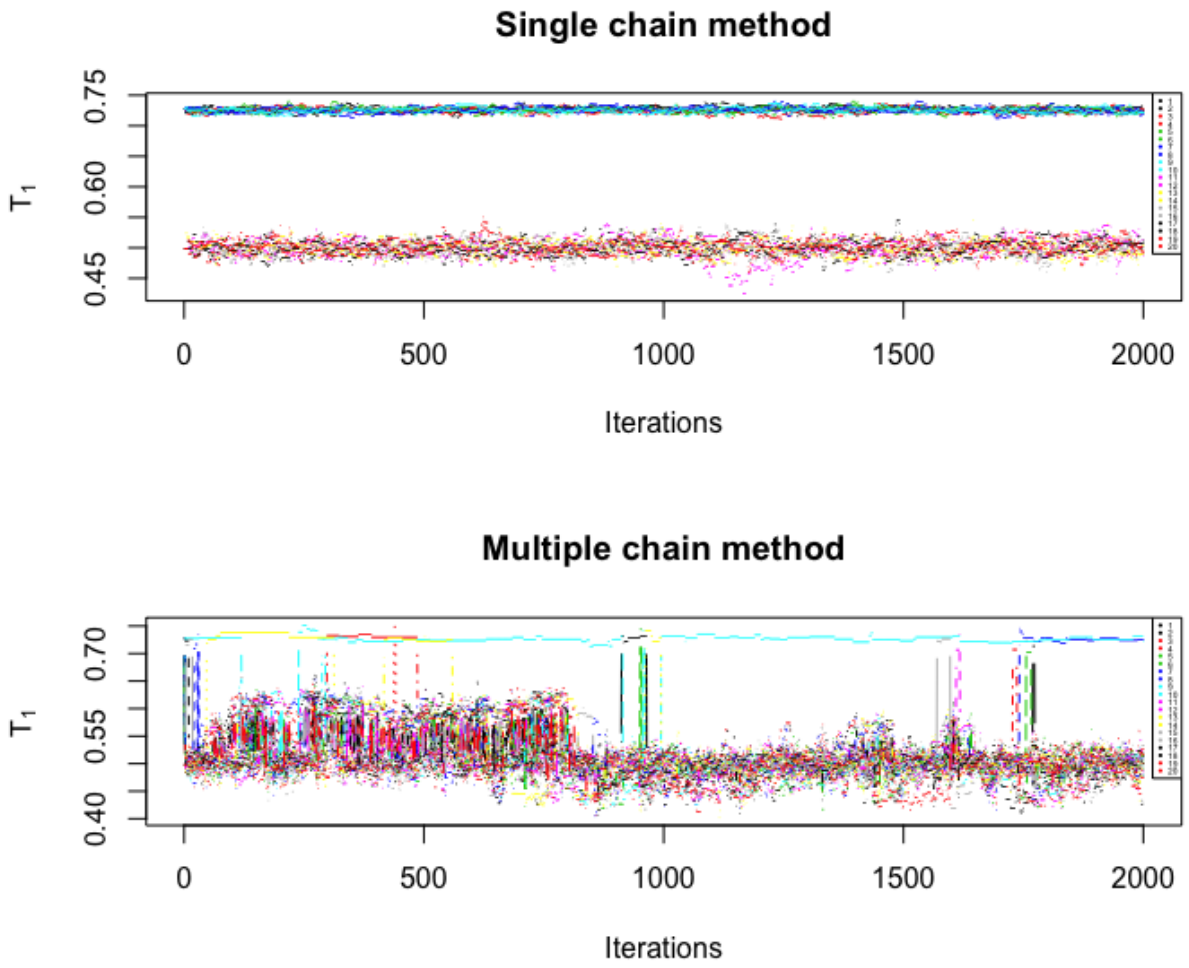

Figure 5.6: Traceplots of T_1 comparing the single chain method (top) and the multiple chain method (bottom) in the 2-T model.

($xsvapec.kT1 + xsvapec.kT2$), *i.e.*, one absorption model $abs1$ and two plasma models $kT1$ and $kT2$. Following the same settings in the 1-T case, the abundance of He is set to 1 and other element abundances are fixed to 0.6. We also link element abundances in $kT2$ to those in $kT1$. By thawing Fe, we have six parameters: N_H, $kT1.T$ as T_1, A_{Fe}, $kT1.N$ as N_1, $kT2.T$ as T_2, and $kT2.N$ as N_2. Parameters of interest are T_1 and T_2 and their associated norms, N_1 and N_2. To begin with, we run 2,000 iterations from two potential modes comparing the single chain method and the multiple chain method. Each starts with 10 chains:

- $T_1 = 0.50$, $T_2 = 0.89$, $N_1 = 0.036$, $N_2 = 0.033$ (major)

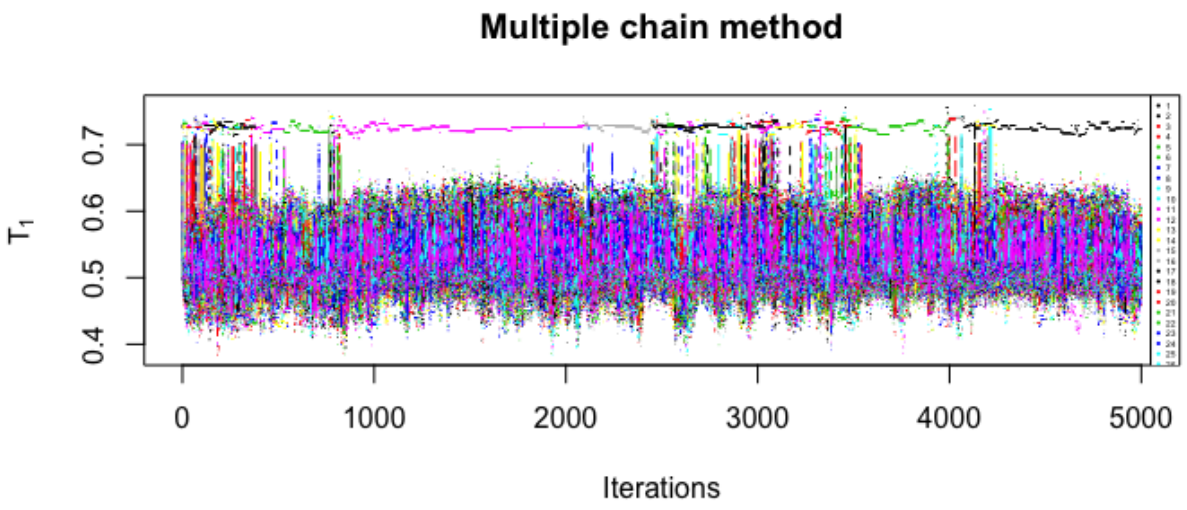

Figure 5.7: Traceplot of T_1 in the multiple chain method in the 2-T model, half starting from the major mode and half from the minor mode.

- $T_1 = 0.72$, $T_2 = 0.08$, $N_1 = 0.060$, $N_2 = 0.061$

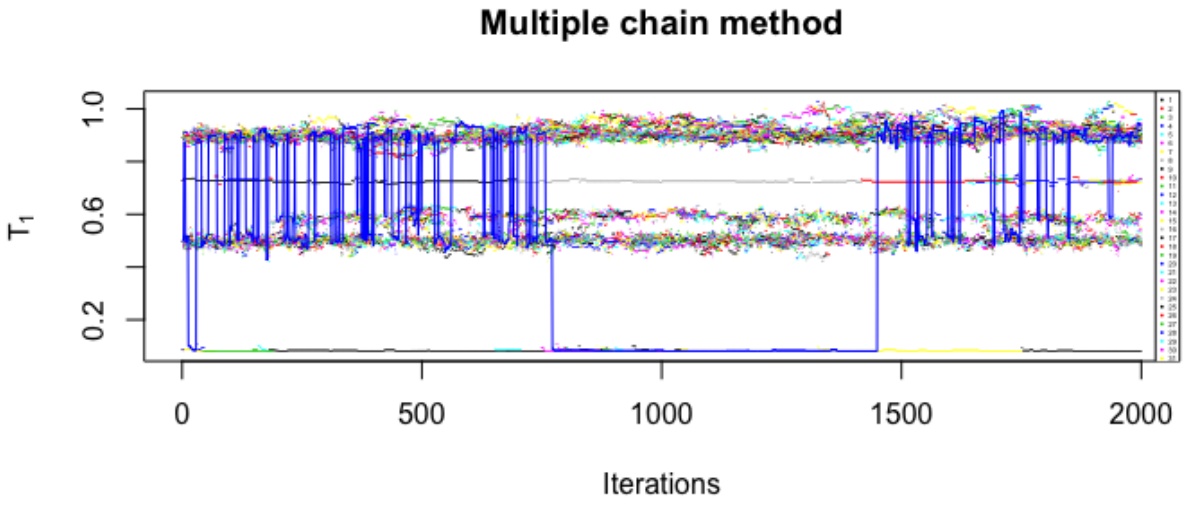

Figure 5.8: Dotplot of T_1 in the multiple chain method in the 2-T model from two potential modes and their mirror images, overlaid by the traceplot of 20-th chain.

These potential modes are identified by Sherpa fit$(\cdot)$ after convergence, where the one that is discovered most frequently is regarded as the major mode. Traceplots of T_1 are shown in Figure 5.6. We can see that the single chain method stays at the mode where the chain

starts, which is not helpful to tell how much proportion one mode contributes to the whole distribution. On the contrary, in the multiple chain method, the chain is able to tell that the minor mode is not important at all compared with the major mode. Similar to the 1-T case, we also demonstrate that our method performs well at deciding which mode is more important. In addition, we find that in our proposed method, the major cluster has not converged yet so that we decide to run it longer. For each mode, we run another 30 chains with 5,000 iterations and draw the traceplot of T_1, shown in Figure 5.7. We can see that: 1) the minor mode is indeed negligible with relative proportion $< 3\%$; 2) the major cluster consists of a band ranging from 0.45 to 0.65. It is suggested that our method finds another nearby mode by itself. To clearly see this, we run another 2,000 iterations from the major mode and the minor mode as well as their mirror images, each with 10 chains:

- $T_1 = 0.50$, $T_2 = 0.89$, $N_1 = 0.036$, $N_2 = 0.033$ (major)
- $T_1 = 0.89$, $T_2 = 0.50$, $N_1 = 0.033$, $N_2 = 0.036$ (major mirror)
- $T_1 = 0.72$, $T_2 = 0.08$, $N_1 = 0.060$, $N_2 = 0.061$ (minor)
- $T_1 = 0.08$, $T_2 = 0.72$, $N_1 = 0.061$, $N_2 = 0.060$ (minor mirror)

Note that label switching is not an issue here, as what we want is to have a better look at how the multiple chain method jumps among modes. Figure 5.8 shows that our method does find a new mode and it is able to jump frequently between the major mode and the new mode. It is also confirmed that the new mode is indeed a mode by Sherpa fit($\cdot$). While Sherpa fit($\cdot$) can occasionally identify the minor mode, the multiple chain method tells us it may not be worth looking into, as it is negligible compared to others. Thus, it is safe to discard it. Actually following this two step process: 1) Sherpa fit($\cdot$) to identify potential modes; 2) the multiple chain method to confirm, we find the following four modes, where the second one is found in the above example:

- $T_1 = 0.50$, $T_2 = 0.89$, $N_1 = 0.036$, $N_2 = 0.033$
- $T_1 = 0.59$, $T_2 = 0.96$, $N_1 = 0.043$, $N_2 = 0.023$
- $T_1 = 0.46$, $T_2 = 0.83$, $N_1 = 0.030$, $N_2 = 0.040$
- $T_1 = 0.61$, $T_2 = 1.02$, $N_1 = 0.046$, $N_2 = 0.018$

We can see that the relative proportion among these four is 60%: 20%: 10%: 10%, by running our proposed multiple chain method. Given these eight thermal components, we can come up with an eight temperature component (8-T) model. By concatenating and freezing [2] these eight temperature values with readjusted norm parameters, the 8-T model can get a better fit to the data in the sense of a higher data likelihood. Yet it is not surprising as an 8-T model has more parameters. To decide which one is the more proper for analysis, among the aforementioned 1-T model, 2-T model, and 8-T model, we turn to pair-wise model comparison, *e.g.*, 2-T model v.s. 8-T model. One possible way to do the comparison is to adopt the Bayesian framework, where we have to carefully specify the proper priors for those parameters and then form the full posterior. In this case, it becomes the same scenario as the Bayesian model selection discussed in 4.1.4, *i.e.*, comparing different mixture models. Although the 8-T model here poses a serious problem in terms of computation (too many parameters for our method to work), the idea is the same that we can apply such model comparison techniques to evaluate model fitting if our goal is to choose the best model.

In summary, our method can be used from two perspectives. First, it can be used as a Bayesian model selection tool to decide which mixture model is more appropriate; secondly and more importantly, it can be served as an exploratory step to identify modes. The strength shown from the second aspect can help determine the importance of those temperature components and decide upon the relative proportion among those modes. Therefore, we can draw the correct inference on the parameters of interest from reasonable model candidates.

[2] Freeze model parameters means they do not vary during a fit.

Chapter 6

Discussion

6.1 Summary

We propose a new population-based MCMC method to sample from multimodal distributions in this thesis, which is designed to alleviate the local trap problem and to improve convergence. Our method initializes multiple chains from dispersed starting values and performs a two-step jump with a novel between-chain jump, which encourages full exploration of the parameter space.

We provide analysis of the related MCMC methods in Chapter 2, and discuss our proposed method in detail in Chapter 3. Our method is designed to help mix between modes rather than find modes. In other words, we require that all modes are predefined and we also assume that they completely separate such that our method can be helpful.

The novelty of our method is that we actually propose a new Metropolis jumping kernel in between-chain jumps that helps the chain to move among modes efficiently. This specific jumping kernel also brings one more feature that, once a mode is visited, it will never be abandoned. After we carefully describe our method, we compare it with other popular meth-

ods in Section 3.4.1, and discuss the pros and cons in terms of both computational cost and parameter tuning. Our proposed method is good at portioning the mass between modes and deciding which mode is more important.

We provide several scenarios in Chapter 4 in which our method can be applied. In Section 4.1 we apply the multiple chain method in computing Bayes factor for Bayesian model selection, where we sample over the joint space created by the model indicators and the parameters for each model. We present a toy example to illustrate in Section 4.1.2, and discuss two real examples of pine data in Section 4.1.3 and galaxy data in Section 4.1.4. In Section 4.2 we apply the multiple chain method to estimate variance components in the mixed effect model. Specifically, we assume a conjugate hierarchical model where the group-level variance is hard to estimate. Compared with the commonly used Gibbs sampler that can easily get stuck, our method is flexible about the prior choice and can explore the parameter space freely. In Section 4.3 we describe the sensor network localization problem and apply the multiple chain method to uncover the locations of unknown sensors. By comparing with three other methods, we find the multiple method helps the chain to better mix among different modes and thus improves the convergence.

In Chapter 5 we focus on a real astronomy application. We apply the multiple chain method on the astronomical data of one star named Capella in order to estimate the parameters of interest, where the model is structured without too many parameters while its target posterior distribution is complicated and costly to evaluate. From discussion in Section 5.2.2, we find our method useful in two aspects. First, it can still be applied in Bayesian model selection to decide which mixture model is more appropriate; secondly, it can be served as an exploratory step to identify modes. The strength shown from the second aspect can help determine the importance of those temperature components and decide upon the relative proportion among those modes.

6.2 Limitations and Future Work

The multiple chain method has some limitations. One issue of the method is that it does not work well in high dimensional cases as discussed in Section 3.4.3. We can also see that from Section 4.3 and Section 5.2.2. The limitation comes from the between chain jumping kernel that we have, which is a mixture of normals centered at other iterates in other chains. This transition kernel suffers from the curse of dimensionality as the higher the dimensional is, the poorer the approximation is. Then it leads to a natural question that why we cannot do one dimension at a time in a Gibbs fashion, and we illustrate in Section 3.4.3 that breaking down the dimensions in a Gibbs fashion does not work, as there is no way for the candidates to jump back so this proposal simply would not even move.

In addition, we want to work more on this between chain jumping kernel. It is based on the independent Hastings proposal and in particular, the choice of $g_0(\cdot)$ in Algorithm 11 is just multivariate normal random walk type kernel for all examples which are discussed in this thesis. For more complicated posterior distributions, we can try other type of distributions to see if it better approximates the target. And even for this specific multivariate normal proposal, there are various ways to modify. For example, it is still not clear how to set the covariance matrix of the multivariate normal properly. Currently we choose the covariance based on the past iterations. It is estimated from the past iterations and that is something connected to adaptive MCMC, which we have discussed in Section 3.1. Another possible way to do this is to set the step size to balance the acceptance rate and how far away the chain can move. If the step size is too small, we see from Section 3.4.2 and Section 4.3 that it is actually not as bad as usual for this type of independent Hastings sampler. If the step size is too large, obviously the chain will not move to anywhere. So probably it is safer to select the fairly small step size. And we further discuss this in Section 3.4.2 and recommend to find a reasonable range of decent acceptance rate and then within that range to select the large step size.

Furthermore, there are many methodology developments that can be done in the future. For example, we discuss the convergence diagnostic in Section 3.4.2, but further work can be done on this type of chains. Usually when we have single chain MCMC, there are two main ways to diagnose convergence. One is to take a look at the autocorrelations to see when they die down. Another is to run multiple chains with diverse staring values and to examine when they mix together, which is done by checking Gelman and Rubin statistic. The issue with applying these techniques to the multiple chain method, in the particular case of Gelman and Rubin statistic, is that it cannot be applied to our method directly. Gelman and Rubin diagnostic is to run multiple chains but here for our method, we already have multiple chains and also jumps between them. Naively applying the diagnostic might mistakenly conclude that the convergence has occurred when in fact it is just normal jumping between-chains that is occurred. Of course one remedy of this problem is by running multiple independent replications of the multiple chain method we have, for instance, run ten independent replications with ten multiple chains for each replication. Within each replication we allow between-chain jumps and between different replications there are no interactions. Then we can apply Gelman and Rubin statistic by computing the summaries for each replication. That would be computational intensive, however, and selection of the static monitored is nontrivial as we have many possible statics.

Besides, the structure of the multiple chains makes it enticing to parallelize the method. However the between-chain jumps make this nontrivial. We are running multiple chains and occasionally they exchange information between chains (between-chain jumps). We may think of how to implement it in a parallel way so that we can take advantage of the computing resources. For example, for each iteration in the within-chain jump, these can be separated in different processors and they do communicate with each other in the between-chain jump. The efficiency of doing this depends on the relative computational time of the within/between-chain jump as well as the communication cost between different processors. Lastly, we can apply the method to more practical situations in the future. Like the astro-

nomical example, suppose there is a highly parametric (the dimension of the parameter space is not extremely high) model with complicated likelihood function that takes much time to evaluate. In such a situation where the posterior surface is expected to be complicated and potentially multimodal, the multiple chain method can be very valuable in reducing the number of iterations required to reach stationarity for MCMC chains. Thus applying this methodology to such type of Bayesian computational problems would be a good direction to go.

Bibliography

Sungjin Ahn, Yutian Chen, and Max Welling. Distributed and adaptive darting Monte Carlo through regenerations. In *Artificial Intelligence and Statistics*, pages 108–116, 2013.

Krishna B Athreya and Gregorio S Atuncar. Kernel estimation for real-valued Markov chains. *Sankhya*, 60(1):1–17, 1998.

Scott Brown and Teresa Head-Gordon. Cool walking: A new Markov chain Monte Carlo sampling method. *Journal of Computational Chemistry*, 24(1):68–76, 2003.

T. Cacoullos and Ch. Charalambides. On minimum variance unbiased estimation for truncated binomial and negative binomial distributions. *Annals of the Institute of Statistical Mathematics*, 27(1):235–244, 1975.

Bradley P Carlin and Siddhartha Chib. Bayesian model choice via Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 473–484, 1995.

Bradley P Carlin and Thomas A Louis. Empirical Bayes: Past, present and future. *Journal of the American Statistical Association*, 95(452):1286–1289, 2000.

Qianshun Cheng, Xu Gao, Ryan Martin, et al. Exact prior-free probabilistic inference on the heritability coefficient in a linear mixed model. *Electronic Journal of Statistics*, 8(2): 3062–3076, 2014.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

Peter Freeman, Stephen Doe, and Aneta Siemiginowska. Sherpa: A mission-independent data analysis application. In *Astronomical Data Analysis*, volume 4477, pages 76–88. International Society for Optics and Photonics, 2001.

Antonella Fruscione, Jonathan C McDowell, Glenn E Allen, Nancy S Brickhouse, Douglas J Burke, John E Davis, Nick Durham, Martin Elvis, Elizabeth C Galle, Daniel E Harris, et al. CIAO: Chandra's data analysis system. In *Observatory Operations: Strategies, Processes, and Systems*, volume 6270, page 62701V. International Society for Optics and Photonics, 2006.

Alan E Gelfand and Adrian FM Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990.

Alan E Gelfand, Sujit K Sahu, and Bradley P Carlin. Efficient parametrisations for normal linear mixed models. *Biometrika*, 82(3):479–488, 1995.

Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, pages 457–472, 1992.

Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 1995.

Andrew Gelman, David A van Dyk, Zaiying Huang, and John W Boscardin. Using redundant parameterizations to fit hierarchical models. *Journal of Computational and Graphical Statistics*, 17(1):95–122, 2008.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In *Readings in Computer Vision*, pages 564–584. Elsevier, 1987.

Charles J Geyer. Markov chain Monte Carlo maximum likelihood. 1991.

Charles J Geyer. Introduction to Markov chain Monte Carlo. *Handbook of Markov Chain Monte Carlo*, 20116022:45, 2011.

Charles J Geyer and Elizabeth A Thompson. Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90 (431):909–920, 1995.

Walter R Gilks and Gareth O Roberts. Strategies for improving MCMC. *Markov Chain Monte Carlo in Practice*, 6:89–114, 1996.

Walter R Gilks, Sylvia Richardson, and David J Spiegelhalter. Introducing Markov chain Monte Carlo. *Markov Chain Monte Carlo in Practice*, 1:19, 1996.

Peter J Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

PJ Green and A O'Hagan. Carlin and Chib do not need to sample from pseudopriors. *Research Rerport 98*, 1, 1998.

Cong Han and Bradley P Carlin. Markov chain Monte Carlo methods for computing Bayes factors: A comparative review. *Journal of the American Statistical Association*, 96(455): 1122–1132, 2001.

Florian Hartig. MCMC chain analysis and convergence diagnostics with coda in R, 2011. URL `https://theoreticalecology.wordpress.com/2011/12/09/mcmc-chain-analysis-and-convergence-diagnostics-with-coda-in-r/`. [Online; accessed 09-June-2018].

Wilfred K Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Alexander T Ihler, John W Fisher, Randolph L Moses, and Alan S Willsky. Nonparametric belief propagation for self-localization of sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):809–819, 2005.

Vinay L Kashyap and Jennifer Posson-Brown. Short timescale coronal variability in Capella. *arXiv preprint arXiv:0709.3093*, 2007.

Robert E Kass and Adrian E Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995.

Robert E Kass, Bradley P Carlin, Andrew Gelman, and Radford M Neal. Markov chain Monte Carlo in practice: A roundtable discussion. *The American Statistician*, 52(2): 93–100, 1998.

Scott Kirkpatrick, Charles D Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Shiwei Lan, Jeffrey Streets, and Babak Shahbaba. Wormhole Hamiltonian Monte Carlo. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

Yaohang Li, Vladimir A Protopopescu, and Andrey Gorin. Accelerated simulated tempering. *Physics Letters A*, 328(4-5):274–283, 2004.

Enzo Marinari and Giorgio Parisi. Simulated tempering: A new Monte Carlo scheme. *EPL (Europhysics Letters)*, 19(6):451, 1992.

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.

NASA/CXC. About Chandra, 2018. URL `http://chandra.harvard.edu/about/`. [Online; accessed 30-August-2018].

Radford M Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6(4):353–366, 1996.

Radford M Neal. Erroneous results in "Marginal Likelihood from the Gibbs output". *Minmeo, University of Toronto*, 1999.

Radford M Neal. Slice sampling. *Annals of Statistics*, pages 705–741, 2003.

Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11), 2011.

GP Patil. Maximum likelihood estimation for generalized power series distributions and its application to a truncated binomial distribution. *Biometrika*, 49(1/2):227–237, 1962.

Christian Robert. Machine Learning, a Probabilistic Perspective, 2014.

Gareth O Roberts and Jeffrey S Rosenthal. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367, 2009.

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

Gareth O Roberts, Jeffrey S Rosenthal, et al. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20–71, 2004.

Kathryn Roeder. Density estimation with confidence sets exemplified by superclusters and voids in the galaxies. *Journal of the American Statistical Association*, 85(411):617–624, 1990.

David Spiegelhalter, Andrew Thomas, Nicky Best, and Wally Gilks. BUGS 0.5* Examples Volume 2 (version ii). *MRC Biostatistics Unit*, 1996.

Matthew Stephens. Bayesian analysis of mixture models with an unknown number of components-an alternative to reversible jump methods. *Annals of Statistics*, pages 40–74, 2000.

Hyungsuk Tak, Xiao-Li Meng, and David A van Dyk. A repelling–attracting Metropolis algorithm for multimodality. *Journal of Computational and Graphical Statistics*, 27(3): 479–490, 2018.

Torsten Bronger. Auriga constellation map, wikimedia commons, 2003. URL `https://commons.wikimedia.org/wiki/File:Auriga_constellation_map.png`. [Online; accessed 28-August-2018].

David A Van Dyk, Alanna Connors, Vinay L Kashyap, and Aneta Siemiginowska. Analysis of energy spectra with low photon counts via Bayesian posterior simulation. *The Astrophysical Journal*, 548(1):224, 2001.

David A van Dyk, Hosung Kang, et al. Highly structured models for spectral analysis in high-energy astrophysics. *Statistical Science*, 19(2):275–293, 2004.

Evan J Williams. *Regression Analysis*. John Wiley & Sons, Incorporated, 1959.