# On a Selection of Advanced Markov Chain Monte Carlo Algorithms for Everyday Use: Weighted Particle Tempering, Practical Reversible Jump, and Extensions

Marcos Carzolio Toledo Arantes

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Statistics

Scotland C. Leman, Chair
Marco Ferreira
Leanna House
Inyoung Kim

June 21, 2016
Blacksburg, Virginia

Keywords: Markov chain Monte Carlo, reversible jump, weighted particle tempering

# On a Selection of Advanced Markov Chain Monte Carlo Algorithms for Everyday Use: Weighted Particle Tempering, Practical Reversible Jump, and Extensions

Marcos Carzolio Toledo Arantes

We are entering an exciting era, rich in the availability of data via sources such as the Internet, satellites, particle colliders, telecommunication networks, computer simulations, and the like. The confluence of increasing computational resources, volumes of data, and variety of statistical procedures has brought us to a modern enlightenment. Within the next century, these tools will combine to reveal unforeseeable insights into the social and natural sciences. Perhaps the largest headwind we now face is our collectively slow-moving imagination. Like a car on an open road, learning is limited by its own rate. Historically, slow information dissemination and the unavailability of experimental resources limited our learning. To that point, any methodological contribution that helps in the conversion of data into knowledge will accelerate us along this open road. Furthermore, if that contribution is accessible to others, the speedup in knowledge discovery scales exponentially.

Markov chain Monte Carlo (MCMC) is a broad class of powerful algorithms, typically used for Bayesian inference. Despite their variety and versatility, these algorithms rarely become mainstream workhorses because they can be difficult to implement. The humble goal of this work is to bring to the table a few more highly versatile and robust, yet easily-tuned algorithms. Specifically, we introduce weighted particle tempering, a parallelizable MCMC procedure that is adaptable to large computational resources. We also explore and develop a highly practical implementation of reversible jump, the most generalized form of Metropolis-Hastings. Finally, we combine these two algorithms into reversible jump weighted particle tempering, and apply it on a model and dataset that was partially collected by the author and his collaborators, halfway around the world. It is our hope that by introducing, developing, and exhibiting these algorithms, we can make a reasonable contribution to the ever-growing body of MCMC research.

# Acknowledgments

I owe the completion of this thesis to many people, but first and foremost, I owe it to my parents, Mari and Papi. Without your unconditional love and support, I would not be the man I am today. I love you.

To my advisor and good friend, Scotland Leman, thank you for your infinite patience and guidance. No one I know is as good-hearted, intelligent, and stubborn as you. I am lucky to have had you as a mentor.

To my first supervisor and good friend, Eric Vance, you changed my life by teaching me how to collaborate and by showing me the world. Thank you for the opportunities you have provided me and your other students.

To all of my friends that have seen me falter and succeed over the past five years, starting and stopping and restarting down numerous paths, I have found my way. Thank you for your kindness and company. I am forever indebted.

*Raffiniert ist der Herrgott, aber boshaft ist er nicht.*

Albert Einstein, 1942

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

In recent decades, Markov chain Monte Carlo (MCMC) algorithms have revolutionized the practice of statistics. Whereas the complexity of statistical models was once limited by the predominant methods of maximum likelihood and asymptotics theory for inference and uncertainty quantification, there is now an abundance of options available to the practitioner for fitting and assessing models, and for digesting the knowledge gained from data. Specifically, a Bayesian can now in many cases simulate from the posterior distribution of not just model parameters, but models themselves. Today, MCMC plays a significant role in many a statistician's toolkit.

The current work seeks to improve the practice of such methods, in particular by contributing a new algorithm for sampling from multimodal distributions called weighted particle tempering (WPT). WPT can be seen as an extension of Metropolis-Hastings sampler that uses multiple Markov chains to help generate the samples from the proposal distribution. The novel algorithm shares similarities with importance sampling (Trotter & Tukey 1956), particle filters (Gordon et al. 1993), parallel tempering (Geyer 1991), and parallel hierarchical sampling (Rigat & Mira 2012). This work also seeks to validate a method to automatically tune and run reversible jump MCMC for a wide range of applications. These contributions add to the rich and expanding literature on practical statistical inference algorithms.

Finally, we combine our work with weighted particle tempering and reversible jump to produce a hybrid algorithm, which we apply to a model for a dataset about children in rural Mozambique.

The broader lesson to be learned from the undertaking herein is that, given a particular research question, the responsibilities of a modern-day statistician are twofold: to make modeling decisions that adequately capture important relationships in the data, and to devise an inference approach for the parameters and models under consideration. The latter involves the selection of estimation procedures (*e.g.* method of moments or maximum likelihood), which in turn often requires algorithms such as an optimizer or MCMC to perform. Often, both of the statistician's roles are tackled at the same time so that modeling and inference are more tractable. Take, for example, the choice of a conjugate prior which facilitates inference and simplifies analysis. In more advanced analyses, the applied statistician usually resorts to sophisticated inferential techniques, at times requiring exotic Monte Carlo algorithms. Multimodal, high-dimensional posteriors pose some of the greatest challenges, where a well-designed Monte Carlo algorithm necessarily exploits the idiosyncrasies of the target distribution, which are learned through trial and error. In this way, the work of the applied statistician becomes at once both an art and a science.

## 1.1 Challenges in Modern Markov Chain Monte Carlo

The goal of MCMC is to produce samples from a given target distribution. Usually used in a Bayesian context, this target distribution is a posterior for parameters conditioned on a dataset $\pi(\theta|Y) = \frac{L(Y|\theta)\pi(\theta)}{\int L(Y|\theta)\pi(\theta)d\theta}$, where the normalizing constant, or the integral in the denominator, is difficult to compute analytically. While there are many ways to sample from a posterior distribution, MCMC achieves this by setting up a Markov chain whose limiting distribution is equal to the target.

There are several theorems that simplify the construction of such a Markov chain. The most

central one, however, is the fact that an irreducible, positive recurrent Markov chain has a unique invariant distribution, which is its stationary distribution. "Detailed balance" is a sufficient condition for having an invariant distribution. Therefore, we are compelled to construct Markov chains that satisfies the detailed balance condition. The works of Nicholas Metropolis Metropolis et al. (1953) and later of Keith Hastings Hastings (1970) provide a framework to do just this. Their efforts have spawned what is now known as the Metropolis-Hastings algorithm, a highly generalizable and versatile algorithm that can be boiled down into iterations over $t = 0, \ldots, N$ of three steps:

0. If $t = 0$, initialize a state $\theta^{(0)} \sim g_0(\theta)$.
1. Sample $\theta^* \sim g(\theta|\theta^{(t-1)})$.
2. Set $\theta^{(t)} = \begin{cases} \theta^* & \text{with probability} \quad \alpha \\ \theta^{(t-1)} & \text{with probability} \quad 1-\alpha \end{cases}$, for some $\alpha$.

In most cases, MCMC methods end up being some variation of this algorithm, with special definitions of the parameter space for $\theta$ as in the case of reversible jump MCMC (Green 1995), special cases of the proposal distribution $g(\theta|\theta^{(t-1)})$, as in, for example, Multi-try Metropolis (Liu et al. 2000), or special acceptance probabilities $\alpha$ that preserve time reversibility.

Markov chain Monte Carlo algorithms face a swath of implementational issues that are, in practice, overcome through various rules of thumb. The so-called practices of "thinning" and "burning in" tackle the problems of autocorrelation in Monte Carlo samples and finding a reasonable starting distribution $g_0(\theta)$, respectively. However, these methods are not easily automated, and they provide no guarantees of optimality. In fact, thinning has been shown to be suboptimal for estimating an integral with reversible Markov chains in Geyer (1992) and with nonreversible Markov chains in MacEachern & Berliner (1994). One should use all of the data generated from Monte Carlo simulations, where computer storage allows.

While in theory a Markov chain satisfying detailed balance will eventually produce samples from the target distribution, in practice the time it takes to produce such samples can

be infeasibly long. In fact, there is no general, established method to test whether such convergence has occurred. As a result, a phenomenon takes place which is referred to by Geyer (2011) as "pseudo-convergence." In particular, if the Markov chain dynamics are poorly designed for the target distribution, the MCMC will appear to have converged when in reality it has not. This phenomenon can take place in any scenario where the proposal distribution is ill-suited for exploring the target distribution and its sample space. One of the predominant causes of pseudo-convergence is a *multimodal* target distribution when using a locally updated MCMC algorithm.

Several fixes have been proposed to deal with sampling from multimodal distributions. Among these are a series of highly sophisticated algorithms that efficiently traverse the sample space to find regions of interest. Of note among these are simulated (Marinari & Parisi 1992, Geyer & Thompson 1995) and parallel tempering (PT) (Geyer 1991) or replica exchange Monte Carlo (Swendsen & Wang 1986); evolutionary Monte Carlo (Liang & Wong 2001); the Equi-Energy Sampler (Kou et al. 2006); Multi-Try Metropolis (Liu et al. 2000); and the Multiset Sampler (Leman et al. 2009). These algorithms are further discussed in later chapters, but it suffices to point out that they each introduce an added layer of implementational complexity in exchange for more efficient sampling. The statistical practitioner therefore is forced to decide between spending more time learning about new algorithms and then coding and debugging or allowing a simpler algorithm to run for a longer time. In contrast, our contribution of weighted particle tempering can take advantage of recent advances in computational horsepower – namely parallel computation – to create a robust, easily tunable and implemented MCMC.

Another significant challenge for MCMC techniques is sampling over varying dimensions. Specifically, a Bayesian is able to place a prior distribution over a model space and learn from data to generate a posterior on these models. However, in most applications, different models have different dimensions and interpretations for their parameters. That is to say, if there is a set $\mathcal{M}$ of models under consideration, and $k$ is the model indicator, then for each $M_k \in \mathcal{M}$, the parameters $\theta_k \in \Theta_k \subseteq \mathbb{R}^{n_k}$ reside in a space of dimension $n_k$. The resulting

posterior distribution over both models and parameters $\pi(M_k, \theta_k|Y)$ is defined in the union of these spaces: $\cup_{M_k \in \mathcal{M}}(M_k \times \Theta_k)$. To sample from this space or average over it, several methods have been introduced, including Gelman & Meng (1998) for bridge sampling, Carlin & Chib (1995) for Gibbs sampling over model space using pseudo priors, and Chen et al. (2012) for Bayes factor approximation. For more, see the comprehensive review of Han & Carlin (2001).

It is not at all clear how to choose a proposal distribution $g(\theta|\theta^{(t-1)})$ that will place a new proposal $\theta^*$ in a region of interest when changing from one model to another, a so-called cross-dimensional jump. This problem puts the burden of MCMC engineering on the practitioners of Green's reversible jump MCMC. Often, practitioners will avoid this issue altogether, resulting in their adoption of other non-probabilistic methods of model selection and averaging, such as BIC or frequentist methods. Our contribution in this field is to perform all of the engineering for the applied statistician beforehand, by defining a process-driven proposal distribution that should work in many practical settings.

We conclude the work with an exposition of a hybrid between weighted particle tempering and our practical implementation of reversible jump, called reversible jump weighted particle tempering. While the algorithm is used for a specific application to a model of child development, it is generalizable and particularly useful for situations in which there is a model selection or averaging component and conditional posteriors are multimodal.

MCMC has a wide range of uses and applications. Our work seeks to expand the reach of this highly significant class of methods. In doing so, it is our hope that inference for sophisticated statistical problems will be alleviated. The efforts detailed in the chapters to follow far from solve one of the aforementioned responsibilities of the statistician: devising an inference approach for the parameters and models under consideration. However, our humble efforts do produce gains in the ease of implementation in cases for which MCMC was once prohibitively complex.

## 1.2 Thesis Structure

The remainder of the thesis is structured in the following way. Chapter 2 gives the relevant historical background and necessary technical details of Markov chain Monte Carlo that we will employ. Chapter 3 introduces our main contribution, weighted particle tempering, developing the theory and establishing its performance in comparison to related algorithms. Chapter 4 demonstrates weighted particle tempering as a versatile algorithm by applying it to two different models and datasets: binary classification trees for breast cancer prediction and decomposable Gaussian graphical models for stock portfolio optimization. Chapter 5 shifts gears and presents an implementation of reversible jump MCMC that reduces the burden of tuning on the user by automating the selection of proposal distribution and eliminating the transformation function. Chapter 6 combines concepts from the previous chapters to create a hybrid algorithm called reversible jump weighted particle tempering, which is then used for inference on a cubic P-spline model to analyze a dataset about the impacts of handpumps for clean water access on child development in rural Mozambique. Finally, chapter 7 ends with some concluding remarks.

## 1.3 Bibliography

Carlin, B. P. & Chib, S. (1995), 'Bayesian Model Choice via Markov Chain Monte Carlo Methods', *Journal of the Royal Statistical Society: Series B (Methodological)* **57**(3), 473–484.

Chen, M.-H., Shao, Q.-M. & Ibrahim, J. G. (2012), *Monte Carlo Methods in Bayesian Computation*, Springer Science & Business Media.

Gelman, A. & Meng, X.-L. (1998), 'Simulating Normalizing Constants: from Importance Sampling to Bridge Sampling to Path Sampling', *Statistical Science* **13**(2), 163–185.

Geyer, C. (2011), Introduction to Markov Chain Monte Carlo, *in* S. Brooks, A. Gelman,

G. L. Jones & X.-L. Meng, eds, 'Handbook of Markov Chain Monte Carlo', Taylor & Francis, pp. 3–48.

Geyer, C. J. (1991), 'Markov Chain Monte Carlo Maximum Likelihood', *Computing Science and Statistics, Proceedings of the 23rd Symposium on the Interface* pp. 156–163.

Geyer, C. J. (1992), 'Practical Markov Chain Monte Carlo', *Statistical Science* **7**(4), 473–483.

Geyer, C. J. & Thompson, E. A. (1995), 'Annealing Markov Chain Monte Carlo with Applications to Ancestral Inference', *Journal of the American Statistical Association* **90**(431), 909–920.

Gordon, N., Salmond, D. J. & Smith, A. F. M. (1993), 'Novel Approach to Nonlinear Non-Gaussian Bayesian State Estimation', *IEE Proceedings on Radar and Signal Processing* **140**(2), 107113.

Green, P. J. (1995), 'Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination', *Biometrika* **82**(4), 711–732.

Han, C. & Carlin, B. P. (2001), 'Markov Chain Monte Carlo Methods for Computing Bayes Factors', *Journal of the American Statistical Association* **96**(455), 11221132.

Hastings, W. K. (1970), 'Monte Carlo Sampling Methods Using Markov Chains and Their Applications', *Biometrika* **57**(1), 97–109.

Kou, S. C., Zhou, Q. & Wong, W. H. (2006), 'Equi-Energy Sampler with Applications in Statistical Inference and Statistical Mechanics', *The Annals of Statistics* **34**(4), 1581–1619.

Leman, S. C., Chen, Y. & Lavine, M. (2009), 'The Multiset Sampler', *Journal of the American Statistical Association* **104**(487), 1029–1041.

Liang, F. & Wong, W. H. (2001), 'Real-Parameter Evolutionary Monte Carlo with Applications to Bayesian Mixture Models', *Journal of the American Statistical Association* **96**(454), 653–666.

Liu, J. S., Liang, F. & Wong, W. H. (2000), 'The Multiple-Try Method and Local Optimization in Metropolis Sampling', *Journal of the American Statistical Association* **95**(449), 121–134.

MacEachern, S. N. & Berliner, L. M. (1994), 'Subsampling the Gibbs Sampler', *The American Statistician* **48**(3), 188–190.

Marinari, E. & Parisi, G. (1992), 'Simulated Tempering: a New Monte Carlo Scheme', *Europhysics Letters* **19**(6), 451–458.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. (1953), 'Equation of State Calculations by Fast Computing Machines', *The Journal of Chemical Physics* **21**(6), 1087–1092.

Rigat, F. & Mira, A. (2012), 'Parallel Hierarchical Sampling: a General-Purpose Interacting Markov Chains Monte Carlo Algorithm', *Computational Statistics & Data Analysis* **56**(6), 1450–1467.

Swendsen, R. H. & Wang, J.-S. (1986), 'Replica Monte Carlo Simulation of Spin-Glasses', *Physical Review Letters* **57**(21), 2607–2609.

Trotter, H. F. & Tukey, J. W. (1956), Conditional Monte Carlo for Normal Samples, *in* 'Symposium on Monte Carlo Methods', Wiley, pp. 64–79.

# Chapter 2

# A Brief History of and Introduction to Markov Chain Monte Carlo

Computers have revolutionized the scientific and statistical communities. Prior to their widespread availability, the majority of data analysis was confined to tractable models for which one could perform wieldy computations with, for example, sufficient statistics. Parameters from these models were inferred directly either through analytic solution, or by appealing to large-sample asymptotics such as the Central Limit Theorem, involving claims such as, "as $n \to \infty$, the sufficient statistic converges in distribution $F_n(T(X_1, \ldots, X_n)) \to F$." While large-sample theory, itself a theoretically remarkable achievement, was crucial for the initial steps in modern empirical science, it was lacking in a few key areas. Data analysis was restricted to the small class of models for which the asymptotics could be developed. Datasets were restricted in size to what could be palpable to human computation. Even an analysis using asymptotics on finite sample sizes is lacking without the proper characterization of convergence rates.

Today, computers are ubiquitous in the practice of statistics. Analyzing brain scans, cosmological data, subatomic collision trajectories, and the human genome are a small selection of the myriad ways in which computers can be used to learn about nature. How did comput-

ers become such a pivotal player in science? Our discussion focuses on one of the greatest algorithms of the past century: Metropolis-Hastings. Almost all forms of MCMC are a variation on the same theme of Metropolis-Hastings, where a distribution is used to generate a candidate, which is accepted or rejected according to some reversible, stochastic rule. Since the introduction of the Metropolis algorithm, dozens of specialized MCMC algorithms have been developed, covering a broad class of problems. Despite the diversity of these algorithms, most of them can trace their roots back to the original work of Metropolis and his coauthors at Los Alamos.

It may be counterintuitive that randomness can help rather than hurt an analysis. Computer algorithms can be used to simulate from a broad range of probability distributions. This is extremely useful for Bayesian statistics, which is exemplified by the characterization of the posterior distribution. However, stochastic simulation can also be used to approximate integrals and perform optimization. In these applications, it is often beneficial to use specialized probability distributions, which are often intractable. Markov chains provide a powerful tool to extend computer simulations to these otherwise inaccessible distributions.

In this chapter, we develop the theory and history of topics most relevant or essential to our work. The rest of the chapter is divided in the following manner. Section 2.1 explores the usefulness and broad applicability of Monte Carlo integration and stochastic optimization. In order to perform Monte Carlo simulations, one must be able to sample from certain probability distribuitions. Markov chain theory provides the basis for Metropolis-Hastings and its relatives, which we explore in section 2.2. More relevant to our work are the suite of tempering and particle methods that opened more sophisticated models up to analysis. These methods are covered in sections 2.3 and 2.4. Finally, we close with an introduction to the most general form of Metropolis-Hastings, Peter Green's reversible jump algorithm, and we discuss the future directions of MCMC in section 2.5.

## 2.1 Monte Carlo for Integration and Stochastic Optimization

The term *Monte Carlo* is used to describe computer algorithms with a deterministic running time, but with a random output. This class of algorithms is named after the famous casino of the same name in Monaco, with the term originating in the 1940s when gambling was illegal. This class of algorithms can be traced even farther back to Ulam and Von Neumann in the early 1940s, with an application to combinatorial optimization: calculation of the probability of winning a game of solitaire. The initial publication on the Monte Carlo method appeared in Metropolis & Ulam (1949), wherein it is revealed the name was suggested by Nicholas Metropolis.

To illustrate Monte Carlo's usefulness, consider the simple problem of computing $\pi$ to some degree of precision. Suppose we only know that the area of a circle is $A = \pi r^2$, and that the area of the unit square is 1. Suppose further that we only have the ability to simulate random numbers from an approximately uniform distribution $Unif(0, 1)$, perhaps by flipping a coin an arbitrary but finite number of times to get the binary expansion of a decimal, $0.010111\ldots$ If we could compute the integral over the region inside of a circle with known radius to give the area, we could solve for $\pi = A/r^2$.

First we inscribe a circle ($r = 0.5$) within the unit square, so that the area of the circle $A$ is the proportion of area inside the unit square that is also inside the unit circle. We can estimate its value by generating random variates $(x_i, y_i) \sim Unif\{(0, 1) \times (0, 1)\}$ and computing the proportion of these points that land inside the circle, or equivalently, compute the proportion of samples for which $\sqrt{(x_i - 0.5)^2 + (y_i - 0.5)^2} \leq 0.5$. Figure 2.1 demonstrates our experimental setup, wherein we take 1,000 samples from the aforementioned uniform distribution and compute the proportion of these samples landing within the circle. The estimated area in this particular simulation is 0.8, yielding an estimate of $\hat{\pi} = 3.2$.

While this is a simple example of numerical integration through Monte Carlo, it in principle

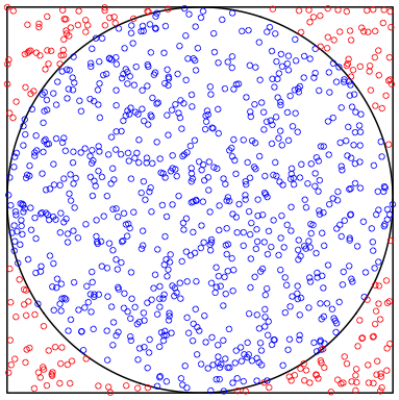


Figure 2.1: 1,000 uniform random numbers landed inside (blue) or outside (red) the circle that inscribes the unit square.

illustrates the core concept of all Monte Carlo integration. Measure theory tells us that to properly define an integral, we need a measure $\mu$. In many cases, this measure can be transformed via normalization into a probability measure $\pi$, which in turn links the integral to an expectation in a probabilistic sense. Suppose the integral we want is $\int g(x)d\mu(x)$, and let $C := \int d\mu(x) < \infty$ so that $\pi(x) := \mu(x)/C$. If we can write $f(x) = g(x)/C$, then $\int g(x)d\mu(x) = \int f(x)d\pi(x) = E(Y)$ where $Y = f(X)$ and $X \sim \pi(x)$. While this equivalence may appear trivial at first glance, the interesting result is that many deterministic integrals can be expressed as the expectation of some appropriately defined random variate. The integral can therefore be approximated via the Central Limit Theorem and Law of Large Numbers by taking a large sample from the expectation's probability distribution and computing the sample mean.

One may wonder why introducing randomness into an optimization algorithm would be desirable. The justification is that, if the optimum resides in a small region with positive measure defined by some tolerance, then running the algorithm for a long stretch of time will find the region with probability 1. Additionally, without any *a priori* knowledge of where the optimum should be, and in situations where the objective function's derivative is

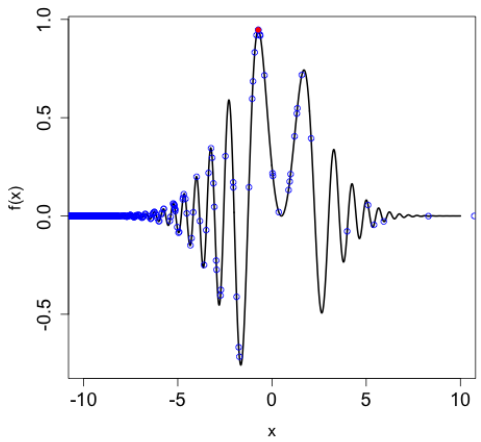



Figure 2.2: Objective function for our stochastic optimization example. The blue points are the candidates drawn from a Cauchy distribution centered at $x = -10$. The red point is the candidate with the largest objective function value.

misleading or unavailable, random perturbations can help the algorithm better explore the search space.

To showcase the value of a randomized search, consider the maximization problem

$$\max_{x \in \mathbb{R}} \sin[(x - 0.5)^2] \exp\left(-\frac{1}{10}x^2\right).$$

One fairly naive way to approximate the maximizer is to evaluate the objective function at a set of candidate locations and choose the location with the highest value. For the general optimization problem, it is not clear how to choose these candidate points. We could select a dense grid of points at which to evaluate the function, but this reduces the optimization problem to selecting how dense this grid should be and over what interval. Also, an exhaustive grid search does not scale well with dimension, as the volume of the space that is searched over grows exponentially. If tuning an optimizer is difficult, a researcher might instead choose to sacrifice computer time and let a random search seek the optimum.

Suppose instead of tuning a grid of candidate points, we sample from a Cauchy distribution centered at $x = -10$ for our candidates. This might be reasonable because all we may know

about the objective function is that the maximum is near $-10$. Figure 2.2 demonstrates how for 1,000 candidate points, we do a decent job of getting close to the optimum. In fact, we could get arbitrarily close to the optimum with a large enough sample of points.

The ideal probability distribution to use for a stochastic optimization would have something to do with the objective function itself. Markov chain Monte Carlo and related methods play a large role in this field, since a probability distribution could be constructed out of the objective function, and draws from this distribution would appear in proportion to the value of the objective. Of particular usefulness is the simulated annealing algorithm developed independently by Kirkpatrick et al. (1983) and Černỳ (1985). In both of these works, the authors applied simulated annealing on the traveling salesman problem, a notoriously difficult optimization, to great success. Simulated annealing is a highly versatile tempering method (section 2.3) that uses the Metropolis-Hastings decision rule (section 2.2) to explore the search space. Other important stochastic optimization algorithms include stochastic gradient descent (see Bertsekas (1999) for example), genetic algorithms (Goldberg 1989), and more recently probability collectives (Wolpert et al. 2013), to name a few.

## 2.2 Markov Chain Monte Carlo

A *Markov chain* is a time-indexed stochastic process where the distribution of the process only depends on the immediately preceding position. More formally, the sequence of random variables $X_1, X_2, \ldots,$ form a Markov chain if $P(X_t|X_1, \ldots, X_{t-1}) = P(X_t|X_{t-1})$. Markov chains are useful for Monte Carlo simulation because, under certain conditions, a Markov chain will have a limiting distribution which may be the distribution of interest for integration, optimization, or simulation, as discussed earlier. Throughout our discussion, we will refer to concepts from chess which will help illustrate certain Markov chain ideas.

Consider a Markov chain on a discrete state space $\Omega$. The state $\omega' \in \Omega$ is said to be *accessible* from $\omega$ if $P(X_t = \omega'|X_s = \omega) > 0$ for time indices $s < t$. If $\omega'$ is accessible from $\omega$ and $\omega$

is accessible from $\omega'$, then the two states are said to be in the same *communicating class*. When the entire state space $\Omega$ is a single communicating class, the Markov chain is said to be *irreducible*, which essentially means that all states are reachable from all other states in finite time. A queen can visit any square on a chess board in a finite number of steps (irreducibility), while a bishop can only visit squares of the same color as its current square. The state space for the white-square bishop, for example, can be reduced to only the white squares.

A state $\omega$ is *periodic* if a return to that state must occur in a multiple of more than one time step, that is if $\min\{t - s : s < t, P(X_t = \omega | X_s = \omega) > 0\} > 1$. If all states in the state space are not periodic, then the Markov chain is said to be *aperiodic*. A knight on a chess board is the only piece that can only revisit the same color square after exactly two moves, making its square's color a periodic state.

Define the random recurrence time $T_\omega := \inf\{t - s : X_t = \omega | X_s = \omega\}$, or the amount of time after $s$ until the Markov chain first revisits state $\omega$. If the expected recurrence time $E(T_\omega) < \infty$, then $\omega$ is said to be *positive recurrent*. If all states in $\Omega$ are positive recurrent, then the Markov chain itself is also positive recurrent. Ignoring queening, a pawn can only move forward, and so its position on the chess board is not recurrent, and the expected recurrence time is therefore infinite.

If the state space is finite, we can write Markovian transitions in terms of matrix operations. Let $P$ be the *probability transition matrix*, the matrix whose $(i, j)$th entry is the probability of transitioning from the $j$th state to the $i$th state in one time step, $P(X_{t+1} = \omega_i | X_t = \omega_j)$. It is easy to verify that if the starting distribution of a Markov chain is the vector $\boldsymbol{\pi}_0$, then the distribution of the chain at time $t$ is just $\boldsymbol{\pi}_t = P^t \boldsymbol{\pi}_0$. The *limiting distribution* is the distribution $\boldsymbol{\pi}^*$ such that as $t \to \infty$, $\boldsymbol{\pi}_t \to \boldsymbol{\pi}^*$, which is invariant to the starting distribution. A *stationary distribution* is a distribution $\boldsymbol{\pi}$ such that $\boldsymbol{\pi} = P\boldsymbol{\pi}$, and so $\boldsymbol{\pi}$ is the eigenvector of $P$ associated with the eigenvalue $\lambda_1 = 1$. If the limiting distribution exists, it is the unique stationary distribution of the Markov chain.

We are interested in constructing a Markov chain such that its limiting distribution is the target distribution. If we can do this, then we can build an algorithm that mimics this Markov chain, let it run for a while, and eventually use its states as samples from the target distribution. First, we must ensure that the stationary distribution exists for the chain. It turns out that if and only if a Markov chain is irreducible and positive recurrent, then it has a stationary distribution. Further, the Basic Limit Theorem (see Chang (2007) for example) states that an irreducible, aperiodic, positive recurrent Markov chain has a limiting distribution.

The *detailed balance* condition, is the following:

$$\pi(X_t = \omega)P(X_{t+1} = \omega'|X_t = \omega) = \pi(X_t = \omega')P(X_{t+1} = \omega|X_t = \omega'), \tag{2.1}$$

for all $\omega, \omega' \in \Omega$. This property essentially states that the Markov chain is just as likely to transition from one state to another as it is to do the reverse, relative to the target distribution. If state $\omega$ is – say – twice as likely to appear as state $\omega'$ under the distribution $\boldsymbol{\pi}$, then a Markov chain satisfying detailed balance is twice as likely to transition from $\omega'$ to $\omega$ as it is to transition from $\omega$ to $\omega'$. The condition is sufficient for the existence of a stationary distribution. In fact, as we shall see, this stationary distribution is $\boldsymbol{\pi}$.

Suppose the probability transition matrix $P$ satisfies detailed balance for some distribution $\boldsymbol{\pi}$. Then for all states $\omega \in \Omega$, the probability of transitioning to $\omega$ from a state randomly selected according to $\boldsymbol{\pi}$ is

$$\begin{aligned}\sum_{\omega' \in \Omega} \pi(X_t = \omega')P(X_{t+1} = \omega|X_t = \omega') &= \sum_{\omega' \in \Omega} \pi(X_t = \omega)P(X_{t+1} = \omega'|X_t = \omega) \\ &= \pi(X_t = \omega),\end{aligned}$$

and therefore $\boldsymbol{\pi} = P\boldsymbol{\pi}$.

One more interesting concept we will use in chapter 3 is the idea of the *spectral gap*. If $1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_{|\Omega|} \geq -1$ is the ordered eigenvalues of a reversible Markov transition matrix $P$, then the spectral gap is $\gamma := 1 - \lambda_2$, or the difference between the first and second

**Algorithm 2.1** Metropolis-Hastings

Initialize $X_0$

**for** $t = 1, \ldots, N$ **do**

$X' \sim g(X'|X_{t-1})$

$\alpha(X_{t-1} \to X') \leftarrow 1 \wedge \frac{\pi(X')g(X_{t-1}|X')}{\pi(X_{t-1})g(X'|X_{t-1})}$

**if** $u \sim Unif(0,1) < \alpha(X_{t-1} \to X')$ **then**

$X_t = X'$

**else**

$X_t = X_{t-1}$

**end if**

**end for**

largest eigenvalues. In a sense, this quantity measures how unique the stationary distribution is. If there were more than one stationary distribution, the spectral gap would be zero, as there would be two eigenvalues equal to one. It can be shown that the spectral gap puts an upper bound on the mixing time, which measures how quickly the Markov chain converges to its stationary distribution. See the excellent resource Levin et al. (2009) for more on this important topic.

Researchers at Los Alamos in the early 1950s needed to study the thermodynamics of a liquid in equilibrium with its gas phase. To do this, they realized they could simply define a Markov chain whose limiting distribution was the same as the distribution of this liquid. The discovery was published in Metropolis et al. (1953), and became known as the Metropolis algorithm. Later in 1970, W. Keith Hastings generalized the algorithm to include arbitrary, rather than symmetric, proposal distributions in Hastings (1970). The Metropolis-Hastings algorithm (MH) is shown in algorithm 2.1.

It is easy to verify that this algorithm preserves detailed balance. From looking at its

formulation, we can see that its probability transition kernel is

$$P(X_{t+1}|X_t) = g(X_{t+1}|X_t)\alpha(X_t \to X_{t+1}) + \mathbb{1}\{X_t = X_{t+1}\} \int_{X' \neq X_t} (1 - \alpha(X_t \to X'))g(X'|X_t)dX' \tag{2.2}$$

where the second term on the right is only present if $X_{t+1} = X_t$, corresponding to the probability of rejecting the proposed sample and staying at the current state. First consider the case where $X_t \neq X_{t+1}$. Without loss of generality, let $X_t$ and $X_{t+1}$ be such that $\alpha(X_t \to X_{t+1}) = 1$. Then in the reverse direction, $\alpha(X_{t+1} \to X_t) = \frac{\pi(X_t)g(X_{t+1}|X_t)}{\pi(X_{t+1})g(X_t|X_{t+1})}$, and we have

$$\pi(X_t)P(X_{t+1}|X_t) = \pi(X_t)g(X_{t+1}|X_t) \tag{2.3}$$

and

$$\begin{aligned} \pi(X_{t+1})P(X_t|X_{t+1}) &= \pi(X_{t+1})g(X_t|X_{t+1})\frac{\pi(X_t)g(X_{t+1}|X_t)}{\pi(X_{t+1})g(X_t|X_{t+1})} & (2.4) \\ &= \pi(X_t)g(X_{t+1}|X_t). & (2.5) \end{aligned}$$

Finally, if $X_t = X_{t+1}$, then of course, $\pi(X_t)P(X_{t+1}|X_t) = \pi(X_{t+1})P(X_t|X_{t+1})$, and so we have proved that the detailed balance condition holds for the Metropolis-Hastings algorithm.

Discussion of the Metropolis-Hastings algorithm would not be complete without a nod to another extremely useful algorithm that is a special case of MH: the Gibbs sampler. Introduced and developed independently from MH for an application in spatial statistics in the paper Geman & Geman (1984), the algorithm allows a multidimensional sample space to be sampled one (or more than one) dimension at a time. Specifically, if $X = (\theta_1, \ldots, \theta_p)$, and if one can draw samples from the full conditionals $\pi(\theta_i|\theta_{j\neq i})$, then all that is required is to sequentially sample from these full conditionals, conditioning on the most recent state of the algorithm. To see that this is a special case of MH, consider that $\pi(X) = \pi(\theta_{j\neq i})\pi(\theta_i|\theta_{j\neq i})$ and $g(X'|X_{t-1}) = \pi(\theta_i'|\theta_{j\neq i,t-1})$, so

$$\begin{aligned} \alpha(X_{t-1} \to X') &= 1 \wedge \frac{\pi(\theta_{j\neq i,t-1})\pi(\theta_i'|\theta_{j\neq i,t-1})\pi(\theta_{i,t-1}|\theta_{j\neq i,t-1})}{\pi(\theta_{j\neq i,t-1})\pi(\theta_{i,t-1}|\theta_{j\neq i,t-1})\pi(\theta_i'|\theta_{j\neq i,t-1})} \\ &= 1. \end{aligned}$$

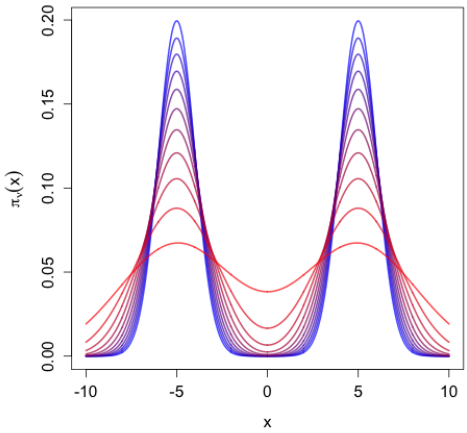



Figure 2.3: Exponentiating a multimodal distribution bridges the gaps between the modes. Here, the exponent ranges from $\nu = 1$ (blue) to $\nu = 0.1$ (red).

## 2.3 Tempering Methods

*Tempering*, and specifically power tempering, in a Monte Carlo context refers to exponentiating the target distribution to a power between zero and one. The term is derived from the metallurgic process of heating a metal and then cooling it to increase its toughness and decrease its hardness. The term is also used in the Monte Carlo literature as a tongue-in-cheek reference to the related stochastic optimization algorithm, simulated annealing. In the field of statistical mechanics and thermodynamics, one of the most common probability distributions is the Boltzmann distribution, $F(x) \propto \exp(-E(x)/kT)$, an exponential of the energy function $E(x)$ divided by the temperature $T$ and Boltzmann constant $k$ (see Chandler (1987) for example). Hence, raising a target distribution to an exponent between zero and one is analogous to increasing the temperature $T$ in the Boltzmann distribution.

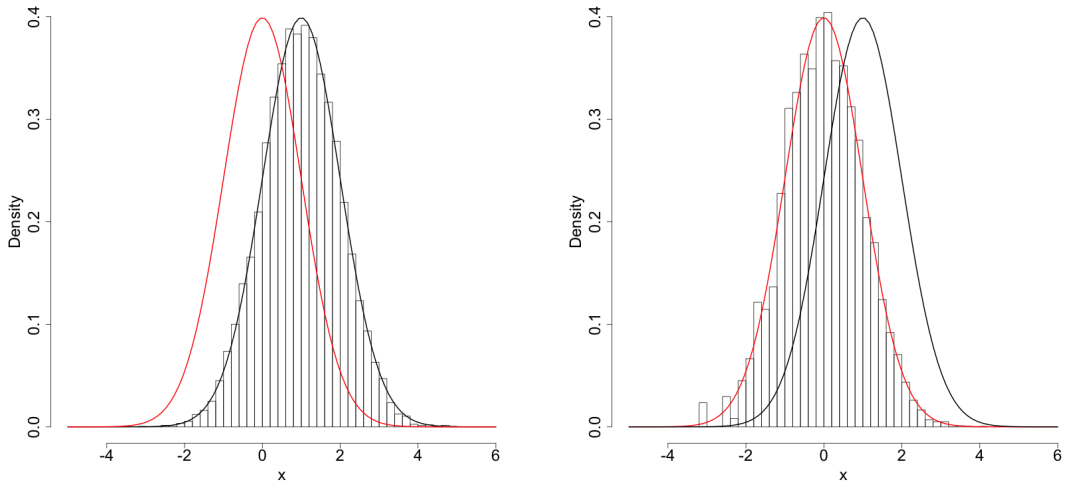
Tempering methods are used in situations where vanilla versions of Metropolis-Hastings do not work efficiently, particularly in applications with multimodal distributions. In these cases, local-update Markov chain samplers such as the random walk Metropolis algorithm can potentially get trapped at local modes, leading to poor mixing and slow convergence.

Tempering can mitigate this problem by bridging gaps between modes and making transitions from one mode to another occur more frequently. Figure 2.3 illustrates how raising a multimodal distribution to an exponent between zero and one in a sense "melts" the distribution, bridging the gaps between modes. A random walk Metropolis would sample more easily from a tempered distribution than from its "cooled" counterpart.

Of course, the goal of MCMC is to sample not from a tempered target distribution, but from the true target, so tempering alone is insufficient to solve the problems caused by multimodal distributions. If sampling from a tempered version of the target distribution is feasible, one option to turn these samples into ones from the true target is to randomly reject them according to the rejection sampling algorithm (von Neumann 1951).

More interesting for our purposes is the suite of algorithms related to what is today called *parallel tempering*. Replica exchange (Swendsen & Wang 1986) and later Metropolis coupled MCMC Geyer (1991) were introduced to analyze Ising models, a class of models in physics where neighboring molecules in a lattice are allowed to interact. Both of these papers independently developed parallel tempering. Simulated tempering, an algorithm closely related to simulated annealing, was introduced shortly after in Marinari & Parisi (1992). The connection between simulated tempering and parallel tempering was developed by Geyer & Thompson (1995). Another algorithm similar to simulated tempering, known as the tempered transition algorithm, was introduced in Neal (1996). More recently, an adaptive form of parallel tempering, which automatically tunes the algorithm, was introduced in Miasojedow et al. (2013). Another recent algorithm called importance tempering is a hybrid of importance sampling Marshall (1956) and simulated tempering, appearing in Gramacy et al. (2010).

These methods are similar in that they use a sequence of tempered distributions that interpolate between the distribution with the smallest exponent and the true, untempered target distribution. Either by swapping or jumping between these distributions, the algorithms are able to produce reversible Markov chains that explore the sample space well, avoiding




Importance Sample Target Approximation

Figure 2.4: 1,000 samples are taken from the importance distribution (pictured left), then weighted and resampled to approximate the target distribution (right).

getting trapped at local modes.

## 2.4 Particle Methods

Particle methods are sequential Monte Carlo algorithms typically used for state space models, where there is a dependence on an unobserved underlying stochastic process. The original particle method, typically referred to as the particle or bootstrap filter, was introduced in Gordon et al. (1993). In essence, the particle filter works by sampling from what we will refer to as a prior distribution (in the sense of a forward filter), then weighting the samples by their likelihood and resampling according to these weights. For the general state space model, this works by sequentially sampling the states from the "current" system equation, or the system equation conditioned on all previous samples, then reweighting them by the observation equation and randomly selecting out of these. This new resampled value – the *particle* – is fed into the next iteration of the algorithm.

The collection of particle methods available rely heavily on the concept of *importance sampling* (Trotter & Tukey 1956). Particles are generated from an importance distribution, and the ratio of target distribution to importance distribution, $\pi/g$, is used to approximate the target and its associated integrals. Suppose we can only generate a sample $X_1, \ldots, X_N \sim g(x)$ but would like to use this sample to approximate an expectation $E_\pi(Y) = \int y\pi(y)dy$, perhaps because $\pi$ is intractable. If we let $w_i \propto \pi(X_i)/g(X_i)$, $i = 1, \ldots, N$ be the set of importance weights, then we can resample with replacement using these weights to approximate the target. By writing $\int y\pi(y)dy = \int y\pi(y)/g(y)g(y)dy$, we see that by the law of large numbers, we can use $1/N \sum_{i=1}^{N} w_i X_i$ to approximate the integral. This can either be computed by taking the weighted average of the sample $X_1, \ldots, X_N$, or by resampling with replacement and taking the average of the resampled observations. Figure 2.4 depicts how a sample from one distribution can be resampled with these importance weights to approximate a target distribution. In essence, particle methods combine this concept of importance weights and resampling with a sequential update to produce samples from a full posterior distribution.

From the particle filter came a very generalizable class of algorithms, most recently including Liu & West (2001), Carvalho et al. (2010), and Andrieu et al. (2010). These methods introduce novel ways to build the proposal distribution for updating the particle in the sequence. The problem that particle methods and their variations have is that each iteration of the algorithm approximates a posterior distribution, and therefore performance decays as these approximations composed of approximations become less accurate. In general, this versatile class of methods share the common step of weighting candidate particles, borrowing from the central concepts of importance sampling. We will be using a similar idea for weighted particle tempering.

## 2.5 Reversible Jump and Beyond

Most MCMC algorithms turn out to be variations on the original Metropolis-Hastings algorithm, usually engineered to have specialized proposal distributions. In Green (1995), Metropolis-Hastings was generalized to a sample space of varying dimension. This algorithm, known as reversible jump Markov chain Monte Carlo, is Metroplis-Hastings in its most general form. The extension to variable-dimensional sampling requires the design of a set of transformation functions and the computation of their Jacobians, a nontrivial task requiring the user of the algorithm to be familiar with cross-model relationships among parameters. Several recent methods have been introduced to automate the selection of transformation functions and proposal distributions for reversible jump. Of note are the works of Ehlers & Brooks (2008) and Carlin & Chib (1995), both of which define conditional probability distributions that can make the leap from one model space to another.

Recently, MCMC has been further extended to applications in which the likelihood is unavailable but simulation from the likelihood is possible. This became necessary because in several applications, evaluating the likelihood may be computationally cumbersome, or because a closed form simply does not exist. Computer simulation experiments are a prime example of this. Approximate bayesian computation (Diggle & Gratton (1984), Rubin (1984)) is an intuitive, recent method for this scenario. The algorithm simulates a parameter draw from the prior distribution, then simulates data from the likelihood conditional on the parameter, and finally accepts or rejects the original parameter sample if the associated simulated data are "close enough" to the observed data.

Much work remains to be done in the field of Monte Carlo for big data applications. At the time of this writing, some of the more advanced methods for big data analysis in a Bayesian setting include variational Bayesian algorithms (Jordan et al. 1999) and cleverly pooling together partial posterior samples that were generated in parallel (Wang & Dunson 2013). It is indeed an exciting time for MCMC research, as modern problems challenge the limitations of this very useful class of methods.

## 2.6 Bibliography

Andrieu, C., Doucet, A. & Holenstein, R. (2010), 'Particle Markov Chain Monte Carlo Methods', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **72**(3), 269–342.

Bertsekas, D. P. (1999), *Nonlinear Programming*, Athena Scientific.

Carlin, B. P. & Chib, S. (1995), 'Bayesian Model Choice via Markov Chain Monte Carlo Methods', *Journal of the Royal Statistical Society: Series B (Methodological)* **57**(3), 473–484.

Carvalho, C. M., Johannes, M. S., Lopes, H. F. & Polson, N. G. (2010), 'Particle Learning and Smoothing', *Statistical Science* **25**(1), 88–106.

Černỳ, V. (1985), 'Thermodynamical Approach to the Traveling Salesman Problem: an Efficient Simulation Algorithm', *Journal of Optimization Theory and Applications* **45**(1), 41–51.

Chandler, D. (1987), *Introduction to Modern Statistical Mechanics*, Oxford University Press.

Chang, J. (2007), Stochastic Processes.
**URL:** *http://www.stat.yale.edu/ pollard/Courses/251.spring2013/Handouts/Chang-notes.pdf*

Diggle, P. J. & Gratton, R. J. (1984), 'Monte Carlo Methods of Inference for Implicit Statistical Models', *Journal of the Royal Statistical Society: Series B (Methodological)* **46**(2), 193–227.

Ehlers, R. S. & Brooks, S. P. (2008), 'Adaptive Proposal Construction for Reversible Jump MCMC', *Scandinavian Journal of Statistics* **35**(4), 677–690.

Geman, S. & Geman, D. (1984), 'Stochastic Relaxation, Gibbs Distributions, and the

Bayesian Restoration of Images', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-6**(6), 721–741.

Geyer, C. J. (1991), 'Markov Chain Monte Carlo Maximum Likelihood', *Computing Science and Statistics, Proceedings of the 23rd Symposium on the Interface* pp. 156–163.

Geyer, C. J. & Thompson, E. A. (1995), 'Annealing Markov Chain Monte Carlo with Applications to Ancestral Inference', *Journal of the American Statistical Association* **90**(431), 909–920.

Goldberg, D. E. (1989), *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley.

Gordon, N., Salmond, D. J. & Smith, A. F. M. (1993), 'Novel Approach to Nonlinear Non-Gaussian Bayesian State Estimation', *IEE Proceedings on Radar and Signal Processing* **140**(2), 107113.

Gramacy, R., Samworth, R. & King, R. (2010), 'Importance Tempering', *Statistics and Computing* **20**(1), 1–7.

Green, P. J. (1995), 'Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination', *Biometrika* **82**(4), 711–732.

Hastings, W. K. (1970), 'Monte Carlo Sampling Methods Using Markov Chains and Their Applications', *Biometrika* **57**(1), 97–109.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S. & Saul, L. K. (1999), 'An Introduction to Variational Methods for Graphical Models', *Machine Learning* **37**(2), 183–233.

Kirkpatrick, S., Gelatt Jr., C. D. & Vecchi, M. P. (1983), 'Optimization by Simulated Annealing', *Science* **220**(4598), 671680.

Levin, D. A., Peres, Y. & Wilmer, E. L. (2009), *Markov Chains and Mixing Times*, American Mathematical Society.

Liu, J. & West, M. (2001), Combined Parameter and State Estimation in Simulation-Based Filtering, *in* A. Doucet, N. d. Freitas & N. Gordon, eds, 'Sequential Monte Carlo Methods in Practice', Springer, pp. 197–223.

Marinari, E. & Parisi, G. (1992), 'Simulated Tempering: a New Monte Carlo Scheme', *Europhysics Letters* **19**(6), 451–458.

Marshall, A. W. (1956), 'The Use of Multi-Stage Sampling Schemes in Monte Carlo Computations', *Symposium on Monte Carlo Methods* p. 123140.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. (1953), 'Equation of State Calculations by Fast Computing Machines', *The Journal of Chemical Physics* **21**(6), 1087–1092.

Metropolis, N. & Ulam, S. (1949), 'The Monte Carlo Method', *Journal of the American Statistical Association* **44**(247), 335–341.

Miasojedow, B., Moulines, E. & Vihola, M. (2013), 'An Adaptive Parallel Tempering Algorithm', *Journal of Computational and Graphical Statistics* **22**(3), 649–664.

Neal, R. M. (1996), 'Sampling from Multimodal Distributions Using Tempered Transitions', *Statistics and Computing* **6**(4), 353–366.

Rubin, D. B. (1984), 'Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician', *The Annals of Statistics* **12**(4), 1151–1172.

Swendsen, R. H. & Wang, J.-S. (1986), 'Replica Monte Carlo Simulation of Spin-Glasses', *Physical Review Letters* **57**(21), 2607–2609.

Trotter, H. F. & Tukey, J. W. (1956), Conditional Monte Carlo for Normal Samples, *in* 'Symposium on Monte Carlo Methods', Wiley, pp. 64–79.

von Neumann, J. (1951), 'Various Techniques Used in Connection with Random Digits', *National Bureau of Standards Applied Mathematics Series* **12**, 3638.

Wang, X. & Dunson, D. B. (2013), 'Parallelizing MCMC via Weierstrass Sampler'.

Wolpert, D. H., Bieniawski, S. R. & Rajnarayan, D. G. (2013), Probability Collectives in Optimization, *in* 'Handbook of Statistics: Machine Learning: Theory and Applications', Vol. 31, Elsevier, p. 6199.

# Chapter 3

# Weighted Particle Tempering: Methodology

**Abstract**

The application of Bayesian methods often requires Metropolis-Hastings or related algorithms to sample from an intractable posterior distribution. In especially challenging cases, such as with strongly correlated parameters or multimodal posteriors, exotic forms of Metropolis-Hastings are preferred for generating samples within a reasonable time. These algorithms require nontrivial and often prohibitive tuning, with little or no performance guarantees. We introduce a new, parallelizable algorithm called weighted particle tempering, which is easily tuned and suitable for a broad range of applications. Weighted particle tempering runs multiple random walk Metropolis chains targeting a tempered version of the target distribution, then weights the iterates and resamples. The algorithm's performance monotonically improves with more of these underlying chains, a feature that simplifies tuning. Using simulation studies, we show that weighted particle tempering outperforms two similar methods: parallel tempering and parallel hierarchical sampling. We also prove that the parallelized form of weighted particle tempering preserves detailed balance in an asymptotic sense.

**Keywords.** Markov chain Monte Carlo, Multi-modality, Parallel algorithm

## 3.1 Introduction

Since the inception of the Metropolis algorithm (Metropolis et al. 1953) and the Gibbs sampler (Geman & Geman 1984), the scientific literature has seen an explosion of Markov chain Monte Carlo (MCMC) techniques for statistical inference. For details, Tierney (1994) is an excellent introduction to MCMC and many of its important features. It suffices to say that the underlying goal of statistical simulation is usually to characterize an unknown or complicated target distribution – usually a posterior. MCMC sets up a Markov chain whose stationary distribution is the distribution in question. Given enough iterations, a properly tuned MCMC will produce dependent random draws from its target distribution. We propose a new MCMC algorithm called weighted particle tempering, whose two major strengths over popular methods are simpler implementation (due to monotonicity in tuning) and scalability via parallelization. Weighted particle tempering performs well in a broad set of applications, and it is therefore an attractive alternative to other algorithms that are often time consuming to tune.

Several issues may arise that hinder the performance of an MCMC algorithm. Multimodal distributions, for example, provide a challenge for locally updated MCMC algorithms, since these algorithms get trapped in local modes. As a result, the researcher must choose between running the MCMC for an impractically long time to reduce Monte Carlo variance, or moving on to a more sophisticated algorithm altogether. Multimodal target distributions appear in many applications, such as in evolutionary or physical systems modeling, time series forecasting, or mixture models. In these scenarios, often the goal of the researcher is to locate the global maximum or identify all local maxima and average over them. In order to accomplish this, a simulation algorithm needs to be able to explore the sample space well, be easily implemented by the applied researcher, and preferably take advantage of the decreasing cost of computational power.

There are many methods that explore multimodal distributions well, starting with replica exchange Monte Carlo (Swendsen & Wang 1986). Metropolis coupled MCMC (Geyer 1991),

more commonly known as parallel tempering, was originally introduced for maximum likelihood inference in Ising models. Simulated tempering (Marinari & Parisi 1992) shortly followed, and its connection to parallel tempering was developed in Geyer & Thompson (1995). A cousin of simulated tempering, the tempered transition algorithm, appeared in Neal (1996). Other important contributions to this field include multi-try Metropolis (Liu et al. 2000), evolutionary Monte Carlo (Liang & Wong 2001), the equi-energy sampler (Kou et al. 2006), and more recently, the multiset sampler (Leman et al. 2009, Kim & MacEachern 2015), and parallel hierarchical sampling (Rigat & Mira 2012), among others. Usually these algorithms make sacrifices with respect to ease of implementation in order to improve performance. While increasing an algorithm's complexity might improve its ability to explore multimodal distributions, doing so typically adds an implementational burden on the researcher. In addition to a boost in performance, a significant advantage of weighted particle tempering is a substantial decrease in the required amount of tuning relative to its competitors.

Parallel tempering was one of the early solutions to the problem of sampling from multimodal distributions. The algorithm allows multiple particles to explore a ladder of tempered target distributions. Tempering flattens the landscape of a distribution so that bridges form between modes. Particles can traverse gaps between modes by visiting a rung on the ladder corresponding to a small tempering exponent (high temperature), then "cooling" back to the target distribution in a process analogous to metallurgic tempering. Mathematically, this is achieved by raising the target distribution to powers of the reciprocals of temperatures in a predefined sequence. The particles swap between adjacent steps on the ladder using Metropolis-Hastings acceptance rules, in such a way that the symmetry of the swap allows the normalizing constants to cancel in the acceptance ratio, eliminating the need to compute them.

One major complication with parallel tempering is the tuning of the temperature ladder, a nontrivial task that requires selecting the number of rungs on the ladder, the step size between rungs, and parameters for the proposal distributions at each of the ladder steps.

Some guiding principles have been established to tune the temperature ladder (Kone & Kofke 2005), while more recent methods have been proposed to adaptively optimize the ladder (Miasojedow et al. 2013). Another issue with parallel tempering is that, despite its name, it is not a computationally parallel algorithm since its chains must communicate with each other at every step. The result is an increase in computation and time, while only samples from the untempered chain are stored.

Simulated tempering was another early, similar algorithm to parallel tempering. Under this framework, only one particle is allowed to explore a temperature ladder. The particle hops from step to step on the temperature ladder using Metropolis-Hastings acceptance rules, a process which requires the often intractable computation of a normalizing constant. Additionally, only samples from the untempered rung of the ladder are retained, requiring a longer running time than parallel tempering.

Some efforts have been made to use the full simulated tempering sample, including samples from the tempered target distribution. Of note in recent literature is a method known as importance tempering (Gramacy et al. 2010), where importance weights are computed for each sample from the target distribution in a simulated tempering run in order to increase the effective sample size used for posterior estimates. This method is related to ours in that weights are computed for each sample, and rather than discarding the samples from tempered chains, they are allowed to (potentially) contribute to the final estimate. While importance tempering provides an overall improvement over naive simulated tempering, its performance hinges on optimal tuning of the temperature ladder, as noted by the authors.

Rather than attempting to improve mixing strictly via the use of temperature-domain methods, the equi-energy sampler allows moves across the energy domain. The sampler depends on the estimation of so-called "energy rings," which allow for improved mixing between the Markov chains of different temperatures. As a result, the equi-energy sampler requires tuning not only of a temperature ladder, but also of the appropriate width of the energy rings. Kou et al. (2006) provide some guiding principles for proper tuning.

Multi-Try Metropolis is another successful technique that introduces an additional layer of tuning. Under the framework of this sampler, the user must specify the number of proposals to be considered at each iteration. However, the proposals are local, resulting in only a slight improvement on the exploration of a multimodal space.

Perhaps the most similar recent method to our own is parallel hierarchical sampling (Rigat & Mira 2012), which is a general way to utilize multiple Markov chains, taking advantage of their different mixing properties and reducing autocorrelation. Parallelization is complicated, however, by the communication among these chains at each step of the algorithm. We discuss how, in contrast, weighted particle tempering can be parallelized to take advantage of the increasingly affordable cost of computation.

Weighted particle tempering provides a simple solution to many of these issues at the increasingly affordable cost of parallel computational power. Tuning the sampler is simplified as there are only two parameters, while introducing more worker nodes will always improve mixing. In this work, we compare our proposed algorithm to parallel tempering and parallel hierarchical sampling because of their similarities and their popularity. Both algorithms require tuning of several chains and their associated parameters, such as proposal standard deviations. Tuning weighted particle tempering is simplified as there are only two parameters, while introducing more underlying chains improves mixing.

## 3.2 Notation

It will be useful to provide some general notation and definitions for the three algorithms we evaluate: parallel tempering, parallel hierarchical sampling, and weighted particle tempering. Each of these algorithms depends on $p$ *underlying particles*, $u_{1,t}, \ldots, u_{p,t}$ that evolve over the iterations $t = 1, \ldots, N$ within their own respective *chains*. The final output of the algorithms is referred to as the *mother chain*, $x_1, \ldots, x_N$. In some cases it will be convenient to refer to the mother chain as the zeroth chain, $u_{0,1}, \ldots, u_{0,N}$. The $(p+1)$-tuple $\{x_t, u_{1,t}, \ldots, u_{p,t}\}$ is a

Markov chain, but the individual particles are themselves notably non-Markovian.

We use $\pi$ to denote the target distribution and $\pi_\nu$ with subscript $\nu \in [0, 1]$ to denote the normalized, *tempered* target. That is $\pi_\nu(x) = \frac{\pi^\nu(x)}{Z(\nu)}$, where

$$Z(\nu) = \int_\Omega \pi^\nu(x)dx < \infty. \tag{3.1}$$

is the normalizing constant and $\Omega$ is the sample space. In certain applications, tempering the target distribution gives a divergent integral in equation 3.1, so the requirement of a finite $Z(\nu)$ is necessary for $\pi_\nu$ to be a proper distribution and essential for parallel tempering and weighted particle tempering to work.

While weighted particle tempering requires a single, tunable tempering exponent $\nu$, weighted particle tempering requires the specification of a *tempering ladder* $1 = \nu_0 > \nu_1 > \ldots > \nu_p \geq 0$. So for parallel tempering, $p$ is the tunable number of underlying particles as well as the length of the tempering ladder, with the zeroth exponent $\nu_0 = 1$ corresponding to the tempering exponent for the mother chain.

Finally, in the algorithms we will present, we use the function $RWM(x, \pi)$ to denote that we perform a single update of particle $x$ with a random walk Metropolis (or related) algorithm targeting $\pi$. The function therefore returns the next step of a random walk Metropolis algorithm given that the current state is $x$ and the target distribution is $\pi$.

## 3.3 Weighted Particle Tempering

Weighted particle tempering is a two-layered algorithm. The first layer contains all the worker nodes, where particles are dispersed over the sample space to explore – using random walk Metropolis – a tempered version of the target distribution. At each iteration of the algorithm, the particles are assigned weights proportional to the target distribution evaluated at their locations. The second layer of the algorithm is the mother node, which picks among the underlying particles according to their pre-computed weights, then accepts or rejects

**Algorithm 3.1** Weighted Particle Tempering

Initialize all particles $\{x_0, u_{1,0}, \ldots, u_{p,0}\}$

**for** $t = 1, \ldots, N$ **do**

Set $w_i \propto \pi^{\delta}(u_{i,t-1})$

$\gamma \leftarrow i$ with probability $w_i$

$\alpha \leftarrow 1 \wedge \frac{\pi^{1-\nu-\delta}(u_{\gamma,t-1})\left[\pi^{\delta}(u_{\gamma,t-1})+\sum_{j\neq\gamma}\pi^{\delta}(u_{j,t-1})\right]}{\pi^{1-\nu-\delta}(x_{t-1})\left[\pi^{\delta}(x_{t-1})+\sum_{j\neq\gamma}\pi^{\delta}(u_{j,t-1})\right]}$

**if** $u \sim Unif(0,1) < \alpha$ **then**

$u_{\gamma,t-1} \leftarrow x_{t-1}$

$x_{t-1} \leftarrow u_{\gamma,t-1}$

**end if**

**for** i = 1,...,p **do**

$u_{i,t} \leftarrow RWM(u_{i,t-1}, \pi_{\nu})$

**end for**

$x_t \leftarrow RWM(x_{t-1}, \pi)$

**end for**

them via the usual Metropolis-Hastings rule.

The algorithm works by initially dispersing its underlying particles across the sample space, which are in turn allowed to explore with respect to the tempered target. Tempering bridges gaps of relatively low density between modes for better exploration while at the same time preserving the locations of modes. As such, particles are able to fluidly explore a flatter version of the target, yet are attracted to high-density regions of the target distribution. In principle, the particles fill the sample space in order to locate modes, while the mother chain has her choice from among these particles, using weights and a Metropolis-Hastings decision rule to properly mix among modes and high-density regions of the space. Weighted particle tempering can be found in algorithm 3.1.

The final samples of interest are stored in the mother chain, $\{x_t\}_{t=1}^{N}$. The fact that the swap move is reversible implies that the algorithm itself preserves detailed balance. It is

worth noting that weighted particle tempering reduces to parallel hierarchical sampling when $\nu = 1$ and $\delta = 0$. Also note that the random walk advance move for the mother chain is akin to an analogous move for the mother chain in parallel tempering (see algorithm 3.2). That is, at each iteration of parallel tempering, all chains are either subjected to a swap or advance move, including the mother chain. In our implementation of parallel tempering, we deterministically alternate between swap and advance. This is to make the algorithms more comparable. In some applications, such a move provides a moderate gain in performance, while in others, the move is not necessary to produce adequate results.

### 3.3.1 Parallelization and Shuffling

While weighted particle tempering is parallelizable, communication between mother and worker nodes at each iteration prohibits the true potential speedup of a fully parallel algorithm. In fact, we suggest running a modified implementation of weighted particle tempering which we describe as follows. The remainder of this chapter and chapter 4 use this parallel implementation.

Rather than swap the randomly selected underlying particle with the mother chain, we will first run all $p$ underlying random walk Metropolis chains independently and in parallel. Doing this will break the Markov structure of the algorithm when proposing a tempered particle to the mother chain. However, for large enough chains, we can randomly permute the samples within these tempered chains so that neighboring samples are approximately independent draws from the tempered target distribution. Finally, the algorithm is modified so that there is no swapping from mother chain to underlying chain, since the underlying samples are only used to generate proposals for the mother. Rather, the proposal is either accepted or rejected.

This implementation of weighted particle tempering requires the storage of the $p$ shuffled Markov chains, limiting the sizes of $p$ and $N$ by the availability of worker nodes and memory, where the requirement for the latter grows at rate $O(pN)$. However, with the increasingly

cheap costs and growing availability of computational power, these restrictions bear almost no impacts on most applications. In section 3.6, detailed balance is shown for applications where consecutive draws from the shuffled underlying chains closely approximate independent draws from the tempered target distribution.

### 3.3.2 Tuning $\delta$ and $\nu$

The main benefit of weighted particle tempering is that for any choice of tuning parameters, increasing the number of worker nodes $p$ will improve performance. This feature shifts the burden of implementation away from tuning the algorithm and towards finding more computational resources.

There is some flexibility in the choice of exponent $\delta$ for the particle weight $w_i$. Using $\delta = 1-\nu$ leads to some nice cancelations in the acceptance ratio $\alpha$, reduces the amount of tuning for the user, and slightly simplifies coding. It is also motivated by principles from importance sampling, since this is the exponent that helps the proposal particle best approximate the target. Using $\delta = 0$ also leads to significant cancelation and is equivalent to uniformly selecting from the $p$ tempered particles. As such, $u_{\gamma,t}$ itself would be a marginal draw from the tempered target. However, setting $\delta = 0$ does not exploit the fact that we have multiple Markov chains exploring the sample space. In this case, as $p$ grows, the marginal distribution of $u_{\gamma,t}$ remains constant, and we see no performance gains with increasing $p$. Conversely, a positive $\delta$ upweights those particles that are in relatively interesting regions of the sample space and takes advantage of the underlying chains. We find that the algorithm works well for many settings of $\delta$, and we default to $\delta = 1$ unless otherwise stated.

The tempering exponent $\nu \in [0, 1]$ should be chosen to facilitate mixing between modes. A small value of $\nu$ will maximize exploration over the sample space, while larger values near one will improve the acceptance rate between the underlying chains and the mother chain. High-dimensional sample spaces exacerbate the discrepancy between the target distribution and the tempered target, so for these problems one should choose larger values of $\nu$. To properly

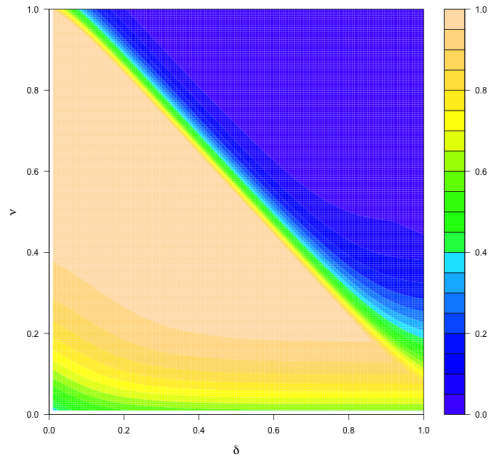



Figure 3.1: Spectral gaps for different tuning levels. A large region of the tunable space is as good, if not better than the end points.

tune this parameter, we recommend running some trial chains to monitor the acceptance rates between mother and worker nodes, choosing the smallest $\nu$ that still produces nominal swap rates that optimize the target distribution's coverage.

To give an example where the optimal $\delta$ and $\nu$ lie in the interior of the space rather than at the endpoints, consider a very peaked target distribution $\pi = (9.9995 \times 10^{-1}, 4.5680 \times 10^{-4}, 1.7310 \times 10^{-8})^\top$. It can be shown that the spectral gap provides bounds on mixing times, where a larger spectral gap results in faster mixing. Figure 3.1 depicts the contours of the spectral gap as a function of $\delta$ and $\nu$ for $p = 2$ underlying particles. In this example, the setting that gives the highest value of the spectral gap is $\delta = 0.025$ and $\nu = 0.985$. Furthermore, $\nu = 1$ and $\delta = 0$ is the setting that reduces to parallel hierarchical sampling when swapping is allowed between mother and worker nodes, and in the following analyses we show our algorithm outperforms parallel hierarchical sampling in several ways for different settings of these parameters.

**Algorithm 3.2** Parallel Tempering

Initialize all particles $\{u_{0,0}, u_{1,0}, \ldots, u_{p,0}\}$

**for** $t = 1, \ldots, N$ **do**

Randomly select a swap index $\gamma = 1, \ldots, p$

$\alpha \leftarrow 1 \wedge \left[\frac{\pi(u_{\gamma-1,t})}{\pi(u_{\gamma,t})}\right]^{\nu_\gamma - \nu_{\gamma-1}}$

**if** $u \sim Unif(0,1) < \alpha$ **then**

$u_{\gamma,t} \leftarrow u_{\gamma-1,t-1}$

$u_{\gamma-1,t} \leftarrow u_{\gamma,t-1}$

**else**

$u_{\gamma,t} \leftarrow u_{\gamma,t-1}$

$u_{\gamma-1,t} \leftarrow u_{\gamma-1,t-1}$

**end if**

**for** $i \notin \{\gamma - 1, \gamma\}$ **do**

$u_{i,t} \leftarrow RWM(u_{i,t-1}, \pi_{\nu_i})$

**end for**

**end for**

## 3.4 Parallel Tempering

Parallel tempering is a $(p+1)$-layered algorithm, where the lower $p$ levels correspond to the underlying tempered chains, and the top layer is the mother chain. At each iteration of the algorithm, particles are either updated with a random walk Metropolis move, or they are swapped with an adjacent particle in the tempering ladder. Parallel tempering is shown in algorithm 3.2. Because of the communication between adjacent chains in the ladder, all $p+1$ particles constitute the Markov chain that must converge to stationarity. If one of the chains is slow to converge, the entire sampler will mix slowly. This is why, as we shall see, the parallelized form of weighted particle tempering will have an advantage over parallel tempering.

**Algorithm 3.3** Parallel Hierarchical Sampling

Initialize all particles $\{u_{0,0}, u_{1,0}, \ldots, u_{p,0}\}$

**for** $t = 1, \ldots, N$ **do**

Randomly select a swap index $\gamma = 1, \ldots, p$

$x_t = u_{\gamma,t-1}$

**for** $i \neq \gamma$ **do**

$u_{i,t} \leftarrow RWM(u_{i,t-1}, \pi)$

**end for**

**end for**

## 3.5 Parallel Hierarchical Sampling

The simplest version of parallel hierarchical sampling is a two-layered algorithm, with the mother chain randomly selecting from among the $p$ underlying chains. More complex implementations build in hierarchies of the same algorithm within itself, but this is outside the scope of our work. At each iteration, one of the underlying particle is randomly selected to swap its position with the mother particle while all other underlying particles are advanced via a random walk Metropolis subroutine, usually with different proposal distributions. Parallel hierarchical sampling is depicted in algorithm 3.3.

## 3.6 Asymptotic Detailed Balance for Weighted Particle Tempering

A key step in the following proof assumes that at each time step, all of the underlying particles are generated from the tempered target distribution independent from their previous states. In particular, the proposal distribution for the underlying chains is $g(\underline{u}|X) = g(\underline{u})$. While this is not true for any finite sample from our proposed algorithm, the condition holds asymptotically, as samples are shuffled over a larger and larger MCMC chain. We first

assume that the condition holds, then discuss the practical implications of finite sample MCMC.

Detailed balance is a sufficient condition for the ergodicity of a Markov chain. From the formulation of our algorithm, the transition kernel is:

$$P(X'|X) = \int \sum_{\gamma=1}^{p} w_\gamma \delta(X' = u_\gamma)\alpha(X, X'|\underline{u}_{-\gamma})g(\underline{u})d\underline{u}, \tag{3.2}$$

where $g(\underline{u}) = \prod_{i=1}^{p} \pi_\nu(u_i)$, $w_\gamma = \pi^\delta(u_\gamma)/\sum_{j=1}^{p} \pi^\delta(u_j)$, and

$$\alpha(X, X'|\underline{u}_{-\gamma}) = 1 \wedge \frac{\pi^{1-\nu-\delta}(X')\left[\pi^\delta(X') + \sum_{j\neq\gamma}\pi^\delta(u_j)\right]}{\pi^{1-\nu-\delta}(X)\left[\pi^\delta(X) + \sum_{j\neq\gamma}\pi^\delta(u_j)\right]}.$$

The summation in the integrand, $\sum_{\gamma=1}^{p} w_\gamma \delta(X' = u_\gamma)\alpha(X, X'|\underline{u}_{-\gamma})$, expresses the fact that $X'$ can only take one of $p$ values, and therefore is a sum over the point masses $\delta(X' = u_\gamma)$ for $\gamma = 1, \ldots, p$. Each of these point masses has probability $w_\gamma$ of being selected.

Switching the order of integration and summation in equation 3.2 gives:

$$\begin{aligned} P(X'|X) &= p\int w_p \delta(X' = u_p)\alpha(X, X'|\underline{u}_{-p})g(\underline{u})d\underline{u} \\ &= p\int \frac{\pi^\delta(u_p)}{\sum_{j=1}^{p}\pi^\delta(u_j)}\delta(X' = u_p)\alpha(X, X'|\underline{u}_{-p})\prod_{i=1}^{p}\pi_\nu(u_i)du_i. \end{aligned}$$

Notice that by symmetry, $\delta(X' = u_p)$ is both a measure on $\mathscr{X}'$ as well as a measure on $\mathscr{U}_p$, the Borel algebra's associated with the next iteration of the algorithm and the current position of the $p$th particle, respectively. We exploit this to evaluate the $p$th dimension of the integral, which is as follows:

$$\begin{aligned} & \int \frac{\pi^\delta(u_p)}{\sum_{j=1}^{p}\pi^\delta(u_j)}\delta(X' = u_p)\alpha(X, X'|\underline{u}_{-p})\pi_\nu(u_p)du_p \\ = & \int \frac{\pi^\delta(u_p)}{\sum_{j=1}^{p}\pi^\delta(u_j)}\alpha(X, X'|\underline{u}_{-p})\pi_\nu(u_p)\delta(u_p = X')du_p \\ = & E_{\delta(u_p=X')}\left(\frac{\pi^\delta(u_p)}{\sum_{j=1}^{p}\pi^\delta(u_j)}\alpha(X, X'|\underline{u}_{-p})\pi_\nu(u_p)\right) \\ = & \frac{\pi^\delta(X')}{\pi^\delta(X') + \sum_{j\neq p}\pi^\delta(u_j)}\alpha(X, X'|\underline{u}_{-p})\pi_\nu(X'). \end{aligned}$$

This leaves us with the following expression for the transition kernel:

$$P(X'|X) = p\frac{\pi^{\nu+\delta}(X')}{Z(\nu)}\int \frac{\alpha(X,X'|\underline{u}_{-p})}{\pi^{\delta}(X')+\sum_{j\neq p}\pi^{\delta}(u_j)}\prod_{i\neq p}\pi_{\nu}(u_i)du_i, \tag{3.3}$$

where $Z(\nu)$ is the normalizing constant of equation 3.1. Notice that the products and sums now range from 1 to $p-1$ as we have extracted the $p$th term from the integrand by taking the expectation with respect to the point mass.

Now let $S = \left\{\underline{u}_{-p} : \alpha(X,X'|\underline{u}_{-p}) = 1\right\}$. This is the set of $(p-1)$-vectors of underlying particles where, in the forward direction $\alpha(X,X'|\underline{u}_{-p}) = 1$, while in the backward direction $\alpha(X',X|\underline{u}_{-p}) < 1$. Its complement, $S^{\mathsf{c}}$, therefore, has the opposite property – specifically that $\alpha(X,X'|\underline{u}_{-p}) < 1$ in the forward direction and $\alpha(X',X|\underline{u}_{-p}) = 1$ in the reverse direction. From equation 3.2 we decompose the forward-direction integral into a sum of two terms:

$$\begin{aligned}
P(X'|X) &= p\frac{\pi^{\nu+\delta}(X')}{Z(\nu)}\left\{\int_S \frac{\prod_{i\neq p}\pi_{\nu}(u_i)du_i}{\pi^{\delta}(X')+\sum_{j\neq p}\pi^{\delta}(u_j)} + \frac{\pi^{1-\nu-\delta}(X')}{\pi^{1-\nu-\delta}(X)}\int_{S^{\mathsf{c}}} \frac{\prod_{i\neq p}\pi_{\nu}(u_i)du_i}{\pi^{\delta}(X)+\sum_{j\neq p}\pi^{\delta}(u_j)}\right\} \\
&= p\frac{\pi^{\nu+\delta}(X')}{Z(\nu)\pi^{1-\nu-\delta}(X)}\left\{\pi^{1-\nu-\delta}(X)\int_S \frac{\prod_{i\neq p}\pi_{\nu}(u_i)du_i}{\pi^{\delta}(X')+\sum_{j\neq p}\pi^{\delta}(u_j)}\right. \\
&+ \left.\pi^{1-\nu-\delta}(X')\int_{S^{\mathsf{c}}} \frac{\prod_{i\neq p}\pi_{\nu}(u_i)du_i}{\pi^{\delta}(X)+\sum_{j\neq p}\pi^{\delta}(u_j)}\right\}.
\end{aligned}$$

In the reverse direction, the set $S$ has the following property:

$$\underline{u}_{-p}\in S \Rightarrow \alpha(X',X|\underline{u}_{-p}) = \frac{\pi^{1-\nu-\delta}(X)\left[\pi^{\delta}(X)+\sum_{j\neq\gamma}\pi^{\delta}(u_j)\right]}{\pi^{1-\nu-\delta}(X')\left[\pi^{\delta}(X')+\sum_{j\neq\gamma}\pi^{\delta}(u_j)\right]}.$$

Therefore, swapping $X$ for $X'$ in equation 3.2 gives us the reverse-direction transition kernel:

$$\begin{aligned}
P(X|X') &= p\frac{\pi^{\nu+\delta}(X)}{Z(\nu)}\left\{\frac{\pi^{1-\nu-\delta}(X)}{\pi^{1-\nu-\delta}(X')}\int_S \frac{\prod_{i\neq p}\pi_{\nu}(u_i)du_i}{\pi^{\delta}(X')+\sum_{j\neq p}\pi^{\delta}(u_j)} + \int_{S^{\mathsf{c}}} \frac{\prod_{i\neq p}\pi_{\nu}(u_i)du_i}{\pi^{\delta}(X)+\sum_{j\neq p}\pi^{\delta}(u_j)}\right\} \\
&= p\frac{\pi^{\nu+\delta}(X)}{Z(\nu)\pi^{1-\nu-\delta}(X')}\left\{\pi^{1-\nu-\delta}(X)\int_S \frac{\prod_{i\neq p}\pi_{\nu}(u_i)du_i}{\pi^{\delta}(X')+\sum_{j\neq p}\pi^{\delta}(u_j)}\right. \\
&+ \left.\pi^{1-\nu-\delta}(X')\int_{S^{\mathsf{c}}} \frac{\prod_{i\neq p}\pi_{\nu}(u_i)du_i}{\pi^{\delta}(X)+\sum_{j\neq p}\pi^{\delta}(u_j)}\right\}.
\end{aligned}$$

Finally, we see that $P(X'|X)/P(X|X') = \pi(X')/\pi(X)$ or equivalently, $\pi(X)P(X'|X) = \pi(X')P(X|X')$. For large, shuffled MCMC samples from the tempered distribution, the proof above applies to the proposed algorithm. This is due to the fact that shuffling breaks the Markov dependence structure of the simulations, while the large sample size presumably ensures the draws are from the tempered distribution. The proposals can therefore be made arbitrarily close to independent draws from the tempered target distribution. We find in practice that in low dimensions, this approximation converges rather quickly.

## 3.7 Simulation Study

In this section, we run a battery of tests for weighted particle tempering and compare its performance to that of parallel tempering and parallel hierarchical sampling. In all tests, the target distribution is some form of a mixture of normals:

$$\pi(x) = \sum_{i=1}^{n} w_i \phi_i(x)$$

The first test (section 3.7.1) demonstrates the empirical rates of convergence to the target distribution in a univariate, bimodal example by measuring the Kolmogorov-Smirnov statistic.

The second test (section 3.7.2) illustrates the gains in weighted particle tempering from placing the right mass across all modes. Specifically, when parallel tempering or parallel hierarchical sampling explores a target distribution with several modes, they are likely to spend long stretches of time at modes due to the dependence across all of their underlying chains, which is induced by swapping. A consequence of this issue is that samples using these algorithms will tend to place too much mass in the modes in which they were stuck the longest. In contrast, the parallel implementation of weighted particle tempering preserves the independence across the underlying chains and avoids this problem at the expense of not sharing information about regions of interest in the sample space between chains during the parallelized exploration steps (see section 3.3.1).

The final test (section 3.7.3) shows that for higher-dimensional problems, weighted particle tempering continues to perform competitively with parallel tempering and parallel hierarchical sampling, provided more computational horsepower. Because weighted particle tempering is significantly simpler to implement than parallel tempering and a generalization of parallel hierarchical sampling, we conclude that weighted particle tempering is an attractive alternative for a wide range of applications.

### 3.7.1 Univariate Mixture

In this section, we empirically show that weighted particle tempering outperforms parallel tempering and parallel hierarchical sampling in generating samples from the target as profiled by the Kolmogorov-Smirnov (KS) test statistic. Our simulations are based on a two-component Gaussian mixture, with component means $\mu_1 = 0$ and $\mu_2 = 10$, component standard deviations $\sigma_1 = 1$ and $\sigma_2 = 2$, and component weights $w_1 = 0.25$ and $w_2 = 0.75$.

We perform these tests by running 100 independent instances of each of the three algorithms. The KS statistic is computed at each iteration across all 100 instances of the algorithms. This statistic gives the maximum distance between the empirical CDF and the true CDF of the target distribution:

$$D_t = \sup_x |G_t(x) - F(x)|,$$

where $G_t(x)$ is the empirical CDF of the sampler computed using the 100 samples at iteration $t$, and $F(x)$ is the target CDF. For this experiment, the target CDF can be written analytically as:

$$F(x) = w_1\Phi_1(x) + w_2\Phi_2(x)$$

where $\Phi_i(x)$ is the normal CDF for the $i$th component of the mixture.

Figure 3.2 gives the LOWESS-fitted curves to the KS statistic for 1,000 iterations of each of the algorithms. LOWESS smoothing is done to reduce the noise in these statistics and demonstrate visually that weighted particle tempering produces a larger sample from the

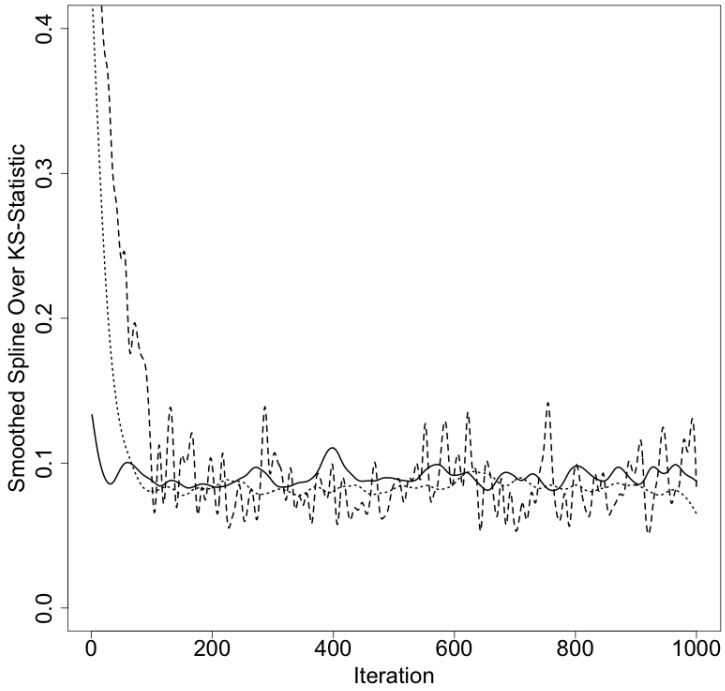



Figure 3.2: Smoothed Kolmogorov-Smirnov (KS) statistics for weighted particle tempering (solid), parallel tempering (dashed), and parallel hierarchical sampling (dotted).

target distribution in less time while parallel tempering and parallel hierarchical sampling require nearly 100 to 200 burnin iterations to converge. We conclude that for a fixed number of iterations, weighted particle tempering produces a larger sample from the target distribution than parallel tempering or parallel hierarchical sampling due to the burnin period.

### 3.7.2 Bivariate Mixture

For the experiment in this section, we use a two-dimensional, five-component mixture of normals as the target distribution. The component weights and centers are in table 3.1. Figure 3.3 depicts samples from single runs of parallel tempering, parallel hierarchical sampling, and weighted particle tempering, showing how the Monte Carlo variance could be higher with parallel tempering and parallel hierarchical sampling due to autocorrelation.

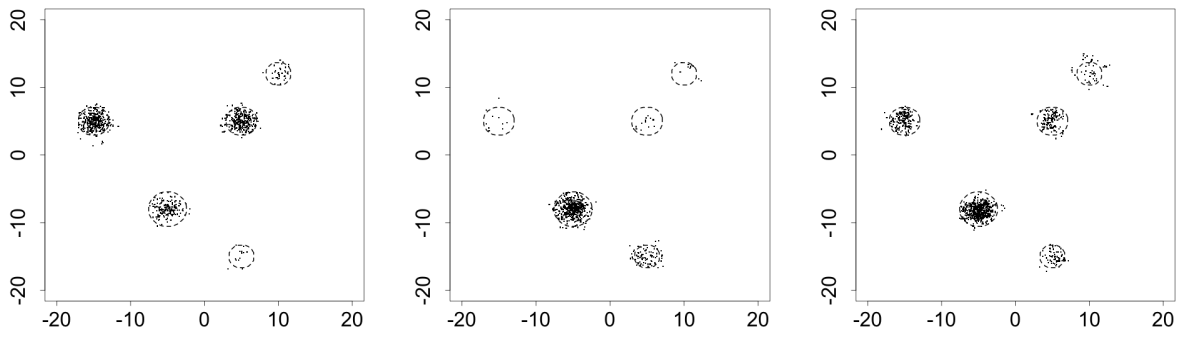



Figure 3.3: Samples (N = 4,000) from a single run of parallel tempering (PT), parallel hierarchical sampling (PHS), and weighted particle tempering (WPT). The dashed contours mark the boundaries of the densest 90% mass region.

Table 3.1: Parameters for the five components of the normal mixture we use as a target distribution in our simulation study. All components have the identity covariance matrix.

| Center | $(-5,-8)$ | $(5,5)$ | $(-15,5)$ | $(10,12)$ | $(5,-15)$ |
|---|---|---|---|---|---|
| Weight | $1/2$ | $1/6$ | $1/6$ | $1/12$ | $1/12$ |

A local-update Metropolis-Hastings algorithm struggles to sample from this distribution because it finds a mode and sticks to it for long stretches of time. While parallel tempering and parallel hierarchical sampling provide remedies for this issue, the dependence structure among their underlying chains cause them to stick to modes for longer than weighted particle tempering, where particles are allowed to explore independently.

We tune parallel tempering over a range of settings, as shown in table 3.2. At each setting of the number of underlying particles $p$, we choose a log-linear sequence of exponents, beginning at $\nu_0 = 1$ and ending at $\nu_p < 1$, as per the recommendation of Kone & Kofke (2005). At each setting of $p$ and $\nu_p$, we run the algorithm 100 times and collect 1,000 samples at each setting of $p$ and $\nu_p$. Then for each of these trials we compute the vector of proportions of samples that land within the five disjoint regions whose union forms the 90% densest probability

Table 3.2: Root mean squared error (RMSE) for parallel tempering (PT) and weighted particle tempering (WPT) at different tuning settings. Light gray marks the minimum RMSE at each exponent, and dark gray is the optimal RMSE.

| | $\nu_p$ (PT) | | | | | | $\nu$ (WPT) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p$ | 0.9 | 0.5 | 0.1 | 0.05 | 0.01 | 0.005 | 0.9 | 0.5 | 0.1 | 0.05 | 0.01 | 0.005 |
| 2 | 0.78 | 0.74 | 0.28 | 0.29 | 0.41 | 0.44 | 0.78 | 0.71 | 0.14 | 0.12 | 0.19 | 0.22 |
| 5 | 0.73 | 0.71 | 0.23 | 0.22 | 0.20 | 0.23 | 0.74 | 0.69 | 0.10 | 0.07 | 0.11 | 0.17 |
| 10 | 0.73 | 0.68 | 0.26 | 0.29 | 0.27 | 0.25 | 0.73 | 0.66 | 0.06 | 0.06 | 0.09 | 0.10 |
| 20 | 0.74 | 0.74 | 0.42 | 0.40 | 0.33 | 0.33 | 0.74 | 0.65 | 0.05 | 0.04 | 0.06 | 0.07 |

region (see figure 3.3), and compare these estimated proportions to their true values, which are roughly 0.48, 0.15, 0.15, 0.06, and 0.06.

Through this process, we find an optimal setting – in the sense of minimum root mean squared error – for parallel tempering of $p = 5$ underlying chains and $\nu_p = 0.01$ as the smallest tempering exponent. Note that this setting is in the interior of the tunable parameter space, so increasing $p$ does not in general lead to improved performance.

To make parallel hierarchical sampling and weighted particle tempering computationally comparable, we also select $p = 5$ for the number of their underlying chains. However, the reader should note that both of these algorithms would perform better with larger $p$. Tuning weighted particle tempering then becomes an issue of selecting the tempering exponent $\nu$, the weighting exponent $\delta$ (we use $\delta = 1$), and the proposal standard deviations for all chains. We try the same values for tempering exponent $\nu$ as we do for $\nu_p$ in parallel tempering and find the optimal setting in terms of minimum root mean squared error to be $\nu = 0.05$.

Tuning parallel hierarchical sampling is comparably simple, but it is not clear how to select the $p$ different proposal standard deviations for the underlying chains. For each of our trials, we randomly generate proposal standard deviations $\sigma_i \sim Exp(1/3)$ for $i = 1, \ldots, p$ because we want to model the uncertainty in how to tune the sampler, and we find that an average

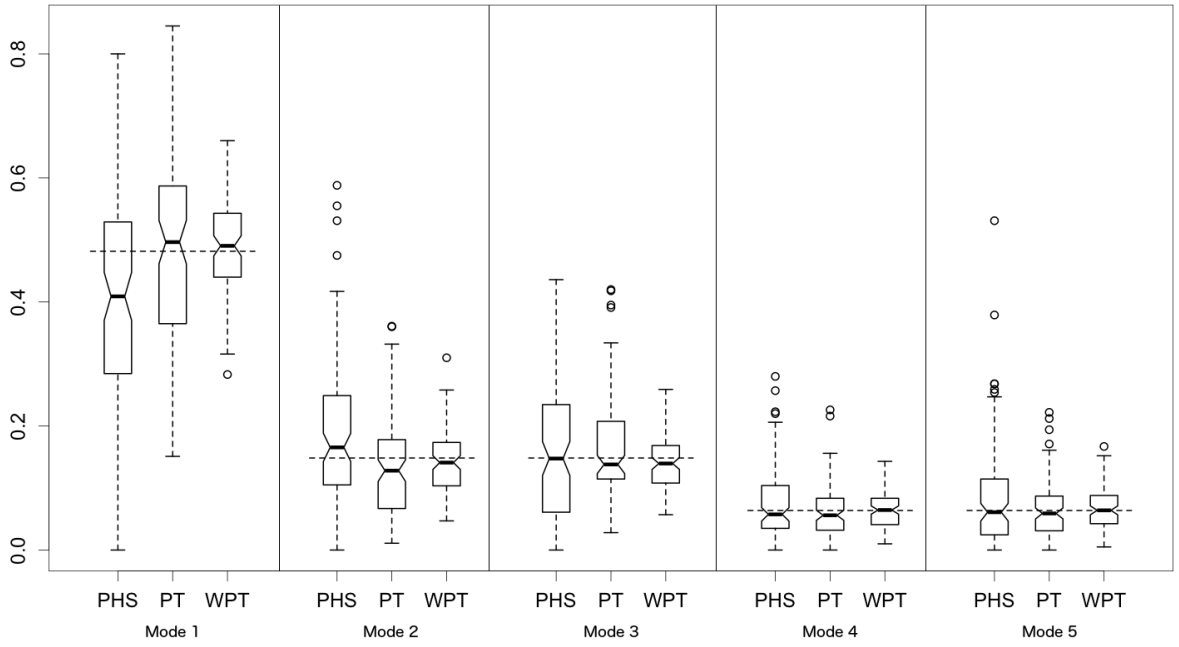



Figure 3.4: Boxplots of the empirical proportions of samples landing in the five modes composing the densest 90% mass regions for 100 runs of both samplers. Dashed horizontal bars indicate the ground truth.

proposal standard deviation of 3 works well.

Once the three algorithms are appropriately tuned, we perform 100 runs, each yielding 1,000 samples. For each run, we calculate the proportion of samples landing in the 90% densest-mass region (denoted by the contours in figure 3.3). Figure 3.4 demonstrates that weighed particle tempering outperforms parallel tempering and parallel hierarchical sampling in the precision of placing the samples within these bounds. However, all three algorithms on average place 90% of their samples inside the union of these disjoint regions, as can bee seen in figure 3.5.

Because parallel tempering and parallel hierarchical sampling share information across all of their underlying chains, the algorithms are able to exploit the locations of the particles as they

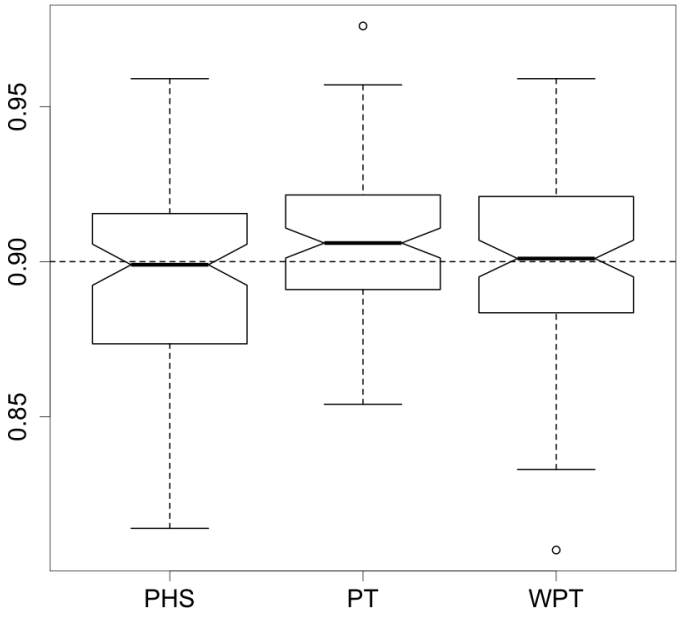



Figure 3.5: Boxplots demonstrating the aggregate placement of samples within the 90% highest-density region for parallel hierarchical sampling (PHS), parallel tempering (PT), and weighted particle tempering (WPT).

explore the sample space. However, this exploitation comes at a cost of inducing dependence among the chains, which means that if they are jointly stuck at a mode or set of modes, they will take a long time to escape, which causes the final samples to be autocorrelated and inflates the Monte Carlo variance. In contrast, the parallel implementation of weighted particle tempering mixes over its several underlying chains *after* they have explored, thereby avoiding this complication.

### 3.7.3 High-Dimensional Mixture

In this section, we discuss weighted particle tempering's limitations when the dimension of the target's sample space is increased. Specifically, our target distribution is a $d$-dimensional,

two-component standard normal mixture, where the components are at opposite corners of the unit hypercube. What we find is that it becomes increasingly difficult to find a suitable weighted particle tempering exponent that allows for *both* good mixing at the underlying particle level *and* good acceptance rates at the mother level. However, initializing the underlying chains independently at random and in an overdispersed manner, and allowing the cold chain to alternate between selecting from tempered chains and making a random-walk Metropolis update provides for improved mixing between the two high-dimensional modes.

To show how the algorithm's deteriorating performance in problem dimension can be mitigated with a larger volume of particles, figure 3.6 demonstrates how the Brooks-Gelman multivariate potential scale reduction factor (MPSRF) (Brooks & Gelman 1998) decreases with an increasing number of underlying chains. The MPSRF compares the output of several parallel instances of an algorithm, and it is used as a stopping rule for convergence. The MPSRF is calculated as follows

$$\hat{R}^p = \max_a \frac{a'\hat{V}a}{a'Wa}$$

where $\hat{V}$ is the estimated covariance matrix across all chains and $W$ is the within-chain covariance matrix. This scalar measure compares the intra- and inter-chain covariances to quantify distance to stationarity. A larger value indicates the MCMC algorithm has not yet converged.

We run weighted particle tempering for different dimensions of the bimodal target distribution and at varying numbers of underlying particles $p$, and for a fixed number of iterations (1,000) we compute the MPSRF. We find that MPSRF increases with problem dimension and decreases with number of underlying particles. Ultimately this means we can remedy the curse of dimensionality with more computational horsepower. Parallel tempering should perform well in these high-dimensional scenarios provided the algorithm is tuned well. However, when it is not clear how to properly tune parallel tempering, weighted particle tempering provides an attractive alternative such that the practitioner knows that he or she can always do better with more worker nodes.

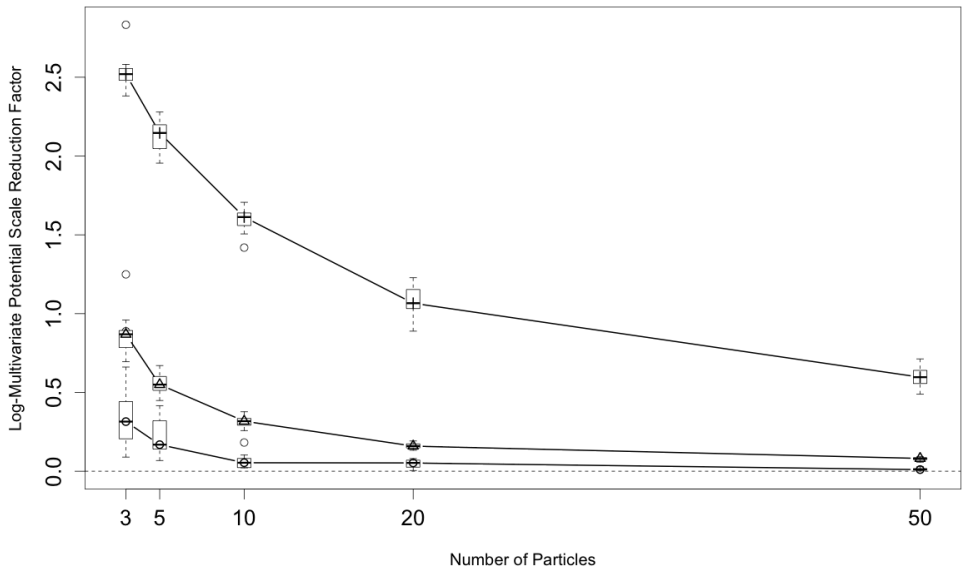



Figure 3.6: MPSRF for different dimensions of the target distribution and for different numbers of underlying particles in weighted particle tempering. Each run is done with the same tempering exponent $\nu = 0.9$, the same number of iterations $N = 1{,}000$, the same proposal distribution standard deviation, and five chains to compute the MPSRF. The circles are the average MPSRF for a 5-dimensional target distribution, the triangles for a 50-dimensional target, and the crosses for a 100-dimensional target.

## 3.8 Discussion

We have proposed a new algorithm that allows for exploration of a complex, intractable distribution with ease given arbitrary computational power. Practitioners applying the algorithm will find it simple to tune and straightforward to implement. We have shown that the algorithm is detail balanced, ensuring convergence to the target distribution, and we have empirically shown it to have some advantages over parallel tempering and parallel hierarchical sampling in the following senses: reduced Monte Carlo variance, simpler tuning, and faster mixing. We see that weighted particle tempering could lose its edge over a well-tuned

parallel tempering as the dimension of the parameter or sample space increases. However, this problem can be mitigated by increasing the number of underlying worker nodes.

Through several numerical experiments, we have shown weighted particle tempering to be a versatile, fast, and precise alternative to parallel tempering and parallel hierarchical sampling. It remains to be theoretically proven, however, under what conditions weighted particle tempering converges to the target distribution faster than its competitors. There is a clear need for advancing the theory of convergence rates for advanced Markov chain Monte Carlo algorithms such as ours, but we have also contributed an asymptotic argument for the detail balanced condition as it relates to the parallelized version of our algorithm. Users of these algorithms will be attracted to weighted particle tempering's simpler implementation and broad applicability.

## 3.9 Bibliography

Brooks, S. P. & Gelman, A. (1998), 'General Methods for Monitoring Convergence of Iterative Simulations', *Journal of Computational and Graphical Statistics* **7**(4), 434–455.

Geman, S. & Geman, D. (1984), 'Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **84**(6), 721–741.

Geyer, C. J. (1991), 'Markov Chain Monte Carlo Maximum Likelihood', *Computing Science and Statistics: Proceedings of the 23rd Symposium Interface* pp. 156–163.

Geyer, C. J. & Thompson, E. A. (1995), 'Annealing Markov Chain Monte Carlo with Applications to Ancestral Inference', *Journal of the American Statistical Association* **90**(431), 909–920.

Gramacy, R., Samworth, R. & King, R. (2010), 'Importance Tempering', *Statistics and Computing* **20**(1), 1–7.

Kim, H. J. & MacEachern, S. N. (2015), 'The Generalized Multiset Sampler', *Journal of Computational and Graphical Statistics* **24**(4), 1134–1154.

Kone, A. & Kofke, D. A. (2005), 'Selection of Temperature Intervals for Parallel-Tempering Simulations', *The Journal of Chemical Physics* **122**(20), 206101.

Kou, S. C., Zhou, Q. & Wong, W. H. (2006), 'Equi-Energy Sampler with Applications in Satistical Inference and Statistical Mechanics', *The Annals of Statistics* **34**(4), 1581–1619.

Leman, S. C., Chen, Y. & Lavine, M. (2009), 'The Multiset Sampler', *Journal of the American Statistical Association* **104**(487), 1029–1041.

Liang, F. & Wong, W. H. (2001), 'Real-Parameter Evolutionary Monte Carlo with Applications to Bayesian Mixture Models', *Journal of the American Statistical Association* **96**(454), 653–666.

Liu, J. S., Liang, F. & Wong, W. H. (2000), 'The Multiple-Try Method and Local Optimization in Metropolis Sampling', *Journal of the American Statistical Association* **95**(449), 121–134.

Marinari, E. & Parisi, G. (1992), 'Simulated Tempering: a New Monte Carlo Scheme', *Europhysics Letters* **19**(6), 451–458.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. (1953), 'Equation of State Calculations by Fast Computing Machines', *The Journal of Chemical Physics* **21**(6), 1087–1092.

Miasojedow, B., Moulines, E. & Vihola, M. (2013), 'An Adaptive Parallel Tempering Algorithm', *Journal of Computational and Graphical Statistics* **22**(3), 649–664.

Neal, R. M. (1996), 'Sampling from Multimodal Distributions Using Tempered Transitions', *Statistics and Computing* **6**(4), 353–366.

Rigat, F. & Mira, A. (2012), 'Parallel Hierarchical Sampling: A General-Purpose Interacting Markov Chains Monte Carlo Algorithm', *Computational Statistics & Data Analysis* **56**(6), 1450–1467.

Swendsen, R. H. & Wang, J.-S. (1986), 'Replica Monte Carlo Simulation of Spin-Glasses', *Physical Review Letters* **57**(21), 2607–2609.

Tierney, L. (1994), 'Markov Chains for Exploring Posterior Distributions', *The Annals of Statistics* **22**(4), 1701–1728.

# Chapter 4

# Weighted Particle Tempering: Applications and Case Studies

**Abstract**

In the previous chapter, we introduced weighted particle tempering and showed that it can outperform parallel tempering and parallel hierarchical sampling through several simulation studies. In this chapter, we explore two case study applications of weighted particle tempering: Bayesian classification trees for the Wisconsin breast cancer dataset and decomposable Gaussian graphical models for financial data. Both of these applications require the sampling of graphs, which is a traditionally difficult problem. In particular, depending on the types of local-update MCMC moves, traversing graph space often requires visiting lower-density graphs before reaching higher-density ones. In this sense, posterior distributions on graphs are often highly multimodal. We showcase the performance of weighted particle tempering, again in comparison to parallel tempering and parallel hierarchical sampling. While all three algorithms have similar performances, the key advantage of weighted particle tempering over the other two is that it is significantly easier to tune.

**Keywords.** Bayesian binary classification tree, decomposable Gaussian graphical model

## 4.1 Introduction

In this chapter, we explore how weighted particle tempering can be applied to a variety of models for Bayesian inference. Specifically, we use weighted particle tempering to learn binary classification trees for breast cancer prediction, and we implement the algorithm to infer decomposable Gaussian graphical models for stock portfolio estimation. We compare our algorithm to parallel tempering (Swendsen & Wang (1986),Geyer (1991)) and parallel hierarchical sampling (Rigat & Mira 2012), as well as Metropolis-Hastings (Hastings 1970) as implemented in Chipman et al. (1998).

The models we use in this section are similar in that they use concepts from combinatorics and graph theory. An excellent introduction to this contemporary field can be found in Harris et al. (2008). Most relevant to our purposes is the fact that the models explore a very large, discrete space. In general, optimization and posterior sampling for large discrete sample spaces are challenging tasks. Aoki et al. (2012) provides a review of Markov chain Monte Carlo methods for this scenario. In particular, the challenges arise when the MCMC update rules define a connected space over which the algorithm can traverse. If this connected space requires that, to get from one state of a certain target distribution value to another state of higher target distribution value, the algorithm must pass a valley of states of lower target distribution value, then the target distribution can be considered multimodal for the algorithm's purposes. Sampling over such a multimodal probability space can be quite difficult, as local-update MCMC methods can get trapped in modes. We find that since weighted particle tempering has a mother chain with a selection of underlying chains from which to choose, the algorithm is well-suited to escape these local traps.

The remainder of this chapter is divided in the following sections. Section 4.2 looks at the canonical Wisconsin Breast Cancer Dataset and how the covariate space can be partitioned to classify tumor cases as benign or malignant. Section 4.3 explores a dataset of log-returns for 20 randomly chosen stocks from the S&P 500, performing covariance regularization through the use of Gaussian graphical models. Finally, we finish with a discussion in section 4.4.

Table 4.1: Variables for the Wisconsin Breast Cancer Dataset

| *Variable* | *Code* |
|---|---|
| Clump thickness | clump |
| Uniformity of cell size | size |
| Uniformity of cell shape | shape |
| Marginal adhesion | adhes |
| Single epithelial cell size | secs |
| Bare nuclei | bare |
| Bland chromatin | bland |
| Normal nucleoli | normal |
| Mitosis | mitosis |
| Class | class (2 = benign, 4 = malignant) |

## 4.2 Bayesian Classification Trees for the Wisconsin Breast Cancer Dataset

### 4.2.1 Data and Model

In this section, we explore the applicability of weighted particle tempering to the same binary classification trees of Chipman et al. (1998), hereafter referred to as CGM after the initials of the three authors. In particular, this model is used for supervised classification in the canonical Wisconsin Breast Cancer Dataset (Wolberg & Mangasarian 1990), which is available at the UCI Machine Learning Repository (Lichman 2013). The data consist of 683 complete cases of cellular characteristics that are useful in predicting the malignancy of a tumor. All of the nine covariates are ordinal variables, each taking an integer value in $\{1, \ldots, 10\}$. The variable names and descriptions are given in table 4.1.

The model consists of a binary tree that gives the probability of a benign tumor given

its cellular characteristics. Terminal nodes on the tree store these probabilities, which are retrieved by filtering data through the tree's internal nodes, beginning at the root node. Each internal node on the tree has a splitting rule that sends an observation to the left or right child node. A splitting rule consists of a splitting variable and value, *e.g.* $\{bare \geq 7\}$, where if a datum falls on such a node, it will be sent to the left child if its value for *bare* is strictly less than 7 and to the right child otherwise.

The space of all binary classification trees for this application is quite large, and fitting these trees to the data to minimize the misclassification rate can lead to deep, complicated, and unwieldy trees – model overfitting – which can be detrimental to out-of-sample prediction. That is why *a priori*, we would like to induce sparsity on the tree structure. To accomplish this, we adopt CGM's generative prior on trees where the probability of splitting an intermediary tree $T$ at terminal node $\eta$ is

$$p_{split}(\eta|T) = \alpha \left(1 + d_\eta\right)^{-\beta}.$$

Here, $d_\eta$ is the depth of node $\eta$ on tree $T$, and $\alpha \in [0, 1]$ and $\beta \geq 0$ are hyperparameters. We choose $\alpha = 0.95$ and $\beta = 1$ to match the work of CGM. One generates a tree from the prior distribution by initiating a root node tree $T_0$ and then recursively iterating through trees until no terminal node is split.

The terminal nodes store sets of probabilities for class labels. In our case there are only two classes: benign or malignant. By using Dirichlet($\boldsymbol{\alpha}_p$) priors for these terminal node probabilities conditioned on the tree $T$, we are able to marginalize out the probabilities and perform inference directly on the tree structure using the posterior $P(T|X, Y) \propto P(T)P(Y|X, T)$. The marginal likelihood is

$$P(Y|X, T) = \left(\frac{\Gamma\left(\sum_k \alpha_{pk}\right)}{\prod_k \Gamma\left(\alpha_{pk}\right)}\right)^b \prod_{i=1}^{b} \frac{\prod_k \Gamma\left(n_{ik} + \alpha_{pk}\right)}{\Gamma\left(n_i + \sum_k \alpha_{pk}\right)},$$

where $n_{ik}$ is the number of observations at the $i$th terminal node belonging to the $k$th class, $n_i = \sum_k n_{ik}$ is the total number of observations at the $i$th terminal node, $k = 1, \ldots, K$, where $K$ is the number of classes, and $b$ is the number of terminal nodes in tree $T$. In our

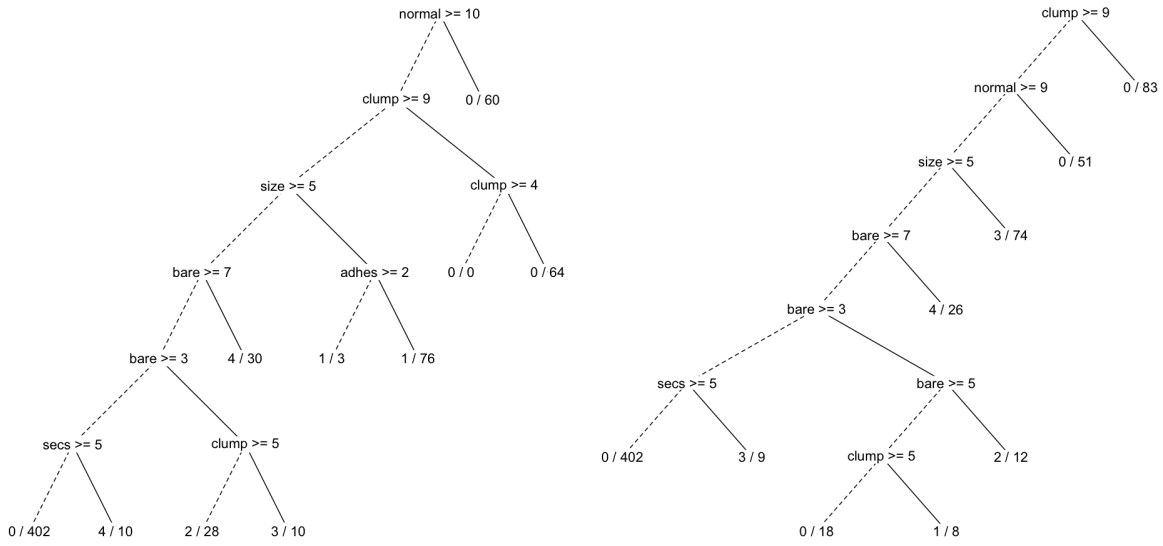
application $K = 2$ since there are only two classes, and we use $\alpha_{p1} = \alpha_{p2} = 1$, matching the prior specification of CGM.

### 4.2.2 Sampling Binary Trees

We sample over tree space using a random walk defined by the same four moves in CGM: *Grow*, *Prune*, *Swap*, and *Change*. *Grow* randomly splits a terminal node into two new terminals, and randomly assigns a splitting rule to the new parent. *Prune* is the reverse of *Grow*, randomly merging two sibling terminal nodes. *Swap* exchanges the splitting rules of parent/child nodes that are both internal nodes, unless the child and its sibling have identical rules, in which case both siblings exchange with the parent. *Change* randomly selects an internal node and assigns a new splitting rule at random. Both *Swap* and *Change* are their own reverse moves.

Exploring tree space is complicated by the existence of multiple modes. CGM take on this problem headlong by running long chains of their Metropolis-Hastings algorithm at multiple starting locations. They initiate the MH algorithm at the single-node tree 500 times and collect 5,000 samples each time. They also initiate the MH algorithm at each of the 10 best (in terms of marginal likelihood) trees found by greedy search using 500 bootstrap samples of the data. Then for each of the 10 starting trees, they run 25 chains of 1,000 iterations. Their large search over this space identifies sparse trees with log-marginal likelihoods as high as -62.2.

In a sense, weighted particle tempering performs these functions automatically, substantially simplifying the search process and producing a larger sample of interesting trees in less time. With only ten tempered particles, a tempering exponent of $\nu = 0.9$ and weighting exponent of $\delta = 1$, we collect 1,000 iterations and are able to find trees with log-marginal likelihoods as high as -60.4. Such a run of weighted particle tempering, performed in parallel, is computationally equivalent to a single run of Metropolis-Hastings with only 11,000 iterations, although the latter is necessarily performed in series.




WPT CGM

Figure 4.1: Highest posterior tree found by weighted particle tempering and CGM's best tree. Dashed (solid) edges indicate the path when the parent splitting rule evaluates as false (true). Terminal nodes display the number of misclassified cases out of total cases arriving at the node.

Figure 4.1 depicts the tree with the highest posterior found by WPT in comparison with one of CGM's most well-rounded trees. While the difference in their log-posteriors is small, it is important to note the discrepancy in amount of computation needed for each algorithm to find their best trees. Additionally, the Metropolis-Hastings algorithm requires multiple restarts. As a result, it is not clear how to combine samples across multiple chains to produce posterior samples.

For comparison, we also use parallel tempering and parallel hierarchical sampling to produce tree samples. With parallel tempering, we use a log-linear sequence of ten tempered levels below the mother chain with a minimum exponent of $\nu_p = 0.5$. With parallel hierarchical sampling, we also use ten underlying chains for consistency among the different methods. While these two algorithms provide significant improvements over CGM's random walk Metropolis-Hastings algorithm in terms of mixing and convergence to interesting trees,

they both suffer from long burnin periods and sticking. As a result, after the 1,000 iterates are collected from both algorithms, the highest posterior tree found by parallel tempering and parallel hierarchical sampling are 1.2 and 5.4 log-posterior units less than that of weighted particle tempering, respectively. Since we are able to save some computation and tuning with weighted particle tempering over its competitors, we conclude that the algorithm is an attractive alternative to the usual workhorses.

## 4.3 Decomposable Gaussian Graphical Models for Financial Data

### 4.3.1 Data and Model

In this section, we analyze financial data for $p = 20$ randomly chosen stocks (table 4.2) and perform covariance regularization through the use of Gaussian graphical models. Our data are log-returns derived from daily price data. That is, if $P_{i,t}$ is the adjusted closing price of stock $i$ on day $t$, then $y_{i,t} = \log(P_{i,t+1}/P_{i,t})$ is the log return associated with purchasing the stock and selling on the following day. We treat the market vector of log-returns $\mathbf{y}_t = (y_{1,t}, \ldots, y_{p,t})$ for $p$ stocks as an independent draw from a multivariate normal distribution with mean zero and covariance matrix $\Sigma_G$. The data are aggregated into a matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$ for the $T = 2{,}200$ business days between January 4, 2007 to September 29, 2015. We ignore issues of irregularities in the time periods between observations, such as weekends and holidays.

The graph $G = \{V, E\}$ is composed of vertices representing each stock with an edge shared between two vertices if the corresponding element in the precision matrix $\Phi = \Sigma_G^{-1}$ is nonzero. That is, $(i, j) \in E$ if and only if $\Phi_{i,j} \neq 0$. We restrict our attention to decomposable graphs, for which computation of the marginal posterior distribution of $G$ is tractable. For more on Gaussian graphical models, we refer the interested reader to the seminal work of Lauritzen

(1996).

We place a prior on $G$ that encourages a sparse number of edges by making the prior edge inclusion probability a function of the number of vertices. Specifically, we place an independent Bernoulli distribution on the edge inclusion indicators, making the prior edge inclusion probability $\beta = 2/(|V|-1)$. For an unrestricted model that admits both decomposable and non-decomposable graphs, the *a priori* expected number of edges would therefore be:

$$\begin{pmatrix} |V| \\ 2 \end{pmatrix} \frac{2}{|V|-1} = |V|.$$

By picking a hyper-inverse Wishart distribution as the conditional prior for the covariance matrix $\Sigma_G$, we are able to marginalize out the covariance matrix and perform direct inference on the graph itself. The marginal posterior of the graph is proportional to the ratio of normalizing constants for the hyper-inverse Wishart distributions in the prior and posterior for $\Sigma_G$ (see Jones et al. (2005) for example).

Our model is therefore:

$$\begin{aligned} p\left(G = \{V, E\}\right) &\propto \frac{2}{|V|-1}^{|E|} \left(1 - \frac{2}{|V|-1}\right)^{\begin{pmatrix} |V| \\ 2 \end{pmatrix}^{-|E|}} \\ \Sigma_G|G &\sim HIW_G(\delta, D) \\ \mathbf{y}_t|\Sigma_G &\sim N\left(\mathbf{0}, \Sigma_G\right), \end{aligned}$$

and as a result the marginal likelihood of $\mathbf{Y}$ given $G$ is:

$$p(\mathbf{Y}|G) = (2\pi)^{-T|V|/2} \frac{h(G, \delta, D)}{h(G, \delta^*, D^*)},$$

where $h(G, \delta, D)$ is the normalizing constant for the hyper-inverse Wishart distribution with degrees of freedom $\delta$ and location matrix $D$. Specifically, this normalizing constant is

$$h(G, \delta, D) = \prod_{p \in P} \frac{|D_{pp}|^{\delta/2}}{2^{\delta|V_p|/2}\Gamma_{|V_p|}(\delta/2)} \Bigg/ \prod_{s \in S} \frac{|D_{ss}|^{\delta/2}}{2^{\delta|V_s|/2}\Gamma_{|V_s|}(\delta/2)}.$$

Here, $P$ is the set of all prime components of the graph $G$, and $S$ is the set of separators. $D_{pp}$ and $D_{ss}$ denote the submatrices of $D$ corresponding to the vertices in prime component $p$ and separator $s$, respectively. $\Gamma_n(\cdot)$ is the multivariate gamma function for dimension $n$. The posterior parameters are $\delta^* = \delta + T$ and $D^* = D + \sum_{t=1}^{T} \mathbf{y}_t \mathbf{y}_t'$, and for prior parameters we use $\delta = p + 1$ and $D = I_{p \times p}$.

### 4.3.2 Sampling Decomposable Graphs

We would ideally take a fully Bayesian approach to inferring the underlying graph in the model above. To do so, we employ a proposal mechanism that can search over the massive space of decomposable graphs and avoid sticking to local modes in the posterior. We use the results of Giudici & Green (1999), which give necessary and sufficient conditions for a decomposable graph, to build a list of candidate edges to either add or delete while preserving decomposability for a move we call *Edge Switch*. Additionally, we perform a move called *Label Swap*, which preserves the topology of the graph but attempts to switch the labels of two vertices. A single random walk step in all of our implementations consists of choosing one of the two proposal moves with equal probability. Both move types are their own reverse moves.

In such an application, collecting an adequate sample of graphs is infeasible due to the size of the sample space. Instead, our goal is to find a tree that gives the largest posterior in the least amount of time. For WPT, we use $\nu = 0.8$ as the tempering exponent, $\delta = 1 - \nu$ as the proposal weight exponent, and $p = 5$ workers. In our PT implementation, we use $\nu_p = 0.8$ as the smallest exponent, then increment in a log-linear fashion to the largest exponent $\nu_0 = 1$, again with $p = 5$ underlying chains. Finally, we implement PHS with $p = 5$ underlying chains. We initialize all three algorithms at the complete graph and generate $N = 2{,}000$ samples with each algorithm.

Figure 4.2 demonstrates the highest posterior graph found by weighted particle tempering. This graph is 0.3 and 0.8 log-posterior units greater than the best graphs found by parallel

tempering and parallel hierarchical sampling, respectively. Again, we see that we achieve similar levels of performance when compared to parallel tempering and parallel hierarchical sampling, but with simplified tuning and with a slight speedup due to parallelization.

## 4.4 Discussion

We have applied weighted particle tempering to two particularly intractable models: binary decision trees for classification and decomposable Gaussian graphical models. Our findings are that weighted particle tempering performs competitively with other algorithms, with the added benefits of simpler tuning (especially in comparison to parallel tempering) and parallelizability. That is, if the algorithm struggles with a particular application, one remedy that is always available is to increase the number of underlying chains. Doing so has minimal impact on computational time if the algorithm is run in parallel.

We have established the versatility and robustness of weighted particle tempering. Further work is now needed to extend the algorithm to multiple tempering exponents for the underlying particles or, similar to parallel hierarchical sampling, building a hierarchy composed of multiple layers of weighted particle tempering. Sampling the underlying tempered distributions can be performed by using weighted particle tempering at even smaller tempering exponents. It would be interesting to see the limits of this algorithm, which has already proven to be highly applicable and useful.

## 4.5 Bibliography

Aoki, S., Hara, H. & Takemura, A. (2012), Markov Chain Monte Carlo Methods over Discrete Sample Space, *in* 'Markov Bases in Algebraic Statistics', Springer, pp. 23–31.

Chipman, H. A., George, E. I. & McCulloch, R. E. (1998), 'Bayesian CART Model Search', *Journal of the American Statistical Association* **93**(443), 935–948.

Geyer, C. J. (1991), 'Markov Chain Monte Carlo Maximum Likelihood', *Computing Science and Statistics: Proceedings of the 23rd Symposium Interface* pp. 156–163.

Giudici, P. & Green, P. J. (1999), 'Decomposable Graphical Gaussian Model Determination', *Biometrika* **86**(4), 785–801.

Harris, J. M., Hirst, J. L. & Mossinghoff, M. J. (2008), *Combinatorics and Graph Theory*, Vol. 2, Springer.

Hastings, W. K. (1970), 'Monte Carlo Sampling Methods Using Markov Chains and Their Applications', *Biometrika* **57**(1), 97–109.

Jones, B., Carvalho, C., Dobra, A., Hans, C., Carter, C. & West, M. (2005), 'Experiments in Stochastic Computation for High-Dimensional Graphical Models', *Statistical Science* **20**(4), 388–400.

Lauritzen, S. L. (1996), *Graphical Models*, Oxford University Press.

Lichman, M. (2013), 'UCI Machine Learning Repository'.
**URL:** *http://archive.ics.uci.edu/ml*

Rigat, F. & Mira, A. (2012), 'Parallel Hierarchical Sampling: A General-Purpose Interacting Markov Chains Monte Carlo Algorithm', *Computational Statistics & Data Analysis* **56**(6), 1450–1467.

Swendsen, R. H. & Wang, J.-S. (1986), 'Replica Monte Carlo Simulation of Spin-Glasses', *Physical Review Letters* **57**(21), 2607–2609.

Wolberg, W. H. & Mangasarian, O. L. (1990), 'Multisurface Method of Pattern Separation for Medical Diagnosis Applied to Breast Cytology.', *Proceedings of the National Academy of Sciences* **87**(23), 9193–9196.

Table 4.2: Twenty randomly selected stocks for our Gaussian graphical model.

| Ticker | Name | Sector |
|---|---|---|
| JWN | Nordstrom | Apparel |
| AN | AutoNation Inc. | Auto |
| TGT | Target Corp. | Discount |
| GME | GameStop Corp. | Electronics |
| ETFC | Etrade Financial | Financial Services |
| K | Kellogg | Food |
| LNC | Lincoln National Corp. | Insurance |
| HIG | The Hartford Group | Insurance |
| ROK | Rockwell Automation Inc. | Machinery |
| DNR | Denbury Resources Inc. | Oil and Gas |
| TSO | Tesoro Corp. | Oil and Gas |
| AGN | Allergan | Pharmaceutical |
| HCP | HCP Inc. | REIT |
| VTR | Ventas Inc. | REIT |
| ALTR | Altera Corp. | Semiconductor |
| LRCX | LRCX | Semiconductor |
| TER | Teradyne Inc. | Semiconductor |
| X | US Steel Corp. | Steel |
| CTSH | Cognizant Tech Corp. | Tech |
| VRSN | VeriSign Inc. | Tech |

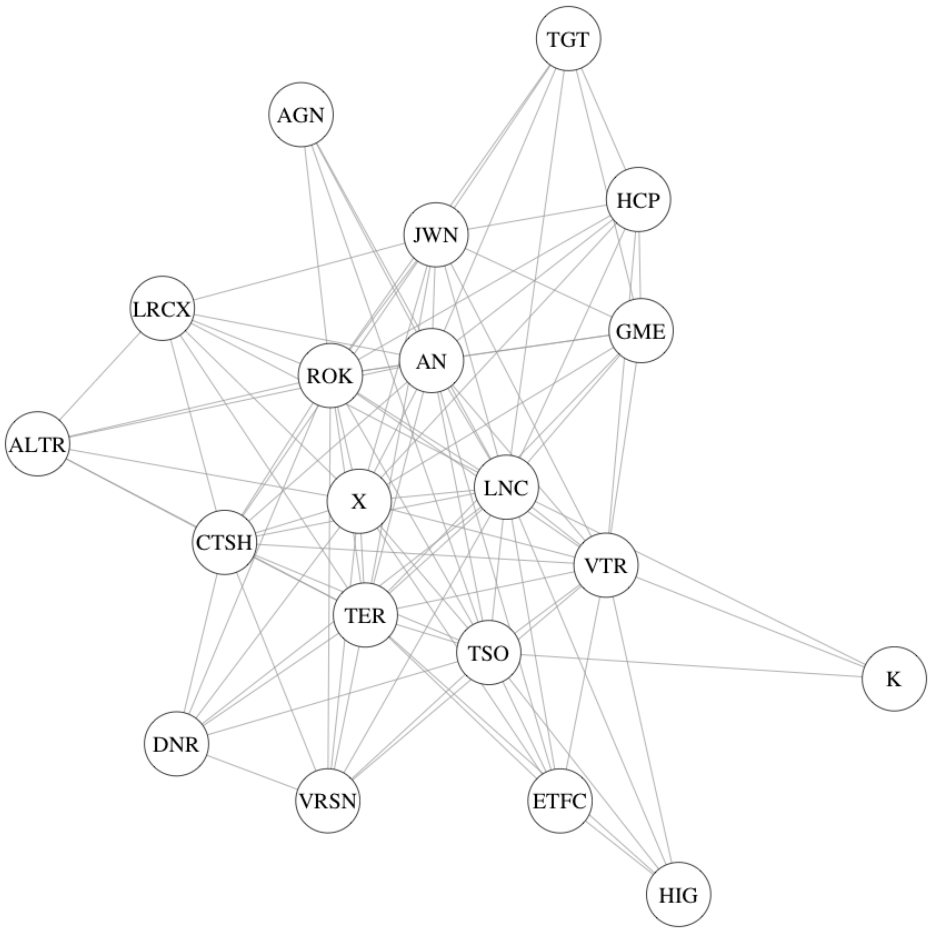



Figure 4.2: Highest posterior graph found by weighted particle tempering. It is 0.3 and 0.8 log-posterior units better than the best graphs found by parallel tempering and parallel hierarchical sampling, respectively.

# Chapter 5

# Practical Reversible Jump Markov Chain Monte Carlo

**Abstract**

Reversible jump Markov chain Monte Carlo (RJMCMC) is an algorithm that provides a way to sample from a posterior distribution on a sample space of varying dimension. This scenario typically surfaces when selecting from or averaging over several models, where different models have different numbers of parameters. However, often times it is not clear how to propose a new sample when moving across dimensions, since there is a possibly complex posterior dependence structure between model indicator and model parameters. Performing such a move requires *a priori* knowledge of this structure. We develop a straightforward, easily implemented method to generate a proposal across model spaces and ultimately perform RJMCMC. The method involves continuously optimizing parameters for each model, then using an independence sampler generating samples near the optima every time a model jump is attempted.

**Keywords.** Bayesian model averaging, growth curves, reversible jump MCMC

## 5.1 Introduction

Reversible Jump MCMC (RJMCMC) was introduced in Green (1995) as a method to tackle the problem of Bayesian model determination. Generally speaking, a researcher may be faced with a countable number of models from which to choose. Bayesian methods allow the researcher to specify a prior over the space of models, and, in theory, extract from the data a posterior distribution over the space. Under certain conditions, this posterior will asymptotically collapse on the "true model," should such a thing exist. However, under finite sample sizes, posterior model probabilities are extremely useful for model averaging (*e.g.* in prediction) or model selection (via the maximum *a posteriori* model), especially under a decision theoretic framework.

While the creation of RJMCMC is in itself a great theoretical accomplishment, its implementation for any arbitrary problem continues to be a practical hurdle. The nontrivial tasks inherent in applying RJMCMC involve tuning proposal distributions for moves both within and across model space, as well as specifying a suitable mapping from one set of dimensions to another. To do this effectively requires a strong understanding of all relationships among models and variables in addition to knowledge of and proficiency in MCMC engineering.

To illustrate the difficulty in tuning an algorithm to perform cross-model moves, figure 5.1 shows a case where the conditional parameter distributions may change drastically in their locations depending on the model under consideration. Such a situation may occur in a set of models where parameters are correlated *a posteriori*. In the figure, there are three models, $M_1, M_2$, and the full model $M_3$, and two parameters, $\alpha$ and $\beta$. If the locations of the parameters do not coincide across models as in this depiction, the RJMCMC user will need to map out exactly how to make these proposals. Specifically, the mode of $\alpha$ under model $M_1$ is greater than that under the full model $M_3$. The researcher must know this prior to tuning his or her RJMCMC algorithm, so that when moving from model $M_1$ to $M_3$, the value of a new $\alpha$ is reduced to bring the proposal closer to a region of high probability.

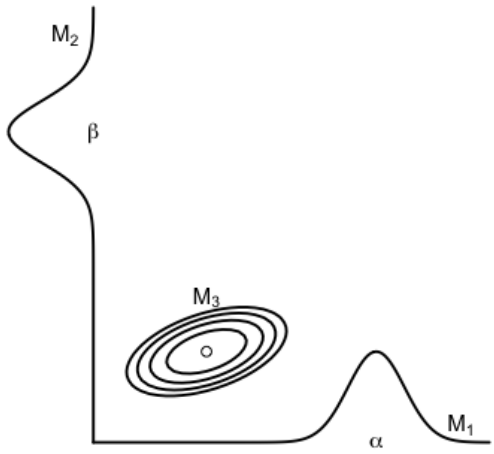



Figure 5.1: Jumping across models $M_1$, $M_2$, and $M_3$ is not straightforward in a situation such as this. The locations of conditional modes change depending on the model. A well-designed RJMCMC implementation should account for this when moving across model space.

Of course within the two decades following the introduction of RJMCMC, significant progress has been made in developing algorithms that search over the model-parameter space efficiently. Brooks et al. (2003) provides an excellent set of examples of RJMCMC using a more specialized method to perform inferences in the context of autoregressive models, Gaussian graphical models, and mixture models. The methods outlined in the work look to center proposals around equal-likelihood parameter values and optimally scale the proposal distribution about these values. One drawback is that the mapping function $h_{k\to k'}$ needs to be specified. The work of Ehlers & Brooks (2008) provides some respite by proposing the use of the posterior conditional distribution over the parameters in the proposed model, conditioned on the parameters of the current model. Such a method, however, requires knowledge or approximation of the full conditionals, a necessity that may be impossible to achieve for the general inference problem.

Other efforts have been made to sidestep the issue of finding a suitable transformation function entirely, among which is the seminal work of Carlin & Chib (1995), wherein the authors achieve model selection by specifying a pseudo-prior for all models and by computing full conditional distributions to perform Gibbs sampling.

The current work extends that of Dellaportas et al. (2002) where a suggested strategy is to jump across models by taking proposals from a distribution centered at a maximizer of the target distribution: "One possible strategy, where appropriate, is to generate parameter values using a normal distribution centred at the maximum likelihood estimate for $\beta_m$ with variance equal to the asymptotic variance of the mle." One might point out that the MLE does not maximize the conditional posterior distribution. However, by the Bernstein-von Mises theorem and under mild conditions, the MLE and maximum *a posteriori* (MAP) estimators converge to each other, which provides some justification for the method.

Our method requires some preprocessing before running the MCMC. First we optimize the conditional posteriors to find the MAP estimator of parameters for each model. Then, whenever possible, we estimate the curvature, or the Hessian, of the conditional log-posteriors at the MAP, and use it as the covariance matrix for proposals. At runtime, our reversible jump algorithm proposes cross-model moves by drawing samples from a normal distribution centered at the MAP and with covariance given by the Hessian. In this way, the method is an independence sampler. One of the main benefits of this method, as we shall see, is that there is no need to devise a clever transformation function and compute its Jacobian.

The remainder of this paper develops the concepts of our very general method for performing RJMCMC. Section 5.2 introduces some notation and the general setting for reversible jump. Section 5.3 outlines our method. Sections 5.4, 5.5, and 5.6 exhibit simulation studies to illustrate the performance of our method, especially in comparison to usual implementations of reversible jump, such as birth/death processes or moment matching. Finally, we wrap up with a discussion in section 6.6.

## 5.2 Reversible Jump MCMC

Reversible jump is an extension of the typical Metropolis-Hastings algorithm to cross-dimensional moves. Specifically, if $M_k \in \{M_1, \ldots, M_P\}$ is a model in a possibly countably

infinite set of candidate models, then $(M_k, x)$ would be a state of an appropriate MCMC algorithm, where $k$ is the model indicator and $x$ is an $n_k$-dimensional parameter vector. The state space for the target distribution will be $\mathcal{X} = \cup_{k=1}^{P}(M_k \times \mathcal{X}_k)$, the union over all models of the model indicators crossed with model sample spaces $\mathcal{X}_k \subset \mathbb{R}$. We will drop the distinction between model $M_k$ and the indicator $k$, as this will simplify notation. Therefore, we consider the state of the Markov chain to be $(k, x)$.

The algorithm accomplishes a move between two models of dimensions $n_k$ and $n_{k'}$ by introducing an auxiliary random vector $u$ of dimension $r$ and a diffeomorphic transformation function $h : \mathbb{R}^{n_k} \times \mathbb{R}^r \to \mathbb{R}^{n_{k'}} \times \mathbb{R}^{r'}$, where $n_k + r = n_{k'} + r'$. This last requirement preserves the total dimension across moves (the so-called "dimension matching" requirement). The auxiliary random vector $u$ needs to be generated by a distribution $g(u|x, k \to k')$ specified by the user, where we define $g(u|x, k \to k')$ to be the proposal distribution when moving from model $k$ to $k'$ and conditioning on the current position $x$. We use the notation $h_{k\to k'}(x, u) = (x', u')$ for the diffeomorphic function, and $h^{-1}_{k\to k'}(x', u') = (x, u)$ for its inverse. A diffeomorphic function is a function whose derivative exists, as does the derivative of its inverse.

In order to preserve detailed balance, RJMCMC is carried out as in algorithm 5.1. This recipe produces a Markov chain that satisfies the detailed balance condition and converges to the target distribution $\pi(k, x)$. In a Bayesian context, this distribution is the posterior of a parameter set and model indicator $(k, \theta)$ that can be factorized – up to a proportionality constant – as $\pi(\theta, k|D) \propto L(D|\theta, k)\pi(\theta|k)\pi(k)$.

## 5.3 Practical RJMCMC

The current method seeks to simplify the process of choosing a set of proposal distributions $g(u'|x, k \to k')$ and transformation functions $h_{k\to k'}(x, u) = (x', u')$ – one for each type of model jump – in order to alleviate the implementational burdens of this otherwise sophisti-

**Algorithm 5.1** Reversible Jump Markov Chain Monte Carlo

Initialize $(k_0, x_0)$

**for** $t = 1, \ldots, N$ **do**

With probability $p(k_{t-1} \to k')$, propose a move from model $k_{t-1}$ to model $k'$

$u \sim g(u|x_{t-1}, k_{t-1} \to k')$

$(x', u') \leftarrow h_{k_{t-1} \to k'}(x_{t-1}, u)$

$\alpha \leftarrow 1 \wedge \frac{\pi(k',x')p(k' \to k_{t-1})g(u'|x',k' \to k_{t-1})}{\pi(k_{t-1},x_{t-1})p(k_{t-1} \to k')g(u|x_{t-1},k_{t-1} \to k')} \left| \begin{array}{cc} \frac{dx'}{dx_{t-1}} & \frac{du'}{dx_{t-1}} \\ \frac{dx'}{du} & \frac{du'}{du} \end{array} \right|$

**if** $v \sim Unif(0,1) < \alpha$ **then**

$k_t \leftarrow k'$

$x_t \leftarrow x'$

**else**

$k_t \leftarrow k_{t-1}$

$x_t \leftarrow x_{t-1}$

**end if**

**end for**

cated algorithm. To do this in a near-automatic way, we choose the transformation functions to yield a Jacobian of one in all applications.

We start by specifying the proposal distribution $g(u|x, k \to k')$ as a normal distribution centered at the MAP (maximum *a posteriori*) estimate for the posterior conditioned on model $k$. We call this location $\mu_{k'} = \arg\max_\theta \log L(D|\theta, k') + \log \pi(\theta|k')$. There is a rich literature and set of methods for finding $\mu_{k'}$ under several conditions, but there is no guarantee of being able to do so in general. For a broad set of applications, we recommend the limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm with box constraints, or "L-BFGS-B" (Broyden (1970), Fletcher (1970), Goldfarb (1970), Shanno (1970), Byrd et al. (1995)). This derivative-free algorithm allows for inequality constraints on the space over which we optimize, and it can produce an estimate of the Hessian of the objective function, which in our case is the conditional posterior.

Once we find $\mu_{k'}$, we ideally compute the Hessian $H$, or matrix of second derivatives, of the function $\log L(D|\theta, k') + \log \pi(\theta|k')$. From the negative Hessian, we compute the nearest positive definite matrix Knol & ten Berge (1989) and call this $\Sigma_{k'}$, which we use as the proposal distribution's covariance matrix.

The optimization routine is performed for each model $k$, with multiple starting values for the optimizer and by storing the optimal outcome over all restarts of the process. As such, this preparation process is easily divided among multiple workers and highly parallelizable. All optimized values are stored and used as the center of the proposal distributions for the reversible jump algorithm in section 5.2, which is therefore $u \sim N(\mu_{k'}, \Sigma_{k'})$.

In theory, an ideal proposal distribution would be one that is most similar to the conditional posterior $\pi(\theta|D, k)$. Our proposal distribution, however, provides a rough approximation to the conditional posterior near the maximum.

Next, we set the transformation function $h_{k\to k'}(x, u) = (u, x)$, essentially reversing the inputs for the outputs. It is easy to see that the dimension matching criterion is achieved in this case, and the Jacobian can be shown to be 1 as follows:

$$\begin{aligned} J &= \begin{vmatrix} \frac{dx'}{dx} & \frac{du'}{dx} \\ \frac{dx'}{du} & \frac{du'}{du} \end{vmatrix} \\ &= \begin{vmatrix} \frac{du}{dx} & \frac{dx}{dx} \\ \frac{du}{du} & \frac{dx}{du} \end{vmatrix} \\ &= \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix} \\ &= 1. \end{aligned}$$

By specifying a proposal distribution that roughly approximates the conditional posterior for all models with a transformation function that yields a Jacobian with a constant value of one regardless of the application, we are able to apply RJMCMC in a wide variety of

practical scenarios. What follows is a set of examples demonstrating its use.

## 5.4 Linear Regression

In this section, we perform variable selection for a multiple linear regression. It turns out that our method is especially well suited for situations where the conditional posterior distributions are normal. This is because the proposal distributions will match exactly to the conditional posteriors. As we shall see, our linear regression scenario has this property.

### 5.4.1 Simulation Design

We simulate $N = 1{,}000$ data points $y_i$ from the following zero-intercept linear model:

$$y_i = \mathbf{x}_i' \boldsymbol{\beta} + \epsilon_i,$$

with $\epsilon_i \sim N(0, 1)$. The full model has $\boldsymbol{\beta} \in \mathbb{R}^{10}$, but we consider all $K = 2^{10} - 1 = 1{,}023$ possible models with at least one nonzero coefficient. The covariate vectors $\mathbf{x}_i$ are independent, uncorrelated standard normal vectors. The true model has only $n = 3$ nonzero coefficients, which are also drawn from the standard normal distribution. All conditional priors on the coefficients are improper flat priors, $\pi(\boldsymbol{\beta}_k|M_k) \propto 1$. Finally, we place equal prior probability on all models.

With this model, we can analytically solve for the posterior distribution over the model space. To do this, we need the marginal likelihood,

$$L(\mathbf{y}|M_k) = \int L(\mathbf{y}|\boldsymbol{\beta}_k, M_k)\pi(\boldsymbol{\beta}_k|M_k)d\boldsymbol{\beta}_k,$$

which we would use to compute the posterior model probabilities:

$$\pi(M_k|\mathbf{y}) = \frac{L(\mathbf{y}|M_k)\pi(M_k)}{\sum_{i=1}^{K} L(\mathbf{y}|M_i)\pi(M_i)}.$$

It can be shown that the marginal likelihood for each model is

$$L(\mathbf{y}|M_k) = \left(\frac{1}{\sqrt{2\pi}}\right)^{N-n_k} |X_k'X_k|^{-1/2} \exp\left\{-\frac{1}{2}\mathbf{y}'(I - X_k(X_k'X_k)^{-1}X_k')\mathbf{y}\right\},$$

where $X_k$ is the covariate matrix for model $M_k$ and $n_k$ is the model dimension. Therefore, we can compute the posterior model probabilities by computing all 1,023 marginal likelihoods and normalizing them, since all models are *a priori* equally likely.

The conditional posterior distribution on the parameters is

$$\pi(\boldsymbol{\beta}_k|\mathbf{y}, M_k) = N\left((X_k'X_k)^{-1}X_k'\mathbf{y}, (X_k'X_k)^{-1}\right),$$

so the maximizer is the mean $(X_k'X_k)^{-1}X_k'\mathbf{y}$, and the negative Hessian of the log-posterior is the covariance $(X_k'X_k)^{-1}$. As such, draws from the proposal distributions using practical reversible jump will be exact draws from the conditional posteriors.

### 5.4.2 Birth/Death Reversible Jump

For a comparison with our practical reversible jump method, we run a birth/death implementation of reversible jump with two possible model jumps, *birth* or *death*. In a birth move, we propose introducing covariate $j$ into the model by first defining $\mathbf{r}(\boldsymbol{\beta}_k) := \mathbf{y} - X_k\boldsymbol{\beta}_k$ as the residuals from the current model and $\mu(\boldsymbol{\beta}_k) := (\mathbf{x}_j'\mathbf{x}_j)^{-1}\mathbf{x}_j'\mathbf{r}(\boldsymbol{\beta}_k)$ as the point estimate for the new coefficient from regressing the residuals on covariate $j$. Next, we define $\sigma(\boldsymbol{\beta}_k)^2 := \mathbf{r}(\boldsymbol{\beta}_k)'(I - \mathbf{x}_j(\mathbf{x}_j'\mathbf{x}_j)^{-1}\mathbf{x}_j')\mathbf{r}(\boldsymbol{\beta}_k)/(N-1)$ to be the estimated variance. The auxiliary random variable for a birth move is $u \sim N(0,1)$. By using the linear transformation $h_{k\to k'}(\boldsymbol{\beta}_k, u) = (\boldsymbol{\beta}_k, u\sigma(\boldsymbol{\beta}_k) + \mu(\boldsymbol{\beta}_k))$, we randomly generate our new candidate. It is easy to see that the Jacobian for this kind of move is $J_{birth} = \sigma(\boldsymbol{\beta}_k)$.

For a death move, we drop a coefficient from the model – call it $\beta_j$. This move does not require an auxiliary random variable, and the transformation is just $h_{k\to k'}(\boldsymbol{\beta}_k, \beta_j) = (\boldsymbol{\beta}_k, (\beta_j - \mu(\boldsymbol{\beta}_k))/\sigma(\boldsymbol{\beta}_k))$, where $\mu(\boldsymbol{\beta}_k)$ and $\sigma(\boldsymbol{\beta}_k)$ are defined in the same way as above. The Jacobian

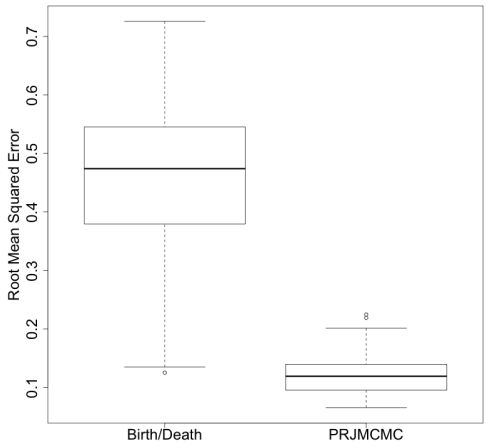



Figure 5.2: Boxplots comparing root mean squared error for estimates of posterior model probabilities for the birth/death algorithm and practical reversible jump.

for the transformation is necessarily the reciprocal of the Jacobian for the birth move, or $J_{death} = 1/\sigma(\boldsymbol{\beta}_k)$.

One more necessary detail is that if a death is proposed when only one coefficient remains in the model – a move that would result in a null model – then the proposal is automatically rejected in the sense that the Markov chain counter advances but the states remain fixed. This precaution ensures that births and deaths happen with equal probability, preventing the need to account for the probabilities of proposing to add or remove a candidate from a varying number of coefficients.

### 5.4.3 Results

To make our comparisons, we run both the practical reversible jump algorithm and the birth/death algorithm 100 times for 1,000 iterations each, then use the samples to estimate posterior model probabilities and compare them to the known posterior model probabilities. Figure 5.2 shows the results of the analysis by visualizing the root mean squared error (RMSE) of both algorithms. In this situation, practical reversible jump is especially well

suited for the analysis, and it shows. With an average RMSE of almost five times greater than that of practical reversible jump, the birth/death algorithm seems to require much larger chains to reduce its Monte Carlo variance. Additionally, the birth/death algorithm requires knowledge that regressing the residuals against the covariate that is entering or exiting the model provides a good transformation. In general, this kind of relationship is not always so obvious.

## 5.5 Poisson / Negative Binomial

In this section, we explore an example from Green & Hastie (2009) that illustrates where practical reversible jump can break down. Just as with any independence sampler, if the proposal distribution does not approximate the target distribution well, the algorithm will be slow to mix. The authors of Green & Hastie (2009) build a reversible jump algorithm to analyze a dataset relating to goal counts in soccer, selecting between a Poisson model and a negative binomial model. Another application of a similar model to a dataset about tumor counts in genetically-engineered mice can be found in Newton & Hastie (2006). We would like to know the ground truth, so our investigation uses simulated data. Keeping the true parameters constant, we generate several datasets of fixed size and use our reversible jump implementation to simulate from the posterior. For comparison, we also simulate from the posterior using the algorithm of Green & Hastie (2009), which we refer to as Green-Hastie.

### 5.5.1 Simulation Design

We simulate $n = 25$ negative binomial data points, $y \sim NB(r, p)$, and consider the following two models:

$$\begin{aligned} M_1 &: y \sim Poisson(\lambda), \\ M_2 &: y \sim NB(r, p). \end{aligned}$$

The Poisson model only has the unknown rate parameter $\lambda$, while the negative binomial (true) model has two parameters, size $r$ and probability $p$. Reversible jump MCMC allows us to sample from the posteriors of both models according to the evidence for each. We purposefully pick the true parameters $r = 10$ and $p = 0.35$ so that the expected value of the data is close to the variance, which will match what is expected of the first two moments for a set of Poisson random variables:

$$\begin{aligned} E(y_i) &= \frac{rp}{1-p} = \frac{10 \times 0.35}{0.65} \approx 5.38, \\ Var(y_i) &= \frac{rp}{(1-p)^2} = \frac{10 \times 0.35}{0.65^2} \approx 8.28. \end{aligned}$$

In doing so, the $n$ observations will be well-approximated by a Poisson distribution, and will encourage the algorithm to explore that model. We use the prior model probabilities of $\pi(M_k) = 0.5$ for $k \in \{1, 2\}$. For the parameters, we have the priors $\lambda \sim Exp(1)$, $r \sim Exp(1/20)$ and $p \sim Unif(0, 1)$.

Every time we generate a dataset, we run 10,000 iterations of both practical reversible jump and the Green-Hastie implementation and store their computed posterior model probabilities. Both algorithms are initialized in the same manner, and both use the same within-model Metropolis-Hastings move. Also, both algorithms have a 0.5 chance of proposing a model jump, so $p(1 \to 2) = p(2 \to 1) = 0.5$.

### 5.5.2 The Green-Hastie Implementation

Our parameterization differs from Green-Hastie, so we present the details in the following. Green-Hastie uses the auxiliary random variable $u \sim N(0, 1)$. When jumping from model $M_1$ to $M_2$, the transformation function is

$$h_{1\to 2}(\lambda, u) = \left(\mu \exp(u), \frac{\lambda}{\mu \exp(u) + \lambda}\right),$$

for some tuning parameter $\mu$. We will show how this parameter affects the Monte Carlo variance of the posterior model probabilities. In the reverse direction, there is no auxiliary

variable, and the transformation function is

$$h_{2\to1}(r,p) = \left(\frac{rp}{1-p}, \log\left(\frac{r}{\mu}\right)\right).$$

Notice that the transformation functions are a form of the usual moment-matching reversible jump.

One could verify that with this choice of functions, the Jacobians are:

$$\begin{aligned} J_{1\to2} &= \frac{1}{\left(1+\frac{\lambda}{\mu\exp(u)}\right)^2}, \\ J_{2\to1} &= \frac{1}{(1-p)^2}. \end{aligned}$$

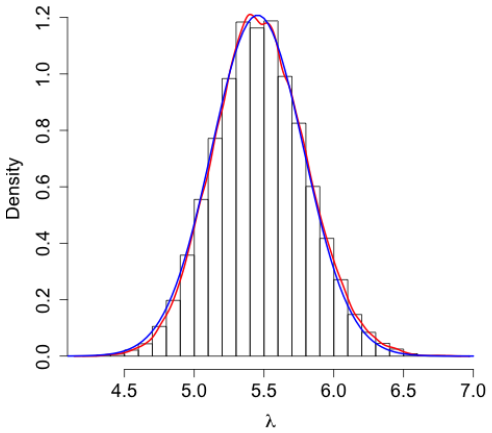
Therefore, the acceptance probabilities are:

$$\begin{aligned} \alpha_{1\to2} &= 1 \wedge \frac{\pi\left(2, \left(\mu\exp(u), \frac{1}{1+\frac{\lambda}{\mu\exp(u)}}\right)\right)}{\pi(1,\lambda)}\sqrt{2\pi}\exp(u^2/2)J_{1\to2}, \\ \alpha_{2\to1} &= 1 \wedge \frac{\pi\left(1, \frac{r(1-p)}{p}\right)}{\pi(2,(r,p))}\frac{1}{\sqrt{2\pi}}\exp\left(-\log(r/\mu)^2/2\right)J_{2\to1}. \end{aligned}$$

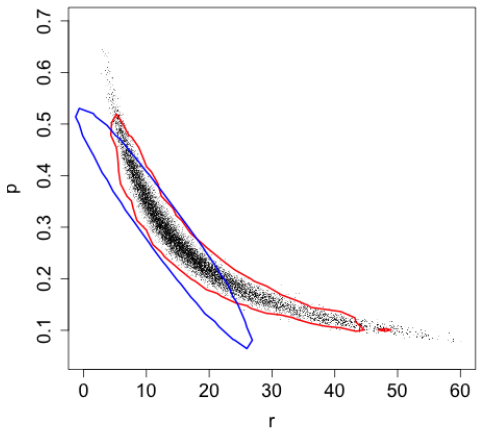
It is unclear if the transformation functions and proposal distribution for Green-Hastie are good ones for this problem, and if so, it is also not clear how to choose the tunable parameter $\mu$. We run the algorithm for various levels of the parameter to show that mixing is affected by the choice of $\mu$, and therefore so is the Monte Carlo variance.

### 5.5.3 Results

Samples from separate Metropolis-Hastings algorithms targeting the two models are shown in figure 5.3 to demonstrate what the conditional posterior distributions should look like. Since the samples in the figure were produced independently, they are unable to give posterior model probabilities. Superimposed on the images in figure 5.3 are the densities of the normal distributions that practical reversible jump uses as the proposal distribution when performing a model jump. From the figure, one can see that proposing into the Poisson model almost




Poisson




Negative Binomial

Figure 5.3: Samples from the conditional posteriors for the Poisson and negative binomial models. Red lines are sample densities. Blue lines are the densities of the normal distributions used to generate proposals for model jumps in practical reversible jump.

exactly generates a sample from the posterior, while proposing into the negative binomial model tends to place a sample within a high density region. Because the negative binomial model's posterior distribution is not well approximated by the normal distribution due to its curved contours, practical reversible jump suffers from slower mixing than Green-Hastie.

In any case, practical reversible jump is able to hop between models well. From a single run of the algorithm, we generate the traceplot for the two parameters $\lambda$ and $r$ in figure 5.4. This figure shows how the algorithm is capable of jumping between models, even if the within-model moves for the negative binomial model wander away from the location of the mass in the Poisson model. In other words, the algorithm is mixing well over the model space. However, careful inspection reveals that samples of the size parameter $r$ are not covering the tails of the distribution as well due to the independence sampling nature of practical reversible jump. Specifically, no sample is larger than 30, yet the conditional posterior distribution reaches as far as 60 in figure 5.3. In this particular simulation, the data only put a 0.5014 posterior probability on the negative binomial model according to

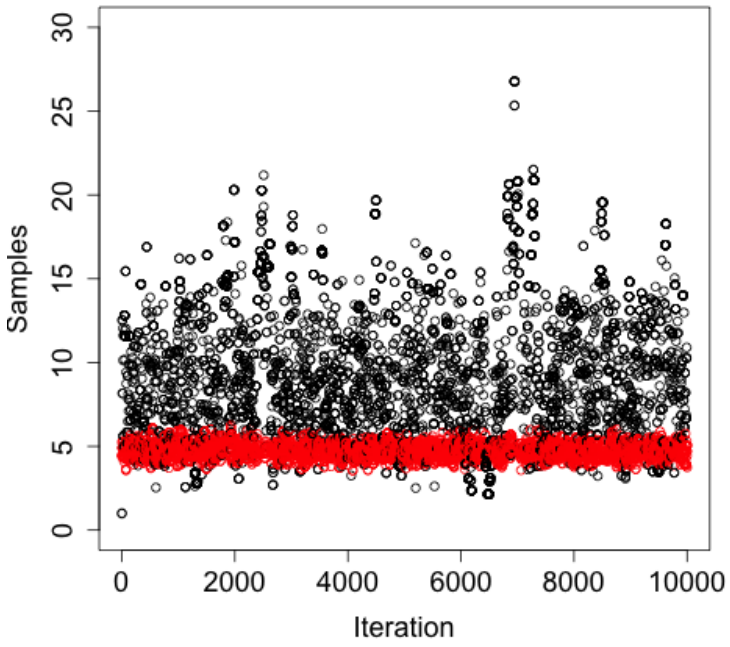



Figure 5.4: Posterior samples from $10,000$ iterations of practical reversible jump. The red denotes samples when the sampler explores the Poisson model ($\lambda$), while black denotes those from the negative binomial model ($r$).

the sample from practical reversible jump, or a 0.5050 posterior probability on the same according to the Green-Hastie sample.

We run 100 repetitions of the Green-Hastie algorithm each at ten different tuning levels for the parameter $\mu$, computing posterior model probabilities in every run. We do the same for practical reversible jump. Since the underlying dataset does not change, the only variability in these estimates is due to Monte Carlo error. Figure 5.5 shows these estimated probabilities and how they can vary as a function of the tuning parameter $\mu$. Since practical reversible jump does not require tuning, there is only one set of estimated probabilities. However, the Monte Carlo variance on the estimated posterior model probabilities is clearly larger for practical reversible jump, due to the fact that the algorithm explores each model for extended stretches of time before it can model jump. One potential solution for this would be to run

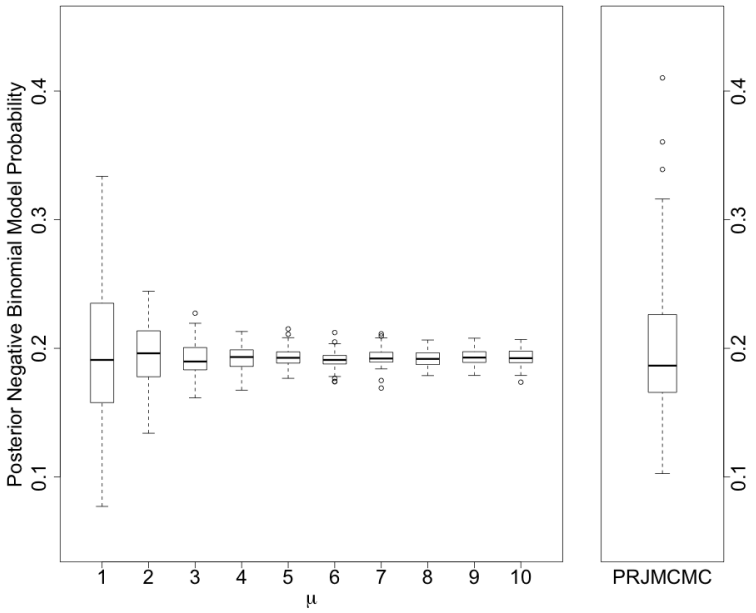



Figure 5.5: Estimated negative binomial model probabilities for Green-Hastie at different levels of the tuning parameter $\mu$ (left) and practical reversible jump (right).

some pilot chains targeting the conditional posteriors, then estimate these distributions with a Gaussian mixture and simulate from the fitted mixture at runtime.

We conclude this section by noting that any independence sampler will mix poorly if the proposal distribution does not approximate the target well. Likewise, practical reversible jump will also suffer if the contours of the conditional posterior distributions are poorly approximated by a normal distribution. The redeeming feature of practical reversible jump is that it is easy to tune, so the user may be satisfied with sacrificing implementation time in exchange for computer time.

## 5.6 Growth Curves

In this section, we illustrate how our method can sample over a set of nonlinear models. Particularly, we consider three parametric growth curves that could explain our simulated data. From the parameterization, it is not clear how to transform model parameters when jumping from one model to another. As such, we are unable to devise a clever transformation function or proposal distribution without significant analytic manipulations of the models. While estimates of our model's parameters are not consistent, we empirically show that estimates of the posterior model probabilities collapse on the true model as the sample size increases.

### 5.6.1 Simulation Design

In the scientific literature, there are several commonly-used sigmoidal curves to model growth data. We are interested in how reversible jump can average over or select from three: the Richards (Richards 1959), Gompertz (see Islam et al. (2002) for an application to forecasting diffusion of innovation), and logistic curves (Verhulst 1845). These nonlinear curves are defined below:

$$\begin{aligned} R(t, \boldsymbol{\theta}_R) &= \theta_{R,1} + \frac{\theta_{R,4} - \theta_{R,1}}{(\theta_{R,3} + \theta_{R,5} e^{-\theta_{R,2} t})^{1/\theta_{R,6}}}, \\ G(t, \boldsymbol{\theta}_G) &= \theta_{G,1} e^{-\theta_{G,2} e^{-\theta_{G,3} t}}, \\ L(t, \boldsymbol{\theta}_L) &= \frac{\theta_{L,1}}{1 + \theta_{L,2} e^{-\theta_{L,3} t}}, \end{aligned}$$

where all parameters are non-negative, and $\dim(\boldsymbol{\theta}_R) = 6$ while $\dim(\boldsymbol{\theta}_G) = \dim(\boldsymbol{\theta}_L) = 3$.

Notice also that the logistic curve is a special case of the Richards curve when $\theta_{R,1} = 0$, $\theta_{R,2} = \theta_{L,3}$, $\theta_{R,3} = 1$, $\theta_{R,4} = \theta_{L,1}$, $\theta_{R,5} = \theta_{L,2}$, and $\theta_{R,6} = 1$. It can also be shown that the Gompertz curve can be recovered from the Richards curve by letting $\theta_{R,6} \to 0^+$, with $\theta_{R,1} = 0$, $\theta_{R,3} = 1$, and $\theta_{R,5} = -1 + (\theta_{R,4}/C)^{\theta_{R,6}}$ for some constant $C$. In this case, $\theta_{R,4} = \theta_{G,1}$, $\log(C/\theta_{R,4}) = \theta_{G,2}$ and $\theta_{R,2} = \theta_{G,3}$.

Table 5.1: True values and summary statistics for Monte Carlo samples of the true model's underlying parameters.

| Parameter | True Value | Posterior Mean (s.e.) |
|---|---|---|
| $\theta_{R,1}$ | 0.7 | 0.79 (0.06) |
| $\theta_{R,2}$ | 0.5 | 0.46 (0.06) |
| $\theta_{R,3}$ | 0.3 | 3.32 (0.58) |
| $\theta_{R,4}$ | 1.2 | 1.38 (0.02) |
| $\theta_{R,5}$ | 0.4 | 2.73 (0.53) |
| $\theta_{R,6}$ | 9 | 5.77 (0.47) |
| $\sigma$ | 0.01 | 0.0086 (0.0003) |

We simulate data from the Richards curve, then allow RJMCMC to sample over all of the three possibilities and report posterior summaries. We start by choosing a sample size of $n = 65$, which we will later vary to demonstrate convergence to the true model with an increasing number of samples. We set the true parameter values to $\boldsymbol{\theta}_R = (0.7, 0.5, 0.3, 1.2, 0.4, 9)'$. Then we uniformly simulate random time points $t_i \sim U(-10, 10)$ to finally create our dataset $y_i \sim N(R(t_i, \boldsymbol{\theta}_R), \sigma = 0.01)$ for $i = 1, \ldots, 65$ (see figure 5.6). We treat $\sigma$ as another unknown parameter. Figure 5.6 shows the resulting dataset along with the MAP estimators for the three models.

Conditional priors for all parameters in the three models are independent exponential distributions with rates $\lambda_M = \frac{1}{\dim(\boldsymbol{\theta}_M)+1}$ for $M \in \{R, G, L\}$. We select this prior to avoid discounting higher dimensional models, even though typically a researcher may want to do so for regularization and parsimony. Finally, we place a uniform prior over the three models, so $\pi(M) = 1/3$ for $M \in \{R, G, L\}$.

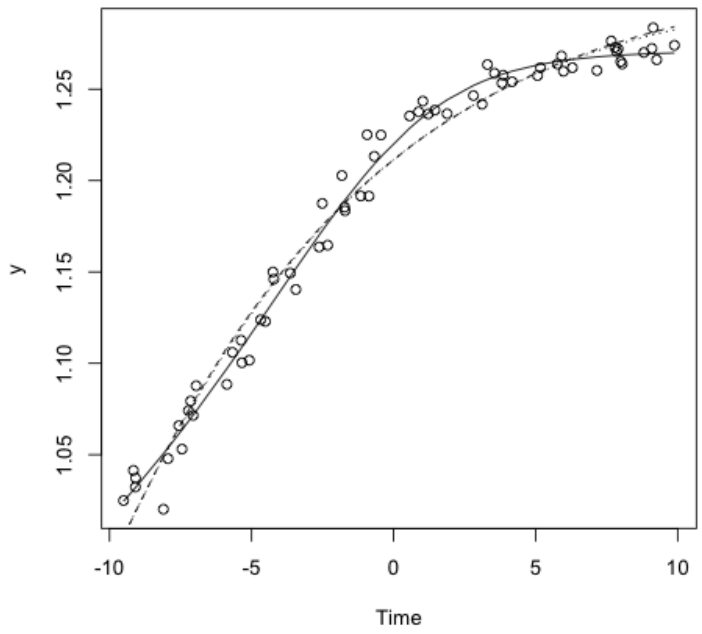



Figure 5.6: Growth curve dataset for $n = 65$. The lines are the maximum *a posteriori* estimates for the Richards (solid), Gompertz (dashed), and logistic (dotted) curves.

### 5.6.2 Results

After 10,000 Monte Carlo iterations, our algorithm computes the following posterior model probabilities: $\pi(R|\mathbf{y}) = 0.86$, $\pi(G|\mathbf{y}) = 0.02$, and $\pi(L|\mathbf{y}) = 0.12$. Table 5.1 shows summary statistics for the posterior samples of model parameters compared to the true values. We see that only two of the seven parameters are accurately estimated within a margin of error of twice the estimated standard error. This is due to the fact that our dataset in the design space (time) is limited to the interval (-10,10), which in turn makes it difficult to estimate all parameters. In fact, estimators in such a situation are not guaranteed to be consistent, meaning they do not converge to the true parameter values as sample size increases.

Figure 5.7 demonstrates that even though the parameter estimates may not be consistent, the model probabilities are. As the dataset size increases, the posterior model probabilities

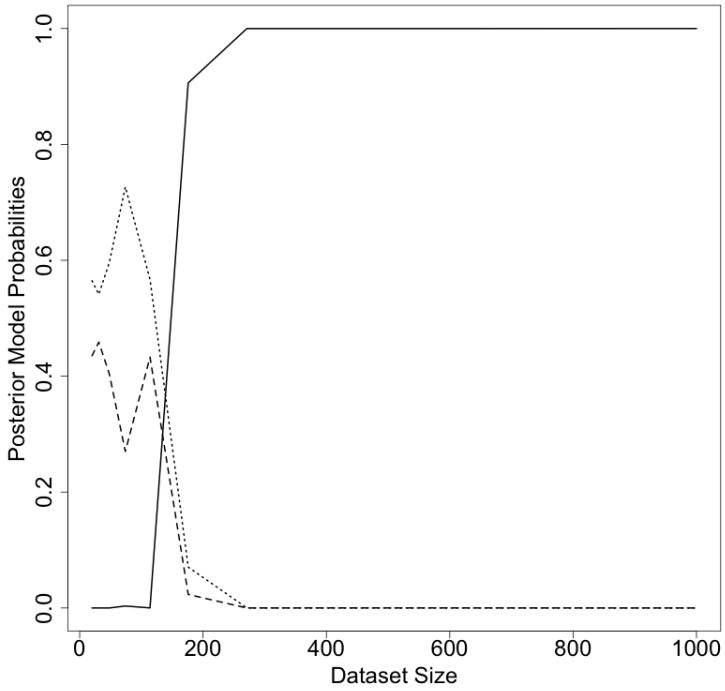



Figure 5.7: Estimated posterior model probabilities collapse on the true Richards model (solid line) as the dataset size increases.

collapse onto the true Richards curve model. As such, practical reversible jump demonstrates its ability to work on an application with a nonlinear model, where parameter relationships across model spaces are difficult to map out. Our method therefore provides an automated way to tune reversible jump in a wide variety of applications.

## 5.7 Discussion

We have developed a very generalizable methodology for performing reversible jump Markov chain Monte Carlo on a broad class of problems. It is our hope that exposing this method will increase the uptake of such a useful algorithm among practitioners. With reversible jump, one has the ability to select over several models in a principled way or create an ensemble

model. The former is generally useful for scientific inquiry, where the researcher is interested in learning which model best fits his or her data. The latter is an excellent practice for prediction and forecasting. Uncertainty over the model space can introduce an added layer of noise. Model averaging helps smooth over the model error, typically improving predictive performance.

Our method of performing reversible jump is particularly useful in cases when cross-model relationships among parameters are unclear or unknown. While these relationships can be derived in simple models such as linear models or with distributions where moments can be matched, the general problem is quite difficult. In such situations, the practitioner may be willing to sacrifice computer time in exchange for developing clever or sophisticated sampling schemes to jump across models. While practical reversible jump suffers when a normal distribution centered at the MAP estimate poorly details the target distributions, it works as a good first-order approximation to other more sophisticated techniques with very little or no tuning required.

## 5.8 Bibliography

Brooks, S. P., Giudici, P. & Roberts, G. O. (2003), 'Efficient Construction of Reversible Jump Markov Chain Monte Carlo Proposal Distributions', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **65**(1), 3–39.

Broyden, C. G. (1970), 'The Convergence of a Class of Double-Rank Minimization Algorithms', *IMA Journal of Applied Mathematics* **6**(1), 76–90.

Byrd, R. H., Lu, P., Nocedal, J. & Zhu, C. (1995), 'A Limited Memory Algorithm for Bound Constrained Optimization', *SIAM Journal on Scientific Computing* **16**(5), 1190–1208.

Carlin, B. P. & Chib, S. (1995), 'Bayesian Model Choice via Markov Chain Monte Carlo Methods', *Journal of the Royal Statistical Society. Series B (Methodological)* pp. 473–484.

Dellaportas, P., Forster, J. J. & Ntzoufras, I. (2002), 'On Bayesian Model and Variable Selection Using MCMC', *Statistics and Computing* **12**(1), 27–36.

Ehlers, R. S. & Brooks, S. P. (2008), 'Adaptive Proposal Construction for Reversible Jump MCMC', *Scandinavian Journal of Statistics* **35**(4), 677–690.

Fletcher, R. (1970), 'A New Approach to Variable Metric Algorithms', *The computer journal* **13**(3), 317–322.

Goldfarb, D. (1970), 'A Family of Variable-Metric Methods Derived by Variational Means', *Mathematics of computation* **24**(109), 23–26.

Green, P. J. (1995), 'Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination', *Biometrika* **82**(4), 711–732.

Green, P. J. & Hastie, D. I. (2009), 'Reversible Jump MCMC', *Genetics* **155**(3), 1391–1403.

Islam, T., Fiebig, D. G. & Meade, N. (2002), 'Modelling Multinational Telecommunications Demand with Limited Data', *International Journal of Forecasting* **18**(4), 605–624.

Knol, D. L. & ten Berge, J. M. (1989), 'Least-Squares Approximation of an Improper Correlation Matrix by a Proper One', *Psychometrika* **54**(1), 53–61.

Newton, M. A. & Hastie, D. I. (2006), 'Assessing Poisson Variation of Intestinal Tumour Multiplicity in Mice Carrying a Robertsonian Translocation', *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **55**(1), 123–138.

Richards, F. (1959), 'A Flexible Growth Function for Empirical Use', *Journal of Experimental Botany* **10**(2), 290–301.

Shanno, D. F. (1970), 'Conditioning of Quasi-Newton Methods for Function Minimization', *Mathematics of computation* **24**(111), 647–656.

Verhulst, P. F. (1845), 'Recherches Mathématiques sur la Loi d'Accroissement de la Population.', *Nouveaux Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles* **18**, 14–54.

# Chapter 6

# Reversible Jump Weighted Particle Tempering: Analyzing the Impacts of Clean Water Access on Child Development

**Abstract**

We analyze the causal impacts on child development of installing handpumps for clean water access in rural villages in the northeastern province of Nampula, Mozambique. Our longitudinal data are used to fit a nonlinear model, which describes the monthly log-growth rates of boys and girls younger than five years of age, accounting for sex, enumerator error, and censoring. We use a Bayesian hierarchical model with variable dimension due to the unknown number of knots in the P-splines. Inference is performed with a novel algorithm called reversible jump weighted particle tempering.

**Keywords.** Child development, improved water sources, P-splines, reversible jump MCMC, weighted particle tempering

## 6.1 Introduction

Access to improved water sources is rare in rural villages of sub-Saharan Africa. In fact, nearly half of all people that use unimproved drinking water sources live in sub-Saharan Africa (UNICEF & Organization 2015). The effects of limited clean water access are varied, from the potential spread of contagions to the additional daily time costs of collecting water from a household's preferred sources. The process of gathering water for daily activities alone has several second-order impacts, including taking time away from child rearing or school attendance, detracting from economically productive activities such as tending to crops, fishing, or selling products, or even cutting into relaxation time.

In this work, we are concerned with the impacts of access to clean water on child development. Specifically, we investigate how drinking potentially contaminated ground water, which may cause bouts of diarrhea and dysentery, could lead to the stunting of a child's growth or gestational development. Presumably, if a child is drinking clean water, he or she will be less susceptible to gastrointestinal disease and consequently experience a more normal development. However, very little evidence exists for this hypothesis. The review and meta-analysis of Fewtrell et al. (2005) finds little support for a reduction in risk of diarrheal disease due to the installation of clean water infrastructure. On the other hand, the opinion piece of Humphrey (2009) hypothesizes that regions with high fecal contamination may have a higher prevalence of sub-clinical gastrointestinal illness that could have a material impact on a child's growth and development. In other words, children may be experiencing gastrointestinal illness that is not severe enough to report, but may still be affecting their development. Other reasons for the lack of evidence for a link between improved water access and diarrhea may be that contagion may spread through other means besides water, such as through contact by hands or from water storage containers that are already contaminated despite a sterile water source (Wright et al. 2004).

The work herein examines child length data collected from two studies administered across two years in a rural province of Mozambique. Several nuances to the data could mask the

effects of handpump access on child development, motivating our use of a Bayesian hierarchical model to tease out these effects while simultaneously estimating model uncertainty. Specifically, we employ cubic P-splines to estimate the log-growth rates of children. Several research efforts have explored the use of components of the model, but to our knowledge, our specific construction has never been implemented before. An introduction to splines and their representation as a linear combination of basis functions, themselves constructed out of splines, can be found in de Boor (1978), de Boor (1986), and Hastie & Tibshirani (1990). A method for free-knot splines is found in Lindstrom (1999), and reversible jump MCMC algorithms are developed for their estimation in DiMatteo et al. (2001) and Lindstrom (2002). In Berry et al. (2002), the authors show how to apply splines to data with measurement error, providing a concise introduction to the Bayesian treatment of P-splines.

Estimation for this model is a nontrivial task, requiring exploration of nonlinear submodels with varying dimension and a moderate number of parameters. As such, a simple reversible jump MCMC procedure would be difficult to tune due to the lack of prior knowledge about the cross-model relationships among parameters. Specifying a proposal distribution and transformation function for model jumps is not straightforward. We develop an efficient reversible jump extension of weighted particle tempering, which allows within-chain updates to explore across the varying-dimension model space. With this novel procedure, we are able to fully characterize the posterior distribution of our model, which in turn allows us to perform Bayesian model averaging and estimate the uncertainty about our overall conclusions.

The remainder of this chapter is divided in the following way. Section 6.2 describes the relevant details of the Mozambique dataset. Section 6.3 gives the background and specifics of the cubic P-splines for log-growth rates model. Section 6.4 delves into the details of the reversible jump weighted particle tempering procedure used to simulate from the posterior. Section 6.5 presents the results and analysis. And section 6.6 concludes with a discussion and some closing remarks.

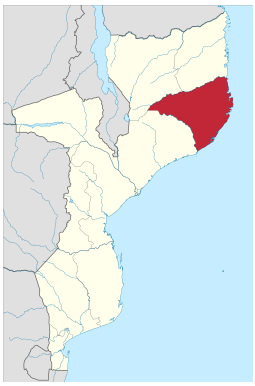


Figure 6.1: Nampula is a province located in the northeast of Mozambique.

## 6.2 The Mozambique Rural Water Supply Activity Dataset

The Millennium Challenge Corporation, an American foreign aid agency established by Congress in 2004, commissioned an independent team of researchers from Virginia Tech and Stanford to perform an evaluation of the impacts of installing handpumps for clean water access in rural villages within the Nampula province of Mozambique (figure 6.1). This impact evaluation (Hall et al. 2014), called the Rural Water Supply Activity (RWSA) study, sought to quantify the socioeconomic and health effects of handpumps on households and individuals who gain access to them. Typically, members of households that are located in these rural villages do not have a source of clean water. Instead, they fetch water from nearby rivers and *poços*, or wells (figure 6.2). Usually these sources are unprotected and susceptible to runoff contamination. Additionally, members of households that use these sources often spend hours per day collecting water, which could detract from more productive or relaxing activities.

In order to investigate these impacts, two studies were performed: a baseline study in 2011 near the end of completion of the first phase of handpumps, and a follow-up study in 2013 after the second phase of handpump installation was completed. The baseline study consisted of a survey of 1,579 households, while the follow-up contained a survey of 1,826 households. 1,147 of these households appeared in both studies, as an effort was made to geotag house-

An unprotected *poço*, or well

A handpump

Figure 6.2: Examples of water sources in the RWSA impact evaluation study.

holds in the baseline and revisit them in the follow-up. The heads of these households answered questionnaires and consented to having their children's lengths measured.

Child lengths were taken by laying a child on a graduated mat (figure 6.3) and noting the length. The measured children were removed from the mat and then replaced on the mat in order to get two separate measurements. In some cases, children were uncooperative for a second measurement, and so only one value was recorded. Some children were longer than the maximum mark on the mat of 99 cm, and so the data are right censored at 99 cm. A combined total of 1,347 children were measured over the studies, with 187 of these children appearing in both the baseline and follow-up. All children measured were between the ages of 0 and 64 months old.

The 19 enumerators had varying degrees of success in measuring children. Figure 6.4 demonstrates this by showing the absolute differences in the two measurements of an individual child on a given visit. The discrepancies between the two measurements can be attributed to measurement error. Some of the enumerators had no discrepancies, while others had

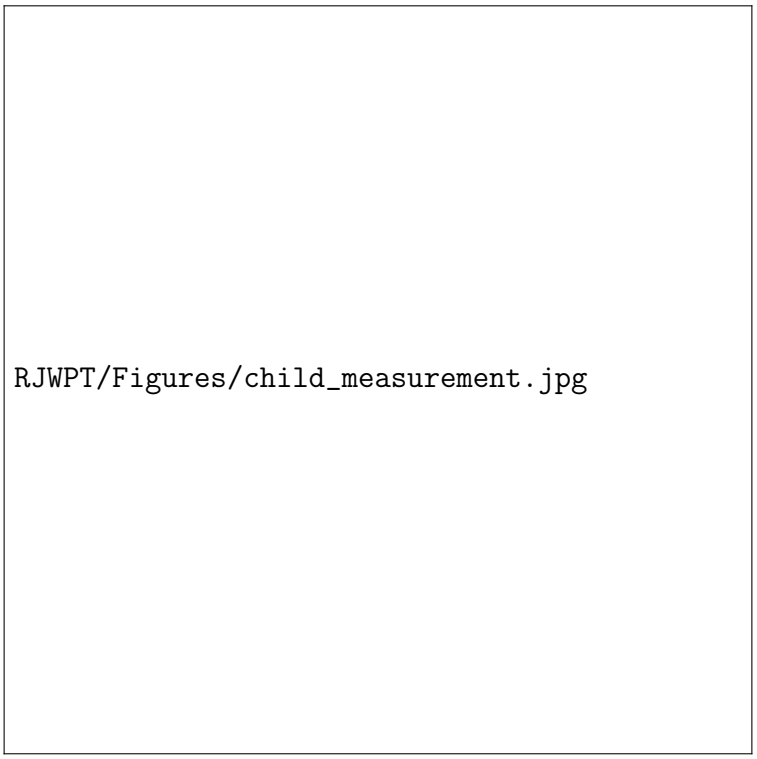



Figure 6.3: A child is measured by laying her on a graduated mat, with each centimeter demarcated from 10 cm to 99 cm.

some large discrepancies, suggesting enumerator error should be considered carefully in the analysis.

Our research interest is to quantify the differences in child development when children have or do not have access to handpumps. Figure 6.5 shows how without considering the nuances of the data, such as censoring, longitudinal repeated measurements, enumerator error, or sex, growth curves for children using and not using the handpumps appear to be indistinguishable.

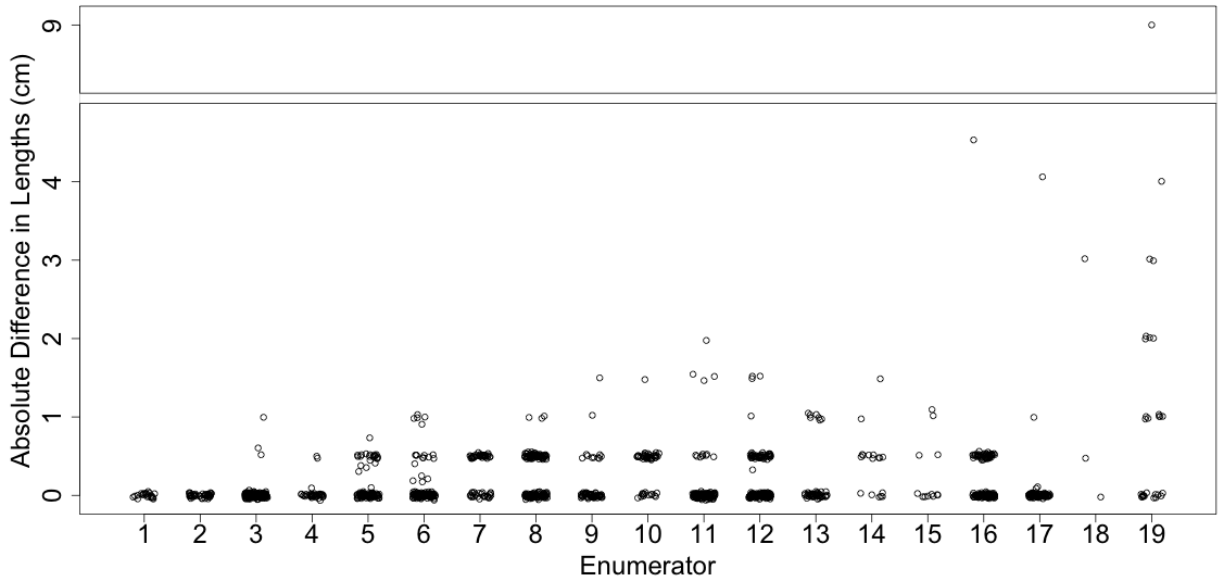



Figure 6.4: Absolute differences between the first and second measurement by enumerator. A random jitter has been added to the individual points for better presentation.

## 6.3 Bayesian Cubic P-Splines for Log-Growth Rates

### 6.3.1 Cubic P-Splines Review

Cubic P-splines are a flexible nonlinear regression technique that we utilize to estimate the log-growth rates of children. We begin with a spline order $q = 3$ for cubic splines and a set of knots, $\nu_1 < \ldots < \nu_k$, with unknown cardinality $k$. Given these knots and the spline order, we can build a set of $n = k - q - 1 \in \mathscr{N}$ linearly independent basis functions $\mathbf{B}_q(x) = \{B_{1,q}(x), \ldots, B_{n,q}(x)\}$ via the following recursion:

$$
\begin{aligned}
B_{i,1}(x) &:= \begin{cases} 1, & \text{if } \nu_i \leq x < \nu_{i+1}, \\ 0, & \text{otherwise,} \end{cases} \\
B_{i,j} &:= \omega_{ij} B_{i,j-1} + (1 - \omega_{i+1,j}) B_{i+1,j-1}, \\
\omega_{ij}(x) &:= \begin{cases} \frac{x - \nu_i}{\nu_{i+j-1} - \nu_i}, & \text{if } \nu_i \neq \nu_{i+j-1}, \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}
$$

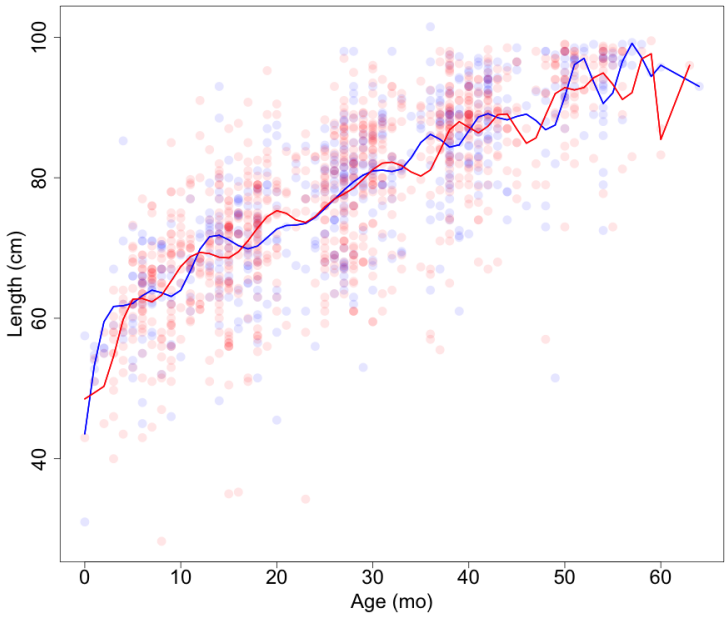



Figure 6.5: A naive implementation of natural cubic splines on children whose households indicated they used a handpump (blue) and did not use a handpump (red) when the measurement was taken.

It is well known that splines can be represented as a linear combination of the above basis functions, hence the term *B-splines*. When the knot locations and number are treated as unknown parameters, this class of models is referred to as *free knot splines*. Lastly, when the number of knots $k$ is less than the number of data points, $N$, this class of models is referred to as *P-splines*, or *penalized splines*. Cubic P-splines combine these features to preserve the simplicity of B-splines, flexibility of free-knot splines, and parsimony of P-splines.

### 6.3.2 Model

We fit the above nonlinear model to the log-growth rates, which ensures that the estimated growth rates are positive. This model can be thought of as a continuous process, which we approximate by discretizing at the monthly level, since we only know every child's age up

to the month of measurement. Let $Y_{i,m,j}$ be the $j$th length measurement of the $i$th child on month $m$ of his or her life. Each child is measured by an enumerator $\eta(i,m) \in \{1,\ldots,19\}$ at most twice within the same visit to the household, so $j \in \{1,2\}$. Some children were measured in both the baseline study and two years later in the follow-up, while others were only measured in one of the two. We use the following model for these measurements:

$$\begin{aligned} Y_{i,m,j} &= L(i,m) + \epsilon_i + \epsilon_{i,m} + \epsilon_{i,m,j}, & (6.1)\\ L(i,m) &= \sum_{a=0}^{m} \left\{\gamma_{i,a}\ell_{hp}(s_i,a) + (1-\gamma_{i,a})\ell_{nhp}(s_i,a)\right\}. & (6.2) \end{aligned}$$

In the formulation above, $\epsilon_i$ stands for the child's deviation from the average length, which has variance $\sigma_c^2$; $\epsilon_{i,m}$ is the measurement error corresponding to the date of the measurement, which has variance $\sigma_d^2$ and also induces correlation for multiple measurements taken of a child on the same visit to the household; and $\epsilon_{i,m,j}$ is the enumerator error with variance $\sigma^2_{\eta(i,m)}$. Hence, each enumerator has an associated variance parameter $\sigma_\eta^2$ governing his or her precision in measuring children. All error terms are normal random variables with zero mean.

The unknown growth curve function $L(i,m)$ is modeled as the accumulation of a child's growth over his or her lifetime, with separate growth rate functions $\ell_{hp}(s_i,a)$ and $\ell_{nhp}(s_i,a)$, which are the amount a child of sex $s_i \in \{m,f\}$ grows at age $a$ if the child's household uses $(hp)$ or does not use $(nhp)$ a handpump. Finally, $\gamma_{i,a}$ is an indicator function for whether or not the child used a handpump in month $a$ of his or her life.

We model the log of the growth rates $\ell_\tau$, $\tau \in \{hp, nhp\}$ with P-splines:

$$\log(\ell_\tau(s_i,a)) = \sum_{j=1}^{n_{s_i,\tau}} \beta^\tau_{j,s_i} B^\tau_{j,s_i}(a). \quad (6.3)$$

By modeling the log-growth rates rather than directly estimating the growth rates, we preserve monotonicity in the growth curve. That is, when these log-growth rates are exponentiated, they become positive, and therefore the model does not permit a decrease in size. This

monotonicity condition cannot be imposed in a vanilla implementation of cubic splines, as can be seen in figure 6.5. Also, a nonlinear model is appropriate because it is well known that children grow faster at younger ages, and that there may be several growth spurts throughout a child's life.

### 6.3.3 Prior Specification

The World Health Organization (WHO) provides standard growth charts for children (de Onis & Onyango 2008), along with thresholds for determining whether or not a child is stunted. These standards were developed through the WHO Multicentre Growth Reference Study (de Onis et al. 2004), a six-year research effort to define benchmarks for child development. We use this study through our prior to inform our own investigation. Specifically, we place a translated Poisson prior on the number of bases, $n - 1 \sim Poisson(\lambda)$. Knots are evenly spaced over the interval (-8,70) months. Then, conditional on the number of knots and knot locations, which uniquely determines the B-splines, we place a normal prior on the basis parameters $\boldsymbol{\beta}$ such that they are centered on the parameters that would minimize the squared error for fitting our model to the WHO growth standard. Let $\mathbf{z}_1$ and $\mathbf{z}_{99}$ be the vectors of the first and 99th percentiles, respectively, of the WHO growth curve standards for the first 61 months of a child's life (ages zero months to 60 months). Define the following:

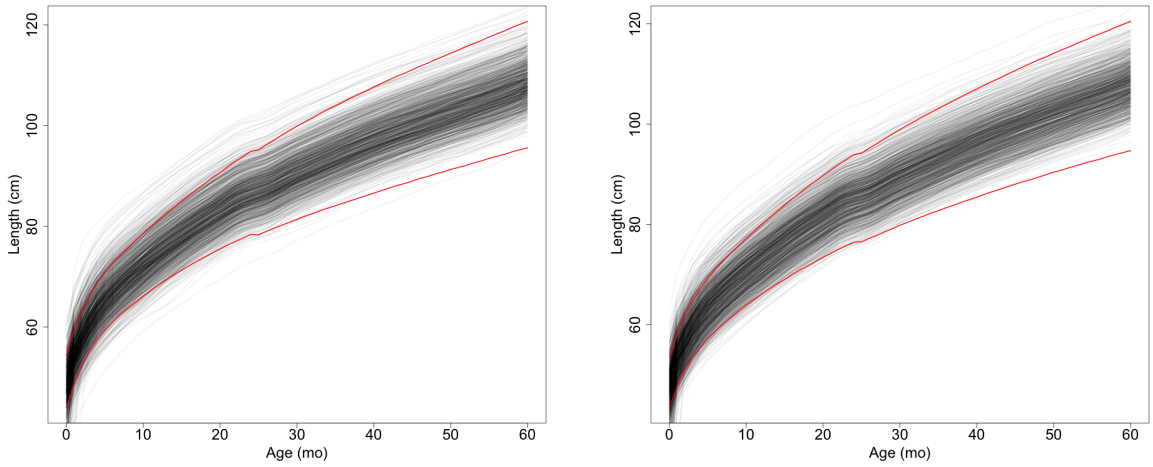
$$\begin{aligned} SSE(\boldsymbol{\beta}, \boldsymbol{\nu}) &:= \sum_{m=0}^{60} \left\{ \left( z_{1,m} - \sum_{i=0}^{m} e^{\sum_{j=1}^{n} \beta_j B_j(i)} \right)^2 + \left( z_{99,m} - \sum_{i=0}^{m} e^{\sum_{j=1}^{n} \beta_j B_j(i)} \right)^2 \right\}, \\ \boldsymbol{\beta}^* &:= \arg\min_{\boldsymbol{\beta}} SSE(\boldsymbol{\beta}, \boldsymbol{\nu}). \end{aligned}$$

Then our prior on $\boldsymbol{\beta}$ conditional on the number of knots and their locations is

$$\boldsymbol{\beta} | \nu_1, \ldots, \nu_k \sim N(\boldsymbol{\beta}^*, \sigma_\beta^2 I).$$

Finally, we use inverse gamma priors on all variance parameters.

Table 6.1 summarizes these priors and their associated hyperparameters. Figure 6.6 demonstrates how the knot prior and the prior on the basis coefficients induce a prior on growth




Boys Girls

Figure 6.6: 1,000 samples from the prior distribution of growth curves for boys (left) and girls (right). 71.4% of the curves for boys and 70.3% of the curves for girls landed entirely between the WHO's first and 99th percentiles (red).

curves, and how this growth curve distribution matches up nicely with the percentiles of the WHO growth standards.

### 6.3.4 Handpump Usage Indicators

Households acquired access to handpumps at different moments in the lives of children in the RWSA dataset. Perhaps the most informative measurements in the dataset for how handpump access affects child growth are the measurements corresponding to children who did not have access to a handpump in the baseline, then did have access when they were revisited in the follow-up or *vice versa*. Our model tracks this important factor through the handpump usage indicator $\gamma_{i,a}$, which is one if child $i$ used a handpump when the child was $a$ months old, and zero otherwise.

These indicators are not directly observed, nor are they identifiable in the model. However, the RWSA dataset provides some information as to what these should be. There are three

Table 6.1: Prior distributions and hyperparameters for the model.

| Parameter | Prior | Hyperparameters |
|---|---|---|
| $n-1$ | $Poisson(\lambda)$ | $\lambda = 26$ |
| $\boldsymbol{\beta}$ | $Normal(\boldsymbol{\beta}^*, \sigma_\beta^2 I)$ | $\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} SSE(\boldsymbol{\beta}, \boldsymbol{\nu})$, $\sigma_\beta = 0.1$ |
| $(1/\sigma_d)^2$ | $Gamma(\alpha_d, \beta_d)$ | $\alpha_d = 1, \beta_d = 1/2$ |
| $(1/\sigma_c)^2$ | $Gamma(\alpha_c, \beta_c)$ | $\alpha_c = 1, \beta_c = 1/7$ |
| $(1/\sigma_\eta)^2$ | $Gamma(\alpha_\eta, \beta_\eta)$ | $\alpha_\eta = 1, \beta_\eta = 1$ |

community types in the dataset, corresponding to when the handpumps were installed. The first community type had its handpump installed prior to the baseline study; these communities are referred to as *Phase I treatment* communities. The second type had its handpump installed between the baseline and follow-up, referred to as *Phase II treatment* communities. The final community type is the *comparison* community, which is the group of communities that never received a handpump.

Households indicated on a questionnaire that they did or did not use the handpump. In some cases, households in treatment communities did not use the handpump due to monetary or political reasons or inconvenience. In rare cases, households in comparison communities had access to a remote handpump. The handpump usage indicators for children in our dataset take these factors into account. Figure 6.7 summarizes the logic for how these indicators are computed for each child.

### 6.3.5 Right-Censored Observations

The proper treatment of a threshold-censored datum is to compute its contribution to the likelihood as the integral of the likelihood over the censored region. Each child in the dataset has a vector of measurements whose dimension can range from one to four. There

are children in the dataset for whom some measurements are censored and others are not. Here we discuss how to generally handle a correlated vector of normal variates where a subset of the variates are censored. Consider the general case $\mathbf{y} \sim N(\boldsymbol{\mu}, \Sigma)$, with the first $m$ observations uncensored, leaving the remaining $n - m$ observations censored. Without loss of generality, consider only *right-censored* cases, where it is only known that $y_i > c$, but it is unknown by how much. We write $\mathbf{y}^\top = \left(\mathbf{y}_u^\top, \mathbf{y}_c^\top\right)$, $\boldsymbol{\mu}^\top = \left(\boldsymbol{\mu}_u^\top, \boldsymbol{\mu}_c^\top\right)$, $\Sigma_{u,u} = \Sigma_{1:m,1:m}$, $\Sigma_{u,c} = \Sigma_{1:m,(m+1):n}$, and $\Sigma_{c,c} = \Sigma_{(m+1):n,(m+1):n}$ to convey that the first $m$ observations are uncensored ($u$), and the remaining $n - m$ are censored ($c$). Then the likelihood can be written as

$$L(\boldsymbol{\mu}, \Sigma | \mathbf{y}_u, y_{m+1} > c, \ldots, y_n > c) = L(\boldsymbol{\mu}_u, \Sigma_{u,u} | \mathbf{y}_u) L(\boldsymbol{\mu}^*, \Sigma^* | y_{m+1} > c, \ldots, y_n > c),$$

where

$$\begin{aligned} \boldsymbol{\mu}^* &= \boldsymbol{\mu}_c + \Sigma_{c,u}\Sigma_{u,u}^{-1}(\mathbf{y}_u - \boldsymbol{\mu}_u), \text{ and} \\ \Sigma^* &= \Sigma_{c,c} - \Sigma_{u,c}^\top \Sigma_{u,u}^{-1} \Sigma_{u,c}. \end{aligned}$$

Then $L(\boldsymbol{\mu}_u, \Sigma_{u,u} | \mathbf{y}_u)$ is the usual multivariate normal density, and

$$L(\boldsymbol{\mu}^*, \Sigma^* | y_{m+1} > c, \ldots, y_n > c) = \int_{z_1 > c} \cdots \int_{z_{n-m} > c} L(\boldsymbol{\mu}^*, \Sigma^* | z_1, \ldots, z_{n-m}) dz_1 \ldots dz_{n-m}. \tag{6.4}$$

The integral in equation 6.4 is computed using the Genz-Bretz algorithm of Genz (1992) and Miwa et al. (2003). While our application only requires the computation of these integrals for a normal likelihood, other algorithms exist for this computation with a t-distribution (see Genz & Bretz (2002) or Genz (2004), for example).

## 6.4 Reversible Jump Weighted Particle Tempering

Reversible jump weighted particle tempering is an extension of weighted particle tempering to cross-model updates. To achieve this, we define a reversible jump move followed by a

sequence of Metropolis-within-Gibbs moves for all within-chain updates. We allow for a swap between the mother chain and any one of its $p$ underlying chains, as is usually the case for weighted particle tempering. Since it is not clear how to propose to a new model space, we continuously perform an optimization of the posterior conditional on the model, so that proposals are generated near the current optimum with a normal distribution.

At iteration $t \in \{0, \ldots, N\}$ for the $i$th chain, where $i \in \{0, \ldots, p\}$, let

$$\boldsymbol{\theta}_{s,\tau}^{(i,t)} = \left\{k_{s,\tau}, \nu_{s,\tau,1}, \ldots, \nu_{s,\tau,k_{s,\tau}}, \boldsymbol{\beta}_{s,\tau}\right\}^{(i,t)}$$

be the set of curve parameters, including knot number, knot locations, and basis coefficients for the log-growth rate of children of sex $s \in \{m, f\}$ and handpump use $\tau \in \{hp, nhp\}$. Let $\boldsymbol{\phi}^{(i,t)} = \{\sigma_d, \sigma_c, \sigma_1, \ldots, \sigma_{19}\}^{(i,t)}$ be the set of standard deviation parameters. Finally, let

$$\Theta^{(i,t)} = \left\{\boldsymbol{\theta}_{m,hp}^{(i,t)}, \boldsymbol{\theta}_{m,nhp}^{(i,t)}, \boldsymbol{\theta}_{f,hp}^{(i,t)}, \boldsymbol{\theta}_{f,nhp}^{(i,t)}, \boldsymbol{\phi}^{(i,t)}\right\}$$

be the full set of parameters. In an effort to avoid cumbersome notation, we drop the superscript $(i, t)$ where it is implied from the context that the same procedure is repeated for all chains $i$ and iterations $t$. We also use the convention that $\Theta^{(0,t)}$ refers to the mother particle at iteration $t$ – samples of which are retained and used as draws from the posterior distribution, $\pi(\Theta)$.

Recall that $\mathscr{N}$ is the set of basis numbers we consider for each of the four log-growth rate curves. Our full model has $\mathscr{N} = \mathbb{N}$, the natural numbers. However, even with finite cardinality, $|\mathscr{N}| < \infty$, we would be required to explore a model space of size $M = |\mathscr{N}|^4$ since each of the four log-growth curves can have a different number of bases. In order to make the problem more tractable, we choose to truncate the basis number set to $\mathscr{N} = \{26, 31\}$, giving $M = 16$ possible models over which to average. We consider these models since some exploratory analyses suggest 30 and 35 knots are within a reasonable range for fitting splines to our dataset, and because increasing the number of models to consider will slow the performance of the optimization routine.

While other reversible jump moves could be layered into a weighted particle tempering al-

**Algorithm 6.1** Reversible Jump Weighted Particle Tempering

Initialize $\Theta^{(i,0)}$ for $i = 0, \ldots, p$

**for** $t = 1, \ldots, N$ **do**

Set $w_i \propto \pi^{\delta}(\Theta^{(i,t-1)})$

$\gamma \leftarrow i$ with probability $w_i$

$\alpha \leftarrow 1 \wedge \frac{\pi^{1-\nu-\delta}(\Theta^{(\gamma,t-1)})\left[\pi^{\delta}(\Theta^{(\gamma,t-1)})+\sum_{j\neq\gamma}\pi^{\delta}(\Theta^{(j,t-1)})\right]}{\pi^{1-\nu-\delta}(\Theta^{(0,t-1)})\left[\pi^{\delta}(\Theta^{(0,t-1)})+\sum_{j\neq\gamma}\pi^{\delta}(\Theta^{(j,t-1)})\right]}$

**if** $u \sim Unif(0,1) < \alpha$ **then**

$\Theta^{(\gamma,t-1)} \leftarrow \Theta^{(0,t-1)}$

$\Theta^{(0,t-1)} \leftarrow \Theta^{(\gamma,t-1)}$

**end if**

**for** i = 0,. . . ,p **do**

$\Theta^{(i,t)} \leftarrow RJMCMC(\Theta^{(i,t-1)}, \pi_{\nu_i})$

**end for**

**end for**

gorithm, our implementation uses an adaptive independence sampler. Each of the $M = 16$ candidate models is optimized by running 10,000 iterations of a Metropolis-within-Gibbs algorithm and choosing the set of sampled parameters that maximizes the conditional posterior. Call this parameter setting $\Theta^*_m$ for model $m$. The optimization is performed in parallel prior to running the reversible jump weighted particle tempering algorithm. At runtime, a model jump is performed by randomly selecting a model and generating a proposal from a normal distribution centered at the optimum for the proposed model with a tunable standard deviation. We leave the details of our implementation for reversible jump weighted particle tempering to the appendix. However, algorithm 6.1 gives a sketch of the overall process. We abbreviate the within-chain update with the function $RJMCMC(\Theta, \pi)$, which is shorthand for the output from a single iteration of reversible jump, initialized at $\Theta$ and targeting $\pi$, that could potentially perform a model jump or within-model move (or both).

Our implementation of reversible jump weighted particle tempering uses $p = 10$ underlying

chains and a tempering exponent of $\nu = 0.9$, which we find to be a good setting for efficient mixing. We run the algorithm for $N = 10{,}000$ iterations. When a new sample is proposed, it is generated from a normal distribution either centered at the MAP estimate (for the reversible jump move in step 3a in the appendix) or centered at the current position of the parameter (as in the set of Metropolis-within-Gibbs moves in step 3b in the appendix). These normal proposal distributions have tunable standard deviations. We use the same proposal standard deviations for the different move types, but this is not a requirement. For all basis parameters $\boldsymbol{\beta}$, we use a proposal standard deviation of 0.01 in the mother chain and 0.04 in the underlying tempered chains. For the enumerator standard deviations $\sigma_\eta$, we use 0.002 in the untempered chain and 0.004 in the tempered chain. For the other standard deviation terms $\sigma_d$ and $\sigma_c$, we use 0.1 in the untempered chain and 0.3 in the tempered chains. With these tunable settings, weighted particle tempering swaps are accepted at a rate of 36.9%, which allows the algorithm to occasionally make larger within-model or cross-model jumps than with the within-chain updates.

## 6.5 Results

Looking at the posterior distribution of growth curves, we find some evidence that not having access to clean water may lead to shorter child lengths at birth. The effect seems to be stronger for boys and statistically insignificant for girls. We hypothesize this to be a result of shorter gestational periods, as drinking potentially contaminated water and contracting the associated gastrointestinal diseases may lead to preterm births. There is evidence in the medical literature of a higher prevalence of preterm births among males than females, as in Zeitlin et al. (2002) and Ingemarsson (2003). While the causes of preterm birth are complex and not yet well understood, there is evidence that certain types of infection are risk factors (Goldenberg et al. 2008).

The MCMC produced samples across three models. The first model has $k = 30$ knots for

all four log-growth rate curves. This model has a posterior probability of 0.28. The second model uses 35 knots for the log-growth rate curve of male handpump users and 30 knots for the rest of the curves, with posterior probability 0.04. Lastly, the third model uses 35 knots for both male log-growth rate curves and 30 knots for both female log-growth rate curves, with posterior probability 0.68. Inferences about growth rate curves were made by averaging over these models.

Figure 6.8 shows for both boys and girls the posterior differences in birth lengths between children born with and without access to a handpump. The estimated posterior mean difference in birth lengths for boys is $E[\ell_{hp}(m,0) - \ell_{nhp}(m,0)] = 2.27$ cm with a 95% credible interval of $[0.23, 4.03]$. For girls, the posterior mean difference is estimated to be $E[\ell_{hp}(f,0) - \ell_{nhp}(f,0)] = 0.27$ cm with a 95% credible interval of $[-2.12, 2.30]$. Additionally, we also estimate that there is only a posterior probability of $P(\ell_{hp}(m,0) \leq \ell_{nhp}(m,0)) = 0.0117$ that the mean birth length for boys born to mothers who use handpump water is less than that of boys born to mothers who do not. A comparison of the hypotheses $\{H_0 : \ell_{hp}(m,0) \leq \ell_{nhp}(m,0)\}$ and $\{H_1 : \ell_{hp}(m,0) > \ell_{nhp}(m,0)\}$ would therefore yield a Bayes factor of 84.5 in favor of $H_1$.

Posterior samples of the log-growth rates can be seen in figure 6.9 for children between 1 month and 60 months of age. Careful inspection of these curves reveals that there are subtle differences between sexes. However, we are more interested in the differences between growth rates for handpump users and nonusers, which is depicted in figure 6.10. Although our model has the ability to estimate monthly differences, no significant difference was found between growth rates after birth for handpump users and nonusers.

The posterior distributions of the standard deviation parameters can be seen in figures 6.11 and 6.12. Figure 6.11 shows the enumerator standard deviations, which should be compared to the raw data in figure 6.4. This comparison reveals that the model is correctly identifying enumerator 19 as the highest variance enumerator and that we are properly controlling for enumerator error. Figure 6.12 depicts the negative posterior correlation between $\sigma_d$ and

$\sigma_c$, which is interpretable as a tradeoff between a child's lifetime deviation from the average growth curve (as in, some people are shorter than average and others are taller than average) and the spread of measurements around the average growth curve by age. In other words, the data can be explained by either a larger spread around the growth curves or by a larger deviation from the average for individual children.

### 6.5.1 Model Validation

After taking the *maximum a posteriori* estimates of all model parameters $\hat{\Theta}_{MAP}$, we use these to compute the z-scores of all observations and compare them with theoretical quantiles for a standard normal distribution. Specifically, if $\mathbf{y}_i$ is the vector of observations for child $i$, we standardize with the transformation $\mathbf{z}_i := \hat{\Sigma}_{i,MAP}^{-1/2}(\mathbf{y}_i - \hat{\mathbf{L}}_{i,MAP})$, where $\hat{\mathbf{L}}_{i,MAP}$ is the MAP estimate of the mean length vector and $\hat{\Sigma}_{i,MAP}$ is that of the covariance matrix. Figure 6.13 shows the resulting quantile-quantile plot. We see that the data are slightly more negatively skewed than the model predicts, and that there is a presence of a few outliers. The bump in the center of the plot suggests there may be unidentified clusters in the data.

Lastly, we compute posterior predictive intervals by taking the average over the handpump use indicators for all children $\bar{\gamma}_a := 1/1344 \sum_{i=1}^{1344} \gamma_{i,a}$. To simulate from the posterior predictive distribution, we create $N = 10{,}000$ draws at each age of a potential new child's length measurement, then compute the monthly 95% credible intervals. This process is depicted in figure 6.14, showing that the actual data fall outside of the intervals near the nominal rate: 6.1% for boys and 5.7% for girls.

While the model seems to be a good fit by inspecting the q-q plot and posterior predictive distributions, it can be improved substantially by collecting better data on when a child used and did not use a handpump throughout his or her life. The model assumes we have this information, ignoring the uncertainty in how the handpump use indicators were selected. There is a chance that monthly developmental differences will in fact be detected if these indicators are measured more accurately.

## 6.6 Discussion

In this work, we have shown that reversible jump weighted particle tempering is capable of simulating from the posterior distribution of a variable dimensional, nonlinear model. The model captures the nuances of the data that stem from repeated measurements over time with censoring and nonlinear monotonic growth rates while accounting for enumerator error and sex differences and while incorporating prior information about growth curves from the WHO Multicentre Growth Reference Study. Our results point to a potential impact of handpump access on the birth lengths of boys – with an estimated difference of 2.27 cm – possibly due to premature birth. Further inquiry is strongly recommended, as water usage activity ought to be tracked at a more granular level to verify these findings.

In fitting our hierarchical model, reversible jump weighted particle tempering has demonstrated itself to be a useful tool to the statistics community. More theoretical work is needed to identify its limitations, but the algorithm's future is promising.

# Appendix

The details for our implementation of reversible jump weighted particle tempering proceed as follows:

1. Initialize all $p+1$ chains by randomly selecting $m$ uniformly from the model indices $1, \ldots, M$, setting parameters equal to their model's MAP estimates $\Theta^*_m$, and setting $t = 0$.

2. Perform a swap between the mother chain and one of the underlying chains:

   (a) Let $w_i \propto \pi(\Theta^{(i,t)})$ and sample the index $\gamma = i$ with probability $w_i$.

   (b) Let $\alpha = 1 \wedge \frac{\pi(\Theta^{(1,t-1)})^\nu \left[\pi(\Theta^{(\gamma,t)}) + \sum_{i \neq \gamma} \pi(\Theta^{(i,t)})\right]}{\pi(\Theta^{(\gamma,t)})^\nu \left[\pi(\Theta^{(1,t-1)}) + \sum_{i \neq \gamma} \pi(\Theta^{(i,t)})\right]}$.

   (c) Set $\Theta^{(1,t)} = \Theta^{(\gamma,t)}$ with probability $\alpha$. Otherwise set $\Theta^{(1,t)} = \Theta^{(1,t-1)}$.

3. For all $p+1$ chains, do the following:

   (a) Perform a reversible jump move:

      i. The chain's model index is currently $m \in 1, \ldots, M$.

      ii. Randomly select a proposal model index $m' \neq m$ with equal probability.

      iii. Generate a proposal $\Theta' \sim N(\Theta^*_{m'}, \Sigma_{m'})$, where $\Theta^*_{m'}$ is the set of parameters that yields the highest posterior conditioned on the proposed model $m'$ of any sample or proposal visited up to this point.

      iv. Let $\alpha = 1 \wedge \frac{\pi(\Theta')^{\nu_i} g(\Theta|\Theta^*_m, \Sigma_m)}{\pi(\Theta)^{\nu_i} g(\Theta'|\Theta^*_{m'}, \Sigma_{m'})}$, where $g(\Theta|\Theta^*_m, \Sigma_m)$ is the density for the normal proposal distribution and $\nu_i$ is one if $i = 1$ and $\nu$ otherwise.

      v. Accept the proposal with probability $\alpha$ by setting $m = m'$ and $\Theta = \Theta'$ and otherwise leaving $m$ and $\Theta$ unchanged.

   (b) Perform a Metropolis-within-Gibbs update of all parameters. For the underlying chains, this update targets the tempered target distribution $\pi(\Theta)^{\nu}$, while for the mother chain it targets the untempered target $\pi(\Theta)$. We execute this with block updates performed in the following order:

      i. Update $\boldsymbol{\beta}_{m,hp}$ and $\boldsymbol{\beta}_{m,nhp}$.

      ii. Update $\boldsymbol{\beta}_{f,hp}$ and $\boldsymbol{\beta}_{f,nhp}$.

      iii. Update $\sigma_\eta$ for $\eta = 1, \ldots, 19$.

      iv. Update $\sigma_d$ and $\sigma_c$.

4. Adapt the independence sampler for reversible jump. Specifically, if any proposal $\Theta'_m$ was generated for which $\pi(\Theta'_m) > \pi(\Theta^*_m)$, set $\Theta^*_m = \Theta'_m$.

5. If $t < N$, increment $t$ and return to step 2.

## 6.7 Bibliography

Berry, S. M., Carroll, R. J. & Ruppert, D. (2002), 'Bayesian Smoothing and Regression Splines for Measurement Error Problems', *Journal of the American Statistical Association* **97**(457), 160–169.

de Boor, C. (1978), *A Practical Guide to Splines*, Springer-Verlag New York.

de Boor, C. (1986), B(asic)-Spline Basics., Technical report, DTIC Document.

de Onis, M., Garza, C., Victora, C. G., Onyango, A. W., Frongillo, E. A. & Martines, J. (2004), 'The WHO Multicentre Growth Reference Study: Planning, Study Design, and Methodology', *Food and Nutrition Bulletin* **25**(1), S15–S26.

de Onis, M. & Onyango, A. W. (2008), 'WHO Child Growth Standards', *The Lancet* **371**(9608), 204–204.

DiMatteo, I., Genovese, C. R. & Kass, R. E. (2001), 'Bayesian Curve-Fitting with Free-Knot Splines', *Biometrika* **88**(4), 1055–1071.

Fewtrell, L., Kaufmann, R. B., Kay, D., Enanoria, W., Haller, L. & Colford, J. M. (2005), 'Water, Sanitation, and Hygiene Interventions to Reduce Diarrhoea in Less Developed Countries: a Systematic Review and Meta-Analysis', *The Lancet Infectious Diseases* **5**(1), 42–52.

Genz, A. (1992), 'Numerical Computation of Multivariate Normal Probabilities', *Journal of Computational and Graphical Statistics* **1**(2), 141–149.

Genz, A. (2004), 'Numerical Computation of Rectangular Bivariate and Trivariate Normal and t Probabilities', *Statistics and Computing* **14**(3), 251–260.

Genz, A. & Bretz, F. (2002), 'Comparison of Methods for the Computation of Multivariate t Probabilities', *Journal of Computational and Graphical Statistics* **11**(4), 950–971.

Goldenberg, R. L., Culhane, J. F., Iams, J. D. & Romero, R. (2008), 'Epidemiology and Causes of Preterm Birth', *The Lancet* **371**(9606), 75–84.

Hall, R., Davis, J., Van Houweling, E., Vance, E., Carzolio, M., Seiss, M. & Russel, K. (2014), 'Impact Evaluation of the Mozambique Rural Water Supply Activity', *Submitted to Millennium Challenge Corporation* .

Hastie, T. J. & Tibshirani, R. J. (1990), *Generalized Additive Models*, CRC Press.

Humphrey, J. H. (2009), 'Child Undernutrition, Tropical Enteropathy, Toilets, and Handwashing', *The Lancet* **374**(9694), 1032–1035.

Ingemarsson, I. (2003), 'Gender Aspects of Preterm Birth', *BJOG: An International Journal of Obstetrics & Gynaecology* **110**(s20), 34–38.

Lindstrom, M. J. (1999), 'Penalized Estimation of Free-Knot Splines', *Journal of Computational and Graphical Statistics* **8**(2), 333–352.

Lindstrom, M. J. (2002), 'Bayesian Estimation of Free-knot Splines Using Reversible Jumps', *Computational Statistics & Data Analysis* **41**(2), 255–269.

Miwa, T., Hayter, A. J. & Kuriki, S. (2003), 'The Evaluation of General Non-Centred Orthant Probabilities', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **65**(1), 223–234.

UNICEF & Organization, W. H. (2015), *Progress on Sanitation and Drinking Water: 2015 Update*, World Health Organization.

Wright, J., Gundry, S. & Conroy, R. (2004), 'Household Drinking Water in Developing Countries: a Systematic Review of Microbiological Contamination Between Source and Point-of-Use', *Tropical Medicine & International Health* **9**(1), 106–117.

Zeitlin, J., Saurel-Cubizolles, M.-J., de Mouzon, J., Rivera, L., Ancel, P.-Y., Blondel, B. & Kaminski, M. (2002), 'Fetal Sex and Preterm Birth: Are Males at Greater Risk?', *Human Reproduction* **17**(10), 2762–2768.

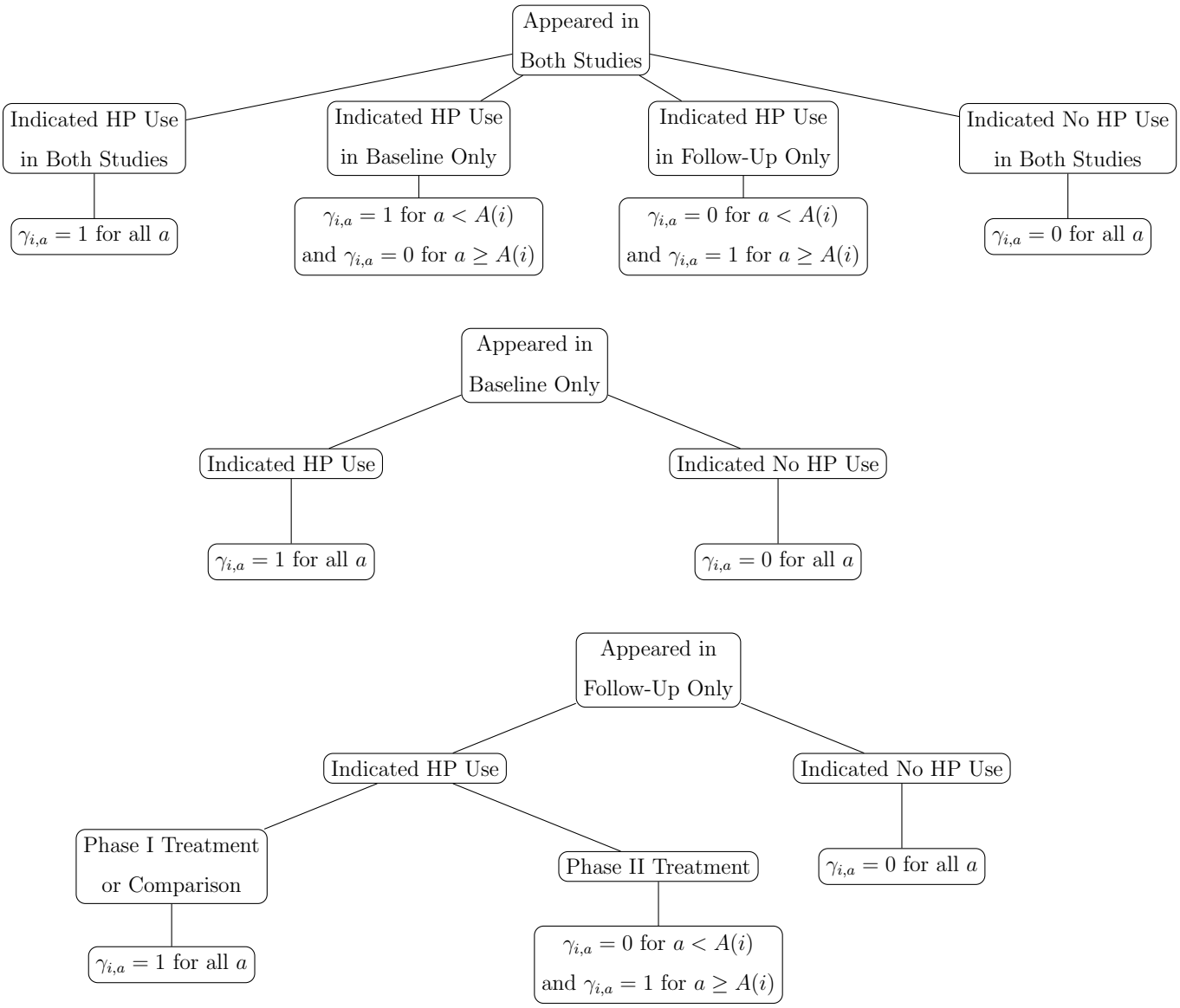



Figure 6.7: Handpump usage indicators for children appearing in both studies (top), only the baseline (middle), or only the follow-up (bottom). $A(i)$ is the age of child $i$ halfway between the dates of the baseline and follow-up studies.

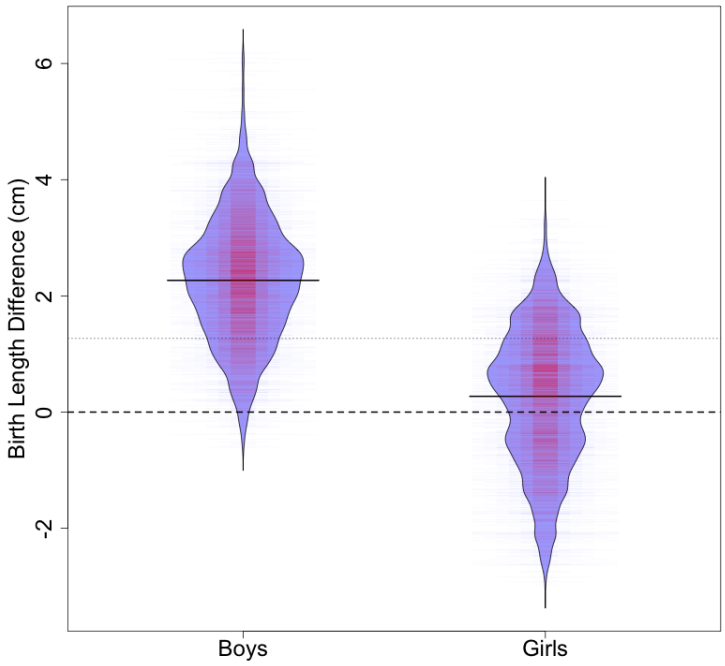



Figure 6.8: Posterior distributions of the mean length differences at birth between children that are born to mothers with and without access to a handpump. Boys born with access to a handpump are 2.52 cm longer than those without. The effect is smaller and statistically insignificant for girls.

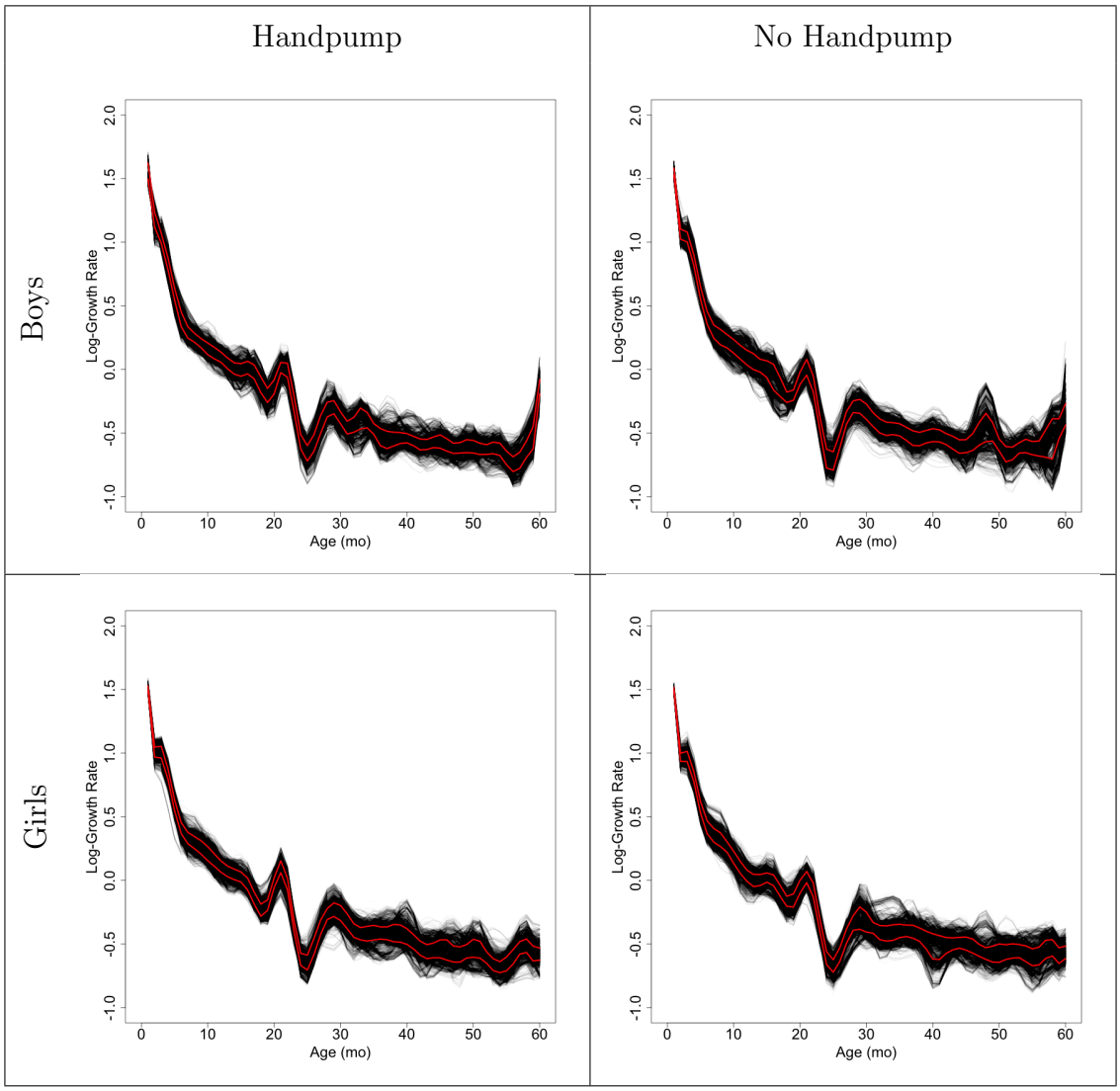
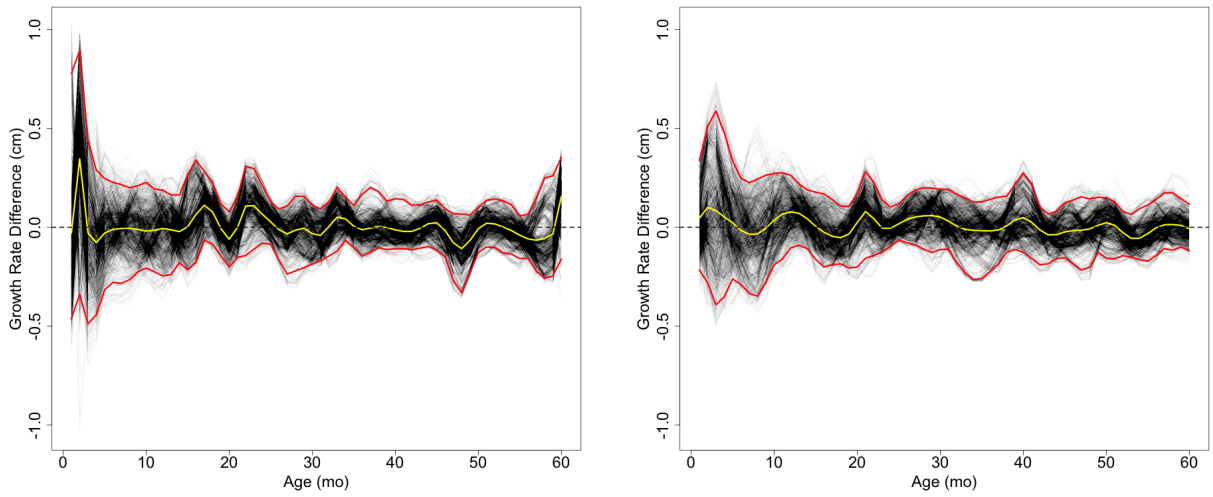



Figure 6.9: Posterior distributions of log-growth rates for households with (left column) and without (right column) access to handpumps, crossed with boys (top row) and girls (bottom row). In red are the 75th and 25th posterior percentiles.




Boys Girls

Figure 6.10: Posterior samples of growth rate differences between children with and without access to a handpump for boys (left) and girls (right). A positive difference indicates a larger growth rate for those with access to a handpump than for those without. Red lines mark the 95% credible intervals, and yellow lines mark the posterior means.

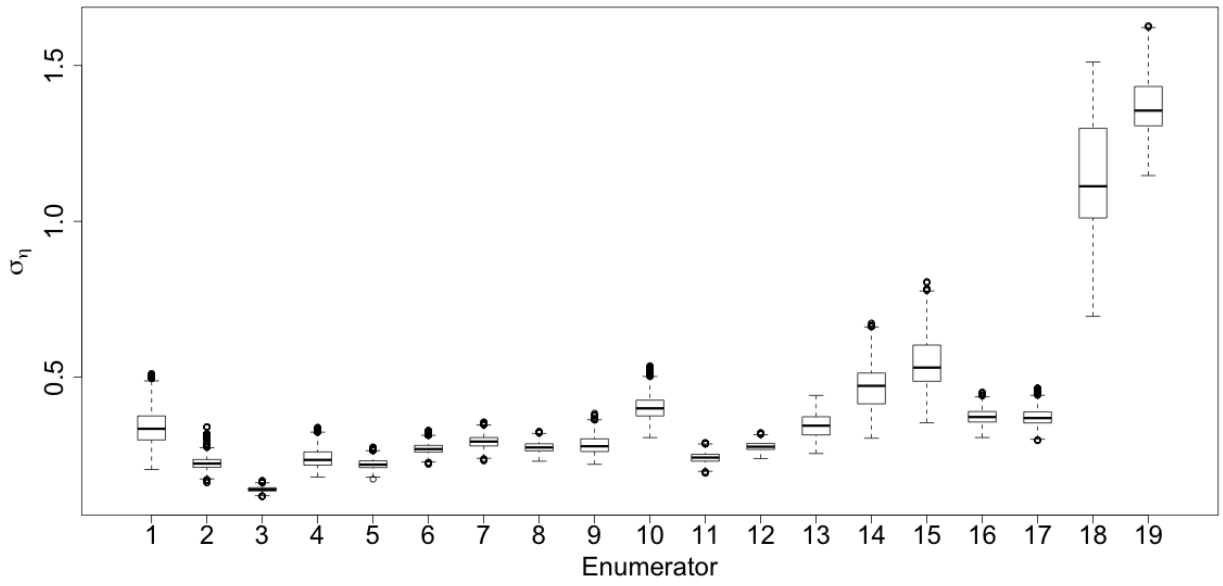



Figure 6.11: Posterior distributions of enumerator standard deviations $\sigma_\eta$. Compare these distributions with the raw data in figure 6.4.

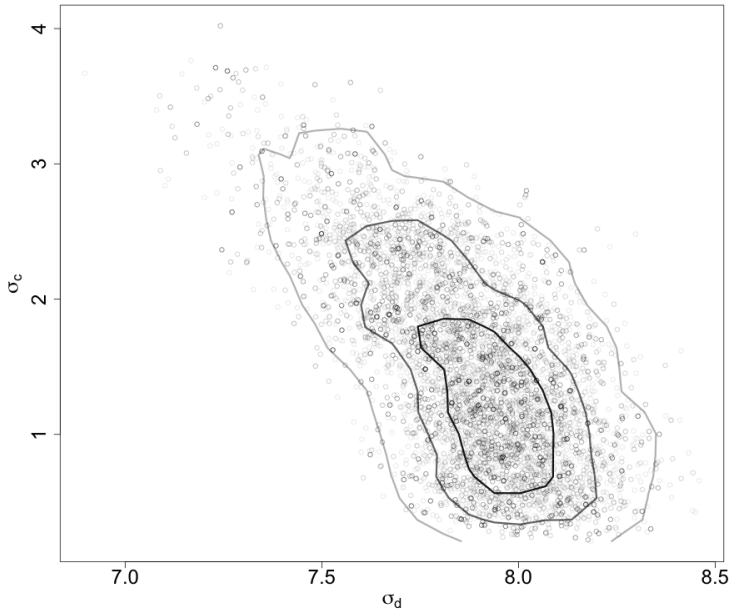



Figure 6.12: Joint posterior distribution of standard deviations for the measurement within date $\sigma_d$ and for that within child $\sigma_c$.

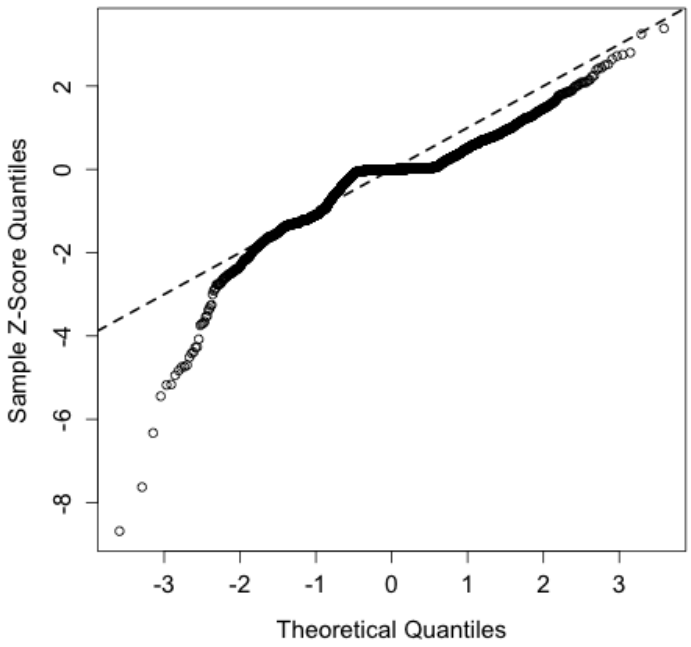
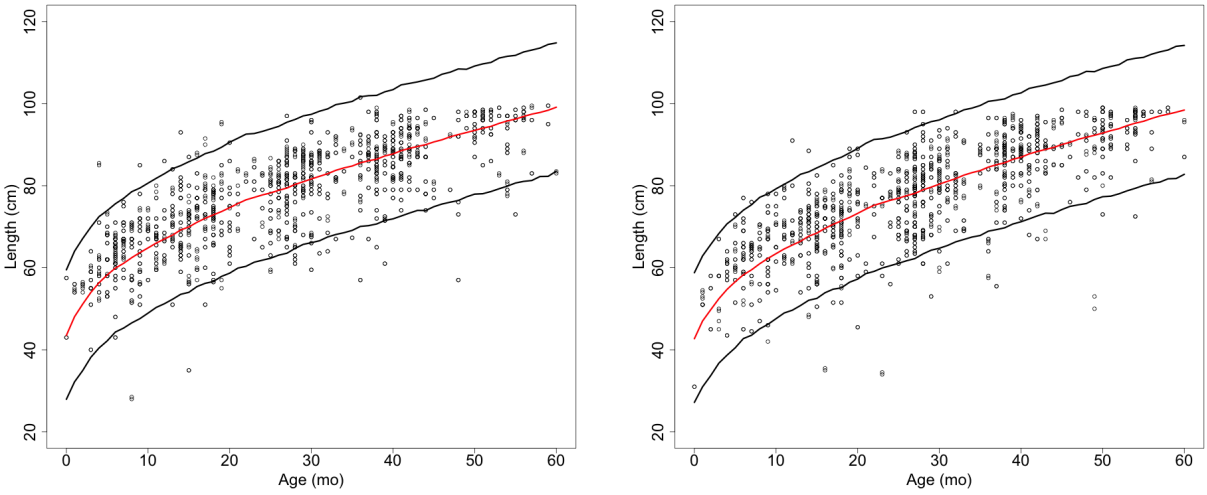



Figure 6.13: Standardized residual q-q plot for the *maximum a posteriori* model and parameter setting. The x-axis is the theoretical quantile for the standard normal distribution, and the dashed line is the 45-degree line.




Boys Girls

Figure 6.14: Comparison of the observed data to the 95% posterior predictive intervals for boys (left) and girls (right). The coverage rates are close to the nominal rate, as 6.1% and 5.7% of observations for boys and girls, respectively, fall outside of the intervals. The red lines mark the posterior means.

# Chapter 7

# Conclusion

We have introduced a novel algorithm called weighted particle tempering, developed a simplifying implementation of reversible jump which alleviates the burden of implementation for the user across several applications, and combined the two algorithms to form a hybrid called reversible jump weighted particle tempering, which we have applied to a nonlinear model discerning the impacts of having clean water access on child development. We have done so in the hope of contributing to the growing collection of available Monte Carlo methods in such a manner that would be accessible to applied researchers. While most of the available methods ignore the obstacles associated with tuning, we have been especially careful to keep this concern as the focus of our work.

There are several future directions in which we could take the efforts provided within this thesis. Weighted particle tempering can be extended to an algorithm with multiple tempering exponents, which would be beneficial for applications in which the target distribution has clusters of modes separated by varying distances. Additionally, one could use weighted particle tempering to sample for underlying chains, inducing a recursive hierarchy similar to parallel hierarchical sampling.

Practical reversible jump could be improved by using cross-model proposal distributions that

better approximate the conditional posterior distributions. A simple way to do this is to fit a density estimator to pilot samples from an MCMC that targets the individual, conditional posteriors. The estimated conditional posterior densities could be updated adaptively while the algorithm produces new samples.

A better analysis of the child lengths dataset would take other covariates into account, including information about diet or gastrointestinal illness symptoms experienced by the children. If another follow up study were performed, it would be good to account for parent heights, have a more detailed breakdown of handpump use, and record gestational time. Such a study would require significantly more resources, but would be well worth the insight that could help foreign aid agencies make high impact decisions.

Our research has opened up new possibilities for modeling complex datasets by allowing efficient exploration for posterior sampling. Armed with new, accessible tools for data analysis, we hope to accelerate humanity's journey to understand the universe and its complicated processes. We optimistically look ahead to the next phase of our endeavors with the anticipation of contributing even more toward this cause.