Chapter 6

Markov chain Monte Carlo

6.1 Basic Markov chain theory

We return again to the goal of estimating an expectation with respect to a distribution π. (We have been using f for this 'target distribution' in the previous chapter; the reason for the switch to π has to do with compatibility with standard notation for Markov chains.) When the algorithms discussed in the previous chapter become inefficient or unusable, Markov chain Monte Carlo is often a viable option. Markov chain Monte Carlo (MCMC) describes any approach where the Metropolis-Hastings algorithm is used to construct a Markov chain with stationary distribution π. The draws from the Markov chain are then used in the same way that iid draws from the distribution π would have been used in the previous chapter — sample averages based on the Markov chain are used to estimate characteristics of π. The popularity of MCMC stems from its general applicability to a wide range of distributions for which classical Monte Carlo methods such as rejection sampling and importance sampling are either impossible or extremely inefficient.

In this section, we will go through some basic Markov chain theory. While this does not purport to be a serious treatment of this vast subject, it is an attempt at providing Markov chain Monte Carlo users with some of the Markov chain theory and ideas relevant to MCMC. Some useful Markov chain theory relevant to MCMC is available at an introductory level in Tierney (1995) and Robert and Casella (2005).

As suggested by Geyer (2000), we refer to all algorithms used for Markov chain Monte Carlo under the umbrella term the "Metropolis-Hastings" or the "Metropolis-Hastings-Green" algorithm, recognizing the contributions of the three seminal papers in this area, Metropolis et al. (1953), Hastings (1970) and Green (1995). Versions of the Metropolis-Hastings algorithm have also been described in several other significant papers, including (Geman and Geman, 1984) in the context of image analysism (Tanner and Wong, 1987), when describing the 'data augmentation' algorithmm (Gelfand and Smith, 1990) for Bayesian inference, and (Geyer and Thompson, 1992) for maxi-

mum likelihood inference. In addition, (Tierney, 1994) lays out much of the important theoretical framework underlying the Metropolis-Hastings algorithm.

6.1.1 Definitions

The following description borrows some descriptions from Jones and Hobert (2001) and Tierney (1995). Consider a Markov chain $X_1, X_2, \ldots$ with state space Ω so $X_i \in \Omega$. Since X is a Markov chain, the Markovian property holds so $X_n, X_{n+1}, \ldots$ and $X_{n-2}, X_{n-3}, \ldots$ are conditionally independent given X_{n-1} for any n,

$$P(X_n \in A \mid X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \ldots) = P(X_n \in A \mid X_{n-1} = x_{n-1}).$$

For a discrete state space, the Markov chain is specified by an **initial distribution** (X_1 is obtained from this) and a **transition probability matrix (t.p.m.)** $\{P_{ij}\}$ where $P_{ij} = Pr(\text{move to state j from state i}) = P(X_n = j \mid X_{n-1} = i)$ for $i, j \in \Omega$. For a continuous state space, the Markov chain is again specified by an initial distribution and a **transition kernel** (or transition density,) which is simply a conditional probabibility density function, $K(x,y) = k(y|x)$ such that

$$P(X_n \in A | X_{n-1} = x) = \int_A k(y|x)dy \ \forall x \in \Omega,$$

where A is any subset of Ω (technically, $A \in \mathcal{B}(\Omega)$ where $\mathcal{B}(\Omega)$ is a Borel $\sigma-$algebra generated by Ω, informally, a collection of all subsets of Ω; discussions of Borel sets can be found in any introductory book on probability and measure theory.) $K^n(x,y) = k^n(y|x)$ is the **n-step transition kernel** where

$$P(X_{i+n} \in A | X_i = x) = \int_A k^n(y|x)dx.$$

Example: A simple example of a Markov chain on a continuous state space is an AR(1) model

$$X_n = \theta X_{n-1} + \epsilon_n, \ \ n = 1, 2, \ldots, \ \ \theta \in (0,1),$$

where $\epsilon_n \overset{iid}{\sim} N(0, \sigma^2)$, $n = 1, 2, \ldots$. It is easy to see that the Markovian assumption holds and the transition kernel for this simple chain is $f(x_n = y \mid x_{n-1})$, the pdf of a Normal density with mean θx_{n-1} and variance σ^2 and:

$$P(X_n \in A \mid X_{n-1} = x_{n-1}) = \int_A f(x_n = y \mid x_{n-1})dy.$$

Stationarity: For a Markov chain on a discrete state space, with transition probability matrix P, if π is a probability mass function such that

$$\pi(x)P_{x,y} = \pi(y),$$

π is the stationary distribution of the Markov chain. For a Markov chain on a continuous state space, if π is a density such that

$$\pi(y) = \int_\Omega k(y|x)\pi(x)dx \tag{6.1}$$

then π is the stationary density for the Markov chain defined by k. The implication of stationarity is that if the current state of the chain (X_n, say) is drawn from π, then marginal density of the next state (X_{n+1}) is also π.

Irreducibility: This simply means the Markov chain has positive probability of reaching the entire state space. In the discrete case the chain is irreducible if:

$$\text{For all } i, j \in \Omega \quad \text{there exists an n such that } P_{ij}^n > 0.$$

In the continuous case, the corresponding notion is **π-irreducibility** which requires that for every set/interval that has positive probability under π, the Markov chain will visit the set with positive probability. Let $\pi(A) = \int_A \pi(x)dx$ (abusing notation slightly). An M.C. is $\pi - irreducible$ if:

$$\text{For all } x \in \Omega, \text{ and all A such that } \pi(A) > 0, P^n(x, A) > 0 \text{ for some n.}$$

That is, any set with positive probability under π is accessible from every point in the state space.
Recurrence: In the discrete case, a Markov chain is recurrent if for all $i \in \Omega$, $P(X_n = i|X_0 = i) = 1$ for some $n < \infty$. In other words, the chain will return to state i after leaving state i in a finite number of steps with probability 1. Equivalently, all state will be visited infinitely often by the Markov chain. In the continuous state space context, recurrence implies that all sets with positive probability under π will be visited infinitely often by the chain, for almost all starting values. Formally, a $\pi - irreducible$ Markov chain is recurrent if for all A such that $\pi(A) > 0$:

1. $Pr(X_n \in A \text{ infinitely often} \mid X_0 = x) > 0$ for all $x \in \Omega$ **and**
2. $Pr(X_n \in A \text{ infinitely often} \mid X_0 = x) = 1$ for all π-almost all x, that is, for all $x \in \Omega$ except possible on a set $\mathcal{C}$ such that $\pi(\mathcal{C}) = 0$.

A recurrent chain is **positive recurrent** if π is a probability distribution. However, the above requirement is less than satisfying since this means that there are a set of values (any $x \in \mathcal{C}$) from where the Markov chain may not reach all A infinitely often. Hence, we define the stronger notion of positive Harris recurrence.

Positive Harris recurrence: A π-irreducible Markov chain is positive Harris recurrent if π is a probability distribution, and for all A with $\pi(A) > 0$,

$$Pr(X_n \in A \text{ infinitely often } \mid X_o = x) = 1, \text{ for all } x \in \Omega$$

Aperiodicity: In the discrete state space context, the **period** of a state $i \in \Omega$ is the greatest common divisor of $\{n \geq 1 \text{ s.t. } P_{ii}^n > 0\}$. If the Markov chain is aperiodic, then $d(i) = 1$ for all $i \in \Omega$. The definition is more technical for continuous state spaces, however some intuition can be borrowed from the discrete state space definition: the Markov chain must be aperiodic for *all* possible partitions of the state space (partitions can be thought of as discretizations of the state space.) Aperiodicity therefore implies that the state space cannot be partitioned in such a way that the Markov chain visits each part of the partition in a systematic manner. A periodic Markov chain, on the other hand, would make a regular (predictable) tour through some partition of the state space.

Ergodicity: A Markov chain that is aperiodic and positive recurrent is said to be ergodic.

Harris ergodicity: All the results we discuss about Markov chain Monte Carlo will hinge on the assumption that the Markov chain we construct is Harris ergodic. If a Markov chain is aperiodic and positive Harris recurrent, it is said to be a Harris ergodic chain.

Reversibility: Another important concept relevant to Markov chain theory, particularly to the development of Markov chain Monte Carlo is the notion of reversibility. A Markov chain is reversible if it is positive recurrent with stationary distribution π and if it satisfies **the detailed balance condition**:

$$\pi(x)K(x, y) = \pi(y)K(y, x). \tag{6.2}$$

The term reversibility refers to the fact that when the chain is stationary, it 'looks' the same running forward as it does running backwards. More concretely, $(X_m, X_{m+1}, \ldots, X_{m+k})$ and $(X_{m+k}, X_{m+k-1}, \ldots, X_m)$ have the same joint distributions. What makes reversibility useful is the following fact: *A chain that is reversible with respect to a distribution π has stationary distribution π.* It turns out that it is easier to construct a Markov chain (by finding the appropriate transition kernel) that is reversible with respect to π, than it is to solve the more general problem of finding a transition kernel that has π as its stationary distribution. This is because solving the integral equation (6.1) is much more difficult than satisfying detailed balance.

6.1.2 MCMC Asymptotics

We can now state the following important result.

Theorem 5. *(Strong Law of Large Numbers for Markov chains): If a Harris-ergodic Markov chain $X_1, X_2, \ldots$ has stationary distribution π, and if $\mu = E_\pi g(x) < \infty$, then*

$$\hat{\mu}_n = \frac{\sum_{i=1}^n g(X_i)}{n} \to \mu \textit{ with probability 1.}$$

Notes:

1. The above result looks like the regular Strong Law of Large Numbers (SLLN) used for classical (i.i.d.) Monte Carlo. Hence, if the stated conditions are satisfied, we can use sample averages based on the values of the chain to estimate expectations with respect to π.

2. An important distinction, however, is that $X_1, \ldots, X_n$ are *dependent* and the X_is are not distributed according to π, unless the chain is already stationary. In fact, the distribution of X_k is different from the distribution of X_j whenever $j \neq k$, and $E(\hat{\mu}_n) \neq \mu$, unless $X_1 \sim \pi$. Hence, the Monte Carlo estimate of μ is *biased* unless the chain is started with a draw from the stationary distribution.

3. It is not easy to estimate Var($\hat{\mu_n}$) due to the dependence among $X_1, \ldots, X_n$.

4. Conditions for establishing that a Central Limit Theorem holds are more complicated and much harder to verify than in the iid case. Hence, theory for MCMC is more complicated than for iid Monte Carlo. We will discuss this in greater detail in Subsection 6.3.

5. The SLLN for Markov chains as stated above holds *regardless of the initial distribution*, that is, regardless of how X_1 was obtained.

In addition to a Strong Law of Large Numbers, a stronger convergence result can be obtained for Harris ergodic chains. But first, another definition:
The **total variational distance** between two distributions π_1 and π_2 is

$$\|\pi_1 - \pi_2\|_{TV} = \sup_{\text{all } A} |\pi_1(A) - \pi_2(A)|$$

where, as before, 'all A' stands for all subsets A of the state space Ω and $\pi_1(A) = \int_A \pi(x)dx$ and $\pi_2(A) = \int_A \pi(x)dx$.

Theorem 6. *(Convergence in total variational distance): If a Harris ergodic Markov chain $X_1, X_2, \ldots$ has stationary distribution π and an n-step transition kernel $k^n(y \mid x)$, then:*

$$\lim_{n\to\infty} \|P^n(x, \cdot) - \pi(\cdot)\|_{TV} = 0$$

for all $x \in \Omega$, where $P^n(x, A) = \int_A k^n(y \mid x)dy$ (that is $P^n(x, A) = P(X_{i+n} \in A | X_i = x)$.)

Convergence in total variational distance implies that for every set A with $\pi(A) > 0$:

$$\lim_{n\to\infty} P^n(x, A) - \pi(A)| = 0,$$

which is convergence in distribution. The implication of the above result is therefore the following: regardless of the starting value, after a long time this Markov chain will produce values that look

approximately like draws from π. Since we may be interested not only in whether the Markov chain converges to the π, but also the *rate* at which converges to π, it is useful to consider two popular rates of convergence as defined according to total variational distance:

- **Geometric ergodicity**: If the rate of convergence of the Markov chain is given by:

  $$\|P^n(x, \cdot) - \pi(\cdot)\|_{TV} \leq M(x)t^n,$$

  where $M(x)$ is a real valued non-negative function of x and $t \in (0, 1)$, the chain is said to be geometrically ergodic.

- **Uniform ergodicity**: If the rate of convergence of the Markov chain is given by:

  $$\|P^n(x, \cdot) - \pi(\cdot)\|_{TV} \leq M^* t^n,$$

  where M^* is finite and $t \in (0, 1)$, the chain is said to be uniformly ergodic. Clearly, uniform ergodicity implies geometric ergodicity.

Theorem 7. *(A Markov chain Central Limit Theorem): Consider a Harris ergodic Markov chain $\{X_n\}$ on Ω with stationary distribution π. For any real valued function g ($g : \Omega \to \mathbb{R}$), consider the following sets of conditions:*

1. *$\{X_n\}$ is geometrically ergodic and $E_\pi|g(x)|^{2+\epsilon} < \infty$ for some $\epsilon > 0$.*

2. *$\{X_n\}$ is geometrically ergodic, reversible and $E_\pi(g(x))^2 < \infty$ for some $\epsilon > 0$.*

3. *$\{X_n\}$ is uniformly ergodic and $E_\pi(g(x))^2 < \infty$.*

If either one of the above conditions is satisfied, the following Markov chain Monte Carlo Central Limit Theorem holds for any initial distribution:

$$\lim_{n\to\infty} \sqrt(n)(\hat{\mu_n} - \mu) \to N(0, \sigma^2). \tag{6.3}$$

Note that the conditions listed here are not the only sufficient conditions for a Markov chain Central Limit Theorem to hold, they just happen to be the easiest ones to state and check (though checking these conditions is not easy in general.) For a more detailed look at the Markov chain Central Limit Theorem see Jones (2004) and Roberts and Rosenthal (2004).

From Theorems 5 and 6, it is apparent that if we can construct a Harris ergodic Markov chain with stationary distribution π, we can estimate characteristics of π. The Metropolis-Hastings algorithm is a very general method for producing just such a chain.

6.2 The Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm describes a method to construct a Harris-ergodic Markov chain $X_1, X_2, \ldots$ with stationary distribution π on state space Ω. Note that π may be a multivariate distribution so X_is may be multidimensional.

6.2.1 'All-at-once' M-H

The simplest version of the Metropolis-Hastings algorithm is easy to describe.

1. Select an initial value X_1 where $X_1 \in \Omega$. The rest of the chain can be simulated in the following way.

2. If $X_n = x$, generate X_{n+1} for $n = 1, 2, \ldots$ as follows:

   (a) Generate a proposal $y \sim q(x, \cdot)$, where $q(x, \cdot)$ is some distribution that may depend on x (the current value of the chain. For example if the proposal is Normal with mean set at current value (x) and variance =3, $q(x, \cdot)$ is the pdf of a Normal with mean x and variance 3.

   (b) Calculate the Metropolis-Hastings acceptance probability $\alpha(x, y)$ if $\pi(x)q(x, y) > 0$:

   $$\alpha(x, y) = \min\left\{1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right\} = \min\left\{1, \frac{h(y)q(y, x)}{h(x)q(x, y)}\right\}, \tag{6.4}$$

   where $h(x), h(y)$ are simply the unnormalized distributions (proportional to π) evaluated at x, y respectively. Note that $\alpha(x, y) = 1$ if $\pi(x)q(x, y) = 0$.

   (c) Accept y with probability $\alpha(x, y)$ else reject y, and 'stay' at x. That is, simulate $U \sim$Unif(0,1) and if $U < \alpha(x, y)$ set $X_{n+1} = y$ else set $X_{n+1} = x$.

Why All-at-once M-H works

We will now discuss why the Metropolis-Hastings algorithm results in a Harris ergodic Markov chain with the desired stationary distribution. Our focus will be on giving general ideas and outlines of proofs. For a more in-depth look at the theory, Tierney (1994) is an ideal reference.

If the following conditions hold, the chain is Harris-ergodic with stationary distribution π (these conditions are sufficient but not necessary):

1. If $q(x, y) > 0$ for all $x, y \in \Omega$, the Markov chain is irreducible.

2. If $P(\pi(x)q(x, y) \leq \pi(y)q(y, x)) < 1$ then Markov chain has positive probability of staying at x and the chain is therefore aperiodic. It is easy to see why this is true. If this condition holds, $\alpha(x, y) = 1$ with probability less than 1 which implies that $\alpha(x, y) < 1$ with probability greater than 0, which implies that there is positive probability of 'staying' at x.

3. The detailed balance condition (6.2)is satisfied.

To see why detailed balance is satisfied, consider the transition kernel resulting from the algorithm, $K(x, y)$. To show detailed balance holds, we need to show:

$$\pi(x)K(x, y) = \pi(y)K(y, x) \quad \text{for all } x, y \in \Omega.$$

From (6.4), $K(x, y) = q(x, y)\alpha(x, y)$ If $x = y$, this is trivially satisfied, so we assume $x \neq y$ and, without loss of generality, that $\pi(y)q(y, x) > \pi(x)q(x, y)$. Then

$$K(x, y) = \pi(x)q(x, y) \min\left\{1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right\} = \pi(x)q(x, y), \tag{6.5}$$

and,

$$\begin{aligned} K(y, x) =& \pi(y)q(y, x) \min\left\{1, \frac{\pi(x)q(x, y)}{\pi(y)q(y, x)}\right\} \\ =& \pi(y)q(y, x)\frac{\pi(x)q(x, y)}{\pi(y)q(y, x)} = \pi(x)q(x, y). \end{aligned} \tag{6.6}$$

Hence, detailed balance holds.

6.2.2 'Variable-at-a-time' M-H

When the dimensions of the distribution π gets large, the simple Metropolis-Hastings algorithm described above may become ineffective and impractical since it is hard to find a proposal q that will work well. Hence, it would be advantageous to have an algorithm that is able to break the problem down into smaller, more manageable blocks. The Variable-at-a-time Metropolis Hastings (VMH) algorithm is one such approach.

Suppose the target distribution, $\pi(\boldsymbol{\theta}), \boldsymbol{\theta} \in \Omega$, is multivariate and $\boldsymbol{\theta}$ can be broken up into k blocks so $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$, where each block may itself be multivariate so the θ_is may be multidimensional. We define the **full conditional distribution** of θ_i by the conditional distribution $\pi(\theta_i \mid \boldsymbol{\theta}_{-i})$, where $\boldsymbol{\theta}_{-i}$ is the $\boldsymbol{\theta}$ vector without θ_i. The full conditional distribution of θ_i, up to a constant of proportionality, is very easy to obtain from the joint distribution $\pi(\boldsymbol{\theta})$ (again, this is something we may only know up to a constant of proportionality.) $\pi(\theta_i \mid \boldsymbol{\theta}_{-i})$ is easily obtained by simply removing all terms in $\pi(\boldsymbol{\theta})$ that do not depend on θ_i. The VMH algorithm proceeds by simply applying the 'all-at-once' Metropolis-Hastings algorithm to each of the k blocks in succession. The simplest version of the VMH algorithm is as follows:

1. Select an initial value $\boldsymbol{\theta}^{(1)} \in \Omega$, the state space. The rest of the chain can be simulated in the following way.

2. For convenience, denote the nth value (state) of the chain, $\boldsymbol{\theta}^{(n)} = \boldsymbol{\theta}$. To obtain the $n+1$st value of the chain, for $i = 1, \ldots, k$, do the following:

   (a) Calculate the full conditional distribution $\pi(\theta_i \mid \boldsymbol{\theta}_{-i})$, where $\boldsymbol{\theta}_{-i}$ contains *the most recently updated values* of the chain, including all the updates from the ongoing $(n+1)$ iteration of the chain.

   (b) Perform a simple ('all-at-once') Metropolis-Hastings update for this full conditional distribution, to obtain the $n+1$st value of the chain for the θ_i block.

Why Variable-at-a-time M-H works

To see why Variable-at-a-time Metropolis-Hastings (VMH) works, we will consider a simple two-block update sampler. Specifically, consider a Markov chain with current state vector $x = (x_1, x_2)$ where x_1, x_2 are two 'blocks' which may be multivariate themselves. Now suppose we have a transition kernel for the first block, $k_{1|2}(x_1, y_1|x_2)$ such that it defines a Markov chain with stationary distribution $\pi_{1|2}(x_1|x_2)$ where $\pi_{1|2}(x_1|x_2)$ is the conditional distribution corresponding to the joint target distribution $\pi(x)$. Similarly, assume the transition kernel for the second block, $k_{2|1}(x_2, y_2|y_1)$ defines a Markov chain with stationary distribution $\pi_{2|1}(x_2|y_1)$. We can think of the transition kernel for both updates (of the first and second blocks in succession) as the composition of two kernels,

$$K((x_1, x_2), (y_1, y_2)) = k_{1|2}(x_1, y_1|x_2) k_{2|1}(x_2, y_2|y_1).$$

We can now show that this transition kernel K has stationary distribution π. We need to show that:

$$\int_\Omega K(x, y)\pi(x)dx = \pi(y)$$

Now,

$$\begin{aligned}\int_\Omega K(x, y)\pi(x)dx &= \int_{x_1}\int_{x_2} K((x_1, x_2), (y_1, y_2))\pi_{1|2}(x_1|x_2)\pi(x_2)dx_1dx_2 \\ &= \int_{x_1}\int_{x_2} K((x_1, x_2), (y_1, y_2))\pi_{1|2}(x_1|x_2)\pi(x_2)dx_1dx_2\end{aligned}$$

6.2.3 Gibbs, Metropolis, Metropolis-Hastings updates

The literature on Markov chain Monte Carlo is vast and the terminology can often be confusing. This short section provides some clarification. The Variable-at-a-time Metropolis-Hastings

(VMH) algorithm as described in Section 6.2.2 produces a Harris-ergodic Markov chain with stationary distribution π by using a Metropolis-Hastings update (corresponding to the 'all-at-once' M-H algorithm) for each block/variable separately. Consider a target distribution $\pi(\boldsymbol{\theta})$ with $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)^T$, where each θ_i is a block (possibly multidimensional.) We will again denote the vector without θ_i by $\boldsymbol{\theta}_{-i}$.

The VMH algorithm involves a series of block updates, an update to each θ_i in turn, to produce each new draw from the Markov chain. These updates can take one of many forms:

1. **Metropolis-Hastings update**: This is the most general kind of update where a proposed value θ_i^* is generated according to a proposal distribution $q(\theta_i, \cdot)$ since the proposal may depend on the current value of θ_i. The Metropolis-Hastings acceptance probability is $\alpha(\theta_i, \theta_i^*)$:
$$\alpha(\theta_i, \theta_i^*) = \min\left\{1, \frac{\pi(\theta_i^*|\boldsymbol{\theta}_{-i})q(\theta_i^*, \theta_i)}{\pi(\theta_i|\boldsymbol{\theta}_{-i})q(\theta_i, \theta_i^*)}\right\}$$

2. **Metropolis update**: When the proposal is symmetric, $q(x, y) = q(y, x)$, the Metropolis-Hastings acceptance probability simplifies to:
$$\alpha(\theta_i, \theta_i^*) = \min\left\{1, \frac{\pi(\theta_i^*|\boldsymbol{\theta}_{-i})}{\pi(\theta_i|\boldsymbol{\theta}_{-i})}\right\}$$

3. **Gibbs update**: When it is possible to directly simulate from the full conditional distribution, as is the case when the full conditinal distribution is a recognized density, the proposal *is* the full conditional distribution, $q(\theta_i, \theta_i^*) = \pi(\theta_i^*|\boldsymbol{\theta}_{-i})$, the Metropolis-Hastings simplifies even further:
$$\alpha(\theta_i, \theta_i^*) = \min\left\{1, \frac{\pi(\theta_i^*|\boldsymbol{\theta}_{-i})}{q(\theta_i, \theta_i^*)}\frac{q(\theta_i^*, \theta_i)}{\pi(\theta_i|\boldsymbol{\theta}_{-i})}\right\} = \min\{1, 1\} = 1.$$
This means that you always accept every proposed draw.

A **Gibbs sampler** is a special case where every update is a Gibbs update, which obviously allows for a much narrower class of algorithms and a **Metropolis algorithm** is one where every update is a Metropolis update. Since a **Metropolis-Hastings algorithm** can involve any combination of the above updates, it seems much easier to simply call all such algorithms Metropolis-Hastings algorithm.

The **Data augmentation algorithm** due to Tanner and Wong (1987) is also a Metropolis-Hastings algorithm, though a specific variant of it where auxilliary random variables are introduced to simplify the form of the distribution of interest. A significant advance in the Metropolis-Hastings algorithm is due to Green (1994) who suggested an algorithm capable of sampling over mixtures of distributions where each component of the mixture could possibly have different dimensions. This algorithm is generally refered to as the **reversible jump Metropolis-Hastings algorithm**. In addition, there is a vast literature on variants of the Metropolis-Hastings algorithm that are useful for particular distributions, such as the **slice sampler** and **simulated tempering** algorithms.

6.2.4 An Example

We now go through an example in detail to fix ideas about how to construct a Metropolis-Hastings algorithm. Our example uses a dataset from the Chandra Orion Ultradeep Project (COUP). More information on this is available at: CASt Chandra Flares data set . The raw data, which arrives approximately according to a Poisson process, gives the individual photon arrival times (in seconds) and their energies (in keV). The processed data we consider here is obtained by grouping the events into evenly-spaced time bins (10,000 seconds width).

Our goal for this data analysis is to identify the change point and estimate the intensities of the Poisson process before and after the change point. We describe a Bayesian model for this change point problem (Carlin and Louis, 2000). Let Y_t be the number of occurrences of some event at time t. The process is observed for times 1 through n and we assume that there is a change at time k, that is, after time k, the event counts are significantly different (higher or lower than before). The mathematical description of the model is provided in change point model (pdf) . While this is a simple model, it is adequate for illustrating some basic principles for constructing a (variable-at-a-time) Metropolis-Hastings algorithm.

Model description: Consider the following hierarchical changepoint model for the number of occurrences Y_i of some event during time interval i with change point k.

$$\begin{aligned} Y_i|k,\theta,\lambda &\sim \text{Poisson}(\theta) \text{ for } i = 1,\ldots,k \\ Y_i|k,\theta,\lambda &\sim \text{Poisson}(\lambda) \text{ for } i = k+1,\ldots,n \end{aligned}$$

Assume the following prior distributions:

$$\begin{aligned} \theta|b_1 &\sim \text{Gamma}(0.5, b_1) \qquad (\text{pdf}=g_1(\theta|b_1)) \\ \lambda|b_2 &\sim \text{Gamma}(0.5, b_2) \qquad (\text{pdf}=g_2(\lambda|b_2)) \\ b_1 &\sim \text{IG}(0,1) \qquad (\text{pdf}=h_1(b_1)) \\ b_2 &\sim \text{IG}(0,1) \qquad (\text{pdf}=h_2(b_2)) \\ k &\sim \text{Uniform}(1,\ldots,n) \qquad (\text{pmf}=u(k)) \end{aligned}$$

Deriving the joint distribution (π): k, θ, λ are conditionally independent and b_1, b_2 are independent.
Assume the Gamma density parameterization $\text{Gamma}(\alpha,\beta) = \frac{1}{\Gamma(\alpha)\beta^\alpha}x^{\alpha-1}e^{-x/\beta}$ and IG (Inverse Gamma) density parameterization $\text{IG}(\alpha,\beta) = \frac{e^{-1/\beta x}}{\Gamma(\alpha)\beta^\alpha x^{\alpha+1}}$

Inference for this model is therefore based on the 5-dimensional **posterior** distribution $\pi(k,\theta,\lambda,b_1,b_2|\mathbf{Y})$ where $\mathbf{Y}=(Y_1,\ldots,Y_n)$. The posterior distribution is obtained *up to a constant* (that is, the normalizing constant is unknown) by taking the product of all the conditional distributions. Thus we have

the 5-dimensional posterior distribution $\pi(k,\theta,\lambda,b_1,b_2|\mathbf{Y})$ The posterior distribution is

$$\begin{aligned} \pi(k,\theta,\lambda,b_1,b_2|\mathbf{Y}) \propto & \prod_{i=1}^{k}\frac{\theta^{Y_i}e^{-\theta}}{Y_i!}\prod_{i=k+1}^{n}\frac{\lambda^{Y_i}e^{-\lambda}}{Y_i!} \\ & \times \frac{1}{\Gamma(0.5)b_1^{0.5}}\theta^{-0.5}e^{-\theta/b_1} \times \frac{1}{\Gamma(0.5)b_2^{0.5}}\lambda^{-0.5}e^{-\lambda/b_2} \\ & \times \frac{e^{-1/b_1}}{b_1}\frac{e^{-1/b_2}}{b_2} \times \frac{1}{n} \end{aligned} \tag{6.7}$$

Note: The reason we have a formula for what π is proportional to (hence $\propto$ rather than =) instead of an exact description of the function is because the missing constant (the normalizing constant) can only be computed by integrating the above function. Fortunately, the Metropolis-Hastings algorithm does not require knowledge of this normalizing constant.

Deriving the full conditional distributions: From (6.7) we can obtain full conditional distributions for each parameter by ignoring all terms that are constant with respect to the parameter. Sometimes these full conditional distributions are well known distributions such as the Gamma or Normal.
Full conditional for θ:

$$\begin{aligned} f(\theta|k,\lambda,b_1,b_2,\mathbf{Y}) &\propto \prod_{i=1}^{k}\frac{\theta^{Y_i}e^{-\theta}}{Y_i!} \times \frac{1}{\Gamma(0.5)b_1^{0.5}}\theta^{-0.5}e^{-\theta/b_1} \\ &\propto \theta^{\sum_{i=1}^{k}Y_i-0.5}e^{-\theta(k+1/b_1)} \\ &\propto \text{Gamma}\left(\sum_{i=1}^{k}Y_i+0.5, \frac{b_1}{kb_1+1}\right) \end{aligned}$$

Full conditional for λ:

$$\begin{aligned} f(\lambda|k,\theta,b_1,b_2,\mathbf{Y}) &\propto \prod_{i=k+1}^{n}\frac{\lambda^{Y_i}e^{-\lambda}}{Y_i!} \times \frac{1}{\Gamma(0.5)b_2^{0.5}}\lambda^{-0.5}e^{-\lambda/b_2} \\ &\propto \text{Gamma}\left(\sum_{i=k+1}^{n}Y_i+0.5, \frac{b_2}{(n-k)b_2+1}\right) \end{aligned}$$

Full conditional for k:

$$\begin{aligned} f(k|\theta,\lambda,b_1,b_2,\mathbf{Y}) &\propto \prod_{i=1}^{k}\frac{\theta^{Y_i}e^{-\theta}}{Y_i!}\prod_{i=k+1}^{n}\frac{\lambda^{Y_i}e^{-\lambda}}{Y_i!} \\ &\propto \theta^{\sum_{i=1}^{k}Y_i}\lambda^{\sum_{i=k+1}^{n}Y_i}e^{-k\theta-(n-k)\lambda}. \end{aligned}$$

Full conditional for b_1:

$$f(b_1|k,\theta,\lambda,b_2,\mathbf{Y}) \propto \frac{1}{b_1^{0.5}}e^{-\theta/b_1} \times \frac{e^{-1/b_1}}{b_1} \propto b_1^{-1.5}e^{-(1+\theta)/b_1} \propto IG(0.5, 1/(\theta+1))$$

Full conditional for b_2:

$$f(b_2|k,\theta,\lambda,b_1|\mathbf{Y}) \propto \times \frac{1}{b_2^{0.5}} e^{-\lambda/b_2} \times \frac{e^{-1/b_2}}{b_2} \propto b_2^{-1.5} e^{-(1+\lambda)/b_2} \propto IG(0.5, 1/(\lambda+1))$$

We are now in a position to run the Metropolis-Hastings algorithm.

Note 1: $\theta, \lambda, b_1, b_2$ all have full conditional distributions that are well known and easy to sample from. We can therefore perform Gibbs updates on them where the draw is from their full conditional. However, the full conditional for k is not a standard distribution so we need to use the more general Metropolis-Hastings update instead of a Gibbs update.

Note 2: The Inverse Gamma density is said to be a **conjugate** prior in this case since it results in a posterior that is also Inverse Gamma and therefore trivial to sample. As such, this density is mathematically convenient (due to its conjugacy property) but does not necessarily result in a better MCMC sampler. Also, it has poorly behaved moments; it may be better to adopt another prior density (such as a Gamma) instead.

The Metropolis-Hastings algorithm:

1. Pick a starting value for the Markov chain, say $(\theta^0, \lambda^0, k^0, b_1^0, b_2^0) = (1, 1, 20, 1, 1)$.
2. Update each variable in turn:
   (a) Sample $\theta^i \sim f(\theta|k,\lambda,b_1,b_2,\mathbf{Y})$ using the most upto date values of k,λ,b_1,b_2 (Gibbs update using the derived Gamma density).
   (b) Sample $\lambda^i \sim f(\lambda|k,\theta,b_1,b_2,\mathbf{Y})$ using the most upto date values of k,θ,b_1,b_2. (Gibbs update using the derived Gamma density).
   (c) Sample $b_1^i \sim f(b_1|k,\theta,\lambda,b_2,\mathbf{Y})$ using the most upto date values of k,θ,λ,b_2. (Gibbs update using the derived Gamma density).
   (d) Sample $b_2^i \sim f(b_2|k,\theta,\lambda,b_1,\mathbf{Y})$ using the most upto date values of k,θ,λ,b_1. (Gibbs update using the derived Gamma density).
   (e) Sample $k \sim f(k|\theta,\lambda,b_1,b_2,\mathbf{Y})$ using the most upto date values of k,θ,λ,b_1,b_2. This requires a Metropolis-Hastings update:
      i. 'Propose' a new value for k, k^* according to a proposal distribution say $q(k|\theta,\lambda,b_1,b_2,\mathbf{Y})$. In our simple example we pick $q(k|\theta,\lambda,b_1,b_2,\mathbf{Y}) = \text{Unif}\{2,\ldots,m-1\}$ where m is the length of the vector (time series) $\mathbf{Y}$.
      ii. Compute the Metropolis-Hastings accept-reject ratio,

$$\alpha(k,k^*) = \min\left(\frac{f(k^*|\theta,\lambda,b_1,b_2,\mathbf{Y})q(k|\theta,\lambda,b_1,b_2,\mathbf{Y})}{f(k|\theta,\lambda,b_1,b_2,\mathbf{Y})q(k^*|\theta,\lambda,b_1,b_2,\mathbf{Y})},1\right)$$

      iii. Accept the new value k^* with probability $\alpha(k,k^*)$, otherwise 'reject' the proposal k^* so that the next value of k remains the same as before.
   (f) You now have a new Markov chain state $(\theta^1, \lambda^1, k^1, b_1^1, b_2^1)$
3. Return to step #2 N-1 times to produce a Markov chain of length N.

6.2.5 MCMC and classical Monte Carlo

Exact (iid) sampling algorithms such as rejection sampling are impractical for most complicated distributions, especially multidimensional ones as already discussed in Section 5.4. Importance sampling is very general in theory but finding reasonable importance functions becomes prohibitively difficult as the dimensions of the distribution increases. With the Metropolis-Hastings algorithm, on the other hand, one needs much less information about the target distribution to construct reasonable proposals and hence a reasonably efficient algorithm. Furthermore, the ability to break down the dimensions of the problem allows one much greater flexibility in the design and implementation of an algorithm for simulating from a high dimensional distribution. In addition, the fact that there is 'memory' built into the algorithm, that is, the fact that the algorithm uses where it has been to simulate the next draw, allows the algorithm to adapt to the local characteristics of the distribution.

Of course, designing a Metropolis-Hastings algorithm that works well for a complicated distribution is still a challenge but the M-H algorithm has allowed statisticians to (relatively) routinely fit much more complicated models than the classical methods allowed. This flexibility does come at a price, namely that theoretical work to assess the quality of estimates becomes much more difficult; on the other hand, the easier algorithms (rejection sampling and importance sampling, for instance), are often impractical when the Metropolis-Hastings algoithm is still practical.

MCMC and Important Sampling in tandem

An important point to note, however, is that Metropolis-Hastings and importance sampling can be used together (they are not mutually exclusive). In fact, this joint usage is the basis of at least two very useful applications: (1) Markov chain Maximum Likelihood, where the likelihood can be evaluated at many different values of the parameter using a single set of draws from a Markov chain, and (2) For studying the sensitivity of the posterior to changes in the prior distribution: a single set of draws from the Markov chain for one set of priors can be reweighted according to several different priors to obtain estimates with respect to the posterior distribution for these new priors. For e.g. if under the first set of priors, you obtain draws from the posterior distribution $\pi_1(\boldsymbol{\theta})$, you can now obtain estimates with respect to a new posterior distribution (based on using a

new set of priors), $\pi_2(\boldsymbol{\theta})$, by using importance sampling since,

$$E_{\pi_2}(\boldsymbol{\theta}) = \int \boldsymbol{\theta}\pi_2(\boldsymbol{\theta})d\boldsymbol{\theta} = \int \boldsymbol{\theta}\frac{\pi_2(\boldsymbol{\theta})}{\pi_1(\boldsymbol{\theta})}\pi_1(\boldsymbol{\theta})d\boldsymbol{\theta},$$

as long as $\pi_1(x) = 0 \Rightarrow \pi_2(x) = 0$ for all x.

6.3 Practical Issues

6.3.1 A rough guide to constructing an MCMC algorithm

We begin with a rough guide of the steps involved in constructing an MCMC algorithm from scratch. Each step describes some of the decisions that need to be made at each stage along with associated tradeoffs.

1. **Derive $h(\boldsymbol{\theta})$, the unnormalized joint distribution** of interest, which is proportional to the target distribution, that is, $h(\boldsymbol{\theta}) \propto \pi(\boldsymbol{\theta})$ where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$.

2. **Identify blocks or components of $\boldsymbol{\theta}$ that would be relatively easy to simulate 'all-at-once'.** This is a very difficult decision to make in general and will typically require some trial and error though, sometimes, the structure of the problem may suggest a blocking strategy. A basic principle to follow is to look for blocks that appear to be highly correlated, as updating these jointly in a sensible fashion will typically make the sampler more efficient. The tradeoff will often be as follows: updating large blocks of parameters will allow the sampler to move around the distribution more efficiently but designing good proposals for large blocks can be difficult *and* updating large blocks of parameters may be computationally more demanding (due to matrix operations that may be involved in the proposal and accept-reject steps of the algorithm.)

3. **Derive full conditional distributions** for each block based on the decision made in Step 2.

4. **Identify any blocks with full conditionals that have a known form.** Use Gibbs updates for these full conditionals. Please note, however, that this choice is only for ease of implementation, not for efficiency reasons; Gibbs updates may or may not be more efficient than Metropolis-Hastings updates. If the sampler works poorly, you should consider a Metropolis-Hastings update instead of a Gibbs update.

5. Construct Metropolis-Hastings updates for the remaining blocks. **There are a large number of possibilities when it comes to determining what M-H update to use.** The first M-H update to try is a simple Metropolis update with a symmetric proposal. It may also be useful to try transformations. For example, if θ_i has positive support, try $\phi_k = \log(\theta_i)$ instead. ϕ_i's full conditional distribution may be easier to deal with and it is easy to transform the sampled value back to θ_i.

6. **Run the M-H algorithm for short trial runs**, printing out lots of intermediate results:

   (a) This can help you make sure there are no obvious errors with your algebra or your programming.

   (b) You can see if you need to make changes to your algorithm. For example: If some of the parameters seem to be highly correlated, you may decide to sample them in a block. If Metropolis updates seem to result in highly autocorrelated samples, try changing the tuning parameters.

   (c) **Save the last draw of each of your trial runs and use it to start your next run.** This is a great way to obtain reasonable starting values. Any value you would not mind having in your sample is a reasonable value to start the sampler at (Geyer 2000).

7. Once obvious problems have been fixed and you have fine tuned your algorithm, **run the chain for as long as possible. If the Monte Carlo standard errors for the estimates of expectations of interest are acceptable, you can stop the chain.** There are many estimates of Monte Carlo standard error. We recommend the `consistent batch means` estimate (Jones et al. (2006)). For computational efficiency, the best approach is to compute the standard error estimates after ever t iterations where t is large enough so the standard error computation does not slow down the algorithm too much. Of course, be aware that this approach works well only when the sampler is working reasonably well. If the Markov chain sampler is poor (slow mixing) and unable to find multiple modes of a multimodal distribution, all methods for deciding when to stop the chain will be ineffective.

8. **Report your final estimates, along with any estimates of error associated with them.** It is useful to also indicate (for future reference) the length of the chain used and what algorithm you finally ended up using.

An important general computing principle is to **always do calculations on the log scale as far as possible.** For instance, for the Metropolis-Hastings accept-reject step, instead of simulating $U \sim$Unif(0,1) and checking if

$$U < \frac{\pi(\theta_i^*|\boldsymbol{\theta}_{-i})q(\theta_i^*, \theta_i)}{\pi(\theta_i|\boldsymbol{\theta}_{-i})q(\theta_i, \theta_i^*)},$$

check whether:

$$\log(U) < \log(\pi(\theta_i^*|\boldsymbol{\theta}_{-i})) + \log(q(\theta_i^*, \theta_i)) - \log(\pi(\theta_i|\boldsymbol{\theta}_{-i})) - \log(q(\theta_i, \theta_i^*)).$$

Of course, this is the same principle that was applied to classical Monte Carlo approaches, leading to much more stable and accurate computations.

6.3.2 Decisions to make

There are several decisions to be made by the user at every stage of the algorithm design and implementation, including, but not limited to:

1. Block updates: How do we decide which blocks of variables to update together? In other words, how do we determine the θ_is above?

2. Proposals: How do we determine proposals for each block update? This is perhaps the decision that is most time consuming and difficult for MCMC users.

3. Tuning proposals and acceptance rates: This is related to choice of proposals. For example, if we decide to use a random walk proposal, say Normal with mean centered at current value and variance of τ, how do we determine τ? What is a reasonable acceptance rate? Based on theoretical results for very simple problems (cf. Rosenthal and Roberts, 1999), 25% is often tossed around as a rule but there are many problems where this is unrealistic or even counter-productive: acceptance rates as high as 70% or as low as 5% may be optimal, for instance. **There are no general rules about optimal acceptance rates for non-toy problems.** A general heuristic is to aim for around 25% acceptance rates, then use other approaches, for instance MCMC standard errors or effective sample sizes, to determine how to change the proposal.

4. How can we determine when to stop the chain? The standard error-based stopping rule discussed previously is among the more reasonable rules available; unfortunately, standard errors may themselves be often prone to large errors, and the difficulty of knowing when a Central Limit Theorem holds is a reason to still be cautious. Here are a few heuristics for determining chain length:

   (a) Track the approximation over time. A stable approximation of the

   (b) Track the MCMC standard errors for each component and run the sampler until the MCMC standard errors for the expectation of interest are small. In some applications there may be good guidance for desirable "small" standard errors. If it is not obvious how to select the threshold, a rule of thumb is to wait until the MCMC standard errors are less than 0.05 of the standard deviation. Plotting the errors may be useful since it will show how/whether the errors are decreasing with the number of iterations.

   There are no methods that guarantee convergence but following *all of the above heuristics* will typically result in reliable approximations based on MCMC.

To make matters worse, there are rarely rules for arriving at sensible ways to do things, only heuristics based on experience. There is therefore a need for fully automated MCMC algorithms. Fortunately, in recent years, some progress has been made in developing MCMC algorithms for classes

of models (distributions.) `WINBUGS` (Spiegelhalter et al., 1991?), where BUGS stands for Bayesian Inference Using Gibbs Sampling, is a remarkable general engine for constructing MCMC samplers based on just a high level specification of a Bayesian model. It automatically calculates conditional distributions and makes algorithm decisions and produces samples. However, for complicated problems, people still typically need to write their own customized MCMC algorithms and it is unusual to even have algorithms that work well for the same model but with different data sets. The open source software `R` (Ihaka and Gentleman, 1998) has lead to the creation of numerous contributed packages, including routines for MCMC for specific models and problems. The contributed packages can be found here: `http://cran.r-project.org/`.

Automation of MCMC algorithms is therefore still an area with plenty of open research questions, since it is typically difficult to construct a sampler with good theoretical properties (say one that is uniformly or geometrically ergodic) that works well for even a relatively small class of problems. Some exciting developments in this research area include slice sampling (Tierney and Mira, 2002, Neal, 2003, Roberts and Rosenthal, 2004?) and variants of the slice sampler, and adaptive Markov chain Monte Carlo approaches. Also, sequential Monte Carlo methods have recently emerged as alternatives to MCMC for some problems.

Burn-in, subsampling, and multiple chains

Burn-in: Since the Markov chain is typically not started with a value from the stationary distribution, the estimates based on the samples are biased. Burn-in is an ad-hoc method used by MCMC practitioners to try to reduce the bias of the estimates by simply removing an arbitrary number of initial values from the chain and using the remainder of the chain to do estimation since for a Markov chain $X_1, X_2, \ldots$, removing the first b draws produces $X_{b+1}, X_{b+2}, \ldots$, which are presumably 'closer' to being draws from the stationary distribution π. Unfortunately, there is no way to know how much bias is being removed (if any) by discarding some initial draws. The only thing we know for sure is that the variability of the estimate has been increased! For more on this, read the thoughtful (and entertaining) discussion on MCMC diagnostics by C.J.Geyer (`http://www.stat.umn.edu/~charlie/mcmc/diag.html`). To quote Geyer (2000) "Any value you do not mind having in your sample is a reasonable starting value." A useful recommendation is to simply start the chain off at the last sample from the previous MCMC run. Since there is often a fair amount of tuning involved in developing an effective Metropolis-Hastings algorithm, these tuning runs can be helpful in providing the initial value for the final run of the sampler.

Subsampling: Many MCMC users are troubled by heavy autocorrelation in their samples, as they should be. One way to deal with this issue is to simply subsample the chain, that is, pick off every

kth sample. For k large enough, this would result in a much less autocorrelated sample. From basic Markov chain theory, the subsampled Markov chain inherits the important properties of the original Markov chain (crucially, it has the same stationary distribution.) *If samples are easy to produce and expensive to process, subsampling can be a simple but useful approach.* For example: If it takes very little time to generate samples from a 10,000 dimensional distribution, and the draws are heavily autocorrelated, it may be worth subsampling the chain just to reduce the burden on storage and processing the samples (computing estimates of different kinds based on the samples.) The flipside is that *if it is expensive to generate samples, and the cost of processing them is not very high, subsampling is wasteful.* Using fewer samples, even if the samples are fairly dependent, is usually going to result in greater variability (reduced accuracy) of the estimates — the Monte Carlo standard errors are going to increase.

Multiple chains: Starting the sampler off at a few different values is a reasonable way to make sure that nothing eggregiously wrong is happening with the sampler. This is primarily useful for diagnosing coding errors and can be helpful (if we are lucky) with figuring out obvious multimodalities in the distribution. Luck comes into the picture because we will need to have chosen starting points that are located appropriately near the different modes to detect them, something that users typically cannot do. *Much more important is the idea of running the chain for as long as feasible, since a long run is likely to eventually pick up unusual features of the target distribution.*

6.3.3 Monte Carlo standard errors

Markov chain Monte Carlo based estimates are routinely reported without estimates of their Monte Carlo standard errors, even though ignoring uncertainty in estimates is obviously very poor statistics. While estimates of Monte Carlo standard errors for i.i.d. Monte Carlo methods are easily obtained, it is much more complicated for MCMC. We suggest the `consistent batch means` estimate (Jones et al. (2006)) for which `R` code is available at `http://www.stat.psu.edu/~mharan/batchmea`
This estimate has nice theoretical properties and is easy to compute.

Consistent batch means

There is a vast literature on sophisticated methods for computing Monte Carlo standard errors for MCMC output. Batch means is one method used to compute Monte Carlo standard errors. Suppose we are interested in estimating an expectation $\mu = E(g(X))$ where X is a draw from a distribution f. The batch means method works as follows: Run the Markov chain $\{X_n\}$ for N=ab iterations (we can assume a and b are integers). Let

$$Y_k = \frac{\sum_{i=(k-1)b+1}^{kb} g(X_i)}{b}$$

If we think of the Markov chain as having been divided into "a" batches of size "b" each, Y_k is then the Monte Carlo estimate of μ based on the kth 'batch'. Let

$$\hat{\sigma}^2 = \frac{b}{a-1}\sum_{k=1}^{a}(Y_k - \hat{\mu})^2.$$

Then the **batch means estimate of Monte Carlo standard error** is, as usual, $\frac{\hat{\sigma}}{\sqrt{N}}$.

A logical question to ask is: How do we pick b (and hence a) ? b (batch size) should be large enough so Y_ks are approximately independent. It turns out that fixing a and b results in an inconsistent estimate of standard error, while having b depend on the chain length (N) in the right way results in a consistent estimate that appears to work well in practice. Jones et al. (2006) call this particular version the **consistent batch means** estimator. In particular, they recommend setting $b = \sqrt{N}$.

Jones et al. (2006) and Flegal et al. (2008) suggest that the standard error based on the consistent batch means method be used to determine when to stop an MCMC simulation. This is an intuitive approach, and corresponds well with the approach used in classical (iid) Monte Carlo methods. The **fixed width MCMC simulation** approach is fairly simple: Determine a desired level of accuracy (in terms of Monte Carlo standard errors) for your parameters. Usual frequentist notions of the desired width of 95% confidence intervals for the parameters can help determine the desired Monte Carlo standard error. Periodically compute standard errors for the parameters using consistent batch means as described above. Stop the chain when all standard errors attain desired levels. Note that one should make sure that the chain is run for a minimum initial length (depending on the problem, observed autocorrelations etc.) so that the standard error estimates themselves are not too unreliable. This approach has the advantage of being theoretically sound and appears to work well in practice (Jones et al. (2006) and Flegal et al. (2008)). However, there are two important caveats to keep in mind: (1) This approach works well if the sampler is 'fast mixing' so it is (as always) important to ensure that the sampler works well in the first place, and (2) Multiple modes can cause additional problems that this approach may not always be able to solve.

Note: It is most often the case that posterior standard deviations or other functions of the mean are of interest, and hence it may be important to assess the standard errors of such quantities. For this, we can use a delta-method approximation (following C.J.Geyer's online vignette for his `R` package mcmcpack.)

6.3.4 Effective Sample Size

When comparing algorithms, one fairly intuitive notion is 'effective sample size'. Consider the simple question: How many independent samples m from π correspond (roughly) to the n *dependent* samples I have obtained from my Metropolis-Hastings sampler which has π as its stationary distribution? For illustrative purposes, consider a Markov chain that is mixing so well that its draws

are practically independent; the number of independent samples corresponding to n draws from this Markov chain may be pretty close to n. Suppose you had a Markov chain with draws that were almost perfectly autocorrelated. Even with n draws you would have effectively obtained not much more than a single sample.

Define autocorrelation time, $\kappa = 1 + 2\sum_{i=1}^{\infty} \rho_k$ where ρ_k is the lag-k autocorrelation. κ therefore captures the extent of autocorrelation in the samples so $\kappa = 1$ if there is no autocorrelation and $\kappa > 1$ if there is autocorrelation. Now, define ESS=N/κ where N=length of the Markov chain and κ is the autocorrelation time. If the samples are independent, $\kappa = 1$ so ESS=N. The other extreme: If the samples have autocorrelation 1 at every lag, that is, only one sample has been repeated N times, the ESS will be $N/N = 1$. In most realistic situations, the ESS will be somewhere between those two extremes.

κ can be estimated using sample autocorrelations from the MCMC chain.

$$\hat{\kappa} = 1 + 2\sum_{i=1}^{k} \hat{\gamma}_i$$

where $\hat{\gamma}_i$ is the lag-i sample autocorrelation from the Markov chain. Even when draws are uncorrelated after a certain lag, the estimated autocorrelations are likely to be non-zero. Hence, after a certain lag, adding estimated autocorrelations would correspond to simply adding noise. It is therefore sensible to stop adding estimated autocorrelations after a certain lag k. k can be determined in many different ways.

A simple method is to simply stop the summation for the first k s.t. $\hat{\gamma}_k < 0.1$. A better method corresponds to the IMSE (initial monotone sequence estimator) in Geyer (1992). Define

$$\hat{\Gamma}_i = \hat{\gamma}_{2i} + \hat{\gamma}_{2i+1}$$

Let m be the largest integer s.t. $\hat{\Gamma}_i > 0$ and $\hat{\Gamma}_i$ is monotone for $i = 1, \ldots m$. Set $k = 2m + 1$.

For a fair comparison between two MCMC algorithms, however, one must take into account the time that it takes to generate draws. Hence, the number of **effective samples per second** (ES/s), is likely to be a very useful practical evaluation of the efficiency of a sampler. Of course, an important practical consideration is also the ease of implementation since the programmer's time is typically more valuable than even computing time. Note that ESS can be obtained from the `coda` and `boa` packages in `R`.

6.4 Monte Carlo Maximum Likelihood

Monte Carlo methods (including Markov chain Monte Carlo) are useful in any problem that involves intractable probability distributions. One particularly useful application of Monte Carlo is to the statistical problem of finding maximum likelihood estimators (MLEs.)

Consider $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$, a family of normalized densities. Now suppose $f_\theta(x) = h_\theta(x)/c(\theta)$ where $c(\theta)$ is an unknown normalizing function. This can happen when there is a need for flexible models for complicated problems. A good example is any distribution that is a member of the exponential family so it can be written (up to a normalizing function) as $h_\theta(x) = e^{t(x),\theta}$ where $t(x)$ is a vector of statistics and θ is an unknown parameter. The exponential family allows for flexible models that may be appropriate for certain problems in genetics and network models, for instance.

Suppose the goal is to compute a Maximum Likelihood estimate for θ,

$$\hat{\theta} = \text{argmax}_{\theta \in \Theta} f_\theta(x).$$

Since we do not know $c(\theta)$, it seems as if this problem is a non-starter. Fortunately, we can use Monte Carlo methods to estimate the MLE, which allows us to then fit the model we want to fit rather than a less satisfactory one purely for computational convenience.

The MCML solution

Pick some $\psi \in \Theta$ such that $h_\psi(x) = 0 \Rightarrow h_\theta(x) = 0$ for almost-all $x \in \Omega$. Notice that this was a requirement for importance sampling (which is not a coincidence since MCML utilizes importance sampling.) Since $f_\psi(x)$ is constant with respect to θ, maximizing $f_\theta(x)$ is equivalent to maximizing $\log\left(\frac{f_\theta(x)}{f_\psi(x)}\right)$ with respect to θ. Denote $\log\left(\frac{f_\theta(x)}{f_\psi(x)}\right)$ by $\ell(\theta)$. We can rewrite $\ell(\theta)$ as follows:

$$\ell(\theta) = \log\left(\frac{h_\theta(x)}{h_\psi(x)}\right) - \log\left(\frac{c(\theta)}{c(\psi)}\right). \tag{6.8}$$

Note that the first term above is completely known, so only the second term poses problems. However, can rewrite the second term as:

$$\begin{aligned}
\log\left(\frac{c(\theta)}{c(\psi)}\right) &= \log\left(\frac{\int h_\theta(x)dx}{c(\psi)}\right) = \log\left(\int \frac{h_\theta(x)}{c(\psi)}dx\right) \\
&= \log\left(\int \frac{h_\theta(x)}{h_\psi(x)/f_\psi(x)}dx\right) = \log\left(\int \frac{h_\theta(x)}{h_\psi(x)}f_\psi(x)dx\right) \\
&= \log\left\{E_{f_\psi}\left(\frac{h_\theta(x)}{h_\psi(x)}\right)\right\}.
\end{aligned} \tag{6.9}$$

The expectation above can be estimated as usual by Monte Carlo. We can simulate $Y_1, \ldots, Y_M \sim f_\psi(\cdot)$ (using rejection sampling, Metropolis-Hastings or other algorithms) and thereby obtain the

approximation,

$$\hat{\ell}(\theta) = \log\left(\frac{h_\theta(x)}{h_\psi(x)}\right) - \log\left\{\frac{1}{M}\sum_{j=1}^{M}\frac{h_\theta(Y_j)}{h_\psi(Y_j)}\right\}. \tag{6.10}$$

Now $\tilde{\theta}$ that maximizes $\hat{\ell}(\theta)$ is a Monte Carlo estimate of the MLE. It is well known that $\tilde{\theta}$ is a poor estimate of the MLE $\hat{\theta}$ if ψ is far from $\hat{\theta}$. Naturally, this is a difficult problem in general since the reason we are using MCML in the first place is because we are unable to obtain the MLE. Therefore, in addition to the usual issues with designing and implementing a good sampler for f_ψ, an important unresolved issue is how to select a good ψ. This is still largely an open problem. Ideas such as path sampling and bridge sampling (cf. the references and discussion in Gelman and Meng, 1998) are ways to improve the estimates.

Example: Consider a toy example where $\mathbf{X} = X_1, X_2, \ldots X_n \overset{iid}{\sim} f_\theta(x)$ where f_θ is the density for an exponential, $f_\theta(x) = \frac{1}{\theta}\exp(-x/\theta)$ so

$$f_\theta(\mathbf{X}) = \frac{1}{\theta^n}\exp\left(-\sum_{i=1}^{n} X_i/\theta\right).$$

The MLE for θ is $\bar{X}$, the sample mean. Now suppose we did not know the normalizing constant so we had to estimate the MLE using MCML. We only know $h_\theta(\mathbf{X}) = \exp\left(-\sum_{i=1}^{n} X_i/\theta\right)$. What if we wanted to find the MLE in such a scenario. If we follow the recipe described above, we simply need to do the following:

1. Select $\psi \in (0, \infty)$.

2. Generate $\mathbf{Y}_1, \ldots, \mathbf{Y}_M \sim f_\psi$. Since this is a toy problem, we can directly simulate each $\mathbf{Y}_j$ by drawing n samples from the exponential density. Typically, we would have to run the Metropolis-Hastings algorithm to simulate the $\mathbf{Y}$s.

3. We can now evaluate our estimate log-likelihood ratio surface, $\hat{\ell}(\theta)$ at any $\theta \in \Theta$:

$$\hat{\ell}(\theta) = \log\left(\frac{h_\theta(\mathbf{X})}{h_\psi(\mathbf{X})}\right) - \log\left\{\frac{1}{M}\sum_{j=1}^{M}\frac{h_\theta(\mathbf{Y}_j)}{h_\psi(\mathbf{Y}_j)}\right\}.$$

   Note that we can do this for any θ by using **the same set of draws from** f_ψ. As you may recall from Section 5.6.1, this is one of the advantages of importance sampling.

4. We can now estimate the MLE using the MCMLE $\tilde{\theta}$, the value that maximizes $\hat{\ell}(\theta)$.