1

*Markov Chain Monte Carlo for Item Response Models**

CONTENTS

1.1 Introduction

Markov Chain Monte Carlo (MCMC) has revolutionized modern statistical computing, especially for complex Bayesian and latent variable models. A recent Web of Knowledge search (Thompson ISI, 2012) for "Markov Chain Monte Carlo" yielded 6,015 articles, nearly half in Statistics, and the rest spread across fields ranging from Computational Biology to Transportation and Thermodynamics. Of these, seventy-two articles appear in *Psychometrika*, *Journal of Educational and Behavioral Statistics* and *Applied Psycho-*

*This work was supported in part by a Graduate Training Grant awarded to Carnegie Mellon University by the US Department of Education, Institute of Education Sciences (#R305B090023). The opinions and views expressed do not necessarily reflect those of IES or the DoEd. The order of authorship is alphabetical.

logical Measurement, and another seventeen in *Psychological Methods*, *Journal of Educational Measurement* and *Educational and Psychological Measurement*. This is remarkable for an estimation method that first appeared in a psychometrics-related journal around 1990.

For all of its success, it dawned on the broader scientific and statistics community slowly. With roots at the intersection of modern computing and atomic weapons research at Los Alamos New Mexico during World War II (Los Alamos National Laboratory, 2012; Metropolis, 1987), MCMC began as a method for doing state calculations in physics (Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller, 1953) and image restoration Geman and Geman (1984), and first came to widespread notice in the field of statistics through the work of Gelfand and Smith (1990), despite earlier pioneering work of Hastings (1970) and Tanner and Wong (1987). A more complete history of MCMC is provided by Robert and Casella (2011), and a very accessible introduction to the method can be found in Chib and Greenberg (1995).

As is evident from its roots (Metropolis, 1987, p. 129, Metropolis et al., 1953), MCMC is not inherently a Bayesian technique. It was used for years as a way of sampling from intractable, often high-dimensional, distributions, and has widespread applications in the integration, estimation and optimization of functions (e.g. Geyer, 1996). Its success in statistics and psychometrics however is driven by the extent to which it makes computation and estimation for complex, novel Bayesian models tractable (e.g. Congdon, 2007; Fox, 2010; Gelman, Carlin, Stern, and Rubin, 2003, etc.). In this chapter we consider the application of MCMC methods to Bayesian IRT and IRT-like models.

1.2 Applied Bayesian Inference

Item response theory (IRT) models, and psychometric models generally, deal fundamentally with multiway data. The most common formulation is two-way data consisting of coded responses U_{pi} of persons $p = 1, \ldots, P$ to items (tasks, stimuli, test questions, etc.) $i = 1, \ldots, I$. IRT provides a family of probabilistic models for the two-way array $\mathcal{U} = [U_{pi}]$ of coded item responses,

$$f(\mathcal{U}|\boldsymbol{\Theta}, \boldsymbol{B}, \boldsymbol{\gamma}) \ , \tag{1.1}$$

given a set $\boldsymbol{\Theta}$ of possibly multidimensional person parameters $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_P)$, a set $\boldsymbol{B}$ of possibly multidimensional item parameters $\boldsymbol{B} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_I)$, and possibly an additional set $\boldsymbol{\gamma}$ of other parameters. (Under mild regularity conditions, e.g. Billingsley, 1995, any parametric model $f(\mathcal{U}; \boldsymbol{\tau})$ for $\mathcal{U}$ expressed in terms of parameters $\boldsymbol{\tau}$ can be identified as a conditional distribution $f(\mathcal{U}|\boldsymbol{\tau})$ for $\mathcal{U}$ given $\boldsymbol{\tau}$. For the Bayesian calculations considered in this chapter, we always assume this identification.)

This basic formulation, and all of the methodology discussed in this chap-

ter, can be modified to account for additional hierarchical structure (Béguin and Glas, 2001; Fox and Glas, 2001; Janssen, Tuerlinckx, Meulders, and de Boeck, 2000; Kamata, 2001; Maier, 2001, for example), multiple time points (Fox, 2011; Studer, 2012, for example), ratings of each item (Mariano and Junker, 2007; Patz, Junker, Johnson, and Mariano, 2002, for example), computerized adaptive testing (Jones and Nediak, 2005; Matteucci and Veldkamp, 2011; Segall, 2002, 2003, for example), missing data (Glas and Pimentel, 2008; Patz and Junker, 1999b, for example), etc. Each of these changes the structure of $\mathcal{U}$—to be a multiway array, a ragged array, etc.—and may also introduce covariates and other features of designed and realized data collection. To keep the discussion focused on fundamental ideas, however, we will mostly consider complete two-way designs in this chapter.

To make the notation concise, we will collect all of the parameters $\boldsymbol{\Theta}$, $\boldsymbol{B}$, $\boldsymbol{\gamma}$ together into a single J-dimensional vector vector $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_J)^T$, so the basic model in Equation 1.1 becomes $f(\mathcal{U}|\boldsymbol{\tau})$. If we supply a prior distribution $f(\boldsymbol{\tau})$, we can write the joint distribution of the data and parameters as

$$f(\mathcal{U}, \boldsymbol{\tau}) = f(\mathcal{U}|\boldsymbol{\tau}) f(\boldsymbol{\tau}) \tag{1.2}$$

and, following Bayes' rule, the posterior distribution of $\boldsymbol{\tau}$ as

$$f(\boldsymbol{\tau}|\mathcal{U}) = \frac{f(\mathcal{U}|\boldsymbol{\tau}) f(\boldsymbol{\tau})}{\int f(\mathcal{U}|\boldsymbol{t}) f(d\boldsymbol{t})} \propto f(\mathcal{U}|\boldsymbol{\tau}) f(\boldsymbol{\tau}) \tag{1.3}$$

as a function of $\boldsymbol{\tau}$.

(Here and throughout, we repeatedly re-use the notation $f()$ to represent various discrete and continuous probability density functions, with the particular role of $f()$ clear from context. In addition, all integrals $\int g(\boldsymbol{x}) f(d\boldsymbol{x})$ should be interpreted as Stieltjes integrals, for example, if $f()$ is a continuous density, this is a usual integral, and if $f()$ is a discrete density, this is a sum.)

Applied Bayesian statistics focuses on characterizing the posterior in Equation 1.3 in various ways. For example we may be interested in

- The posterior mean, or *expected a posteriori (EAP)* estimate of $\boldsymbol{\tau}$,

  $$E[\boldsymbol{\tau}|\mathcal{U}] = \int \boldsymbol{\tau} f(d\boldsymbol{\tau}|\mathcal{U})$$

  or perhaps the posterior mean of a function $E[g(\boldsymbol{\tau})|\mathcal{U}]$, or

- The posterior mode, or *maximum a posteriori (MAP)* estimate of $\boldsymbol{\tau}$,

  $$\text{argmax}_{\boldsymbol{\tau}} f(\boldsymbol{\tau}|\mathcal{U})$$

  or

- A *credible interval (CI)*, that is, a set A of parameter values $\boldsymbol{\tau}$ such that

  $$P(\boldsymbol{\tau} \in A|\mathcal{U}) = 1 - \alpha$$

  for some fixed probability $1 - \alpha$, or

- A *graph* or other characterization of the shape of $f(\boldsymbol{\tau}|\mathcal{U})$ as a function of (some coordinates of) $\boldsymbol{\tau}$,

and so forth.

1.3 Markov Chain Monte Carlo—General Ideas

The essential problem is to learn about the posterior distribution $f(\boldsymbol{\tau}|\mathcal{U})$, and this problem is made difficult by the need to compute the integral in the denominator of Equation 1.3. In most applications this integral must be computed numerically rather than analytically, and is usually quite high-dimensional—the dimension J of the parameter space is at least as large as $P + I$, the number of persons plus the number of items. Difficult numerical integration can often be sidestepped by Monte Carlo methods, and Markov Chain Monte Carlo (MCMC) offers a straightforward methodology for generating samples from (approximately) the posterior distribution $f(\boldsymbol{\tau}|\mathcal{U})$ in well-behaved Bayesian models, without directly calculating the integral in Equation 1.3.

The essential idea is to define a stationary Markov chain $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \ldots$ with states $\mathcal{M}_k = (\boldsymbol{\tau}^{(k)})$, and transition kernel

$$\kappa(\boldsymbol{t}^{(0)}, \boldsymbol{t}^{(1)}) = P[\mathcal{M}_k = (\boldsymbol{t}^{(1)})|\mathcal{M}_{k-1} = (\boldsymbol{t}^{(0)})] \ , \forall k \ ,$$

the probability of moving to a new state $\boldsymbol{t}^{(1)}$ given the current state $\boldsymbol{t}^{(0)}$, with *stationary distribution* $\pi(\boldsymbol{t})$, defined by

$$\int \kappa(\boldsymbol{t}^{(0)}, \boldsymbol{t}^{(1)})\pi(d\boldsymbol{t}^{(0)}) = \pi(\boldsymbol{t}^{(1)}) \ . \tag{1.4}$$

A common sufficient condition for Equation 1.4 to hold is *detailed balance* or *reversibility*, that is,

$$\pi(\boldsymbol{t}^{(0)})\kappa(\boldsymbol{t}^{(0)}, \boldsymbol{t}^{(1)}) = \pi(\boldsymbol{t}^{(1)})\kappa(\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(0)}) \ . \tag{1.5}$$

The Markov chain is said to be π-*irreducible* if it has positive probability of entering any set A for which $\pi(A) = \int_A \pi(d\boldsymbol{t}) > 0$. It is said to be periodic if there are portions of the state space that it can visit only at regularly spaced intervals; otherwise it is *aperiodic*. If a Markov chain has stationary distribution $\pi()$ as in Equation 1.4 and it is π-irreducible and aperiodic then the distribution of $\mathcal{M}_k$ will converge as, $k \to \infty$, to $\pi()$; see Tierney (1994) for details.

If we can define the transition kernel $\kappa(\boldsymbol{t}^{(0)}, \boldsymbol{t}^{(1)})$ so that in Equation 1.4 $\pi(\boldsymbol{t}^{)} = f(\boldsymbol{t}|\mathcal{U})$, then, after throwing away the first K observations—the "burn-in" period before the distribution of $\mathcal{M}_k$ has converged to $\pi(\boldsymbol{t})$—the remaining

1. Sample $\boldsymbol{\tau}_1^{(k)}$ from $f(\boldsymbol{\tau}_1|\boldsymbol{\tau}_2^{(k-1)},\ldots,\boldsymbol{\tau}_H^{(k-1)},\mathcal{U})$;
2. Sample $\boldsymbol{\tau}_2^{(k)}$ from $f(\boldsymbol{\tau}_2|\boldsymbol{\tau}_1^{(k)},\boldsymbol{\tau}_3^{(k-1)},\ldots,\boldsymbol{\tau}_H^{(k-1)},\mathcal{U})$;
3. Sample $\boldsymbol{\tau}_3^{(k)}$ from $f(\boldsymbol{\tau}_3|\boldsymbol{\tau}_1^{(k)},\boldsymbol{\tau}_2^{(k)},\boldsymbol{\tau}_4^{(k-1)},\ldots,\boldsymbol{\tau}_H^{(k-1)},\mathcal{U})$;

$\vdots$

H. Sample $\boldsymbol{\tau}_H^{(k)}$ from $f(\boldsymbol{\tau}_H|\boldsymbol{\tau}_1^{(k)},\boldsymbol{\tau}_2^{(k)},\boldsymbol{\tau}_3^{(k)},\ldots,\boldsymbol{\tau}_{H-1}^{(k)},\mathcal{U})$.

FIGURE 1.1
A generic blocked-MCMC algorithm based on the partition $(\boldsymbol{\tau}_1,\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H)$ of $\boldsymbol{\tau}$ into H blocks of parameters.

"good" observations $(\boldsymbol{\tau}^{(1)}) = \mathcal{M}_{K+1}, (\boldsymbol{\tau}^{(2)}) = \mathcal{M}_{k+2},\ldots,(\boldsymbol{\tau}^{(M)}) = \mathcal{M}_{K+M}$ can be treated like (dependent) draws from $f(\boldsymbol{\tau}|\mathcal{U})$.

For example, an EAP estimate of any integrable function $g(\boldsymbol{\tau})$ can be obtained simply as

$$\int_{\boldsymbol{\tau}} g(\boldsymbol{\tau}) f(d\boldsymbol{\tau}|\mathcal{U}) \approx \frac{1}{M}\sum_{m=1}^{M} g(\boldsymbol{\tau}^{(m)}) \ ;$$

with convergence guaranteed as $M \to \infty$. Similarly, an approximate graph of $f(\boldsymbol{\tau}|\mathcal{U})$ can be constructed as a (smoothed) histogram of the sample $\boldsymbol{\tau}^{(m)}, m = 1,\ldots,M$, and CI's and MAP estimates can be computed from the graph. More sophisticated methods leading to more efficient and stable estimates are also available, and should be used in practice; see for example Section 1.6.

Constructing the transition kernel $\kappa(\boldsymbol{t}^{(0)},\boldsymbol{t}^{(1)})$, so that the stationary distribution of the Markov chain is the posterior distribution $f(\boldsymbol{\tau}|\mathcal{U})$, is remarkably straightforward. For example, let $(\boldsymbol{\tau}_1,\boldsymbol{\tau}_2)$ be a disjoint partition of the parameter vector $\boldsymbol{\tau}$ into two blocks of parameters. Then a short calculation verifying Equation 1.4 shows that

$$\kappa(\boldsymbol{\tau}^{(0)},\boldsymbol{\tau}^{(1)}) = f(\boldsymbol{\tau}_1^{(1)}|\boldsymbol{\tau}_2^{(0)},\mathcal{U}) f(\boldsymbol{\tau}_2^{(1)}|\boldsymbol{\tau}_1^{(1)},\mathcal{U}) \tag{1.6}$$

has stationary distribution $f(\boldsymbol{\tau}|\mathcal{U})$.

More broadly, let $(\boldsymbol{\tau}_1,\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H)$ be any fixed, disjoint partition of the parameter vector $\boldsymbol{\tau}$ into $H \leq J$ blocks. In Figure 1.1 we define a sampling scheme to move from $\mathcal{M}_{k-1} = (\boldsymbol{\tau}_1^{(k-1)},\boldsymbol{\tau}_2^{(k-1)},\ldots,\boldsymbol{\tau}_H^{(k-1)})$ to $\mathcal{M}_k = (\boldsymbol{\tau}_1^{(k)},\boldsymbol{\tau}_2^{(k)},\ldots,\boldsymbol{\tau}_H^{(k)})$ in the Markov chain.

The conditional densities on the right in Figure 1.1 are called *complete conditionals*, because they express the distribution of each partition element $\boldsymbol{\tau}_h$ conditional on *all* other parameters and data in the model. To make the notation more concise we often abbreviate the conditioning variables in a

complete conditional as "*rest*", and write $f(\boldsymbol{\tau}_h|rest)$. An extension of the calculation showing that the kernel in Equation 1.6 has $f(\boldsymbol{\tau}|\mathcal{U})$ as its stationary distribution shows that the kernel consisting of the product of the complete conditionals,

$$\begin{aligned}\kappa(\boldsymbol{\tau}^{(k-1)},\boldsymbol{\tau}^{(k)}) &= f(\boldsymbol{\tau}_1^{(k)}|\boldsymbol{\tau}_2^{(k-1)},\ldots,\boldsymbol{\tau}_H^{(k-1)},\mathcal{U}) \\ &\times f(\boldsymbol{\tau}_2^{(k)}|\boldsymbol{\tau}_1^{(k)},\boldsymbol{\tau}_3^{(k-1)},\ldots,\boldsymbol{\tau}_H^{(k-1)},\mathcal{U}) \times f(\boldsymbol{\tau}_3^{(k)}|\boldsymbol{\tau}_1^{(k)},\boldsymbol{\tau}_2^{(k)},\boldsymbol{\tau}_4^{(k-1)},\ldots,\boldsymbol{\tau}_H^{(k-1)},\mathcal{U}) \\ &\times \quad\cdots\quad \times f(\boldsymbol{\tau}_H^{(k)}|\boldsymbol{\tau}_1^{(k)},\boldsymbol{\tau}_2^{(k)},\boldsymbol{\tau}_3^{(k)},\ldots,\boldsymbol{\tau}_{H-1}^{(k)},\mathcal{U}) \\ &= f(\boldsymbol{\tau}_1^{(k)}|rest)\times f(\boldsymbol{\tau}_2^{(k)}|rest)\times f(\boldsymbol{\tau}_3^{(k)}|rest)\times\cdots\times f(\boldsymbol{\tau}_H^{(k)}|rest)\end{aligned} \tag{1.7}$$

also has $f(\boldsymbol{\tau}|\mathcal{U})$ as its stationary distribution.

Note that each complete conditional density is proportional to the joint density as a function of its block of parameters, for example

$$\begin{aligned}f(\boldsymbol{\tau}_1|\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H,\mathcal{U}) &= \frac{f(\mathcal{U}|\boldsymbol{\tau}_1,\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H)f(\boldsymbol{\tau}_1,\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H)}{\int_{\tau_1} f(\mathcal{U}|d\boldsymbol{\tau}_1,\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H)f(d\boldsymbol{\tau}_1,\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H)} \\ &\propto f(\mathcal{U}|\boldsymbol{\tau}_1,\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H)f(\boldsymbol{\tau}_1,\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H)\end{aligned} \tag{1.8}$$

as a function of $\boldsymbol{\tau}_1$, holding the other blocks $\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H$ and the data $\mathcal{U}$ fixed. Thus, when the likelihood $f(\mathcal{U}|\boldsymbol{\tau}_1,\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H)$ and prior $f(\boldsymbol{\tau}_1,\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H)$ factor into a product of terms involving separate blocks of the partition $(\boldsymbol{\tau}_1,\boldsymbol{\tau}_2,\ldots,\boldsymbol{\tau}_H)$, it is easy to "pick out" a function proportional to the complete conditional, by simply retaining those terms in the joint density that depend on $\boldsymbol{\tau}_1$. Examples of this idea will be presented in Section 1.7.

It should be noted that there is nothing special about the order in which parameters are sampled in Figure 1.1, or even the fact that each parameter is sampled once per complete step of the algorithm. In designing an MCMC algorithm, we are free to change this *scan order* in any way that we wish, as long as each parameter is visited a non-vanishing fraction of the time. A basic result discussed by Hastings (1970, p. 102) and Tierney (1994, p. 1710), and referred to as the "product of kernels" principle by Chib and Greenberg (1995, p. 332), guarantees that all such scan orders will have the same stationary distribution. We will return to this idea in Section 1.8.7.

In Sections 1.4, 1.5 and 1.6 we discuss common, generic approaches to constructing transition kernels for MCMC Markov chains and making inferences from them, and we illustrate these methods in Sections 1.7 and 1.8. It is not necessary to routinely include the forms of complete conditionals or the methods used to sample from them in published journal articles, except when the complete conditionals or sampling methodology deserve special theoretical or pedagogical attention.

1.4 Gibbs Sampling

An MCMC algorithm is simplest to implement when the complete conditionals in Figure 1.1 can be written in closed form, and can be sampled from directly. In this case, the MCMC algorithm is called a *Gibbs sampler*. This is the kind of sampling scheme first made popular in the statistics literature (Gelfand and Smith, 1990), and it remains the focal method today.

When directly sampling from a complete conditional is not possible, standard alternate Monte Carlo methods can be used. If $f(t)$ is a density that is difficult to sample from, several standard methods can be used, for example:

- If the cumulative distribution function (cdf)

$$F(t) = \int_{-\infty}^{t} f(s)ds$$

  and its inverse $F^{-1}(u)$ can be calculated in closed form, then it is easy to see that $T^* = F^{-1}(U)$, where $U \sim Unif(0,1)$, will have $f(t)$ as its density. This is called *inversion sampling*.

- Suppose $g(t)$ is a density that is easy to sample from, with the property that $f(t) < Cg(t)$ for some constant C and all t. The density $g(t)$ is called the *proposal density*. Let $T^* \sim g(t)$ and $U \sim Unif(0,1)$. If $UCg(T^*) \leq f(T^*)$ then it can be shown, with a little calculus, that T^* has $f(t)$ as its density. Otherwise we reject it and sample another pair (T^*, U), repeating until we accept a T^*. This is called *rejection sampling*.

- Rejection sampling clearly is most efficient (not many rejections per accepted T^*) when $C \approx 1$. However, finding $m(t)$ that is easy to sample from with $C \approx 1$ can be difficult. If $f(t)$ is a log-concave density, then $\log f(t)$ can be enclosed in a piecewise linear function which, when exponentiated, can serve as $m(t)$. This piecewise linear function can be improved (so that C comes closer to 1) each time T^* is rejected, as outlined in Gilks and Wild (1992); the resulting method is *adaptive rejection sampling*.

Early versions of the BUGS/WinBUGS program relied primarily on these three methods, for one-variable-at-a-time complete conditionals, to automatically create Gibbs samplers for a variety of Bayesian models (Lunn, Spiegelhalter, Thomas, and Best, 2009).

An advantage of rejection sampling, especially for sampling from complete conditionals that are known only proportionally as in Equation 1.8, is that $f(t)$ need only be known up to a constant of proportionality. It is easy to see that any required normalizing constant for $f(t)$ can be absorbed into the constant C: $f(t)/B < Cg(t)$ if and only if $f(t) < BCg(t)$, and $UCg(T^*) \leq f(T^*)/B$ if and only if $UBCg(T^*) \leq f(T^*)$. Other direct sampling methods, such as such as slice sampling (Neal, 2003) and importance sampling (Ripley, 1987) also have this property.

1.5 Metropolis-Hastings

A second popular mechanism for generating an MCMC algorithm is known as the *Metropolis-Hastings* algorithm (Chib, Greenberg, and Chiband, 1995; Hastings, 1970; Metropolis et al., 1953). The Metropolis-Hastings algorithm actually involves a different transition kernel than the one outlined in Equation 1.8 but one with the same stationary distribution. Results reviewed in Tierney (1994) guarantee that using Metropolis-Hastings sampling for some complete conditionals and Gibbs sampling for others maintains $f(\boldsymbol{\tau}|\mathcal{U})$ as the stationary distribution.

For the complete conditional $f(\boldsymbol{\tau}_k|rest)$, to implement a Metropolis-Hastings step we first sample $\tau_k^* \sim g_m(\boldsymbol{\tau}_k|\boldsymbol{\tau}_k^{(m-1)})$, where $g_m(\boldsymbol{\tau}_k|\boldsymbol{\tau}_k^{(m-1)})$ is a *proposal density*, which will be discussed further below. Then we calculate the acceptance probability

$$\alpha^* = \min\left\{\frac{f(\boldsymbol{\tau}_k^*|rest)g_m(\boldsymbol{\tau}_k^{(m-1)}|\boldsymbol{\tau}_k^*)}{f(\boldsymbol{\tau}_k^{(m-1)}|rest)g_m(\boldsymbol{\tau}_k^*|\boldsymbol{\tau}_k^{(m-1)})}, 1\right\} \tag{1.9}$$

and generate $U \sim Unif(0,1)$. If $U \leq \alpha^*$, we set $\boldsymbol{\tau}_k^{(m)} = \boldsymbol{\tau}_k^*$, otherwise we set $\boldsymbol{\tau}_k^{(m)} = \boldsymbol{\tau}_k^{(m-2)}$.

The proposal density $g_m(\boldsymbol{\tau}_k|\boldsymbol{\tau}_k^{(m-1)})$ can be chosen to be any convenient density. Two common choices are

- *Independence M-H.* Take the proposal density $g_m(\boldsymbol{\tau}_k|\boldsymbol{\tau}_k^{(m-1)}) = g_m(\boldsymbol{\tau}_k)$, independent of $\boldsymbol{\tau}_k^{(m-1)}$.
- *Normal random walk M-H.* Take the proposal density $g_m(\boldsymbol{\tau}_k|\boldsymbol{\tau}_k^{(m-1)}) = n(\boldsymbol{\tau}_k|\boldsymbol{\mu} = \boldsymbol{\tau}_k^{(m-1)}, \boldsymbol{\Sigma})$, a normal density with mean $\boldsymbol{\tau}_k^{(m-1)}$ and variance matrix $\boldsymbol{\Sigma}$.

Note that if $g_m()$ is constant (or symmetric) in $\boldsymbol{\tau}_k$ and $\boldsymbol{\tau}_k^{(m-1)}$, then the $g_m()$ terms drop out of Equation 1.9, and the algorithm tends to move toward the mode of $f(\boldsymbol{\tau}_k|rest)$.

It is also worth noting that if $g_m(\boldsymbol{\tau}_k|\boldsymbol{\tau}_k^{(m-1)}) = f(\boldsymbol{\tau}_k|rest)$ then the M-H step reduces to a Gibbs step: we are sampling from $f(\boldsymbol{\tau}_k|rest)$ and always accepting τ_k^* since in this case all terms cancel in Equation 1.9, leaving $\alpha^* = 1$. In this sense, Gibbs sampling is in fact a special case of Metropolis-Hastings. A typical MCMC algorithm will intersperse Gibbs steps—for complete conditionals that can be sampled directly—with Metropolis-Hastings steps—for complete conditionals that cannot be sampled directly. This class of algorithms is called, somewhat inaccurately, *Metropolis-Hastings within Gibbs*.

Results reviewed in Rosenthal (2011) suggest that the MCMC algorithm will achieve good mixing and convergence to the stationary distribution, if

$g_m(\boldsymbol{\tau}_k|\boldsymbol{\tau}_k^{(m-1)})$ is chosen so that if $\boldsymbol{\tau}_k$ is one dimensional, the rate at which τ_k* is accepted is approximately 0.44, and if $\boldsymbol{\tau}_k$ is of dimension d, the acceptance rate should fall to 0.234 for $d \approx 5$ or more. In fact, the mixing and rate of convergence tend to be good as long as the acceptance rate is roughly between 0.1 and 0.6 (Rosenthal, 2011, Figure 4.5).

For example, the normal random walk proposal density for unidimensional τ_k is

$$g_m(\tau_k|\tau_k^{(m-1)}) = \frac{1}{\sqrt{2\pi\sigma_g^2}} e^{-\frac{1}{2}(\tau_k - \tau_k^{(m-1)})^2/\sigma_g^2} .$$

The proposal variance σ_g^2 is a tuning parameter than can be adjusted until the acceptance rate is in the rough neighborhood of 0.44.

Rosenthal (2011) outlines a general theory for adaptive MCMC, in which details of the Markov Chain can be tuned "on the fly" without undermining convergence to the stationary distribution. One such adaptation would be to adjust the variances of normal random walk M-H steps to achieve the above target acceptance rates. Indeed, when M-H steps were introduced into WinBUGS (Lunn et al., 2009), they were introduced in an adaptive form, so that σ_g^2 was adjusted toward a target acceptance rate between 0.2 and 0.4.

1.6 Tricks of the Trade

1.6.1 Initial State, Burn-In, and Convergence

From Equation 1.4, it is clear that the best choice for an *initial state* for the Markov chain would be a draw from the stationary distribution, since then there would be no burn-in segment. We cannot do this (if we could, we wouldn't need MCMC!) but it does suggest starting the chain somewhere near the center of the posterior density $f(\boldsymbol{\tau}|\mathcal{U})$. The default starting values in WinBUGS are draws from the prior distribution; this only makes sense if the prior and the posterior are centered similarly.

In practice, one often proceeds by simulating starting values from normal, t, or similar distributions, centered at provisional parameter estimates (e.g., maximum likelihood or method of moments estimates from a simpler model), overdispersed relative to provisional standard errors. MCMC is fundamentally a local search algorithm, and starting the algorithm from several well-dispersed initial states can help to ensure that the parameter space is fully explored. In addition, starting multiple chains at well-dispersed initial states facilitates correct interpretation of convergence indices such as the $\hat{R}$ statistic (Brooks and Gelman, 1998; Gelman and Rubin, 1992). On the other hand we have found IRT and similar models to be somewhat delicate with respect to starting values, a point we will discuss further below in Sections 1.8.5 and 1.8.6.

As discussed in Section 1.3, the *burn-in* segment of a Markov chain is

the initial segment of the chain before it has *converged*—that is, before samples from the chain are like samples from the stationary distribution $\pi(\boldsymbol{\tau}) \equiv f(\boldsymbol{\tau}|\mathcal{U})$. These samples of the chain should be discarded; only samples after burn-in are useful for inference about the posterior distribution $f(\boldsymbol{\tau}|\mathcal{U})$.

Although many methods have been proposed as heuristics for assessing convergence (Cowles and Carlin, 1996), only three or four methods stand out as convenient enough and informative enough to have become standard practice. Apart from perfect sampling (Fill, 1998) and regenerative methods (Mykland, Tierney, and Yu, 1995), there are no guaranteed methods of determining when an MCMC algorithm has "converged" to its stationary distribution. When developing or applying an MCMC algorithm, one should always look at trace plots, or summaries of trace plots, and autocorrelation function (acf) plots. These are basic graphical tools for assessing whether the MCMC algorithm is functioning well, and they are also useful for assessing burn-in, mixing and convergence to the stationary distribution.

Trace Plots

Trace plots or time-series plots are simply plots of $g(\boldsymbol{\tau}^{(m)})$, as a function of m, $m = 1, \ldots, M$. Most often, $g(\boldsymbol{\tau}) = \tau_j$, one of the parameters in the model. But it could be anything. To aid visual detection of some of these problems, it is useful to add a horizontal line at the mean or median of the graphed values, and/or a smoothed running mean or median curve.

Ideal plots look like white (unpatterned) noise centered at the posterior mean or median, as in Figure 1.13. On the other hand, Figure 1.2 illustrates four common problems in trace plots:

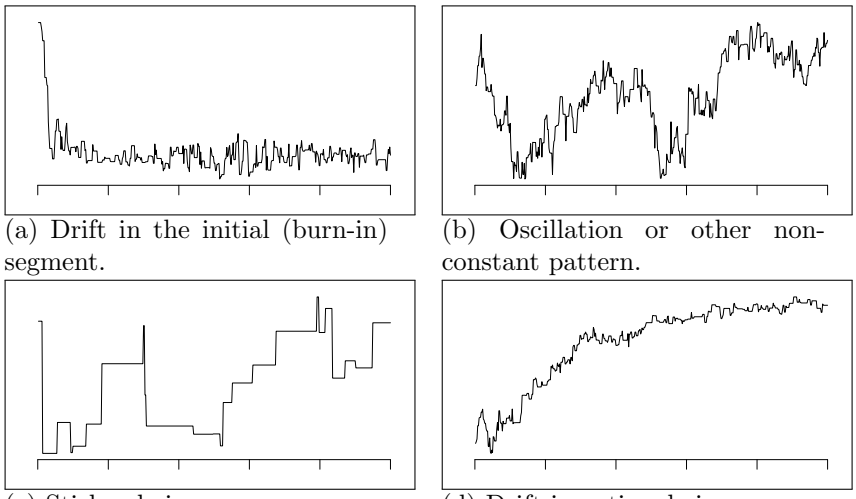
(a) An initial segment that looks like it "drifted in from somewhere else";

(b) A low-frequency cycle or other pattern in the graph (e.g. the white noise is following a large sinusoid rather than being centered on a horizontal line);

(c) Excessive "stickiness", that is, the trace plot stays constant for several steps before moving to a new value;

(d) An overall trend or drift upwards or downwards across the whole trace plot.

Problem (a) is due to the early part of the chain not being very near the stationary distribution. The usual fix is to throw away the initial (burn-in) part of the chain, and use the stable remaining part for inference. Problem (b) is often due to excessive dependence between steps in the MCMC algorithm. This can happen with parameters that are sampled with random-walk Metropolis-Hastings steps if the proposal variance is too small, for example. It can also happen with parameters that are sampled with Gibbs steps. One can live with this problem if MCMC sampling is fast enough that one can take many many



(a) Drift in the initial (burn-in) segment.

(b) Oscillation or other non-constant pattern.

(c) Sticky chain.

(d) Drift in entire chain.

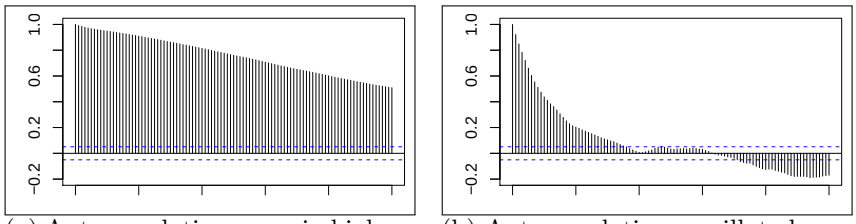
FIGURE 1.2
A collection of trace plots indicating various problems.

more steps M to get precise estimates of posterior quantities. The alternative is to redesign the chain so as to reduce this dependence. Problem (c) is often seen in parameters that are sampled with Metropolis-Hastings steps, and it is simply due to too low an acceptance rate, e.g., too high a proposal variance in random-walk Metropolis-Hastings. Tune or change the proposal density to fix this. Problem (d) may be a special case of problems (a) or (b), which one would discover by running the chain longer to see what happens. If the drift persists even after running the chain a very long time, it may be evidence of problems in the underlying statistical model, Equation 1.2. For example, the posterior density $f(\boldsymbol{\tau}|\mathcal{U})$ may not be sufficiently peaked around a dominant posterior mode (e.g., not enough data), or there may be an identification problem (e.g., the MCMC samples for one parameter drift toward $+\infty$, and the MCMC samples for another drift toward $-\infty$ but their sum stays constant).

Autocorrelation plots

Autocorrelation plots and cross-correlation plots are useful for directly assessing dependence between steps of the chain. Let $\boldsymbol{\tau}^{(m)}$, $m = 1, 2, 3, \ldots$ be the output from the MCMC algorithm, and consider functions $g_1(\boldsymbol{\tau})$ and $g_2(\boldsymbol{\tau})$. The *autocorrelation function* (acf) for $g_1(\boldsymbol{\tau})$ is

$$\rho_k^{g_1} = \frac{\text{Cov}\left(g_1(\boldsymbol{\tau}^{(m)}), g_1(\boldsymbol{\tau}^{(m+k)})\right)}{\text{Var}\left(g_1(\boldsymbol{\tau}^{(m)})\right)}$$



(a) Autocorrelations remain high.

(b) Autocorrelations oscillate between values near and far from 0.

FIGURE 1.3
Two common problems evident in autocorrelation plots.

and similarly for $g_2(\boldsymbol{\tau})$, and the cross-correlation function is

$$\sigma_k^{g_1 g_2} = \frac{\text{Cov}\left(g_1(\boldsymbol{\tau}^{(m)}), g_2(\boldsymbol{\tau}^{(m+k)})\right)}{\sqrt{\text{Var}\left(g_1(\boldsymbol{\tau}^{(m)})\right)\text{Var}\left(g_2(\boldsymbol{\tau}^{(m+k)})\right)}}$$

(neither function depends on m if the Markov chain is stationary). The acf plot is a plot of $\hat{\rho}_k^{g_1}$ as a function of k, and the cross-correlation plot is a plot of $\hat{\sigma}_k^{g_1 g_2}$. These estimates are usually straightforward method-of-moments estimates; and are reasonably useful if k is not large compared to M. Note that $\hat{\rho}_0^{g_1}$ is forced to be 1, and then $\hat{\rho}_k^{g_1}$ should fall toward zero in absolute value as k increases. The cross-correlation plot should exhibit similar behavior, except that $\sigma_k^{g_1 g_2}$ is not constrained to be equal to 1. To aid in interpreting the plots, it is useful to add a horizontal line at zero correlation to the graph, and also add lines indicating the rejection region for a test of the hypothesis that $\rho_k^{g_1} = 0$, or the hypothesis that $\sigma_k^{g_1, g_2} = 0$.

Problematic behavior in acf and cross-correlation plots include

(a) $\rho_k^{g_1}$ (or $\sigma_k^{g_1 g_2}$) remains significantly different from zero for all observable values of k;

(b) $\rho_k^{g_1}$ (or $\sigma_k^{g_1 g_2}$) oscillates between values close to zero and values far from zero;

examples are shown in Figure 1.3. Both problems (a) and (b) reflect excessive autocorrelation (or cross correlation) in the Markov chain. These problems, especially persistent positive autocorrelation, are often associated with problems (a), (b) or (c) in Figure 1.2, for example. One can live with this problem if MCMC sampling is fast enough that one can take many many more steps M (and perhaps discard an initial burn-in segment) to get precise estimates of posterior quantities. The alternative is to redesign the chain so as to reduce this dependence. For example, a "good" autocorrelation plot can be seen later in the chapter, in Figure 1.13.

Single and Multiple Chain Methods

There exist several heuristic methods for assessing convergence of an MCMC algorithm to its stationary distribution, based on a single run of the algorithm. A visual inspection of trace plots and autocorrelation plots may be enough: if the trace plots look like white noise centered at the mean or median of the output, and the acf plot drops quickly to zero and stays there, we may have some confidence that the chain has reached its stationary distribution. As a quantitative check, one can break a long chain into two (or more) parts and compare posterior mean and variance estimates, density plots of the posterior, etc., from the different parts of the chain; if the chain has converged and the parts of the chain are long enough, the posterior summaries should be similar. A more formal measure of convergence along the same lines is the time series diagnostic of Geweke (1992).

Another approach to assessing the convergence of an MCMC algorithm is to generate three or more chains from different starting values. If the trace plots overlap one another to a great extent after a suitable burn-in segment is discarded, this is some assurance that the algorithm is mixing well and has reached the stationary distribution. If the chains converge quickly (have short burn-in segments) and not much autocorrelation, then samples from the multiple chains can be pooled together to make inferences about the posterior distribution. Since the chains do not depend on each other in any way, they can be run in parallel on either multiple cores or multiple computers to generate output faster. If convergence to the stationary distribution is slow, it may be more efficient to run one long chain, to avoid spending computation on the burn-in for multiple chains.

An informal quantitative assessment of convergence using multiple chains would be to compare posterior mean and variance estimates, posterior density plots, etc., across the multiple chains. The $\hat{R}$ statistic (Brooks and Gelman, 1998; Gelman and Rubin, 1992) formalizes and quantifies this idea, essentially by examining the ratio of between-chain variation to within-chain variation. When all the chains have achieved stationarity, the between- and within-variation will be comparable; otherwise the between-variation is likely to be larger than the within-variation. A common heuristic is to declare convergence if $\hat{R} < 1.1$.

The above methods for assessing convergence of the MCMC algorithm for univariate functions $g(\boldsymbol{\tau})$, as well as several methods for assessing multivariate convergence, are implemented in the software package BOA (Bayesian output analysis) of Smith (2007), available at http://www.public-health.uiowa.edu/boa), which derives from the methodology surveyed in Cowles and Carlin (1996). The graphical and multiple chain methods above are also readily available in WinBUGS and R interfaces to WinBUGS such as `R2WinBUGS` (Gelman, Sturtz, Legges, Gorjanc, and Kerman, 2012) and `rube` (Seltman, 2010).

Thinning and Saving Values From the Converged Chain

It is fairly common practice to "thin" the Markov chain in some way, for example, to retain only every m_{thin}^{th} step of of the converged Markov chain. Thinning is useful for reducing autocorrelation, and for reducing the amount of MCMC data one has to store and calculate with. A common procedure to determine the thinning interval m_{thin} is to examine autocorrelation plots from a preliminary run with no thinning, and then for the final run take m_{thin} to be an interval at which the autocorrelation is near zero. Because of uncertainty of estimation in the acf plot, however, there will still be some autocorrelation in the retained MCMC samples, and this should be addressed using a method like that of Equation 1.11 in Section 1.6.2, to understand the precision of inferences from the MCMC output. Because thinning involves throwing away perfectly good draws from the posterior distribution, some authors advocate against thinning to reduce autocorrelation. For example, in large interesting models, requiring small autocorrelations can result in $m_{thin} = 50$ or 100 or more; 98% or more of the computation is then wasted. In such cases it may be possible (and is desirable) to reformulate the model or MCMC algorithm in order to reduce autocorrelation and make the thinning interval and hence computational efficiency more reasonable.

Once you are satisfied that the chain has converged to its stationary distribution, and the burn-in segment has been discarded, the problem remains to decide how many steps of the chain to save, for inference about the posterior distribution. One rule of thumb is to keep roughly 1000 steps from the chain or chains, after burn-in and thinning if any of this is done. For example if one wishes to calculate 2.5^{th} and 97.5^{th} percentiles in order to report equal-tailed 95% posterior credible intervals, 1000 is a minimal sample size. A more principled approach might be to monitor the Monte Carlo standard error (discussed below in Section 1.6.2) as the length of the chain grows: we want the Monte Carlo standard error to be much smaller than the posterior standard error, and small enough to ensure a couple of digits of accuracy in posterior mean estimates.

It does not seem necessary to publish trace plots, acf plots, or Monte Carlo standard errors routinely in journal articles (except when one is trying to make a point about the behavior of the MCMC algorithm), but they should be available in on-line supplements for referees or interested readers. Numerical evidence of good mixing and convergence to the stationary distribution (Brooks and Gelman, 1998; Gelman and Rubin, 1992; Geweke, 1992) is useful for published papers.

1.6.2 Monte Carlo Standard Error and Posterior Uncertainty

Two different uncertainty calculations are important to carry out in MCMC estimation of posterior quantities. For specificity, suppose we are interested in

estimating a function of $\boldsymbol{\tau}$, $g(\boldsymbol{\tau})$, using MCMC output $\boldsymbol{\tau}^{(m)}$, $m = 1, \ldots, M$. For example, to estimate the k^{th} univariate parameter in $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_K)$, we would take $g(\boldsymbol{\tau}) = \tau_k$.

- *Monte Carlo Uncertainty:* Suppose we are interested in estimating the posterior mean

$$E[g(\boldsymbol{\tau})|\mathcal{U}] \approx \frac{1}{M} \sum_{m=1}^{M} g(\boldsymbol{\tau}^{(m)}) \equiv \overline{g}$$

  How accurate is $\overline{g}$ as an estimate of $E[g(\boldsymbol{\tau})|\mathcal{U}]$? We would like to compute a Monte Carlo standard error, SE_{MCMC} and provide a Monte Carlo confidence interval, e.g. $\overline{g} \pm t^* SE_{MCMC}$. We can make this Monte Carlo estimate more precise (reduce SE_{MCMC}) simply by increasing the MCMC sample size M, without collecting any more data.

  Confidence intervals expressing MCMC estimation error are more readily interpretable to the extent that the central limit theorem (CLT) applies to the chain $\boldsymbol{\tau}^{(m)}$, $m = 1, 2, 3, \ldots$. If the chain is reversible, that is, satisfies Equation 1.5, then a CLT can be obtained (Geyer, 2011). If not, the CLT depends on *geometric ergodicity* (Flegal and Jones, 2011), which says essentially that the difference (or, technically, the total variation distance) between the distribution of $\boldsymbol{\tau}^{(m)}$ and $f(\boldsymbol{\tau}|\mathcal{U})$ tends to zero like r^m for some number $r \in (0, 1)$, and a moment condition such as $E[g(\boldsymbol{\tau})^{2+\delta}|\mathcal{U}] < \infty$. Many, but not all, common MCMC algorithms satisfy this or stronger sufficient conditions for the CLT to hold.

- *Posterior Uncertainty:* As an inference about $\boldsymbol{\tau}$ from the data, we may be interested in summarizing the posterior distribution of $g(\boldsymbol{\tau})$ using the posterior mean $E[g(\boldsymbol{\tau})|\mathcal{U}]$ and the posterior standard error $SE_{post} = \sqrt{\text{Var}\,(g(\boldsymbol{\tau})|\mathcal{U})}$. For example we may wish to report a posterior credible interval such as $E[g(\boldsymbol{\tau})|\mathcal{U}] \pm t^* \cdot SE_{post}$ for some suitable cutoff t^*. We can make our inference about $g(\boldsymbol{\tau})$ more precise (reduce SE_{post}) by collecting more data. Increasing the MCMC sample size M can make our estimates of $E[g(\boldsymbol{\tau})|\mathcal{U}]$ and SE_{post} more precise, but it can't structurally improve the precision of our inference about $g(\boldsymbol{\tau})$. Intervals based on estimates of $E[g(\boldsymbol{\tau})|\mathcal{U}]$ and SE_{post} are only interpretable, to the extent that a posterior CLT holds as more data is collected (e.g. Chang and Stout, 1993; Walker, 1969).

  If there is any concern about the shape of the posterior density, it is better to construct credible intervals directly from the posterior density. For example, the equal-tailed credible interval running from the 0.025 posterior quantile to the 0.975 posterior quantile is guaranteed to produce an interval with 95% posterior probability, regardless of whether the posterior CLT holds or not.

To estimate SE_{MCMC} we must first estimate $\sigma_g^2 = \text{Var}\,(g_m)$, the variance of the sampled values of the Markov chain (which is due to both the shape

of the posterior distribution and Markov sampling). Because the $\boldsymbol{\tau}^{(m)}$ are dependent, the naive estimator

$$\hat{\sigma}^2_{naive} = \frac{1}{M-1}\sum_{m=1}^{M}(g(\boldsymbol{\tau}^{(m)}) - \overline{g})^2 \tag{1.10}$$

is seldom adequate. Several valid methods are currently in use (Flegal and Jones, 2011; Geyer, 2011) but we will focus on only one: the method of *overlapping batch means (OLMB)* (Flegal and Jones, 2011). Define b_M to be the batch length, and define batches $B_1 = (\boldsymbol{\tau}^{(1)}, \boldsymbol{\tau}^{(2)}, \ldots, \boldsymbol{\tau}^{(b_M)})$, $B_2 = (\boldsymbol{\tau}^{(2)}, \boldsymbol{\tau}^{(3)}, \ldots, \boldsymbol{\tau}^{(b_M+1)})$, $B_3 = (\boldsymbol{\tau}^{(3)}, \boldsymbol{\tau}^{(4)}, \ldots, \boldsymbol{\tau}^{(b_M+2)})$, etc. There are $M - b_M + 1$ such batches in an MCMC run of length M. Now let

$$\begin{aligned}
\overline{g}_j &= \frac{1}{b_M}\sum_{\boldsymbol{\tau}^{(m)}\in B_j} g(\boldsymbol{\tau}^{(m)}) \text{ , and} \\
\hat{\sigma}^2_{OLBM} &= \frac{Mb_M}{(M-b_M)(M-b_M+1)}\sum_{j=1}^{M-b_M+1}(\overline{g}_j - \overline{g})^2 \text{ .}
\end{aligned} \tag{1.11}$$

Flegal and Jones (2011) argue that under conditions similar to those needed for the CLT to apply, $\hat{\sigma}^2_{OLBM}$ will be a consistent estimator of σ^2_g, as long as $B_M \approx M^{1/2}$. In that case, we can take

$$SE_{MCMC} \approx \sqrt{\hat{\sigma}^2_{OLBM}/M} \text{ .}$$

The MCMC confidence interval is then $\overline{g} \pm t^* \cdot SE_{MCMC}$ where t^* is a cutoff from a t-distribution with $M - b_M$ degrees of freedom, although for many practical purposes simply taking $t^* = 2$ gives an adequate impression (since the degrees of freedom $M - b_M$ are likely to be quite large). An alternative method with simpler-to-calculate non-overlapping batch means is described in Flegal, Haran, and Jones (2008); the non-overlapping approach described here however is generally thought to be more efficient.

When the Markov chain is ergodic it is reasonable to estimate $SE_{post} \approx \sqrt{\sigma^2_{naive}}$ in the sense that the right-hand side will converge to the left as M grows. To know how accurate the estimate is, one can exploit the identity $\mathrm{Var}\,(g(\boldsymbol{\tau})|\mathcal{U}) = E[(g(\boldsymbol{\tau}) - E[g(\boldsymbol{\tau})|\mathcal{U}])^2|\mathcal{U}] = E[g(\boldsymbol{\tau})^2|\mathcal{U}] - E[g(\boldsymbol{\tau})|\mathcal{U}]^2$, separately estimating

$$\begin{aligned}
E[g(\boldsymbol{\tau})|\mathcal{U}] &\approx \frac{1}{M}\sum_{m=1}^{M} g(\boldsymbol{\tau}^{(m)}) \equiv \overline{g} \text{ , and} \\
E[g(\boldsymbol{\tau})^2|\mathcal{U}] &\approx \frac{1}{M}\sum_{m=1}^{M} g(\boldsymbol{\tau}^{(m)})^2 \equiv \overline{g^2} \text{ ,}
\end{aligned}$$

and monitoring the MCMC standard error of both estimates to make sure that we do not get excessive cancellation in the difference $\overline{g^2} - \overline{g}^2$. An MCMC

standard error for $\widehat{SE}_{post} = \sqrt{\sigma^2_{naive}}$ can then be obtained using the delta method (e.g., Serfling, 1980, p. 124).

When the posterior distribution is not approximately normal, intervals based on (estimates of) $E[g(\boldsymbol{\tau})|\mathcal{U}]$ and SE_{post} may be misleading; a better measure of posterior uncertainty can be based on quantiles of the posterior distribution, estimated as empirical quantiles of the MCMC sequence $g(\boldsymbol{\tau}^{(m)})$, $m = 1, \ldots, M$. For example, one might report the empirical median as a point estimate for $g(\boldsymbol{\tau})$ and the empirical 0.025 and 0.975 quantiles as an approximate equal-tailed 95% credible interval. Flegal and Jones (2011) also give methods for estimating the MCMC standard errors of these quantiles.

MCMC standard errors seldom find their way into published empirical papers. Largely speaking there may be little harm done by this, as long as authors can assure themselves and their readers (or referees) that the magnitude of the posterior uncertainty overwhelms the magnitude of the MCMC uncertainty, as is almost always the case. MCMC standard errors would be important to report in cases where they are comparable in size to posterior standard errors, or in cases where estimating a posterior quantity (such as $E[g(\boldsymbol{\tau})|\mathcal{U}]$, $\text{Corr}\,(\tau_1, \tau_2|\mathcal{U})$, etc.) is of primary interest.

Those who wish to apply MCMC methods in practical testing settings should note carefully that Monte Carlo uncertainty and posterior uncertainty are qualitatively different and do not carry equal weight in evaluating a method's utility. For example, Monte Carlo uncertainty is probably unacceptable in test scoring, unless it can be reduced to a level well below the number of digits used in reporting scores. Stakeholders in testing will not find it acceptable to think that pass/fail decisions or rankings might be sensitive to such factors as the seed of the random number generator, or the length of the chain that was run to obtain the estimate. This is perhaps not entirely unique to MCMC, and we know that maximum-likelihood-based scoring has its own tolerances and sensitivities. MCMC is much more readily usable in item/test calibration, where any imprecision due to estimation method is reflected in the "level playing field" that all examinees encounter when answering test questions. If we consider that such matters as 'adequate yearly progress' for schools might be sensitive to Monte Carlo error in calibration and/or linking, there is still reason (as if there were not enough reasons already) to be concerned. Addressing these types of concerns is on the one hand just good psychometric practice (know the precision of any instruments and make decisions accordingly), but on the other hand might be important for adoption of MCMC for IRT models to move from the research literature into the professional practice.

1.6.3 Blocking

As should be clear from our discussion at the end of Section 1.3, it is not necessary for the complete conditional densities to be univariate densities for each single parameter in the model. If some parameters are known (from theory, or from preliminary estimation) to be highly correlated in the posterior

$f(\boldsymbol{\tau}|\mathcal{U})$, then it makes sense to group them together into a block and sample them together. Posterior correlation corresponds to a ridge in the posterior distribution, and it is usually more efficient to explore the ridge with MCMC steps parallel to the ridge (as would be the case for blocked MCMC steps) rather than steps parallel to the coordinate axes (as would be the case for one-parameter-at-a-time MCMC steps).

If a block of parameters can be sampled in a Gibbs step, the form of the complete conditional for that block will have the appropriate association structure in it. If the block must be sampled using a M-H step, a common strategy is to use a normal random walk proposal, with a proposal variance-covariance matrix that reflects the anticipated posterior correlation in that block. If "good" proposal distributions to update blocks of parameters can be found, block-sampled MCMC can provide a significant improvement over one-parameter-at-a-time MCMC (Doucet, Briers, and Sénécal, 2006).

1.6.4 Data Augmentation

Suppose we have a model of the form Equation 1.2 for a set of parameters $\boldsymbol{\tau}_1$ of interest, $f(\mathcal{U}, \boldsymbol{\tau}_1) = f(\mathcal{U}|\boldsymbol{\tau}_1)f(\boldsymbol{\tau}_1)$, and we wish to calculate the posterior mean $E[g(\boldsymbol{\tau}_1)|\mathcal{U}]$ for the function $g(\boldsymbol{\tau}_1)$. According to the results of Section 1.3 we could do this directly with the output $\boldsymbol{\tau}_1^{(m)}$, $m = 1, \ldots, M$, from an MCMC algorithm for the posterior density $f(\boldsymbol{\tau}_1|\mathcal{U})$, as

$$E[g(\boldsymbol{\tau}_1)|\mathcal{U}] = \int g(\boldsymbol{\tau}_1) f(d\boldsymbol{\tau}_1|\mathcal{U}) \approx \frac{1}{M} \sum_{m=1}^{M} g(\boldsymbol{\tau}_1^{(m)})$$

for any integrable function $g()$.

On the other hand, suppose we expand the model to include additional parameters $\boldsymbol{\tau}_2$, $f(\mathcal{U}, \boldsymbol{\tau}_1, \boldsymbol{\tau}_2) = f(\mathcal{U}|\boldsymbol{\tau}_1, \boldsymbol{\tau}_2)f(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2)$, and we obtain an MCMC sample $(\boldsymbol{\tau}_1^{(m)}, \boldsymbol{\tau}_2^{(m)})$, $m = 1, \ldots, M$, from the expanded posterior $f(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2|\mathcal{U})$. We can still calculate

$$\begin{aligned} E[g(\boldsymbol{\tau}_1)|\mathcal{U}] &= \int g(\boldsymbol{\tau}_1) f(d\boldsymbol{\tau}_1|\mathcal{U}) = \int\int g^*(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2) f(d\boldsymbol{\tau}_1, d\boldsymbol{\tau}_2|\mathcal{U}) \\ &\approx \frac{1}{M} \sum_{m=1}^{M} g^*(\boldsymbol{\tau}_1^{(m)}, \boldsymbol{\tau}_2^{(m)}) = \frac{1}{M} \sum_{m=1}^{M} g(\boldsymbol{\tau}_1^{(m)}) \end{aligned}$$

for any integrable $g()$, where $g^*()$ is simply the function $g^*(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2) = g(\boldsymbol{\tau}_1)$. This shows that the act of "throwing away" some coordinates of an MCMC sample is equivalent to integrating them out of a marginal expected value calculation (Patz and Junker, 1999a).

Moreover, the calculations above suggest a strategy that is often successful: when it is difficult to construct an efficient, well-mixing Markov chain for $f(\boldsymbol{\tau}_1|\mathcal{U})$, it may be much easier to do so for $f(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2|\mathcal{U})$. As long as

$$f(\boldsymbol{\tau}_1|\mathcal{U}) = \int f(\boldsymbol{\tau}_1, d\boldsymbol{\tau}_2|\mathcal{U})$$

then we can just "throw away" $\boldsymbol{\tau}_2^{(m)}$, $m = 1, \ldots, M$ and use $\boldsymbol{\tau}_2^{(m)}$, $m = 1, \ldots, M$ to make inferences on the marginal density $f(\boldsymbol{\tau}_1|\mathcal{U})$.

This strategy is called *data augmentation*, first formally suggested by Tanner and Wong (1987), and implemented for the first time in IRT with the with normal ogive model by Albert (1992). Indeed, we know that the normal ogive item response function for dichotomous responses

$$P[U_{pi} = 1|a_i, b_i, \theta_p] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{a_i(\theta_p - b_i)} e^{-t^2/2} dt$$

obtains for U_{pi}, if U_{pi} is defined with the condensation function (Bartholomew and Knott, 1999; Maris, 1995) of $Z_{pi} \sim N(0, 1)$,

$$U_{pi} = \begin{cases} 0, & \text{if } Z_{pi} < -a_i(\theta_p - b_i) \\ 1, & \text{if } Z_{pi} \geq -a_i(\theta_p - b_i) \end{cases} .$$

The parameters a_i, b_i and θ_p would play the role of $\boldsymbol{\tau}_1$, and Z_{pi} would play the role of the data augmentation variables $\boldsymbol{\tau}_2$. This is the basis of the data augmentation approach to analyzing multivariate binary and polytomous probit models (Albert and Chib, 1993), as well as the data augmentation approach to estimating the normal ogive IRT model (Albert, 1992; Fox, 2010; Thissen and Edwards, 2005). Recently progress has been made on an efficient data augmentation scheme for MCMC estimtation of logistic models as well (Polson, Scott, and Windle, 2013).

1.6.5 Rao-Blackwellization

Again consider the model $f(\mathcal{U}, \boldsymbol{\tau}_1, \boldsymbol{\tau}_2) = f(\mathcal{U}|\boldsymbol{\tau}_1, \boldsymbol{\tau}_2) f(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2)$, where $\boldsymbol{\tau}_1$ is of primary interest, and let us examine the problem of estimating $E[g(\boldsymbol{\tau}_1)|\mathcal{U}]$. We have

$$\begin{aligned} E[g(\boldsymbol{\tau}_1)|\mathcal{U}] &= \int\int g(\boldsymbol{\tau}_1) f(d\boldsymbol{\tau}_1, d\boldsymbol{\tau}_2|\mathcal{U}) = \int \left[\int g(\boldsymbol{\tau}_1) f(d\boldsymbol{\tau}_1|\boldsymbol{\tau}_2, \mathcal{U}) \right] f(d\boldsymbol{\tau}_2|\mathcal{U}) \\ &\approx \frac{1}{M} \sum_{m=1}^{M} \int g(\boldsymbol{\tau}_1) f(d\boldsymbol{\tau}_1|\boldsymbol{\tau}_2^{(m)}, \mathcal{U}) \end{aligned}$$

This gives an alternative to

$$E[g(\boldsymbol{\tau}_1)|\mathcal{U}] \approx \frac{1}{M} \sum_{m=1}^{M} g(\boldsymbol{\tau}_1^{(m)})$$

for estimating $E[g(\boldsymbol{\tau}_1)|\mathcal{U}]$. By analogy with the Rao-Blackwell theorem of mathematical statistics, in which conditioning an estimator on the sufficient statistic reduces the variance of the estimator, the so-called *Rao-Blackwellized* (Casella and Robert, 1996) estimator

$$E[g(\boldsymbol{\tau}_1)|\mathcal{U}] \approx \frac{1}{M} \sum_{m=1}^{M} \int g(\boldsymbol{\tau}_1) f(d\boldsymbol{\tau}_1|\boldsymbol{\tau}_2^{(m)}, \mathcal{U}) \tag{1.12}$$

is often better behaved than the simple average. Of course in order to use the Rao-Blackwellized estimator, one has to be able to calculate $\int g(\boldsymbol{\tau}_1) f(d\boldsymbol{\tau}_1 | \boldsymbol{\tau}_2^{(m)}, \mathcal{U})$ many times. This is not usually a problem if the integral can be obtained in closed form or via simple numerical integration; this is often the case if $g()$ is a relatively simple function and conjugate or partially conjugate priors are used (so that the complete conditional in Equation 1.12 can be identified by conjugacy).

A related technique can be used to estimate marginal densities smoothly. Instead of estimating $f(\boldsymbol{\tau}_1|\mathcal{U})$ using a histogram or smooth density estimate using $\boldsymbol{\tau}_1^{(m)}$, $m = 1, \ldots, M$, one can recognize that

$$f(\boldsymbol{\tau}_1|\mathcal{U}) = \int f(\boldsymbol{\tau}_1, d\boldsymbol{\tau}_2|\mathcal{U}) = \int f(\boldsymbol{\tau}_1|\boldsymbol{\tau}_2, \mathcal{U}) f(d\boldsymbol{\tau}_2, \mathcal{U}) \approx \frac{1}{M} \sum_{i=1}^{M} f(\boldsymbol{\tau}_1|\boldsymbol{\tau}_2^{(m)}, \mathcal{U}) \, .$$

Again, this latter estimator may be more stable than the simple histogram estimator.

1.7 Item Response Theory Models

The generic model in Equation 1.1 can usually be rewritten in IRT applications as

$$f(\mathcal{U}|\boldsymbol{\tau}) = \prod_{p=1}^{P} \prod_{i=1}^{I} f(U_{pi}|\boldsymbol{\theta}_p, \boldsymbol{\beta}_i)$$

where the product over p is due to the *experimental independence* assumption in IRT (Lord and Novick, 1968) and the product over i is due to IRT's *local independence* assumption. It is also typical to specify independent priors for all parameters, so that the generic joint distribution in Equation 1.2 has the form

$$\begin{aligned} & f(\mathcal{U}|\boldsymbol{\tau}) f(\boldsymbol{\tau}) \\ &= \prod_{p=1}^{P} \prod_{i=1}^{I} f(U_{pi}|\boldsymbol{\theta}_p, \boldsymbol{\beta}_i) \prod_{p=1}^{P} f_p(\boldsymbol{\theta}_p|\boldsymbol{\lambda}_\theta) \prod_{i=1}^{I} f_i(\boldsymbol{\beta}_i|\boldsymbol{\lambda}_\beta) f(\boldsymbol{\lambda}_\theta) f(\boldsymbol{\lambda}_\beta) \\ &= \prod_{p=1}^{P} \left\{ \prod_{i=1}^{I} f(U_{pi}|\boldsymbol{\theta}_p, \boldsymbol{\beta}_i) f_i(\boldsymbol{\beta}_i|\boldsymbol{\lambda}_\beta) \right\} f_p(\boldsymbol{\theta}_p|\boldsymbol{\lambda}_\theta) f(\boldsymbol{\lambda}_\theta) f(\boldsymbol{\lambda}_\beta) \end{aligned} \tag{1.13}$$

where $\boldsymbol{\lambda}_\theta$ and $\boldsymbol{\lambda}_\beta$ are hyperparameters for $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$, respectively, and $f(\boldsymbol{\lambda}_\theta)$ $f(\boldsymbol{\lambda}_\beta)$ are their hyperprior distributions.

To be concrete, consider the two-parameter logistic (2PL) IRT model, with dichotomous responses $U_{pi} = 1$ if response i from person p is coded as correct

or positive; and $U_{pi} = 0$ if the response is coded as incorrect or negative. In this case $\theta_p \in \Re$ is unidimensional and $\boldsymbol{\beta}_i = (a_i, b_i)$ has two components. The item response function (IRF) is

$$P_i(\theta_p; a_i, b_i) = P[U_{pi} = 1|\theta_p, a_i, b_i] = \frac{\exp(a_i(\theta_p - b_i))}{1 + \exp(a_i(\theta_p - b_i))} \ ,$$

and the density for U_{pi} becomes

$$f(u_{pi}|\theta_p, \boldsymbol{\beta}_i) = P_i(\theta_p; a_i, b_i)^{u_{pi}}(1 - P_i(\theta_p; a_i, b_i))^{1-u_{pi}} \ .$$

In Equation 1.13 we will also use the normal prior distributions for θ_p and b_i and a log-normal prior distribution for a_i,

$$\begin{aligned} f_p(\theta_p|\boldsymbol{\lambda}_\theta) &= n(\theta_p|\mu_\theta, \sigma_\theta^2) && (1.14)\\ f_i(a_i|\boldsymbol{\lambda}_a) &= n(\ln a_i|\mu_a, \sigma_a^2)/a_i && (1.15)\\ f_i(b_i|\boldsymbol{\lambda}_b) &= n(b_i|\mu_b, \sigma_b^2) \ , && (1.16) \end{aligned}$$

where $n(x|\mu, \sigma^2)$ is the normal density with mean μ and variance σ^2. (To be clear, $\boldsymbol{\lambda}_\theta = (\mu_\theta, \sigma_\theta^2)$, $\boldsymbol{\lambda}_a = (\mu_a, \sigma_a^2)$, etc.)

Although $f_p(\theta_p|\boldsymbol{\lambda}_\theta)$ plays the mathematical role of a prior distribution in the model, it is usually interpreted in IRT as the population distribution for θ. For this reason it is not usually taken to be "flat" or uninformative, but rather its shape tells us directly about the distribution of proficiency in the population. Specifying a prior $f(\boldsymbol{\lambda}_\theta)$ allows us to estimate that shape in terms of the parameters $\boldsymbol{\lambda}_\theta$. When $f_p(\theta_p|\boldsymbol{\lambda}_\theta) = n(\theta_p|\mu_\theta, \sigma_\theta^2)$, it is common to place prior distributions on μ_θ and σ_θ^2. For the purposes of illustrating an MCMC algorithm here, we will assume $\mu_\theta = 0$ and $\sigma_\theta^2 \sim IG(\alpha_\theta, \beta_\theta)$, an inverse-gamma distribution with parameters α_θ and β_θ.

We have now fully specified a Bayesian IRT model, elaborating Equation 1.13 as follows.

$$\begin{aligned} U_{pi} &\overset{indep}{\sim} \text{Bernoulli}\,(\pi_{pi}) && (1.17)\\ \ln\frac{\pi_{pi}}{1+\pi_{pi}} &= a_i(\theta_p - b_i) && (1.18)\\ \theta_p &\overset{iid}{\sim} \text{Normal}\,(0, \sigma_\theta^2) && (1.19)\\ a_i &\overset{iid}{\sim} \text{Log-Normal}\,(\mu_a, \sigma_a^2) && (1.20)\\ b_i &\overset{iid}{\sim} \text{Normal}\,(0, \sigma_b^2) && (1.21)\\ \sigma_\theta^2 &\sim \text{Inv-Gamma}\,(\alpha_\theta, \beta_\theta) && (1.22)\\ \mu_a, \sigma_a^2, \sigma_b^2, \alpha_\theta, \beta_\theta & \text{ assigned in Section 1.8 below} && (1.23) \end{aligned}$$

for all $p = 1, \ldots, P$ and all $i = 1, \ldots, I$. Here, $U_{pi} \overset{indep}{\sim} \text{Bernoulli}\,(\pi_{pi}$ means that the U_{pi}'s are independent with densities $\pi_{pi}^{u_{pi}}(1-\pi_{pi})^{1-u_{pi}}$, $\sigma_\theta^2 \sim$ Inv-Gamma $(\alpha_\theta, \beta_\theta)$ means that σ_θ^2 has inverse-gamma density $IG(\sigma_\theta^2|\alpha_\theta, \beta_\theta)$,

and similarly Equations 1.19, 1.20 and 1.21 correspond to the density specifications in Equations 1.14, 1.15 and 1.16.

If responses U_{pi} are missing completely at random (MCAR) or missing by design, the corresponding terms may simply be omitted from the products in Equation 1.13 and from Equations 1.17 through 1.23. More complex forms of missingness require additional modeling and may change the structure of the model and complete conditionals (e.g. Glas and Pimentel, 2008).

Complications of this basic framework involve richer latent space structure, and/or dependence induced by constraints on the parameters. MCMC algorithms for Linear logistic test models (De Boeck and Wilson, 2004; Scheiblechner, 1972), latent class models (Lazarsfeld and Henry, 1968), and some conjunctive cognitive diagnosis models (Haertel, 1989; Junker and Sijtsma, 2001; Macready and Dayton, 1977) received early consideration in Junker (1999) and have since been pursued in detail by de la Torre and Douglas (2004); Roussos, DiBello, Stout, Hartz, Henson, and Templin (2007); Sinharay and Almond (2007) and many others. Random effects models for "testlets" are considered in Bradlow, Wainer, and Wang (1999), for example, and for families of automatically generated items are considered by Johnson and Sinharay (2005); Sinharay, Johnson, and Williamson (2003).

The basic model of Equation 1.13 can be expanded to additional "levels" of hierarchical structure, by elaborating on the prior densities $f(\boldsymbol{\lambda}_\theta)$ and $f(\boldsymbol{\lambda}_\beta)$, to account for additional covariates, dependence due to administrative clustering such as students within classroom, and so forth. Early explorations of this idea include Kamata (2001); Maier (2001). In addition the IRT model itself can be embedded in other hierarchical/Bayesian models, for example MIMIC models. Both of these ideas are nicely developed in Fox (2011).

There is usually not much information available about the item parameters, a_i and b_i in the case of the 2PL model here, and so flat prior distributions are usually chosen for them; we will discuss specific choices below in Section 1.8. Further hierarchical structure is also possible, and is discussed at length in recent monographs such as De Boeck and Wilson (2004) and Fox (2010).

We will discuss particular choices for α_θ and β_θ in Equation 1.23, below in Section 1.8; it is usual to take choices such as $\alpha_\theta = \beta_\theta = 1$, indicating a fairly flat prior, which will allow the data to determine σ^2_θ, and hence the shape of the normal population distribution $f_p(\theta_p|\boldsymbol{\lambda}_\theta)$. For other prior choices for the parameters of the normal θ distribution, see for example Casabianca and Junker (vol. 2, chap 19). Non-normal (e.g. Sass, Schmitt, and Walker, 2008; van den Oord, 2005) and non-parametric (e.g. Karabatsos and Sheu, 2004; Miyazaki and Hoshino, 2009; Woods, 2006; Woods and Thissen, 2006) choices for $f_p(\theta_p|\boldsymbol{\lambda}_\theta)$ also appear in the literature.

The factorization displayed in Equation 1.13 into terms involving different blocks of parameters, facilitated by the independence assumptions of IRT, makes the complete conditionals rather uncomplicated. From Equations 1.17 through 1.23 it it follows that the complete conditional densities for the indi-

vidual parameters will be

$$
\begin{aligned}
f(\theta_p|rest) &\propto \prod_{i=1}^{I} P_i(\theta_p;a_i,b_i)^{u_{pi}}(1-P_i(\theta_p;a_i,b_i))^{1-u_{pi}} n(\theta_p|0,\sigma_\theta^2) \ , \\
&\quad \forall p = 1,\ldots,P \qquad (1.24) \\
f(a_i|rest) &\propto \prod_{p=1}^{P} P_i(\theta_p;a_i,b_i)^{u_{pi}}(1-P_i(\theta_p;a_i,b_i))^{1-u_{pi}} n(\ln a_i|\mu_a,\sigma_a^2)/a_i \ , \\
&\quad \forall i = 1,\ldots,I \qquad (1.25) \\
f(b_i|rest) &\propto \prod_{p=1}^{P} P_i(\theta_p;a_i,b_i)^{u_{pi}}(1-P_i(\theta_p;a_i,b_i))^{1-u_{pi}} n(b_i|0,\sigma_b^2) \ , \\
&\quad \forall i = 1,\ldots,I \qquad (1.26) \\
f(\sigma_\theta^2|rest) &\propto \prod_{p=1}^{P} n(\theta_p|0,\sigma_\theta^2) IG(\sigma_\theta^2|\alpha_\theta,\beta_\theta) \\
&= IG\left(\sigma_\theta^2 \,\middle|\, \alpha_\theta + \frac{P}{2}, \beta_\theta + \frac{1}{2}\sum_{p=1}^{P}\theta_p^2\right) \qquad (1.27)
\end{aligned}
$$

with σ_a^2, σ_b^2, α_θ and β_θ to be fixed at values we specify.

The complete conditionals in Equations 1.24, 1.25 and 1.26 almost always lead to Metropolis-Hastings sampling (Section 1.5), because the product-Bernoulli form of the likelihood is typically not reducible to a simple exponential family or similar distribution. Data augmentation using the normal ogive/probit model in place of the logistic curve (Albert, 1992; Fox, 2010) is attractive because they do lead to Gibbs sampling; see for example Thissen and Edwards (2005) for a sense of the trade-offs between these strategies. Our choice of an inverse-Gamma distribution for σ_θ^2, on the other hand, leads to a conjugate inverse-Gamma complete conditional distribution, so that this part of the algorithm can be implemented using Gibbs sampling (Section 1.4).

It is interesting to note that the complete conditional for σ_θ^2 does not depend on the data, but only $\boldsymbol{\theta}_p$. This is typical for parameters at higher levels in a hierarchical model. It is also interesting to note that, since I is usually much smaller than P, there is much more information to estimate α_i, β_i and σ_θ^2 than to estimate θ_p. However, computational problems (e.g. floating point underflow) are also more common with complete conditionals that involve more factors.

1.8 Implementing an MCMC Algorithm

Conceptually, MCMC is easy: identify complete conditionals, develop a sampling scheme like that of Figure 1.1, run one or more chains for "long enough", use the "converged" part of the chains to make inferences about the posterior distribution. As suggested in Section 1.7, IRT models usually suggest a natural separation of parameters into single-parameter blocks with fairly natural Gibbs and M-H steps, and this is a good place to start.

In this section we will sketch a fruitful approach for developing an MCMC algorithm, using the model and complete conditionals developed in Section 1.7. For concreteness, code is presented in R (R Development Core Team, 2012), but the general recipe will apply to any other programming language, choice of sampling algorithm to implement, or choice of model with suitable translation. After illustrating the development of the algorithm, we will also illustrate its application briefly to item response from a mathematics assessment administered to a US national sample of fifth grade students. Further details are available in an online supplement to this chapter (VanHoudnos, 2012).

1.8.1 Simulating Fake Data

In order to check that an estimation algorithm is working properly, it is useful to see if the algorithm can recover the true parameter values in one or more simulated "test" data sets. Figure 1.4 displays R code, based on Equations 1.17, 1.18 and 1.19, for simulating dichotomous item responses for $P = 2000$ persons and $I = 30$ items, using the 2PL model, with discrimination parameters a_i sampled randomly from a Unif$[0.5, 1.5]$ distribution, and difficulty parameters b_i equally spaced running from -3 to $+3$. Persons' proficiency parameters θ_p were sampled randomly from a normal distribution $n(\theta|\mu_\theta, \sigma_\theta^2)$ with mean $\mu_\theta = 0$ and standard deviation $\sigma_\theta = 1.25$. The R function `plogis(x)` computes $1/(1 + \exp(-x))$; below in Figure 1.10 we will also use the inverse function `qlogis(p)` which computes $\log p/(1 - p)$.

After simulating the data it is a good idea to check to see if the simulation was successful, by checking that quantities estimated from the data agree with the simulation setup. In the case of a new model for which MCMC will be the first estimation algorithm, it is useful to check that the simulated data is consistent with predictions from the simulation model. For example we can check that moments (means, variances, covariances, etc.) of the simulated data agree with the same moments of the simulating model. Although the 2PL IRT model is itself a well-known model, we illustrate this approach in Figure 1.5. The left panel compares the theoretical values

$$\pi_i = \int_{-\infty}^{\infty} \frac{\exp\{a_i(\theta - b_i)\}}{1 + \exp\{a_i(\theta - b_i)\}} \cdot n(\theta|0, \sigma_\theta^2)\, d\theta$$

```
# Set the random-number generator seed,
# to make the results reproducible
set.seed(314159)

# set the number of items I and persons P
I.items    <- 30
P.persons  <- 2000

# set the fixed item and population parameters
a.disc     <- 1 + runif(I.items,-0.5,0.5)
b.diff     <- seq(-3,3,length=I.items)

mean.theta <- 0
sig2.theta <- (1.25)^2

# generate thetas and the I x P matrix of response probabilities
theta.abl  <- rnorm(P.persons, mean=mean.theta, sd=sqrt(sig2.theta))
term.1     <- outer(theta.abl, a.disc)
term.2     <- matrix(rep(a.disc*b.diff, P.persons), nrow=P.persons,
                byrow=TRUE)
P.prob     <- plogis(term.1-term.2)  ### 1/(1 + exp(term.2 - term.1))

# generate the 0/1 responses U as a matrix of Bernoulli draws
U          <- ifelse(runif(I.items*P.persons)<P.prob,1,0)
```

FIGURE 1.4
R code to generate fake 2PL data for testing our MCMC algorithm, based on Equations 1.17, 1.18 and 1.19.

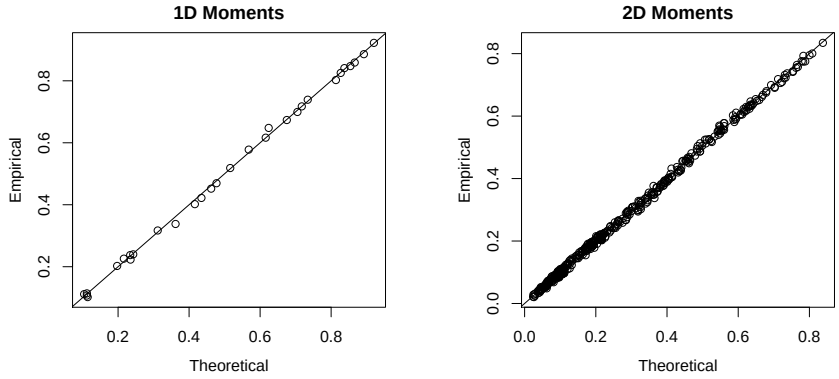


FIGURE 1.5
Checking the fake-data simulation by verifying that empirical moments from the simulated data match theoretical moments from the simulation model. The left panel compares theoretical and empirical estimates of $\pi_i = P[U_i = 1]$, and the right panel compares theoretical and empirical estimates of $\pi_{ij} = P[U_i = 1 \ \& \ U_j = 1]$.

with their empirical estimates

$$\hat{\pi}_i = \frac{1}{P} \sum_{p=1}^{P} U_{pi}$$

The right panel compares the theoretical values

$$\pi_{ij} = \int_{-\infty}^{\infty} \frac{\exp\{a_i(\theta - b_i)\}}{1 + \exp\{a_i(\theta - b_i)\}} \cdot \frac{\exp\{a_j(\theta - b_j)\}}{1 + \exp\{a_j(\theta - b_j)\}} \cdot n(\theta|0, \sigma_\theta^2)\, d\theta$$

with their empirical estimates

$$\hat{\pi}_{ij} = \frac{1}{P} \sum_{p=1}^{P} U_{pi} U_{pj}$$

See VanHoudnos (2012) for computational details. Of course other such "moment matching" checks are possible, and should be chosen to check any features of the simulated data that might be in doubt.

In the case of a familiar model such as the 2PL IRT model for which many other estimation algorithms have been written, we can also check to see that parameters are recovered by a known algorithm, such as the R package `ltm` (Rizopoulos, 2006). This is illustrated in Figure 1.6: the left panel compares theoretical discrimination parameters a_i with their ML estimates from `ltm`; the right panel makes the same comparison for difficulty parameters b_i. Both panels employ a linear equating using the means and standard deviations

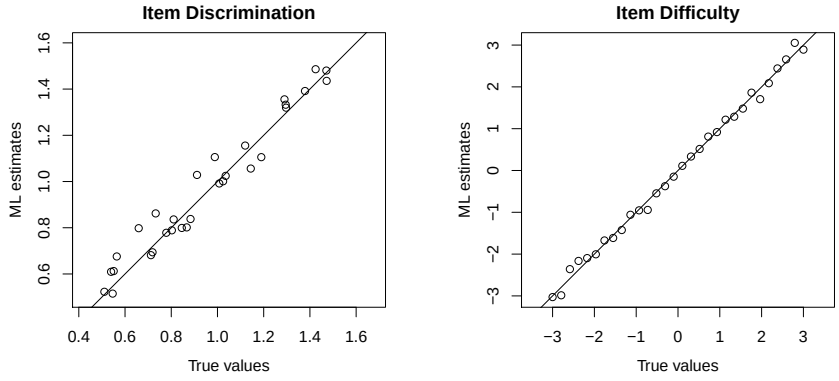


FIGURE 1.6
Checking the fake-data simulation by verifying that the simulation model parameters can be recovered by maximum likelihood (ML). The left panel compares theoretical and ML-estimated values of the discrimination parameters a_i and the right panel makes the same comparison for the difficulty parameters b_i.

of the true and estimated b_i's, as suggested by Cook and Eignor (1991) for example, to account for scale indeterminacy in estimating the IRT model. (While this particular transformation is unique to the particular model we are working with here, it is important to keep similar latent space indeterminacies in mind when working with other IRT and IRT-like models.) Figures 1.5 and 1.6 suggest that we have successfully simulated fake data from the intended 2PL model; we will use this data to test the MCMC algorithm as we develop it.

1.8.2 The MCMC Algorithm Shell

Before building the individual sampling functions that generate Monte Carlo samples from the complete conditionals in Equations 1.24 through 1.27, it is helpful to think about the context in which they will operate. Figure 1.7 displays the two main components of a working one-variable-at-a-time MCMC sampler.

- `blocked.mcmc.update` collects together four functions, `sample.th`, `sample.a`, `sample.b` and `sample.s2` that will be used to implement one full step of the one-variable-at-a-time MCMC algorithm outlined in Figure 1.1. We will discuss the definitions of these four functions below in Section 1.8.3.

  `U.data` is the matrix $\mathcal{U}$ of item responses. `cur` is a "list" (an R data structure) that contains the current state of the Markov chain (the current set of sampled parameter values), as well as auxiliary information such as hy-

```
blocked.mcmc.update  <- function( U.data, cur){
  cur <- sample.th(U.data, cur)  # Equation 1.24
  cur <-  sample.a(U.data, cur)  # Equation 1.25
  cur <-  sample.b(U.data, cur)  # Equation 1.26
  cur <- sample.s2(U.data, cur)  # Equation 1.27
  return(cur)
}
```

```
run.chain.2pl <- function(M.burnin, M.keep, M.thin, ... ){

  ...

  # Burn-in phase: do not keep these results
  if( M.burnin > 0 ) {
    for(ii in 1:M.burnin ) {
      cur <- blocked.mcmc.update(U.data, cur)
  }}

  # Converged phase: Keep these results after thinning
  for (m in 1:M.keep) {
    # Skip the "thinned" pieces of the chain
    if( M.thin > 1 ) {
      for( ii in 1:(M.thin-1) ) {
        cur <- blocked.mcmc.update(U.data, cur)
    }}
    # Generate a "kept" update and save its results
    cur <- blocked.mcmc.update(U.data, cur)
    chain.keep[,m] <- c( cur$th, cur$a, cur$b, cur$s2 )
  }

  ...

  return( chain.keep )
}
```

FIGURE 1.7
The R function `blocked.mcmc.update` collects together four sampling routines that implement the one-variable-at-a-time MCMC algorithm of Figure 1.1, for a 2PL IRT model. `cur` defines the current state of the Markov Chain. Each function call `sample.th` , `sample.a`, etc. implements a set of draws from complete conditionals; the definitions of these functions will be described below in Section 1.8.3. The R function `run.chain.2pl` shows how `blocked.mcmc.update` is used to implement a complete MCMC algorithm. Some housekeeping details have been elided; see Figure 1.8 for these details.

perparameters, Metropolis-Hastings proposal variances and acceptance rate information. The effect of a call to `blocked.sample.mcmc` is to take one step in the Markov chain, that is, to replace the old sampled parameter values with new ones.

- `run.chain.2pl` is a sketch of the main function that "runs" the entire MCMC algorithm. Certain housekeeping details are omitted for clarity, and replaced with "`...`" in Figure 1.7; they can be seen in detail in Figure 1.8.

  The algorithm runs in two phases. In the *burn-in phase*, `M.burnin` steps of the Markov chain are sampled, and simply thrown away, since the Markov chain is assumed not to have converged yet. In the *converged phase*, `M.keep` × `M.thin` steps are sampled, and every $\texttt{M.thin}^{th}$ set of sampled parameter values is kept. The resulting matrix of `M.keep` samples from the posterior distribution is returned in the matrix `chain.keep`.

A complete version of `run.chain.2pl` is shown in Figure 1.8, for completeness. Details present in Figure 1.8, but omitted in Figure 1.7, include:

- Additional information passed to `run.chain.2pl`: `U.data`, the data matrix of item responses; `hyperpars`, the vector of hyperparameters for the prior distributions according to Equation 1.23; vectors `th.init`, `a.init`, `b.init` and `s2.init` for all model parameters; Metropolis-Hastings proposal variances `MH.th`, `MH.a`, `MH.b`; and a switch `verbose` to indicate whether to report M-H acceptance rates.
- Setting up the "current state" list `cur` and initializing the model parameters in `cur` to the initial values passed to `run.chain.2pl`.
- Setting up the matrix `chain.keep` in which to keep values of the Markov chain once it has converged. Each column of `chain.keep` contains one complete set of model parameters drawn from the posterior distribution.
- Code to report `Average acceptance rates` for the M-H portions of the algorithm.

1.8.3 Building the Complete Conditionals

We suggest building and testing each of the parameter-sampling functions `sample.th`, `sample.a`, `sample.b` and `sample.s2` shown in Figure 1.7 separately. While testing one function, say `sample.th`, the other functions should simply return the values they are given, effectively holding those parameters fixed at their initial values, to isolate the behavior of `sample.th` itself. For example, `sample.a` could be given the provisional definition `sample.a <- function(U.data, cur) { return(cur) }`, and identical definitions could be used for `sample.b` and `sample.s2`. These provisional definitions would be replaced with actual MCMC-sampling code as each `sample.*` function is developed (see VanHoudnos, 2012, for details).

```
run.chain.2pl <- function(M.burnin, M.keep, M.thin,
                          U.data, hyperpars,
                          th.init, a.init, b.init, s2.init,
                          MH.th, MH.a, MH.b, verbose=FALSE) {

  # Define and initialize the list of things to keep track of in the
  # "current state" of the chain -- see text for detailed explanation
  cur     <- list(th=th.init, a=a.init, b=b.init, s2=s2.init,
                  hyperpars=hyperpars,
                  MH = list(th=MH.th,    a=MH.a,    b=MH.b),
                  ACC= list(th=0,th.n=0, a=0,a.n=0,  b=0,b.n=0 ))

  # Define matrix to store MCMC results...
  chain.keep <- matrix( NA, ncol = M.keep,
                        nrow = length(th.init) + length(a.init)
                               + length(b.init) + length(s2.init) )
  rownames(chain.keep) <- c( paste('theta.abl', 1:length(th.init)),
                             paste('a.disc',  1:length(a.init)),
                             paste('b.diff',  1:length(b.init)),
                             'sig2.theta')

  # Burn-in phase: do not keep these results
  if( M.burnin > 0 ) {
    for(ii in 1:M.burnin ) {
      cur <- blocked.mcmc.update(U.data, cur)
  }}

  # Converged phase: Keep these results after thinning
  for (m in 1:M.keep) {
    # Skip the "thinned" pieces of the chain
    if( M.thin > 1 ) {
      for( ii in 1:(M.thin-1) ) {
        cur <- blocked.mcmc.update(U.data, cur)
    }}
    # Generate a "kept" update and save its results
    cur <- blocked.mcmc.update(U.data, cur)
    chain.keep[,m] <- c( cur$th, cur$a, cur$b, cur$s2 )

    if ( m %% 100 == 0) {
      print(m)
      # Adaptive tuning would go here.
    }

  }

  if (verbose) {
    cat(paste("Average acceptance rates:",
                "\n theta.abl:", round(cur$ACC$th / cur$ACC$th.n ,3),
                "\n a.disc:   ", round(cur$ACC$a  / cur$ACC$a.n  ,3),
                "\n b.diff:   ", round(cur$ACC$b  / cur$ACC$b.n  ,3),"\n"))
  }

  return( chain.keep )
}
```

FIGURE 1.8
Function definition for a complete MCMC algorithm for fitting a 2PL model to data. See text for discussion of the various parts of this function.

For illustration, we show a definition for the function `sample.th`, in Figure 1.9. We sample from the complete conditional shown in Equation 1.24, using a random-walk M-H sampler as described in Section 1.5. All probability calculations have been done on the log scale, to avoid numerical underflow on the computer, which is common when multiplying many probabilities together (for larger problems, logarithmic calculations typically have to be combined with other strategies such as additive offsets, to avoid underflow; see for example Monahan, 2011, Chapter 2).

In Figure 1.9, `log.prob` is an auxiliary function that will be used several times in `sample.th`, `sample.a` and `sample.b`. It returns a $P \times I$ matrix of terms

$$\log \left[P_i(\theta_p; a_i, b_i)^{u_{pi}} (1 - P_i(\theta_p; a_i, b_i))^{1-u_{pi}} \right]$$

which are useful in calculating the log-likelihoods following Equations 1.24, 1.25 and 1.26.

`sample.th` is the main function for sampling θ_p's from the complete conditional densities in Equation 1.24. In `sample.th`,

- `old` contains the "old" current state of the Markov chain, and `cur` contains the "new" current state after `sample.th` is finished. Both variables have the structure of `cur` defined in `run.mcmc.2pl` in Figure 1.8.

- We have exploited the high degree of separation of parameters available in IRT models (due to independence across persons and items in the IRT likelihood) to string together the P calculations needed in Equation 1.24 into a small number of implicit vector calculations in R; this not only shortens the programming task, it also speeds up R's execution. In other models not enjoying such separation, these calculations would have to be done separately and sequentially.

- The line `th.star <- rnorm(P.persons,th.old,th.MH)` implements the normal random-walk proposal draws

  $$\theta_p^* \overset{indep}{\sim} N(\theta_p^{(m-1)}, \sigma_p^2),\ p = 1, \ldots, P$$

  where the value of σ_p is specified in `th.MH`.

- The next five assignment statements calculate the vector of M-H acceptance ratios α_p^* for each θ_p (Equation 1.9), using the implicit vector calculations available in R.

- `acc.new` contains a vector of 0's and 1's: 0 in the p^{th} position if the corresponding θ_p^* is not accepted, and 1 if it is.

- Finally, the current state `cur` of the Markov chain is updated with each accepted θ_p^*, the running sum of acceptances, `cur$ACC$th`, is updated with the average acceptance rate across all θ_p's, and the number of updates, `cur$ACC$th.n`, is incremented.

```
log.prob <- function(U.data, th, a, b) {
  term.1     <- outer(th, a)
  term.2     <- matrix(rep(a*b, P.persons), nrow=P.persons,
                byrow=TRUE)
  P.prob <- plogis(term.1 - term.2)

  log.bernoulli <- U.data*log(P.prob) + (1-U.data)*log(1-P.prob)
  return(log.bernoulli)
}
```

```
sample.th <- function(U.data, old) {
 th.old    <- old$th
 MH.th     <- old$MH$th
 P.persons <- length(th.old)
 th.star   <- rnorm(P.persons,th.old,MH.th)

 log.cc.star <- apply(log.prob(U.data, th.star, old$a, old$b),1,sum) +
                   log(dnorm(th.star,0,sqrt(old$s2)))
 log.cc.old  <- apply(log.prob(U.data, th.old, old$a, old$b),1,sum) +
                   log(dnorm(th.old,0,sqrt(old$s2)))
 log.prop.star <- log(dnorm(th.star,th.old,MH.th))
 log.prop.old  <- log(dnorm(th.old,th.star,MH.th))

 log.alpha.star <- pmin(log.cc.star + log.prop.old - log.cc.old -
                    log.prop.star,0)

 acc.new <- ifelse(log(runif(P.persons))<=log.alpha.star, 1, 0)

 cur          <- old
 cur$th       <- ifelse(acc.new==1, th.star, th.old)
 cur$ACC$th   <- old$ACC$th + mean( acc.new )
 cur$ACC$th.n <-  cur$ACC$th.n + 1

 return(cur)
}
```

FIGURE 1.9
Definition of `sample.th`, for sampling from the complete conditionals for θ's, using a normal random-walk Metropolis-Hastings strategy. An auxiliary function `log.prob` which is used several times in `sample.th` and other complete-conditional samplers, is also defined here. See text, and VanHoudnos (2012), for details.

Note that in this algorithm, we are accumulating only average acceptance counts across all P θ_p parameters, rather than individual acceptance rates for each θ_p. In well-behaved IRT models with no missing data in the rectangular array $\mathcal{U}$, this is a workable strategy; in more challenging situations, it may be necessary to keep track of acceptance rates for each θ_p individually and use separate M-H proposal variances for each θ_p (expanding both `old$MH$th` and `old$MH$ACC` to I-dimensional vectors).

After some trial runs to make sure we are happy with `sample.th`, we would move on to developing `sample.a`, `sample.b`, and `sample.s2`. For `sample.a` and `sample.b`, a strategy similar to that of Figure 1.9 can be used. Since `sample.s2` will be a Gibbs step, simpler direct sampling from the inverse-Gamma distribution in Equation 1.27 can be implemented. More details on all four sampling functions are supplied in VanHoudnos (2012).

1.8.4 Tuning Metropolis-Hastings

The Metropolis-Hastings samplers `sample.th`, `sample.a` and `sample.b` must also be tuned—that is, proposal variances `MH.th`, `MH.a` and `MH.b` must be chosen to get workable acceptance rates. If the acceptance rate is too high, the chain may not explore very much of the posterior distribution because it is taking too many small, incremental steps. The resulting trace plot may look like plot (b) in Figure 1.2. If the acceptance rate is too low, the chain will also not explore much of the posterior distribution because most proposals are rejected and it stays in place. The resulting trace plot may look like plot (c) in Figure 1.2. In both the too high and too low regions of acceptance rate, the autocorrelation will be high (Figure 1.3). For a single parameter at a time, the optimal acceptance rate is near 0.44; in practice rates between 0.2 and 0.6 are usually reasonable (Rosenthal, 2011).

In Figure 1.10 we show one possible strategy for tuning the M-H samplers. In the figure, we only keep track of the average acceptance rate for θ's, the average rate for a's and the average rate for b's. It is possible that some individual parameters do not have reasonable acceptance rate. We could instead tune acceptance rates for each θ_p, a_i and b_i individually, by recording separate acceptance rates for each of them, and using different component values in `th.MH`. Note that tuning one M-H sampler can affect the acceptance rates of the others.

The results of three tuning runs using the code illustrated in Figure 1.10 are shown in Table 1.1. The first run (first row of the table) shows unacceptably low average acceptance rates, especially for the a and b parameters. The second run (second row) shows acceptable but still somewhat low average acceptance rates for all three sets of parameters. The third run (third row) shows nearly optimal average acceptance rates.

In addition to changing a "tuning parameter" for a specific proposal distribution, it can also be advantageous to change the proposal distribution itself, in order to more efficiently explore the posterior. Typically, the proposal distri-

```
set.seed(314159)

M.burnin <- 250
M.keep <- 1000
M.thin <- 1

# See Section 1.8.5 for details of prior specification
hyperpars <- list( mu.a         = 1.185,
                   s2.a         = 1.185,
                   s2.b         = 100,
                   alpha.th     = 1,
                   beta.th      = 1 )

# use naive method of moments estimators as initial values for
# theta's and b's, and set the initial values of a's and s2 to 1.
th.tmp       <- qlogis(0.98*apply(U,1,mean) + .01)
a.tmp        <- rep(1,I.items)
b.tmp        <- qlogis(0.98*apply(U,2,mean) + .01)
s2.tmp       <- 1

# specify the MH tuning parameters
MH.th <- .75; MH.a <- .15; MH.b <- .10

tune.run <- run.chain.2pl(M.burnin, M.keep, M.thin, U, hyperpars,
                     th.tmp, a.tmp,  b.tmp, s2.tmp,
                     MH.th,  MH.a, MH.b, verbose=TRUE)
#Average acceptance rates:
# theta.abl: 0.416
# a.disc:    0.47
# b.diff:    0.436
```

FIGURE 1.10
Example of tuning the completed sampler by iteratively adjusting the values of `MH.th`, `MH.a`, and `MH.b`. The table presents additional sampling runs that were used to find the final tuned values.

TABLE 1.1
Results of tuning runs as in Figure 1.10 for various values of tuning parameters.

MH.th	Acceptance	MH.a	Acceptance	MH.b	Acceptance
2.	0.218	1.	0.076	1.	0.072
1.	0.352	0.25	0.306	0.25	0.226
0.75	0.416	0.15	0.470	0.10	0.436

bution would be chosen to mimic the posterior distribution before running the chain, or it can be adaptively adjusted to mimic the posterior distribution as the whole MCMC algorithm runs (see also the discussion of choice of proposal distributions in Section 1.5 above). In the case of one-dimensional complete conditionals, one adaptive approach is to simply adjust the variance of the proposal distribution as the chain runs in order to zero in on optimal acceptance rates. Such adaptations could be built into the function `run.chain.2pl` in Figure 1.7. When the complete conditional is multidimensional, the adaptive procedure would try to mimic the covariance structure of the posterior distribution. Unfortunately these methods are beyond the scope of the chapter; see Rosenthal (2011) and the references therein for an accessible discussion.

1.8.5 Testing on Simulated Data

We tried the algorithm out on data simulated as in Figure 1.4. We used the prior distributions listed in Equations 1.20–1.22 with the hyperparameters listed in Figure 1.10.

The hyperparameters were chosen to create uninformative priors. The b_i prior is a normal with variance $\sigma_b^2 = 100$, which is uninformative because it is very flat for typical values of item difficulty. We similarly dispersed the log-normal prior of the a_i values by matching the 95% quantile of a more familiar distribution: a positive truncated normal distribution with a variance of 100. A quick calculation shows that 95% of the mass of a truncated normal with a variance of 100 is below 19.6 units, and that a log-normal with a mode of 1 must have $\mu_a = \sigma_a^2 = 1.185$ to also have 95% of its mass below 19.6 units. Values larger than 1.185 will further flatten the prior, but since the discrimination parameters are typically less variable than difficulty parameters it is not useful to further broaden the discrimination prior. Unlike the previous two priors, which were more uniformative for larger values of the hyperparameters, the inverse-gamma prior for σ_θ^2 becomes less informative as α and β become smaller as can be seen from complete conditional in Equation 1.27. We chose the minimum values of $\alpha = \beta = 1$. A more detailed discussion of these prior specifications can be found in VanHoudnos (2012).

We simulated three chains from starting values that were overdispersed from the method of moments starting values shown in Figure 1.10. On one hand, in order to get trustworthy evidence of convergence from the Gelman-Rubin convergence statistic R, and moreover any iterative search algorithm should be started from several different initial values, to gain evidence that algorithm does not get "stuck" in non-optimal solutions. On the other hand, we have found that MCMC for IRT can be sensitive to excessively poor starting values: typically much of the IRT likelihood and posterior is extremely flat, and starting the MCMC algorithm in a flat part of the posterior distribution will cause the chain to wander a very long time, and/or become numerically unstable, before converging on a local or global mode.

We note in passing that, although simulating multiple chains is extremely

valuable—both for diagnosing problems with the MCMC algorithm and for confirming that the chain has converged to the stationary distribution—it is extremely slow computationally, especially if the chains must be generated sequentially. VanHoudnos (2012) suggests some methods for generating the chains in parallel, by exploiting the fact that most modern personal computers are equipped with multiple CPU cores.

To generate starting values for the chains we began by mildly jittering the method of moments estimates in Figure 1.10. We further spread the starting values out by exploiting the location and scale indeterminacy of logistic IRT models, transforming these jittered values by randomly chosen location and scale values which did not change the probabilities of correct responses.

However, even this mild dispersion caused the chains to converge very slowly, especially for parameters with little support in the data such as σ_θ^2 and the largest item discrimination parameter ($\max a_i$). To speed convergence we modified the scan order of the Metropolis-Hastings algorithm to reduce autocorrelation and cross-correlation in the chain, without incurring a large computational cost, by repeatedly sampling item parameters, holding person parameters fixed, and vice-versa (see Section 1.8.7 below for details). This improved stability of the Markov chain and sped convergence to the stationary distribution. Figure 1.11 shows typical behavior of the chain, for three sets of overdispersed starting values, using the modified scan order.

Care in starting values and scan order here is partly due to numerical instability in the expression `log.bernoulli <- U.data*log(P.prob) + (1-U.data)*log(1-P.prob)` in the top panel of Figure 1.9, especially when `P.prob` is close to 0 or 1. In R, a more numerically stable expression would be `log.bernoulli <- log(P.prob^U.data) + log((1-P.prob)^(1-U.data))`; using this expression instead, we are able to use much more overdispersed starting values, for example. For additional details, see VanHoudnos (2012).

From trace plots like Figure 1.11, as well as acf plots, we chose `M.burnin` = 3,000, `M.thin` = 1, and `M.keep` = 3,000 for the final run of the improved scan order chain. It converged nicely: the Gelman-Rubin convergence statistic $\hat{R} \leq 1.055$ for all parameters and every MCMC standard error was at least an order of magnitude smaller than the associated posterior standard error. A comparison of posterior quantiles with the equated true values from the simulated fake-data is shown in Figure 1.12, confirming that our algorithm works well. Additional details, such as the code to generate the graphs, can be found in VanHoudnos (2012).

1.8.6 Applying the Algorithm to Real Data

We analyzed data collected as a part of the national standardization research for the Comprehensive Test of Basic Skills, Fifth Edition (CTBS/5), published by CTB/McGraw-Hill (1996). A mathematics test comprised of 32 multiple-choice items and 11 constructed-response items was administered to a U.S. nationally representative sample of 2171 Grade 5 students. All multiple-choice

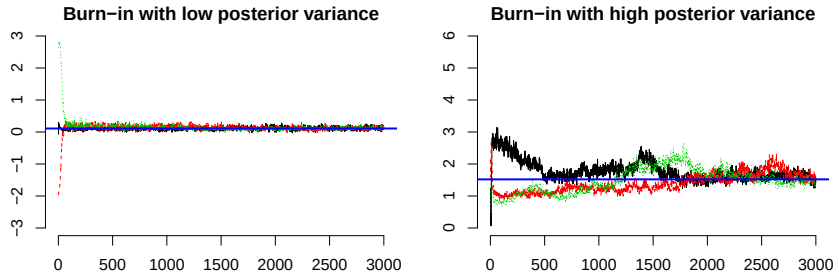


FIGURE 1.11
Trace plots demonstrating the differences in burn-in times between a parameter with small posterior variance, which burns-in within 100 iterations, and a parameter with large posterior variance, which burns-in after 2000 iterations. The equated true values of each parameter are plotted as a horizontal line. See text, and VanHoudnos (2012) for details on the code used to setup, run, and visualize the overdispersed chains.

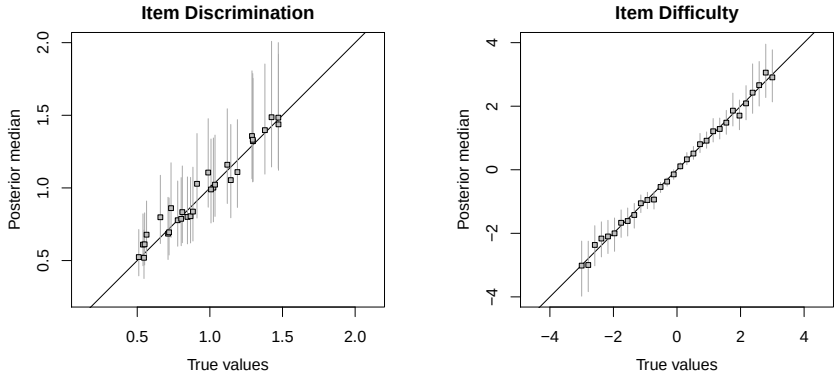


FIGURE 1.12
Checking the MCMC algorithm by verifying that the simulation model parameters can be recovered. The left panel compares equated theoretical and MCMC-estimated values of the discrimination parameters a_i and the right panel makes the same comparison for the difficulty parameters b_i. The error bars are equated 95% posterior quantiles.

item responses were dichotomously scored, and all constructed responses were professionally evaluated on rubrics that ranged from zero to a maximum of between 2 and 5 points. For this illustration, we analyzed only the 32 multiple-choice questions. There are no missing data.

In principle all aspects of the model described in Equations 1.17–1.23 should be reconsidered when new data is considered. Especially the form of the prior distributions and hyperparameters in Equations 1.20–1.23 should receive special care, since they can affect parameter and latent proficiency estimates even if we keep the form of the IRT model in Equations 1.17–1.19 the same. For the purposes of illustration in this chapter, however, we used the same prior distributions and chosen hyperparameters $\mu_a = 1.185$, $\sigma_a^2 = 1.185$, $\sigma_b^2 = 100$, $\alpha_\theta = 1$, $\beta_\theta = 1$. The example Grade 5 dataset had properties very similar to the simulated "fake data" above (indeed, we designed the "fake data" to mimic the real data), so it was very straightforward to analyze with our improved scan order MCMC algorithm, prior distributions, and so forth.

After a 6,000 iteration burn-in from overdispersed starting values the sampler had clearly converged both from the inspection of trace plots and the calculation of $\hat{R}$ statistics for each of the parameters, which were all less than 1.05. Even though the chain was properly tuned and it had clearly converged, the autocorrleation of some parameters was still quite high as can be seen in Figure 1.13. To bring the autocorrelation down to non-significant levels, the chain would have had to be thinned to keep only every 150^{th} iteration, which would have left only 67 samples per 10,000 iterations. However, this extreme thinning is unnecessary: in all cases the MCMC error as estimated with overlapping batch means was very small compared to the posterior variance.

Figure 1.14 compares the posterior quantile estimates from pooling all three converged chains to maximum likelihood estimates. Note that in all cases the two sided 95% credible intervals contained the ML estimates. There is also evidence in Figure 1.14 that CI's for a's and b's were wider where they were difficult to estimate (larger magnitude a's and b's).

1.8.7 Miscellaneous Advice

As we have seen, setting up a functioning MCMC algorithm for an IRT model is not difficult. In practice, the MCMC algorithm may be computationally slow, may converge slowly to the $f(\boldsymbol{\tau}|\mathcal{U})$, and/or may exhibit excessive autocorrelation/cross-correlation. Each of these takes some care to correct, in practice.

Computational Improvements & Restarting

If the chain is mixing reasonably well but is executing slowly, then some computational adjustments may be needed. Adapting M-H proposal distributions dynamically (Rosenthal, 2011), or replacing M-H steps with adaptive rejection (Gilks and Wild, 1992) or Gibbs (Albert, 1992; Fox, 2010) sampling steps, can

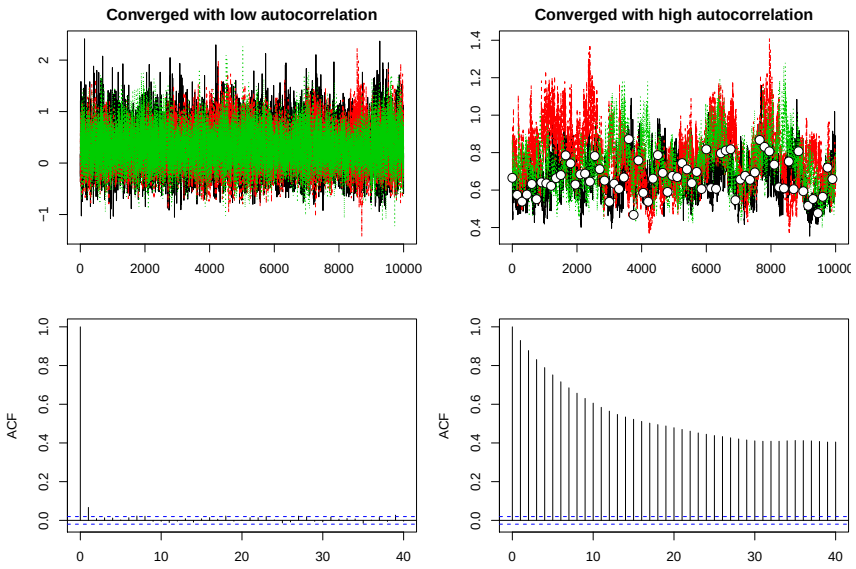


FIGURE 1.13
Converged trace plots on the real data example with the autocorrelation functions calculated from the black chains. The left plots are of a typical person ability parameter (θ_p); the right plots, a typical item discrimination parameter (a_i). The 67 white circles are values that would be kept if the chain were thinned so that its autocorrelation would be as low as that of the left example chain (`M.thin`= 150).

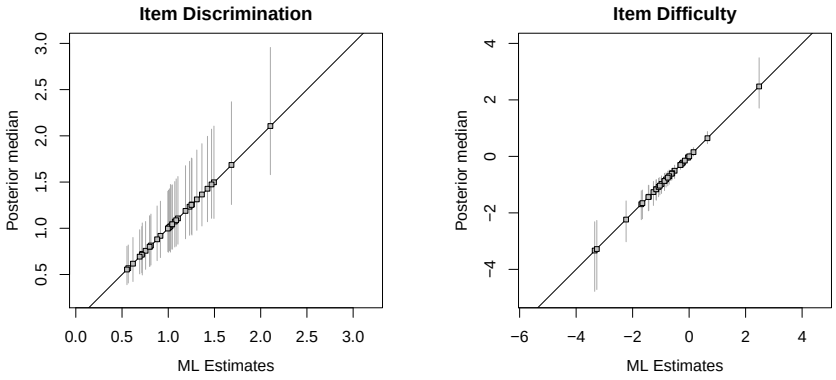


FIGURE 1.14
Posterior medians compared to equated ML estimates. The error bars are equated 95% posterior quantiles.

speed up computation. Avoiding time-intensive matrix calculations, precomputing quantities that will be used repeatedly, and/or pre-marginalizing over nuisance parameters, can also help. Parallel computing is an obvious speed-up tool: simply taking advantage of modern hardware to run multiple instances of existing code simultaneously on a multi-core CPU speeds up computation by a simple division of labor (e.g. VanHoudnos, 2012); running a single chain on multiple CPUs is also possible, if one can identify or construct regeneration states (e.g. Brockwell and Kadane, 2005) in the Markov chain. More recently efficient and highly accurate approximate methods for running a single MCMC algorithm on multiple computers has been developed (Scott, Blocker, Bonassi, Chipman, George, and McCulloch, 2013).

The order in which the complete conditionals in Figure 1.1 are sampled is called the *scan order* of the MCMC algorithm. *Fixed scan* algorithms sample in the fixed order that the blocks are numbered—1, 2, 3, ..., H, as in Figures 1.1 and 1.8. Although fixed-scan algorithms do not satisfy the reversibility condition of Equation 1.5, they still do have $f(\boldsymbol{\tau}|\mathcal{U})$ as the stationary distribution, as suggested in the discussion of Equation 1.6. Reversible kernels tend to have better asymptotic properties—it is easier to prove asymptotic normality, and to calculate or estimate convergence rates. *Palindromic scan* algorithms samples the complete conditional in order and then in reverse order, to produce one Markov chain step, and is reversible. *Random permutation scan* algorithms sample from the complete conditionals in random order for each step of the Markov chain, and *random scan* algorithms simply sample from a random complete conditional H times, to produce one step. Roberts and Sahu (1997) argue that fixed scan Gibbs samplers generally have the fastest convergence rates, though random scan methods will be faster for some parameterizations of models with linear mean structure.

Sometimes the scan order needs to be tailored to the specific features of the model or data one is working with. For example in the running IRT example in this chapter, we found that with a simple fixed order

$$\theta_p \to a_i \to b_i \to \sigma_\theta^2, \tag{1.28}$$

samples of the parameter σ_θ^2 were highly affected by variability of samples of θ_p's and in turn samples of the a and b parameters; this slowed the convergence of the chain considerably. We found that the MCMC algorithm stabilized more quickly if we could somewhat isolate samples of θ's and σ_θ^2 from samples of a's and b's, and vice-versa, by sampling k sets of θ_p's and σ_θ^2 for each set of a's and b's. Thus, the scan order

$$\left(\theta_p \to \sigma_\theta^2\right)^k \to \left(a_i \to b_i\right)^k$$

will still generate an MCMC algorithm that converges to the true posterior distribution, as discussed in Section 1.3, but will converge much faster. The value of k controls a tradeoff between reduced autocorrelation and cross-correlations in the Markov chain, and computational time. By trial and error we found $k = 7$ to be a reasonable choice.

Although it is not a computational improvement per se, it also is very good practice to save the random number generator seed (or some other sufficient description of the current state of the simulated Markov chain) at regular intervals during an MCMC run. That way, if the simulation is interrupted (by a power failure for the computer running the algorithm, by the need to move to another computer with more memory or a faster processor, etc.) then the simulation can begin from the last saved state rather than from the initial state. This can also be useful in reproducing "bad" behavior to help diagnose computational errors, slow mixing, etc.

Blocking, Reparametrization, and Decorrelation Steps

If the chain is mixing slowly and/or exhibiting large autocorrelation or cross-correlation, several remedies are possible. Graves, Speckman, and Sun (2011) survey a number of strategies for dealing with slowly mixing or highly autocorrelated or cross-correlated chains, many of which were first gathered together by Gilks and Roberts (1996) and Roberts and Sahu (1997). The most common are blocking, reparametrization, and decorrelation steps.

One suggestion often made is to try a different *blocking* scheme, that is, choose different disjoint subsets of parameters $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_H$ in Figure 1.1. However, the item parameters for each item are natural blocks that often exhibit strong posterior dependence. These and other groups of parameters that exhibit strong dependence can be revealed in shorter preliminary runs of of a less-than-optimal MCMC algorithm. When blocks of dependent parameters are grouped together and sampled as a block, overall mixing can improve (though of course the dependence between parameters within blocks remains). Exact conditions under which blocking does or does not help, for normal complete conditionals (which are often approximately true for other complete conditionals, by a CLT-like argument), can be found in Roberts and Sahu (1997); see also Doucet et al. (2006).

Another way to deal with dependence between parameters is to *reparametrize* the model so that the new parameters are less dependent. Parameters with a normal or nearly normal posterior distribution can be rotated, via linear transformation, to be uncorrelated for example. This does not remove the dependence among the original parameters, but an MCMC algorithm for the transformed parameters may run better (and the results transformed back to the original parametrization if needed). Hierarchical centering (Gilks and Roberts, 1996; Roberts and Sahu, 1997) involves expressing prior distributions at one level as centered at the hyperparameters at the next level. It is particularly suited to models with linear mean structure such as hierarchical linear models (HLMs), generalized HLMs and HLM-like models, and has the effect of both making parameters at level 2 independent and concentrating more data on estimates at level 3. Data augmentation (Section 1.6.4) simplifies complete conditionals and often reduces both computational time and chain autocorrelation (Fox, 2010), but the gains are not always obvious

and probably deserve empirical investigation for each new class of models one encounters (see, for example, Thissen and Edwards, 2005).

Finally, a common source of poor behavior in an MCMC algorithm is that the underlying model is overparametrized, or nearly so. The Markov chain will tend to walk along the ridge in the posterior distribution induced by the overparametrization. In that case Graves et al. (2011) suggest that inserting occasional M-H steps with proposals that are perpendicular to the ridge can reduce autocorrelation in the chain and lead to better exploration of the parameter space. A simpler idea along the same lines (Tierney, 1994) is to insert occasional independence M-H steps as *decorrelation steps*. More general considerations along these lines lead to stochastic relaxation (Geman and Geman, 1984) and simulated annealing (Bertsimas and Tsitsiklis, 1993) algorithms.

Model Fit

Applying a sophisticated model to complex data is more than defining the model, as in Section 1.7, and successfully implementing an estimation algorithm, as in Section 1.8. Assessing the fit of the model, informally while working with the estimation algorithm, and formally after estimation is complete, is an essential part of statistical and psychometric practice.

It will sometimes be the case that the model fits the data so poorly that no MCMC algorithm will converge or give sensible answers. MCMC is particularly sensitive to the shape of the posterior distribution: if the posterior is well peaked around a well-defined maximum, MCMC works quite well. If the posterior is relatively flat with ridges and/or poorly defined maximas, as can happen when the model fits the data poorly, then the MCMC sample will tend to wander without converging, often to the edge of the parameter space.

Formal methods can be used to compare the fit of competing models, if more than one model has been fitted to the data. There is a large literature on Bayesian model comparison, focusing mainly on Bayes factor calculations (Han and Carlin, 2001; Kass and Raftery, 1995; Wasserman, 2000) and fit indices such as AIC (Akaike, 1977; Burnham and Anderson, 2002), BIC (Kass and Wasserman, 1995; Schwarz, 1978), and DIC (Ando, 2007; Celeux, Forbes, Robert, and Titterington, 2006; Spiegelhalter, Best, Carlin, and van der Linde, 2002).

Model criticism is the process of isolating particular features of a model that can be changed to improve the fit of the model; the process is similar in spirit to model modification indices in structural equations modeling, for example. A model criticism method particularly suited to MCMC calculation is posterior predictive checking (Gelman et al., 2003; Gelman, Meng, and Stern, 1996; Lynch and Western, 2004). Essentially, one takes the output of the MCMC algorithm and uses it to generate more fake data from the fitted model, and compares some summary of the fake data with the same summary of the real data used to fit the model. If the fake data summaries look like the

real data summary, this gives confidence that the model is capturing whatever feature of the data the summary was intended to focus on. If not, the difference between the real and fake data summaries usually suggest ways to improve the model. A full discussion is beyond the scope of this chapter; see Sinharay, Johnson, and Stern (2006) or Sinharay (vol. 2, chap. 19) for an introduction to posterior predictive checking of psychometric models.

1.9 Discussion

In this chapter, we have reviewed some theory and useful methodology for implementing Markov Chain Monte Carlo (MCMC) algorithms for inferences with Bayesian IRT models. These ideas were illustrated using a standard 2PL IRT model.

MCMC has been applied successfully in many IRT and IRT-related contexts. To give only a few examples, data-augmentation-based MCMC for normal ogive models (Albert, 1992) has been extended to hierarchical IRT models (Fox, 2010), testlet models (Bradlow et al., 1999), and other settings. Other applications of MCMC include estimation of hierarchical rater models Casabianca, Junker, and Patz, vol. 1, chap. 27, models for test compromise (Segall, 2002; Shu, 2010), and models involving change over time (Studer, 2012) in IRT and cognitive diagnosis models (CDM's, e.g. Junker and Sijtsma, 2001; Rupp, Templin, and Henson, 2010). In a related direction, Weaver (2008) gives an example of MCMC applied to a detailed computational production model for cognitive response data, tied to a particular cognitive architecture (Anderson and Lebiere, 1998).

For modest-sized applications with fairly standard models, it makes sense to work with existing standard software, rather than developing an MCMC algorithm from scratch. WinBUGS (Johnson, vol. 3, chap. 22; Lunn et al., 2009) and its cousins OpenBugs (Spiegelhalter, Thomas, Best, and Lunn, 2012) and JAGS (Plummer, 2012a) are readily available and can be used to fit and make inferences with a wide variety of standard models. R packages such as `R2WinBUGS` (Gelman et al., 2012), `rube` (Seltman, 2010), `BRugs` (Ligges, 2012) and `rjags` (Plummer, 2012b) make it convenient to access the power of WinBUGS and its cousins from R. For example, Ayers (2009) sketches a flexible approach to fitting various IRT models and CDM's using WinBUGS from R. A number of other packages for R (Geyer and Johnson, 2012; Hadfield, 2010; Martin, Quinn, and Park, 2009) and other computational platforms (Daumé III, 2007; Patil, Huard, and Fonnesbeck, 2010; Stan Development Team, 2012) allow for faster computation for specific models, exploration of specific techniques for improving MCMC, and so forth. For a review, see Rusch, Mair, and Hatzinger (vol. 3, chap. 21).

For developing new models, or performing MCMC computations in new or

complex settings, it can be preferable to develop MCMC algorithms by hand: one then has greater flexibility concerning sampling methods, scan order, parallelization, diagnostics, etc., and these can be quite important in creating an efficient and trustworthy algorithm. In addition, the methods illustrated here can be applied not only in R but also in C++ or any other computational language, and this can by itself produce significant speed-ups (see for example VanHoudnos, 2012). The monographs of Gilks, Richardson, and Spiegelhalter (1996) and Brooks, Gelman, Jones, and Meng (2011) provide good entry points into the large literature on established MCMC methodology as well as more novel related approaches.

Bibliography

Akaike, H. (1977), "On entropy maximization principle," in *Applications of Statistics*, ed. Krishnaiah, P., Amsterdam: North-Holland, pp. 27–41.

Albert, J. H. (1992), "Bayesian estimation of normal ogive item response curves using Gibbs sampling," *Journal of Educational Statistics*, 17, 251–269.

Albert, J. H. and Chib, S. (1993), "Bayesian analysis of binary and polychotomous response data," *Journal of the American Statistical Association*, 88, 669–679.

Anderson, J. R. and Lebiere, C. (1998), *The atomic components of thought*, Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Ando, T. (2007), "Bayesian predictive information criterion for the evaluation of hierarchical Bayesian and empirical Bayes models," *Biometrika*, 94, 443–458.

Ayers, E. (2009), "Using R to Write WinBUGS COde," Tech. rep., American Institutes for Research, Washington DC, http://www.stat.cmu.edu/~eayers/BUGS.html.

Bartholomew, D. J. and Knott, M. (1999), *Latent variable models and factor analysis (Kendalls Library of Statistics, No. 7, 2nd. ed.*, New York, NY: Edward Arnold.

Béguin, A. A. and Glas, C. A. (2001), "MCMC estimation and some model-fit analysis of multidimensional IRT models," *Psychometrika*, 66, 541–561.

Bertsimas, D. and Tsitsiklis, J. (1993), "Simulated annealing," *Statistical Science*, 8, 10–15.

Billingsley, P. (1995), *Probability and Measure, 3rd Edition*, New York, New York, USA: Wiley-Interscience.

Bradlow, E., Wainer, H., and Wang, X. (1999), "A Bayesian Random Effect Model for Testlets," *Psychometrika*, 64, 153–168.

Brockwell, A. and Kadane, J. B. (2005), "Identi fication of regeneration times in MCMC simulation with application to adaptive schemes," *Journal of Statistical Computation and Graphics*, 14, 436–458.

Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L. (eds.) (2011), *Handbook of Markov Chain Monte Carlo*, Boca Raton FL: Chapman & Hall/CRC.

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

Burnham, K. and Anderson, D. (2002), *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*, New York: Springer.

Casella, G. and Robert, C. P. (1996), "Rao-Blackwellization of sampling schemes," *Biometrika*, 83, 81–94.

Celeux, G., Forbes, F., Robert, C., and Titterington, D. (2006), "Deviance Information Criteria for Missing Data Models (with discussion)," *Bayesian Analysis*, 651–674 (disc. 675–706).

Chang, H.-H. and Stout, W. (1993), "The asymptotic posterior normality of the latent trait in an IRT model," *Psychometrika*, 58, 37–52.

Chib, S. and Greenberg, E. (1995), "Understanding the Metropolis-Hastings Algorithm," *The American Statistician*, 49, 327–335.

Chib, S., Greenberg, E., and Chiband, S. (1995), "Understanding the metropolis-hastings algorithm," *American Statistician*, 49, 327–335.

Congdon, P. (2007), *Bayesian Statistical Modelling, 2nd Edition*, New York: Wiley.

Cook, L. L. and Eignor, D. R. (1991), "IRT Equating Methods," *Educational Measurement: Issues and Practice*, 10, 37–45.

Cowles, M. K. and Carlin, B. P. (1996), "Markov Chain Monte Carlo Convergence Diagnostics : A Comparative Review," *Journal of the American Statistical Assocation*, 91, 883–904.

CTB/McGraw-Hill (1996), *Comprehensive Test of Basic Skills, Fifth Edition*, Monterey, CA: Author.

Daumé III, H. (2007), "HBC: Hierarchical Bayes Compiler," Tech. rep., http://hal3.name/HBC/.

De Boeck, P. and Wilson, M. E. (2004), *Explanatory item response models: A generalized linear and nonlinear approach*, New York: Springer-Verlag.

de la Torre, J. and Douglas, J. (2004), "Higher-order latent trait models for cognitive diagnosis," *Psychometrika*, 69, 333–353.

Doucet, A., Briers, M., and Sénécal, S. (2006), "Efficient Block Sampling Strategies for Sequential Monte Carlo Methods," *Journal of Computational and Graphical Statistics*, 15, 693–711.

Fill, J. A. (1998), "An interruptable algorithm for perfect sampling via Markov chains," *Annals of Applied Probability*, 8, 131–162.

Flegal, J. and Jones, G. (2011), "Implementing MCMC: Estimating with confidence," in *Handbook of Markov Chain Monte Carlo*, eds. Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L., Boca Raton FL: Chapman & Hall/CRC, chap. 7, pp. 175–197.

Flegal, J. M., Haran, M., and Jones, G. L. (2008), "Markov Chain Monte Carlo: Can We Trust the Third Significant Figure?" *Statistical Science*, 23, 250–260.

Fox, J. (2010), *Bayesian item response modeling: Theory and applications*, New York: Springer.

Fox, J. P. (2011), "Joint Modeling of Longitudinal Item Response Data and Survival (Keynote Address)," Keynote Address, Escola de Modelos De Regressao. Fortaleza, Brazil.

Fox, J.-P. and Glas, C. A. (2001), "Bayesian Estimation of a Multilevel IRT Model Using Gibbs Sampling," *Psychometrika*, 66, 271–288.

Gelfand, A. E. and Smith, A. F. M. (1990), "Sampling-based approaches to calculating marginal densities," *Journal of the American Statistical Association*, 85, 398–409.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2003), *Bayesian Data Analysis*, New York, New York, USA: Chapman & Hall, 2nd ed.

Gelman, A., Meng, X.-L., and Stern, H. S. (1996), "Posterior Predictive Assessment of Model Fitness Via Realized Discrepancies (with discussion)," *Statistica Sinica*, 6, 33–87.

Gelman, A. and Rubin, D. B. (1992), "Inference from iterative simulation using multiple sequences," *Statistical Science*, 7, 457–511.

Gelman, A., Sturtz, S., Legges, U., Gorjanc, G., and Kerman, J. (2012), "R2WinBUGS: Running WinBUGS and OpenBUGS from R / S-PLUS," http://cran.r-project.org/web/packages/R2WinBUGS/index.html, version 2.1-18.

Geman, S. and Geman, D. (1984), "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images." *IEEE transactions on pattern analysis and machine intelligence*, 6, 721–41.

Geweke, J. (1992), "Evaluating the accuracy of sampling-based approaches to calculating posterior moments," in *Bayesian Statistics 4*, eds. Bernado, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M., Oxford, UK: Clarendon Press, pp. 169–194.

Geyer, C. J. (1996), "Estimation and Optimization of Functions," in *Markov Chain Monte Carlo in Practice*, eds. Gilks, W. R., Richardson, S., and Spiegelhalter, D. J., London: Chapman and Hall, pp. 241–258.

— (2011), "Introduction to Markov Chain Monte Carlo," in *Handbook of Markov Chain Monte Carlo*, eds. Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L., Boca Raton FL: Chapman & Hall/CRC, chap. 1, pp. 3–48.

Geyer, G. J. and Johnson, L. T. (2012), "mcmc: Markov Chain Monte Carlo (R package)," Tech. rep., http://www.stat.umn.edu/geyer/mcmc/.

Gilks, W. R., Richardson, S., and Spiegelhalter, D. J. (eds.) (1996), *Markov Chain Monte Carlo in Practice*, London: Chapman and Hall.

Gilks, W. R. and Roberts, G. O. (1996), "Strategies for improving MCMC," in *W. R*, London: Chapman and Hall, pp. 89–114.

Gilks, W. R. and Wild, P. (1992), "Adaptive Rejection Sampling for Gibbs Sampling," *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 41, 337–348.

Glas, C. A. W. and Pimentel, J. (2008), "Modeling nonignorable missing data in speeded tests," *Educational and Psychological Measurement*, 68, 907–922.

Graves, T. L., Speckman, P. L., and Sun, D. (2011), "Improved Mixing in MCMC Algorithms for Linear Models," *Journal of Computational and Graphical Statistics*.

Hadfield, J. (2010), "MCMC Methods for Multi-Response Generalized Linear Mixed Models: The MCMCglmm R Package," *Journal of Statistical Software*, 33, 1–22, http://www.jstatsoft.org/v33/i02/.

Haertel, E. H. (1989), "Using restricted latent class models to map the skill structure of achievement items," *Journal of Educational Measurement*, 26, 301–321.

Han, C. and Carlin, B. P. (2001), "MCMC Methods for Computing Bayes Factors: a Comparative Review," *Journal of the American Statistical Association*, 96, 1122–1132.

Hastings, W. (1970), "Monte Carlo samping methods using Markov chains and their applications," *Biometrika*, 57, 97–109.

Janssen, R., Tuerlinckx, F., Meulders, M., and de Boeck, P. (2000), "A Hierarchical IRT Model for Criterion-Referenced Measurement," *Journal of Educational and Behavioral Statistics*, 25, 285–306.

Johnson, M. S. and Sinharay, S. (2005), "Calibration of Polytomous Item Families Using Bayesian Hierarchical Modeling," *Applied Psychological Measurement*, 29, 369–400.

Jones, D. H. and Nediak, M. (2005), "Item Parameter Calibration of LSAT Items Using MCMC Approximation of Bayes Posterior Distributions," Tech. Rep. Computerized Testing Report 00-05, Law School Admission Council, Newton, PA, http://www.lsac.org/lsacresources/Research/CT/pdf/CT-00-05.pdf.

Junker, B. (1999), "Some statistical models and computational methods that may be useful for cognitively-relevant assessment," Tech. rep., Department of Statistics, Carnegie Mellon University, http://www.stat.cmu.edu/~brian/nrc/cfa/.

Junker, B. W. and Sijtsma, K. (2001), "Cognitive Assessment Models with Few Assumptions, and Connections with Nonparametric Item Response Theory," *Applied Psychological Measurement*, 25, 258–272.

Kamata, A. (2001), "Item analysis by the hierarchical generalized linear model," *Journal of Educational Measurement*, 38, 79–93.

Karabatsos, G. and Sheu, C. (2004), "Order-Constrained Bayes Inference for Dichotomous Models of Unidimensional Nonparametric IRT," *Applied Psychological Measurement*, 28, 110–125.

Kass, R. E. and Raftery, A. E. (1995), "Bayes Factors," *Journal of the American Statistical Association*, 90, 773–795.

Kass, R. E. and Wasserman, L. (1995), "A reference Bayesian test for nested hypotheses and its relation to the Schwarz criterion," *Journal of the American Statistical Association*, 90, 928–934.

Lazarsfeld, P. F. and Henry, N. W. (1968), *Latent structure analysis*, Boston: Houghton Mifflin.

Ligges, U. (2012), "Package BRugs," Tech. rep., Technische Universität Dortmund, Germany, http://cran.r-project.org/web/packages/BRugs/index.html.

Lord, F. M. and Novick, M. R. (1968), *Statistical theories of mental test scores*, Reading, MA: Addison-Wesley, with contributions by Allan Birnbaum.

Los Alamos National Laboratory (2012), "History Center," Tech. rep., http://www.lanl.gov/history/.

Lunn, D., Spiegelhalter, D., Thomas, A., and Best, N. (2009), "The BUGS project: Evolution, critique and future directions," *Statistics in Medicine*, 28, 3049–3067.

Lynch, S. M. and Western, B. (2004), "Bayesian Posterior Predictive Checks for Complex Models," *Sociological Methods & Research*, 32, 301–335.

Macready, G. B. and Dayton, C. M. (1977), "The use of probabilistic models in the assessment of mastery," *Journal of Educational Statistics*, 2, 99–120.

Maier, K. (2001), "A Rasch hierarchical measurement model," *Journal of Educational and Behavioral Statistics*, 26, 307–330.

Mariano, L. T. and Junker, B. (2007), "Covariates of the rating process in hierarchical models for multiple ratings of test items," *Journal of Educational and Behavioral Statistics*, 32, 287–314.

Maris, E. (1995), "Psychometric latent response models," *Psychometrika*, 60, 523–547.

Martin, A., Quinn, K., and Park, J. (2009), "MCMCpack: Markov Chain Monte Carlo (R package)," Tech. rep., http://CRAN.R-project.org/package=MCMCpack.

Matteucci, M. and Veldkamp, B. V. (2011), "On the use of MCMC CAT with empirical prior information to improve the efficiency of CAT," Tech. rep., AMS Acta, Università di Bologna, http://amsacta.unibo.it/3109/.

Metropolis, N. (1987), "The beginning of the Monte Carlo method," *Los Alamos Science (1987 Special Issue dedicated to Stanisaw Ulam)*, 125–130.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953), "Equations of state calculations by fast computing machines," *J. Chem. Phys.*, 21, 1087–1091.

Miyazaki, K. and Hoshino, T. (2009), "A Bayesian Semiparametric Item Response Model with Dirichlet Process Priors," *Psychometrika*, 74, 375–393.

Monahan, J. F. (2011), *Numerical Methods of Statistics*, New York, NY: Cambridge University Press.

Mykland, P., Tierney, L., and Yu, B. (1995), "Regeneration in Markov Chain Samplers," *Journal of the American Statistical Association*, 90, 233–241.

Neal, R. (2003), "Slice sampling," *Annals of Statistics*, 31, 705–767.

Patil, A., Huard, D., and Fonnesbeck, C. J. (2010), "PyMC: Bayesian Stochastic Modelling in Python," *Journal of Statistical Software*, 35, http://www.jstatsoft.org/.

Patz, R. J. and Junker, B. W. (1999a), "A Straightforward Approach to Markov Chain Monte Carlo Methods for Item Response Models," *Journal of Educational and Behavioral Statistics*, 24, 146–178.

— (1999b), "Applications and Extensions of MCMC in IRT: Multiple Item Types, Missing Data, and Rated Responses," *Journal of Educational and Behavioral Statistics*, 24, 342–366.

Patz, R. J., Junker, B. W., Johnson, M. S., and Mariano, L. T. (2002), "The hierarchical rater model for rated test items and its application to large-scale educational assessment data," *Journal of Educational and Behavioral Statistics*, 27, 341–384.

Plummer, M. (2012a), "JAGS Version 3.2.0 user manual," Tech. rep., http://mcmc-jags.sourceforge.net.

— (2012b), "Package rjags," Tech. rep., http://cran.r-project.org/web/packages/rjags/index.html.

Polson, N. G., Scott, J. G., and Windle, J. (2013), "Bayesian inference for logistic models using Pólya-Gamma latent variables," *Journal of the American Statistical Association*, 108, 1339–1349, available at http://arxiv.org/pdf/1205.0310.pdf.

R Development Core Team (2012), "R: A Language and Environment for Statistical Computing," Tech. rep., R Foundation for Statistical Computing, Vienna, Austria, http://www.r-project.org/.

Ripley, B. D. (1987), *Stochastic Simulation*, New York: Wiley & Sons.

Rizopoulos, D. (2006), "ltm: An R package for latent variable modelling and item response theory analyses," *Journal of Statistical Software*, 17, 1–25, http://www.jstatsoft.org/v17/i05/.

Robert, C. and Casella, G. (2011), "A Short History of Markov Chain Monte Carlo: Subjective Recollections from Incomplete Data," *Statistical Science*, 26, 102–115.

Roberts, G. and Sahu, S. K. (1997), "Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler," *Journal of the Royal Statistical Society. Series B (Methodological)*, 59, 291–317.

Rosenthal, J. S. (2011), "Optimal proposal distributions and adaptive MCMC," in *Handbook of Markov Chain Monte Carlo*, eds. Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L., Boca Raton FL: Chapman & Hall/CRC, chap. 4, pp. 93–111.

Roussos, L. A., DiBello, L. V., Stout, W., Hartz, S. M., Henson, R. A., and Templin, J. H. (2007), "The Fusion Model Skills Diagnosis System," in *Cognitive Diagnostic Assessment for Education: Theory and Applications*, eds. Leighton, J. and Gierl, M., Cambridge University Press.

Rupp, A., Templin, J., and Henson, R. (2010), *Diagnostic Assessment: Theory, Methods, and Applications*, New York: Guilford.

Sass, D. A., Schmitt, T. A., and Walker, C. M. (2008), "Estimating Non-Normal Latent Trait Distributions within Item Response Theory Using True and Estimated Item Parameters," *Applied Measurement in Education*, 21, 65–88.

Scheiblechner, H. (1972), "Das Lernen und Lösen komplexer Denkaufgaben [The Learning and Solving of Complex Reasoning Items]," *Zeitschrift für Experimentelle und Angewandte Psychologie*, 3, 456–506.

Schwarz, G. E. (1978), "Estimating the dimension of a model," *Annals of Statistics*, 6, 461–464.

Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H., George, E., and McCulloch, R. (2013), "Bayes and big data: the consensus Monte Carlo algorithm," in *EFaBBayes 250 conference (Vol 16)*, available at http://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/41849.pdf.

Segall, D. (2002), "An item response model for characterizing test compromise," *Journal of Educational and Behavioral Statistics*, 27, 163–179.

Segall, D. O. (2003), "Calibrating CAT Pools and Online Pretest Items Using MCMC Methods," in *Annual Meeting of the National Council on Measurement in Education*, Chicago IL.

Seltman, H. (2010), "R Package rube (Really Useful WinBUGS Enhancer)," http://www.stat.cmu.edu/~hseltman/rube/, version 0.2-13.

Serfling, R. (1980), *Approximation Theorems of Mathematical Statistics*, New York: Wiley.

Shu, Z. (2010), "Detecting test cheating using a deterministic, gated item response theory model," Ph.D. thesis, The University of North Carolina at Greensboro, Greensboro NC.

Sinharay, S. and Almond, R. G. (2007), "Assessing Fit of Cognitive Diagnostic Models A Case Study," *Educational and Psychological Measurement*, 67, 239–257.

Sinharay, S., Johnson, M., and Stern, H. (2006), "Posterior Predictive Assessment of Item Response Theory Models," *Applied Psychological Measurement*, 30, 298–321.

Sinharay, S., Johnson, M. S., and Williamson, D. M. (2003), "Calibrating Item Families and Summarizing the Results Using Family Expected Response Functions," *Journal of Educational and Behavioral Statistics*, 28, 295–313.

Smith, B. J. (2007), "boa: An R Package for MCMC Output Convergence Assessment and Posterior Inference," *Journal of Statistical Software*, 21, 1–37.

Spiegelhalter, D., Thomas, A., Best, N., and Lunn, D. (2012), "OpenBUGS User Manual," Tech. rep., http://www.openbugs.info.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., and van der Linde, A. (2002), "Bayesian measures of model complexity and fit (with discussion)," *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 64, 583–639.

Stan Development Team (2012), "Stan Modeling Language: Users Guide and Reference Manual. Version 1.0." Tech. rep., http://mc-stan.org/.

Studer, C. E. (2012), "Incorporating Learning Over Time into the Cognitive Assessment Framework," Ph.D. thesis, Carnegie Mellon University, Pittsburgh PA.

Tanner, M. A. and Wong, W. H. (1987), "The calculation of posterior distributions by data augmentation (with discussion)," *Journal of the American Statistical Association*, 82, 528–550.

Thissen, D. and Edwards, M. C. (2005), "Diagnostic Scores Augmented Using Multidimensional Item Response Theory: Preliminary Investigation of MCMC Strategies," in *Annual Meeting of the National Council on Measurement in Education*, Montreal, Canada.

Thompson ISI (2012), "Web of Knowlege," Tech. rep., http://wokinfo.com/.

Tierney, L. (1994), "Markov chains for exploring posterior distributions," *The Annals of Statistics*, 22, 1701–1728.

van den Oord, E. J. C. G. (2005), "Estimating Johnson Curve Population Distributions in MULTILOG," *Applied Psychological Measurement*, 29, 45–64.

VanHoudnos, N. (2012), "Online Supplement to "Markov Chain Monte Carlo for Item Response Models"," Tech. rep., Department of Statistics, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, http://mcmcinirt.stat.cmu.edu/.

Walker, A. M. (1969), "On the asymptotic behaviour of posterior distributions," *J. R. Statist. Soc.*, 31, 80–88.

Wasserman, L. (2000), "Bayesian Model Selection and Model Averaging," *Journal of Mathematical Psychology*, 44, 92–107.

Weaver, R. (2008), "Parameters, Predictions, and Evidence in Computational Modeling: A Statistical View Informed by ACTR," *Cognitive Science*, 1349–1375.

Woods, C. M. (2006), "Ramsay-curve item response theory (RC-IRT) to detect and correct for nonnormal latent variables," *Psychological Methods*, 11, 253–270.

Woods, C. M. and Thissen, D. (2006), "Item response theory with estimation of the latent population distribution using spline-based densities," *Psychometrika*, 71, 281–301.