4

Introduction to Markov chain Monte Carlo methods

4.1 Bayesian computation

4.1.1 Single-parameter models

As discussed in Chapter 3, the posterior distribution contains all the information needed for Bayesian inference. In all of the examples encountered thus far there is a *single* unknown parameter, whose posterior distribution might be graphed to provide a complete picture of the current state of knowledge arising from the data and prior information. More generally, though, we wish to calculate numeric summaries of the posterior distribution via integration, e.g., $E[\theta|y] = \int_\theta \theta p(\theta|y)\, d\theta$. In the *conjugate* examples considered so far, the posterior distribution is available in closed form and so the required integrals are straightforward to evaluate. However, outside the conjugate family of models, the posterior is usually of non-standard form (although we can always write down its density function to within a constant of proportionality). As a consequence, at least some of the integrals required for summarising the distribution are difficult.

Various methods are available for evaluating such integrals. In cases where we can sample directly from the posterior, such as in conjugate problems, we could use Monte Carlo simulation (if we wished to venture beyond standard results). More generally, however, we could try to obtain an approximation to the posterior density that is analytically tractable, for example, assuming asymptotic normality of the posterior or more complex techniques such as Laplace's method (see, for example, Carlin and Louis (2008); Gelman et al. (2004) for further details). Alternatively, numerical integration methods can be used (Davis and Rabinowitz (1975); Press et al. (2002), Ch. 4). Standard techniques include Gaussian quadrature, or a form of (non-iterative) Monte Carlo integration, which differs from the form described in § 1.4. There we could obtain a direct sample from $p(\theta|y)$ — here we cannot, so we would integrate by sampling points uniformly from the region to be integrated over, averaging the values of the integrand at those points, finally multiplying by the size of the region.

Here, however, we focus exclusively on the class of iterative methods known

as Markov chain Monte Carlo (MCMC) integration (Gelfand and Smith, 1990; Geman and Geman, 1984; Metropolis et al., 1953; Hastings, 1970). These are by far the most powerful and flexible class of algorithms available for Bayesian computation, though see § 4.6 for a brief discussion of situations where MCMC is not well suited. We first present an example in which the single parameter of interest has a non-standard posterior, to illustrate the ease with which complex integrals can be evaluated using MCMC in BUGS. Later, after discussing *multi-parameter* models, we will describe the types of MCMC algorithm used by BUGS for performing such computations.

Example 4.1.1. *Surgery (continued): non-conjugate inference*

Suppose we observe the number of deaths y in a given hospital for a high-risk operation. Let n denote the total number of such operations performed and suppose we wish to make inferences regarding the underlying *true* mortality rate, θ, say. The likelihood, up to a constant of proportionality, is given by

$$p(y|\theta) \propto \theta^y (1-\theta)^{n-y}.$$

Note that θ must lie between 0 and 1, and suppose that to impose this constraint we choose a non-conjugate, normal prior for the logistic transform of θ:

$$\text{logit}\,\theta = \log\left(\frac{\theta}{1-\theta}\right) \sim \text{Normal}(\mu, \omega^2)$$

$$\Rightarrow p(\theta) = \frac{1}{\theta(1-\theta)} \times \frac{1}{\omega\sqrt{2\pi}} \exp\left\{-\frac{1}{2\omega^2}(\text{logit}\,\theta - \mu)^2\right\}.$$

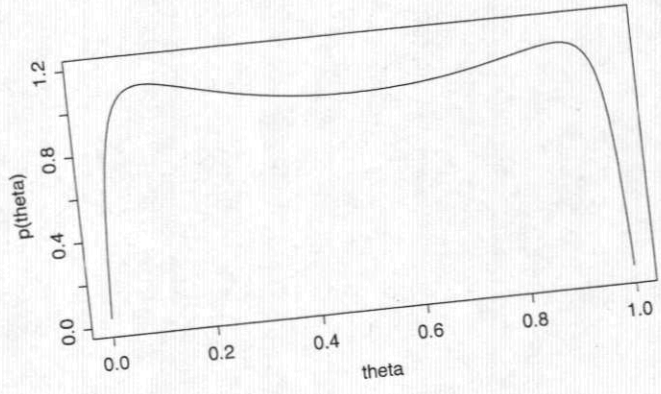

FIGURE 4.1
Prior density for θ in the case where $\text{logit}\,\theta \sim \text{Normal}(0, 2.71)$.

Figure 4.1 shows the prior density for θ with $\mu = 0$ and $\omega^2 = 2.71$, which correspond to a good approximation of the standard logistic density (Appendix C

and § 5.2.5), which would be uniform on the scale of θ. Multiplying prior and likelihood together gives the posterior

$$p(\theta|y) = A \times \theta^{y-1}(1-\theta)^{n-y-1} \exp\left\{-\frac{1}{2\omega^2}(\text{logit}\,\theta - \mu)^2\right\},$$

where A is the normalising constant required to make the density integrate to 1. A is analytically intractable, but even if we knew A, the posterior expectation

$$E[\theta|y] = A \times \int_0^1 \theta^{y}(1-\theta)^{n-y-1} \exp\left\{-\frac{1}{2\omega^2}(\text{logit}\,\theta - \mu)^2\right\} d\theta$$

would still be intractable.

In Example 3.3.1, where a conjugate Beta(.,.) prior was specified for θ, we were able to derive the posterior in closed form and perform Monte Carlo integration directly by specifying that closed form as a sampling distribution in BUGS. With a normal prior on $\text{logit}\,\theta$, however, there is no closed-form posterior. In such cases BUGS can perform *Markov chain* Monte Carlo integration instead if we simply specify the likelihood and prior separately. Suppose $y = 10$ and $n = 100$:

```
y             <- 10
n             <- 100
#############################
y              ~ dbin(theta, n)
logit(theta) <- logit.theta
logit.theta    ~ dnorm(0, 0.368)  # precision = 1 / 2.71
```

The software knows how to derive the posterior distribution and subsequently sample from it. The resulting samples are used as in standard Monte Carlo integration to compute various posterior summaries, e.g.,

```
node  mean   sd      MC error 2.5%    median 97.5%  start sample
theta 0.1081 0.03029 3.387E-4 0.05725 0.1052 0.1744 1001  10000
```

Hence the posterior mean is 0.108 and an approximate 95% credible interval for θ is (0.0573, 0.174). Given the prior distribution and the observed data, we can be 95% sure that the "true" mortality rate lies between 0.0573 and 0.174. These results are very similar to those obtained in the conjugate case with θ assigned a fully uniform Beta$(1, 1)$ prior, as opposed to the approximately uniform prior shown in Figure 4.1: mean = 0.108, interval = (0.0557, 0.175).

4.1.2 Multi-parameter models

More generally we are interested in models with more than one unknown parameter. As the number of parameters increases, however, it is increasingly difficult to identify a conjugate prior, to the extent that for all but the simplest of problems the *joint* posterior distribution is of non-standard form. In

addition, the integrals required for inference become high dimensional. For example, suppose we have a joint posterior distribution for the vector of unknowns $\theta = \{\theta_1, \ldots, \theta_k\}$. We often want to base inference on the *marginal* posterior of a subset of the parameters: the marginal posterior for θ_1, say, is

$$p(\theta_1|y) = \int_{\theta_2} \cdots \int_{\theta_k} p(\theta|y)\, d\theta_2 \ldots d\theta_k.$$

In such cases MCMC is often the *only* suitable method of integration.

Example 4.1.2. *A multi-parameter model*

Suppose we have observed data y_i, $i = 1, \ldots, n$, which we believe arise from a heavy-tailed Student-t distribution with unknown mean μ, unknown inverse-scale-squared r, and unknown degrees of freedom d (see Appendix C.1). Further suppose that we specify independent Normal(γ, ω^2) and Gamma(α, β) priors for μ and r, respectively, and an independent discrete-uniform prior for d on the set $\{2, 3, \ldots, 30\}$. The *joint* posterior distribution is given by

$$\begin{aligned} p(\mu, r, d|y) \propto \left\{ \frac{\Gamma(\frac{d+1}{2})}{\Gamma(\frac{d}{2})} \sqrt{\frac{r}{d\pi}} \right\}^n \prod_{i=1}^n \left\{ 1 + \frac{r}{d}(y_i - \mu)^2 \right\}^{-(d+1)/2} & \quad \text{[likelihood]} \\ \times \exp\left\{ -\frac{1}{2\omega^2}(\mu - \gamma)^2 \right\} & \quad \text{[prior } \mu\text{]} \\ \times r^{\alpha-1} \exp(-\beta r) & \quad \text{[prior } r\text{]} \\ \times 1/29 & \quad \text{[prior } d\text{]}, \end{aligned}$$

which is certainly of non-standard form! Suppose we wish to make *marginal* inferences about the unknown degrees of freedom d. Then we need

$$p(d|y) = \int_\mu \int_r p(\mu, r, d|y)\, dr d\mu,$$

which is intractable. In BUGS we simply specify the likelihood and each prior as follows:

```
for (i in 1:n) {y[i]    ~ dt(mu, r, d)}
mu                      ~ dnorm(gamma, inv.omega.squared)
r                       ~ dgamma(alpha, beta)
d                       ~ dcat(p[])
p[1]                    <- 0
for (i in 2:30) {p[i] <- 1/29}
```

The software then uses Markov chain Monte Carlo to generate samples from the joint posterior distribution $p(\mu, r, d|y)$. These can be used to make arbitrary inferences about the joint posterior or, by simply ignoring samples not pertaining to the variable(s) of interest, to make marginal inferences about any subset of the parameters. For example, we first generate a toy data set by simulating $n = 100$

values from a t-distribution with $\mu = 0$, $r = 1$, and $d = 4$. We then fit the above BUGS model, with "vague" priors given by $\gamma = 0$, $\omega = 100$, and $\alpha = \beta = 10^{-3}$ (see Chapter 5 for discussion of why these might be suitable choices), and initial values `list(mu = 0, r = 1, d = 10)`, to obtain

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
d	12.82	7.584	0.1648	3.0	11.0	29.0	1	100000
mu	0.04393	0.09752	5.455E-4	-0.1467	0.0431	0.2368	1	100000
r	1.339	0.3203	0.005169	0.8774	1.282	2.123	1	100000

Hence, we can immediately infer that the degrees of freedom has a (marginal) posterior median of 11 and a 95% credible interval of [3, 29]. Visual inspection of the posterior (Figure 4.2) reveals that there is limited information in the data regarding d but that the mode, 5, is close to the true value of 4.

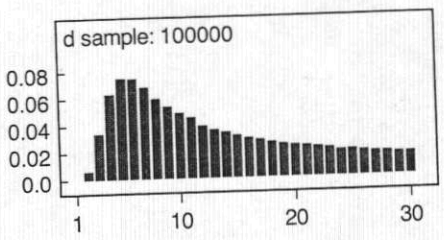

FIGURE 4.2
Approximate posterior distribution for number of degrees of freedom from analysis of 100 observations from $t(0, 1, 4)$.

4.1.3 Monte Carlo integration for evaluating posterior integrals

As we have seen in §1.4 we can calculate arbitrary summaries of interest for a given distribution by Monte Carlo integration. Hence, assuming we can obtain a sample of realisations from the joint posterior $p(\theta|y)$, we have an entirely general method for evaluating the integrals necessary for Bayesian inference. So how might we obtain a sample from $p(\theta|y)$?

For all but the most tractable of posterior distributions, this cannot be done directly. However, a number of algorithms exist for *indirect* sampling from non-standard distributions. In general, these methods work by sampling values from an approximate distribution and then correcting or adjusting the values so that they better resemble a sample from the true distribution of interest. The book by Ripley (1987) offers the interested reader a thorough

account of many such methods. Amongst the most widely used are importance sampling and rejection sampling. These are non-iterative algorithms, in the sense that the same approximation to the *target* distribution is used throughout. However, to use them for Bayesian computation necessitates finding a density that is a good approximation to the (log) joint posterior *and* that is easy to sample from directly. For many realistically complex Bayesian models, this is difficult or impossible to do (using generic methods that do not have to be tuned to specific applications).

The alternative is to use an *iterative* algorithm, in which a single realisation from the approximating distribution is drawn at each iteration, but the approximate distribution is *improved at each step*. Once the approximating distribution is sufficiently close to the target (i.e., the joint posterior), successive draws from this distribution can be considered to form a sample from the joint posterior of interest. Hence as the iterations proceed, the approximating distribution can be thought of as *converging* towards the posterior. Theorems exist which prove that if the approximating distribution is set up in a certain way (essentially so that the successive realisations form a Markov chain with appropriate transition probabilities — see below), then this convergence will occur almost surely as T (the number of iterations) $\to \infty$ (Tierney, 1994; Roberts and Rosenthal, 2004; Robert and Casella, 2004; Asmussen and Glynn, 2011). In practice, of course, only a finite number of iterations is possible, and as we shall see in §4.4, deciding at which point the approximating distribution is *close enough* to the target posterior is crucial when using these methods for Bayesian inference.

4.2 Markov chain Monte Carlo methods

As hinted above, one of the most reliable and general methods for choosing a suitable iterative approximating distribution for sampling from complex Bayesian posterior distributions is to use a Markov chain. Formally, a sequence of random variables $X^{(0)}, X^{(1)}, X^{(2)}, \ldots$ forms a Markov chain if, for all t, the distribution of the $t+1^{th}$ variable in the sequence is given by

$$X^{(t+1)} \sim p_{\text{trans}}(x|X^{(t)} = x^{(t)}), \tag{4.1}$$

that is, conditional on the value of $X^{(t)}$, the distribution of $X^{(t+1)}$ is independent of all other preceding values, $X^{(t-1)}, \ldots, X^{(0)}$. The right-hand side of (4.1) is called the *transition distribution* of the Markov chain and defines the *conditional* probability of moving to any particular new value given the current value of the chain. Subject to fairly general regularity conditions (including irreducibility and aperiodicity, see Cox and Miller (1965)), the *marginal* (or *un*conditional) distribution of $X^{(t+1)}$ will converge to a unique *stationary dis-*

tribution as $t \to \infty$. In simple terms, this means that although each variable in the chain depends directly on its predecessor, eventually (as t increases) we reach a point such that for practical purposes, all subsequent values are distributed *marginally* according to the same fixed distribution, which, crucially, is independent of the starting value $X^{(0)}$. In other words, the chain eventually forgets where it started and conforms to an underlying "equilibrium" distribution.

So how does this help us to generate realisations of θ from the joint posterior distribution $p(\theta|y)$ in a Bayesian analysis? Replacing the random variable X above by the random vector θ, the answer is to choose a transition distribution suitable for generating (from an arbitrary initial state $\theta^{(0)}$) a sequence of realisations $\theta^{(1)}, \theta^{(2)}, \theta^{(3)}, \ldots$ whose unique stationary distribution is the joint posterior of interest $p(\theta|y)$. The marginal distributions of the $\theta^{(t)}$s ($t = 1, 2, 3, \ldots$) play the role of the approximating distributions discussed earlier, with the approximation becoming successively closer to the target posterior as the Markov chain converges to its stationary distribution.

Many methods exist for designing and sampling from such transition distributions, and their suitability depends on the nature of the joint posterior distribution to be explored. As Bayesian models have become more and more sophisticated, so people have invented cleverer and cleverer algorithms for constructing efficient Markov chains to sample from required posterior distributions. Inevitably there is a trade-off between the generality of a particular method and its ability to sample efficiently from complex, high-dimensional densities through fine-tuning. Here we focus on the main algorithms used by the BUGS software, which, of necessity, are designed to be robust in a wide range of applications rather than optimised for specific cases. Some strategies and tricks for improving the efficiency of BUGS simulations in certain situations are discussed, for example, in §6.1, §10.5, §11.2.

Note that, in general, MCMC methods generate a *dependent* sample from the joint posterior of interest, since each realisation depends directly on its predecessor. We can still use this sample as the basis for Monte Carlo integration, however, as all of the results discussed in §1.4 still hold.

4.2.1 Gibbs sampling

The Gibbs sampler (Geman and Geman, 1984; Gelfand and Smith, 1990; Casella and George, 1992) is one of the most widely used algorithms for simulating Markov chains. It is a special case of the Metropolis–Hastings algorithm (Metropolis et al., 1953; Hastings, 1970) and generates a multi-dimensional Markov chain by splitting the vector of random variables θ into subvectors (often scalars) and sampling each subvector in turn, conditional on the *most recent* values of all other elements of θ. The algorithm proceeds as follows. Let the vector of unknowns θ consist of k sub-components, i.e., $\theta = (\theta_1, \theta_2, \ldots, \theta_k)$:

1. Choose arbitrary starting values $\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_k^{(0)}$ for each component,

where subscripts denote sub-components of θ and superscripts denote the iteration number (iteration zero being the initial state of the Markov chain).

2. Sample new values for each element of θ by cycling through the following steps:

   - Sample a new value for θ_1, from the *full conditional distribution* of θ_1 given the most recent values of all other elements of θ and the data:
   $$\theta_1^{(1)} \sim p(\theta_1|\theta_2^{(0)}, \theta_3^{(0)}, \ldots, \theta_k^{(0)}, y).$$
   - Sample a new value $\theta_2^{(1)}$ for the second component of θ, from its full conditional distribution $p(\theta_2|\theta_1^{(1)}, \theta_3^{(0)}, \ldots, \theta_k^{(0)}, y)$. Note that as a new value for θ_1 has already been sampled, it is this "most recent" value that is conditioned upon, together with the starting values for all other elements of θ.
   - ...
   - Sample $\theta_k^{(1)}$ from $p(\theta_k|\theta_1^{(1)}, \theta_2^{(1)}, \ldots, \theta_{k-1}^{(1)}, y)$.

   This completes one iteration of the Gibbs sampler and generates a new realisation of the vector of unknowns, $\theta^{(1)}$.

3. Repeat stage 2 many times, always conditioning on the most recent values of other parameters, to obtain a sequence of dependent realisations of the vector of unknowns $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(T)}$ (where T is typically of the order of many thousands).

Figure 4.3 graphically illustrates the algorithm in the case of a hypothetical two-parameter problem ($k = 2$). The beauty of Gibbs sampling is that simulation from a complex, high-dimensional joint posterior distribution is reduced to a sequence of algorithms for sampling from one- or low-dimensional distributions. As we shall see in the next subsection, these univariate or low-dimensional full conditional distributions can usually be simplified by exploiting the conditional independence structure of the model, and in many cases are available in closed form (see Example 4.2.2 below), in which case direct sampling is straightforward using a specialized, distribution-specific random number generator (Ripley, 1987).

4.2.2 Gibbs sampling and directed graphical models

Suppose the model of interest can be represented as a directed acyclic graph with stochastic nodes $\mathcal{G}$ and directed links $\mathcal{L}$. As discussed in § 2.1.2, the conditional independence assumptions expressed through the DAG structure allow us to write $p(\mathcal{G}) = \prod_{v \in \mathcal{G}} p(v|\text{pa}[v])$ (Lauritzen et al., 1990). That is, the joint distribution of all nodes is given by the product, over all nodes, of the

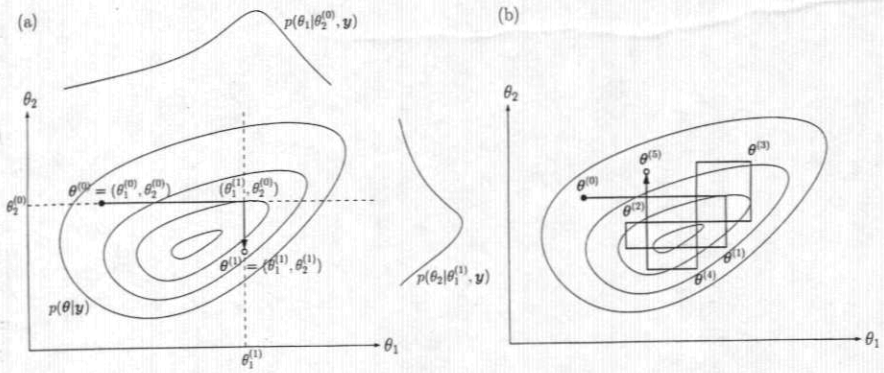

FIGURE 4.3
(a) First iteration of the Gibbs sampler for an illustrative two-parameter (bivariate) problem. The contours show the "height" of the true bivariate posterior distribution $p(\theta|y) = p(\theta_1, \theta_2|y)$. The starting point of the Gibbs sampler $\theta^{(0)} = (\theta_1^{(0)}, \theta_2^{(0)})$ is shown by the solid dot, and the pair of values $(\theta_1^{(1)}, \theta_2^{(1)})$ sampled in the first iteration is shown by the open circle. The univariate density projected onto the top horizontal axis shows the full conditional distribution $p(\theta_1|\theta_2^{(0)}, y)$, which is obtained by taking a horizontal "slice" through the joint posterior distribution at the value $\theta_2 = \theta_2^{(0)}$ (indicated by the horizontal dashed line). A new value for θ_1 ($\theta_1^{(1)}$) is generated from this full conditional, and then a "slice" parallel to the θ_2 axis is taken through the joint posterior at $\theta_1 = \theta_1^{(1)}$ (vertical dashed line). This gives the univariate full conditional $p(\theta_2|\theta_1^{(1)}, y)$, which is shown projected onto the right-hand vertical axis, and from which a new value for θ_2 ($\theta_2^{(1)}$) is sampled. (b) First five iterations of the Gibbs sampler shown in (a). Note that the sampler always moves parallel to the axes.

assumed distribution of each node conditional on its parents; in other words, the product of all distributional assumptions. The set of all unknowns θ and the set of all data y together form a partition of $\mathcal{G}$, and so

$$p(\theta, y) = \prod_{v \in \mathcal{G}} p(v|\text{pa}[v]). \tag{4.2}$$

From the definition of conditional probability, $p(\theta|y) = p(\theta, y)/p(y)$, which is proportional to $p(\theta, y)$ when considered as a function of θ. Hence the joint posterior can be obtained trivially for any DAG, up to a constant of proportionality. Similarly, *any* conditional distribution involving all nodes in the graph is also proportional to $p(\theta, y)$. Thus for any unobserved node (or set of nodes) θ_i, say, the full conditional distribution $p(\theta_i|\theta_{\setminus i}, y)$, where $\theta_{\setminus i}$ denotes "all elements of θ except θ_i," is proportional to $p(\theta, y)$ and can therefore be expressed as the right-hand side of (4.2). However, we are seeking to identify

a distribution in θ_i and so any factor in (4.2) not involving θ_i can be ignored, since it forms part of the normalising constant. Hence we obtain

$$p(\theta_i|\theta_{\setminus i}, y) \propto p(\theta_i|\text{pa}[\theta_i]) \times \prod_{v \in \text{ch}[\theta_i]} p(v|\text{pa}[v]), \tag{4.3}$$

and so the full conditional is dependent only on $\text{pa}[\theta_i]$, $\text{ch}[\theta_i]$ and all *co-parents* of θ_i's children. Collectively these three sets of nodes form a neighbourhood in the graph around θ_i known as the *Markov blanket*; θ_i is *conditionally independent* of all other nodes in the graph, *given* the Markov blanket. The subsequent derivation of a closed form for the full conditional (where available) is exactly analogous to the derivation of a closed-form posterior in conjugate, single-parameter models. The first term on the right-hand side of (4.3) plays the role of the prior distribution and is referred to as the "prior component." The product in (4.3) plays the role of the likelihood and is known as the "likelihood component." Finally, as we are conditioning on the most recent values of all other nodes, it is as if those nodes have *known* values, and so there is effectively only one unknown, θ_i.

Example 4.2.1. *The Markov blanket*

Consider the directed acyclic graph shown in Figure 4.4. Suppose we wish to derive the full conditional distribution for node C. This is proportional to the product of distributions for C and all of its children, conditional on their parents, i.e.,

$$p(\text{C}|\text{A}, \text{B}, \text{D}, \ldots, \text{I}) \propto p(\text{C}|\text{A}, \text{B}) \times p(\text{E}|\text{C}, \text{D}) \times p(\text{F}|\text{C}, \text{D}),$$

since $\text{ch}[\text{C}] = \{\text{E}, \text{F}\}$. The Markov blanket for C is given by its parents, its children, and all co-parents of its children, in this case $\{\text{A}, \text{B}, \text{D}, \text{E}, \text{F}\}$. *Given* the Markov blanket, a node is conditionally independent of all other nodes in the graph (G, H, and I here), and so $p(\text{C}|\text{A}, \text{B}, \text{D}, \ldots, \text{I})$ can be rewritten as $p(\text{C}|\text{A}, \text{B}, \text{D}, \text{E}, \text{F})$.

Example 4.2.2. *Full conditional distributions*

Suppose we observe data $y_1, \ldots, y_n$, assumed to form a random sample from a normal distribution with unknown mean μ and unknown precision τ. Further suppose that we specify independent priors on μ and τ as follows:

$$\mu \sim \text{Normal}(\gamma, \omega^2); \quad \tau \sim \text{Gamma}(\alpha, \beta).$$

Note that these are the conjugate priors for the cases in which the precision and the mean, respectively, have known values (see Table 3.1). Note also, however, that the joint prior $p(\mu, \tau) = \text{Normal}(\gamma, \omega^2) \times \text{Gamma}(\alpha, \beta)$ does *not* lead to a closed-form joint posterior. Thus, in order to make inferences about μ and τ, we might run a Gibbs sampler. We first choose arbitrary starting values, $\mu^{(0)} = 5$ and $\tau^{(0)} = 10$, say, and then an equally arbitrary updating order, μ then τ, say.

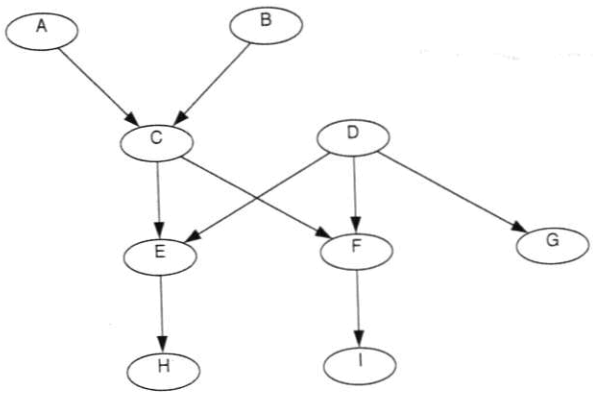

FIGURE 4.4
Directed acyclic graph for Example 4.2.1, showing parent–child relationships between nine nodes.

At iteration t of the Gibbs sampler we draw $\mu^{(t)} \sim p(\mu|\tau^{(t-1)}, y)$ and $\tau^{(t)} \sim p(\tau|\mu^{(t)}, y)$, where the full conditional sampling distributions can be derived from the DAG shown in Figure 4.5. The full conditional for μ is proportional to the prior for μ multiplied by the distribution of each child of μ conditional on that child's parents. From the graph, $\mathrm{ch}[\mu] = \{y_i, i = 1, \ldots, n\}$, and so

$$p(\mu|\tau^{(t-1)}, y) \;\propto\; \exp\left\{-\frac{1}{2\omega^2}(\mu-\gamma)^2\right\} \times \exp\left\{-\frac{\tau^{(t-1)}}{2}\sum_{i=1}^{n}(y_i-\mu)^2\right\}.$$

Conditioning on $\tau = \tau^{(t-1)}$ is essentially the same as assuming τ to be known. Hence this is exactly the same type of calculation as is required for deriving the single-parameter posterior (for μ) in the unknown mean, known precision/variance case — see §3.3.2. Therefore,

$$p(\mu|\tau^{(t-1)}, y) \;=\; \text{Normal}\left(\frac{\tau^{(t-1)}\sum y_i + \omega^{-2}\gamma}{n\tau^{(t-1)} + \omega^{-2}},\; \frac{1}{n\tau^{(t-1)} + \omega^{-2}}\right).$$

Similarly, $\mathrm{ch}[\tau] = \{y_i, i = 1, \ldots, n\}$, and so

$$p(\tau|\mu^{(t)}, y) \;\propto\; \tau^{\alpha-1}\exp\{-\beta\tau\} \times \tau^{\frac{n}{2}} \exp\left\{-\frac{\tau}{2}\sum_{i=1}^{n}(y_i-\mu^{(t)})^2\right\},$$

which is the same as in the known mean, unknown precision case presented in Table 3.1. Hence

$$p(\tau|\mu^{(t)}, y) \;=\; \text{Gamma}\left(\alpha + \frac{n}{2},\; \beta + \frac{1}{2}\sum_{i=1}^{n}(y_i-\mu^{(t)})^2\right).$$

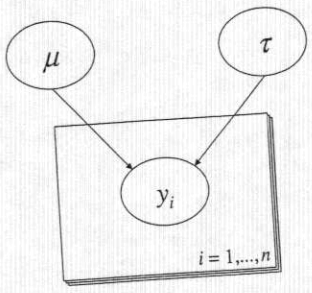

FIGURE 4.5
Directed acyclic graph depicting the assumption $y_i \sim \text{Normal}(\mu, \tau^{-1})$, $i = 1, \ldots, n$. Note the use of rectangular "plates" to denote repetition, i.e., the "loop" over i.

4.2.3 Derivation of full conditional distributions in BUGS

Equation (4.3) demonstrates that only *local* knowledge of the graph is needed to derive the full conditional distribution of any node (or group of nodes). Indeed, in order to derive all of the full conditionals required for Gibbs sampling on any DAG, we simply need to know how each node is related to its parents and which nodes are its children. BUGS stores this information in the form of an object-oriented version of the specified graph, with "objects" representing nodes and "pointer variables" linking nodes together, in particular linking each node to its children. The software uses an expert system to visit each node in turn and classify the form of the full conditional by considering the "compatibility" of the prior component (the distribution of the node itself, conditional on its parents) with each term in the likelihood component (the distribution of each child conditional on its parents). If the full conditional is available in closed form then a specialized, distribution-specific algorithm will be used to derive and sample from that closed form. If a closed form is not available, however, the expert system chooses a more general algorithm from a range of suitable methods, based on any important features of the full conditional that might have been gleaned from the aforementioned compatibility considerations, e.g., whether the density is "log-concave," or has infinite support, say. Some of the more commonly used alternative sampling methods are discussed briefly in the following subsection; the reader is referred to Lunn et al. (2000) and Lunn et al. (2009b) for further details regarding the internal workings of BUGS.

4.2.4 Other MCMC methods

As mentioned above, if direct sampling from the full conditional is not possible, then the WinBUGS software implements a number of alternative, more general algorithms, including "slice" sampling (Neal, 2003), Metropolis sam-

pling (Metropolis et al., 1953; Hastings, 1970), and various types of rejection method, such as "adaptive rejection sampling" (Gilks, 1992; Gilks and Wild, 1992) — see Lunn et al. (2000) or the WinBUGS manual for more details. Further algorithms are available in the OpenBUGS implementation of BUGS, for example, delayed rejection (Green and Mira, 2001) and hybrid/Hamiltonian methods (Duane et al., 1987; Neal, 2010) — please see the OpenBUGS documentation. Note that such methods are used only as a means of updating full conditionals within a Gibbs sampling scheme; use of the Metropolis algorithm, for example, thus leads to a type of sampling known as Metropolis-within-Gibbs.

Metropolis algorithms We do not wish to go into the details of various sampling methods in this text, but we feel it is instructive to provide an overview of Metropolis-based algorithms since they lead to a characteristic form of output that could be perceived as erroneous by the inexperienced user. Metropolis algorithms work by first sampling values from a *proposal* distribution, which approximates the relevant full conditional (or, more generally, *target* distribution) but is easy to sample from. An *acceptance probability* α is then calculated for the proposed value, and a Uniform$(0, 1)$ random number u is drawn to convert that probability into an accept/reject decision (reject if $u > \alpha$, accept otherwise). If the proposed value is to be accepted then the Markov chain moves to that value, otherwise it remains at its current value. This leads to a Markov chain that stays in the same place for a number of iterations, on a regular basis, particularly if the proposal distribution does not approximate the target distribution well (giving typically low values for α). If we plot a continuous line joining successive samples together, then the resulting *trace* plot — see §4.4.1 — can have the appearance of a cityscape (a metropolis perhaps), with the roofs of buildings corresponding to multiple successive rejections (see Figure 4.6, for example). Despite the chain containing multiple repeated values, it may be used, for inference, in exactly the same way as if all values had been sampled directly, as long as sufficient time is allowed for the chain to fully explore the target distribution.

In principle it is possible to use the Metropolis algorithm to jointly update the entire posterior distribution in one go, rather than using the alternating conditional updating scheme of the Gibbs sampler. However, this often requires problem-specific fine-tuning to obtain a multivariate proposal distribution that is a good approximation to the joint posterior of interest; hence, such an approach is not used in any implementation of BUGS, although note that various types of "block updating," where a group of variables is updated together according to their joint full conditional, are possible. Even when used within a Gibbs sampling scheme to sample from univariate full conditionals, the Metropolis algorithm requires some tuning to obtain efficient simulations. This tuning process has been fully automated in WinBUGS and is known as the *adapting phase* (see also §12.4.3, and the WinBUGS manual). Samples

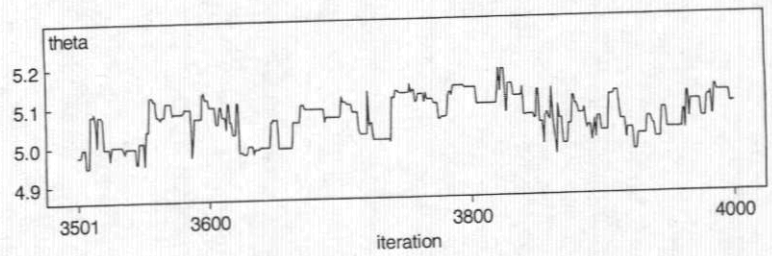

FIGURE 4.6
Trace plot, whereby successive samples for a single parameter are joined together by a continuous line (plotted against iteration number). The plot illustrates how output from Metropolis-based samplers contains multiple sequences of repeated values, which can give the plot the appearance of a cityscape.

generated during this phase, however, cannot be considered to arise from the posterior distribution, and so they must be discarded before performing any Monte Carlo integration. The "slice" sampler used in all BUGS implementations also fine-tunes itself during an initial adapting phase, and the same rule about discarding samples generated during this phase applies. These and other issues relating to the interpretation of output from different MCMC algorithms will be further discussed as they arise in context.

4.3 Initial values

Before any MCMC method can be started, we must first initialise the Markov chain, i.e., provide starting values for each unknown parameter in the statistical model. WinBUGS and OpenBUGS can often do this automatically, by sampling from the assumed ("prior") distribution of each parameter. However, such distributions may have very large variances, say, to reflect a lack of prior knowledge, and wildly inappropriate starting values may result. The disparity between such initial values and those values supported by the posterior often then causes the program to crash. Hence we have the facility to specify initial values manually instead. JAGS, on the other hand, when asked to generate initial values automatically, chooses a "central" value from the distribution, such as the mean or median, which avoids this problem. In WinBUGS initial values can be supplied as a list after the model description or in a separate file (see §12.4.2). In practice a mixture of these two strategies typically works well, with some values specified by the user and the remaining values generated automatically by the BUGS engine. All of the remaining examples in this

book have been run with a mixture of manually specified and automatically generated initial values. Except in cases where the manually specified values have been given explicitly, it will not generally be possible for the reader to reproduce *exactly* any results quoted using their own code, although results should agree up to Monte Carlo error — see §4.5.

Example 4.3.1. *Initial values*
Consider the multi-parameter Student-t problem described in Example 4.1.2. There are three unknown parameters, each of which must be initialised. The prior for μ, the mean of the Student-t distribution, is Normal(γ, ω^2). With $\gamma = 0$ and $\omega = 100$ as in Example 4.1.2, typical values from this prior lie in the range $(-20000, 20000)$. Hence we might prefer to simply set $\mu^{(0)} = 0$ rather than sample from the prior and obtain 14,603, say, which is unlikely to be a good starting point! The prior for r is Gamma$(0.001, 0.001)$, which WinBUGS cannot even sample from (due to numerical inaccuracies caused by the small size of the first parameter), and so a user-specified initial value is essential, e.g., $r^{(0)} = 1$. We may also provide our own starting value for the degrees-of-freedom parameter, e.g., $d^{(0)} = 8$, but any value from the discrete-uniform prior on $\{2, 3, \ldots, 30\}$ would represent a reasonable starting point and so we might prefer to let WinBUGS generate this automatically. The user-specified values for μ and r can be provided in the form of a list given after the model description:

```
list(mu = 0, r = 1)
```

In Example 2.1.2, we described the basic steps for running a model in WinBUGS. In this example, we would now need to load this list of initial values in Step 7, as follows, which was previously ignored.

- Highlight the word `list` by double-clicking on it, and click `load inits` in the `Specification Tool`.

Initial values for the remaining parameters (d in this case) are generated automatically using the *gen inits* command (see also §12.4.3); note that this must be used *after* the user-specified values have been loaded.

4.4 Convergence

As discussed in §4.2, if we simulate realisations from a Markov chain transition distribution, then under broad conditions, eventually the simulated values will be marginally distributed according to the Markov chain's unique stationary distribution. To fully exploit this fact we need to be able to detect when the

marginal behaviour of the chain is sufficiently close to stationarity, so that we can harvest all subsequent realisations as a dependent sample from the stationary distribution; note that the initial, non-stationary portion of the chain is referred to as the "burn-in."

It is important to note that convergence here is to a distribution, not to a fixed value. In practice we diagnose convergence retrospectively, by guessing for how long to run the simulation (we can always keep going if our initial guess proves insufficient) and then trying to determine if some latter portion of the chain can be considered stationary. We typically have to run the simulation well beyond the point of convergence, as detecting stationarity (or a lack thereof) requires a substantial sample size, as does accurate inference. The actual post-convergence sample size required depends very much on the *efficiency* of the Markov chain. We will discuss efficiency more formally in §4.5, but we note here that, basically, the number of samples required, both for detecting (non-) stationarity and for drawing accurate inferences, increases with an increasing level of serial dependence in the chain (also known as serial- or auto-correlation).

So why might we have a significant level of autocorrelation in the chain? First note that all Markov chains, by definition, exhibit at least some serial correlation. Second, imagine the two-parameter Gibbs sampler depicted in Figure 4.3 and suppose there is a high degree of correlation in the target distribution, such that the contours form narrow ellipses along the line $y = x$, say. The Gibbs sampler is capable of exploring the entire joint posterior but can only move perpendicular to the axes. Hence the shape of the distribution prevents the sampler from taking anything other than small steps, and so successive values of each variable are close together. All Metropolis-based sampling algorithms also induce a level of autocorrelation in the sample, due to their tendency to remain in the same place for a number of iterations, as discussed in §4.2.4.

4.4.1 Detecting convergence/stationarity by eye

Our task is made easier by the fact that it is often straightforward to detect (lack of) convergence *informally* by eye. Figure 4.7 illustrates this via several *history* (or *trace*) plots, whereby a continuous line joining successive realisations of a specific variable is plotted against Gibbs iteration number. The various plots comprising the figure are discussed below.

Our model may contain many parameters but it is not feasible to visually examine more than one at a time. This does not preclude convergence diagnosis, however, since we can simply assume convergence at the point where all parameters have reached stationarity — note that it is quite normal for different parameters to converge at different rates. Many realistically complex models, however, have hundreds or even thousands of parameters, and performing an individual assessment for each one may be impractical. In such cases it is important to at least assess all parameters (and functions thereof)

of interest. It is also prudent to examine at least a random selection of the remaining parameters.

As demonstrated by Figure 4.7(a), a Markov chain that has reached stationarity should look like a random scatter about a stable mean value. In addition, if plotted from the point of convergence onwards, a stationary Markov chain that contains sufficient information for reliable inferences, informally speaking, has the appearance of a "fat hairy caterpillar" (see Figure 4.7(b)). However, if the Markov chain has a "snake-like" appearance instead (Figure 4.7(c)), this does not necessarily mean that it hasn't converged, since there may simply be a high degree of autocorrelation in the sample. Hence the time scale over which the chain is plotted may be too short to clearly demonstrate stationarity, meaning that more samples are required. If we were to continue sampling indefinitely, repeatedly redrawing the entire history plot within the same area, it would eventually take on the required form, i.e., that of a fat caterpillar (see Figure 4.7(d)), since any "periodicity" would be "squeezed" to the point where it looked like random scatter — note that this also applies to regions of any Metropolis output where rejections have occurred. Markov chains generated using the Gibbs sampler often converge surprisingly quickly, within the first few iterations, say. However, it is important to note that the rate of convergence can be very slow, so slow in fact that a Markov chain may appear stationary even when it is not (the apparently "stable" mean may be drifting very slowly).

Of course, there is no reason why we cannot simulate two or more Markov chains and *pool* the resulting samples (after convergence) for making inferences. Such an approach provides us with a very effective means of checking convergence. If the multiple chains are given widely differing initial states, then we can be reasonably confident that stationarity has been reached when the chains come together and start behaving similarly. Figure 4.7(e) illustrates this by showing the same chain as in Figure 4.7(a) but with another chain, initialised at a different starting point, superimposed. This approach forms the basis of perhaps the most reliable *formal* convergence diagnostic, which is discussed in the following subsection. We note here, however, that running multiple chains can be inefficient, particularly if the rate of convergence is slow, since we effectively have to wait for convergence multiple times. This problem is exacerbated by the need to use widely differing initial states, which increase confidence in the diagnosis but often also substantially increase the number of iterations required to reach convergence.

4.4.2 Formal detection of convergence/stationarity

Numerous techniques for formally diagnosing convergence can be found in the literature. Many of these have been implemented within the CODA (Best et al., 1995; Plummer et al., 2006) and BOA (Smith, 2000) software packages for R and S-Plus, which are designed to take BUGS output as input. No method should be used blindly, however, as none can provide conclusive

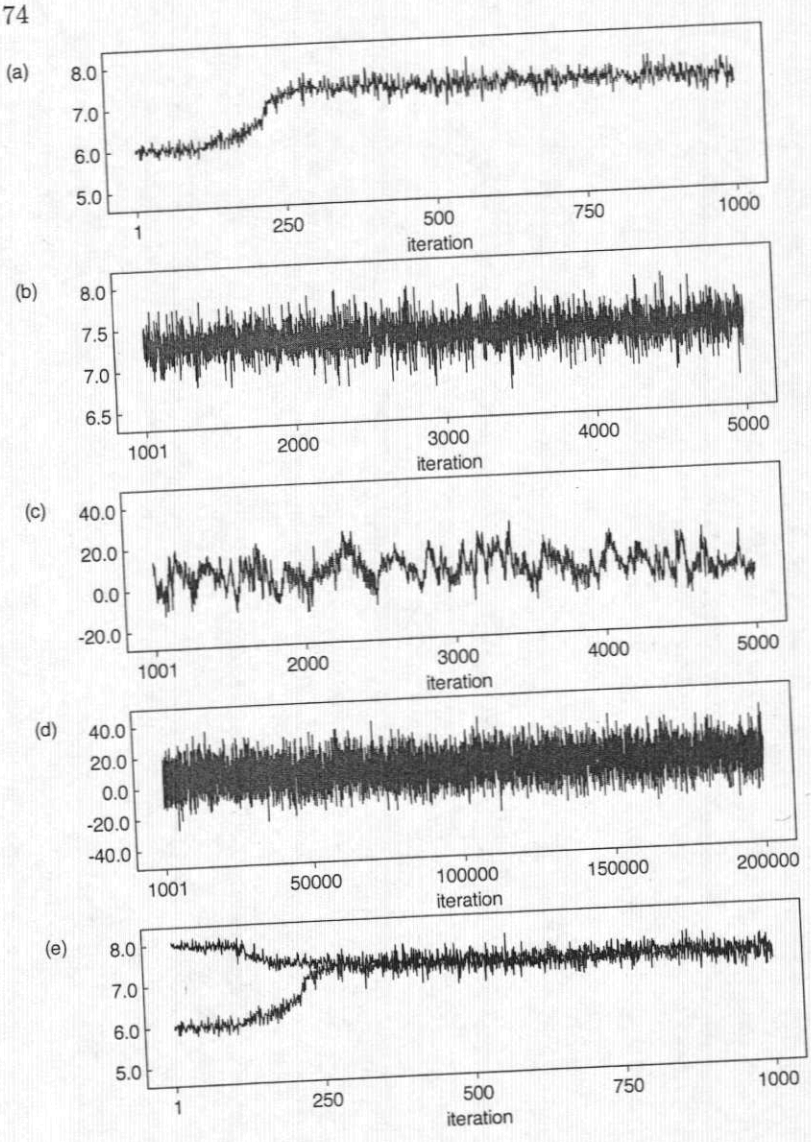

FIGURE 4.7
History/trace plots depicting five different scenarios. A continuous line joining successive realisations of a given parameter is plotted against Gibbs iteration number: (a) After convergence to its stationary distribution, a Markov chain typically looks like a random scatter about some stable mean value (iteration ~250 onwards); (b) A converged chain that contains sufficient information for accurate inferences looks like a "fat hairy caterpillar;" (c) A "snake"-like chain may have converged but contains too much serial-/auto-correlation for accurate inferences to be drawn — more samples required; (d) The same chain as in (c) but extended to iteration 200,000 and plotted over the same width — the chain now takes on the required form; (e) The same chain as in (a) but with another chain, initialised at a different starting point, superimposed.

proof of convergence and all are fallible. Indeed they are invariably founded on a null hypothesis of convergence and as such are designed only to detect *non*-convergence. Different methods examine/highlight different stochastic features of the chain(s), and so it is prudent to always consider a range of methods. For any given method, it is usually possible to construct an example whereby that method will fail to detect non-convergence but others may succeed. The reader is referred to Cowles and Carlin (1996) and Mengersen et al. (1999) for detailed reviews of the various methods available. One method that seems to stand out as being particularly effective, in our experience, is that originally proposed by Gelman and Rubin (1992) and subsequently modified in Brooks and Gelman (1998). Here multiple chains starting at "overdispersed" initial values* are simulated and convergence is assessed by comparing within- and between-chain variability. This is the only convergence diagnostic currently implemented in WinBUGS (and OpenBUGS), but we reiterate the wide range of methods available in BOA and CODA.

Again, we examine only one variable at a time, although multivariate extensions of the approach do exist (Brooks and Gelman, 1998). Suppose we simulate M chains, each of length $2T$, with a view to assessing the degree of stationarity in the final T iterations. We take as a measure of posterior variability the width of the $100(1-\alpha)\%$ credible interval for the parameter of interest, e.g., $\alpha = 0.2$. From the final T iterations we calculate the empirical $100(1-\alpha)\%$ credible interval for each chain. We then calculate the average width of these intervals across the M chains and denote this by W. Finally, we compute the width B of the empirical $100(1-\alpha)\%$ credible interval based on all MT samples pooled together. The ratio $\hat{R} = B/W$ of pooled to average interval widths should be > 1 if the starting values are suitably overdispersed; it will also tend to 1 as convergence is approached, and so we can assume convergence for practical purposes if $\hat{R} < 1.05$, say.

Rather than calculating a single value of $\hat{R}$, we can examine the behaviour of $\hat{R}$ over iteration time by performing the above procedure repeatedly for an increasingly large fraction of the total iteration range, ending with all of the final T iterations contributing to the calculation as described above. Brooks and Gelman (1998) propose splitting the total iteration range $(1, \ldots, 2T)$ into Q batches of length a and calculating $\hat{R}(q)$, $B(q)$, and $W(q)$ based on the latter halves of iterations $1, \ldots, qa$, for $q = 1, \ldots, Q$. A plot of $\hat{R}(q)$, $B(q)$, and $W(q)$ against some appropriate function of q then allows us not only to assess the rate of convergence of $\hat{R}$ to 1, but also to check that both B and W have stabilised, which is crucial for a reliable diagnosis. WinBUGS chooses $a = \max(100, \lfloor 2T/100 \rfloor)$† and plots against the starting iteration of each range, $\lfloor qa/2 \rfloor + 1$, so that the plot directly indicates the vicinity of the

*In this context "overdispersed" means that the initial values are more variable than they would have been if they had somehow been drawn from the target posterior.

†$\lfloor x \rfloor$ denotes the largest integer not greater than x.

point of convergence.

Example 4.4.1. *Brooks–Gelman–Rubin diagnostic*
Consider the two converging Markov chains shown in Figure 4.7(e). Clearly convergence occurs at around 250 iterations. Figure 4.8 shows the corresponding Brooks–Gelman–Rubin (BGR) diagnostics for the iteration ranges 51–100, 101–200, 151–300, 201–400, ..., 501–1000 ($2T = 1000$, $a = 100$, $Q = 2T/a = 10$). These are plotted against the starting iteration of each range ($\lfloor qa/2 \rfloor + 1 = 51$, 101, 151, 201, ..., 501) so that the approximate point of convergence can be read directly off the figure. The $\hat{R}$ line suggests convergence at around 200 iterations. However, note that the B and W lines do not stabilise until slightly later.

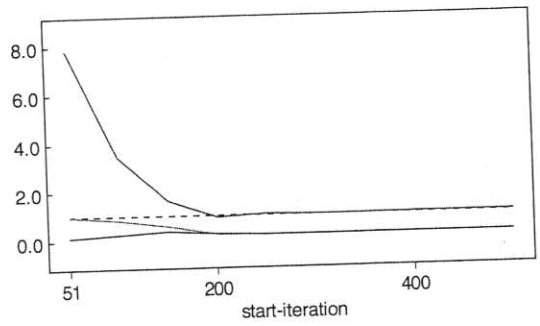

FIGURE 4.8
$\hat{R}(q)$, $B(q)$ and $W(q)$ for the two Markov chains shown in Figure 4.7(e), plotted against the starting iteration, $\lfloor qa/2 \rfloor + 1$, of each range for which the diagnostics are calculated. The upper line represents $\hat{R}$, which should converge to the value 1 (indicated by the horizontal dashed line). The upper and lower of the other two lines represent B and W, respectively. For plotting purposes, so that they can be clearly seen on the same scale as $\hat{R}$, these are normalised such that the maximum estimated interval width is equal to 1.

This strategy works because both the length of the chains used in the calculation and the start iteration are always increasing. Hence we will always eventually (with an increasing sample size) discard any "burn-in" iterations and also include a sufficient number of stationary samples to conclude convergence.

4.5 Efficiency and accuracy

For a given sample size, the accuracy of our inferences is dependent on the efficiency of our posterior sample, which decreases with an increasing level of autocorrelation. One way of increasing efficiency is to reparameterise the model so that the posterior correlation between parameters is reduced (see, for example, §6.1). However, identifying such a parameterisation is not always straightforward. Another way to improve efficiency is to perform a process known as *thinning* whereby only every νth value from the Gibbs sampler is actually retained for inference (the rest are still generated but are subsequently discarded). However, this only represents an efficiency gain in terms of storing and post-processing the sample: for the same computational cost of simulation, the full sample will always contain more information and hence lead to better accuracy.

4.5.1 Monte Carlo standard error of the posterior mean

The easiest way in which to improve accuracy is to simply increase the posterior sample size, but what sample size should we choose to achieve a specific level of accuracy? Indeed, how should we define accuracy? Suppose we are interested in estimating the posterior expectation of some (scalar-valued) function $g(\theta_i)$. Further suppose we have T posterior samples for θ_i and denote our Monte Carlo estimate, based on those T samples, by

$$\bar{g}_T = \frac{1}{T}\sum_{t=1}^{T} g(\theta_i^{(t)}).$$

As discussed in §1.4, if the samples $\theta_i^{(1)}, \ldots, \theta_i^{(T)}$ were independent, then the *Central Limit Theorem* would provide us with a (Monte Carlo) standard error for $\bar{g}_T$ of $\sqrt{Var[g(\theta_i)|y]/T}$, which of course could be estimated by $s/\sqrt{T}$ where s^2 is the sample variance:

$$s^2 = \frac{1}{T-1}\sum_{t=1}^{T}\left\{g(\theta_i^{(t)}) - \bar{g}_T\right\}^2.$$

Central limit theorems also exist for *dependent* samples (see Jones (2004), for example). In particular, $\bar{g}_T$ tends, in distribution, to Normal$(E[g(\theta_i)|y], \rho/T)$ as $T \to \infty$, for some positive constant ρ. We cannot use the sample variance to estimate ρ as $\rho \neq Var[g(\theta_i)|y]$ when the sample is dependent. Instead we split the sample into Q batches of length a and assume that a is sufficiently large that the central limit theorem approximately holds for each batch:

$$\bar{g}_{a,q} = \frac{1}{a}\sum_{t=(q-1)a+1}^{qa} g(\theta_i^{(t)}) \sim_{\text{approx}} \text{Normal}\left(E[g(\theta_i)|y], \frac{\rho}{a}\right), \quad q = 1, \ldots, Q.$$

Thus the batch means, $\bar{g}_{a,q}$, $q = 1, \ldots, Q$, form a sample from some distribution with variance ρ/a. Hence

$$\rho \approx \hat{\rho} = \frac{a}{Q-1} \sum_{q=1}^{Q} (\bar{g}_{a,q} - \bar{g}_T)^2 ,$$

and so the standard error of $\bar{g}_T$ can now be approximated, by $\sqrt{\hat{\rho}/T}$. This "Monte Carlo standard error" (MCSE) tells us how accurately the mean of our Monte Carlo samples for g estimates the *true* posterior expectation of g. It should not be confused with the posterior standard deviation, which reflects the inherent uncertainty about g given the model and the data.

The above is known as the *batch means* method of calculating the MCSE. There do exist other approaches, in particular methods adapted from time series analysis (e.g., Geweke (1992)), but BUGS implements batch means due to its simplicity. The value chosen for a is given by $\lfloor \sqrt{T} \rfloor$ (Chien, 1988). In cases where multiple Markov chains have been run, we split each chain in the way described above and the number of batches becomes MQ, where M is the number of chains.

4.5.2 Accuracy of the whole posterior

If we were only interested in accurate inference about the posterior mean of $g(\theta)$, we could simply run the chain for long enough that $\bar{g}_T \pm \delta \times$MCSE ($\delta = 2$ typically) agree to the desired number of significant figures. However, we would usually want to characterise the whole posterior distribution of $g(\theta)$ and present a credible interval or posterior probability. Raftery and Lewis (1992) describe a general method for determining whether a posterior tail probability is estimated to within a particular degree of accuracy with a specified probability. This is implemented in the CODA package for R and S-Plus. For example, about 4000 *independent* samples after convergence are sufficient to estimate the cumulative distribution function of the 2.5% quantile of a well-behaved posterior within ± 0.005 with probability 0.95. Reported 95% credible intervals would then have actual posterior probability between 0.94 and 0.96. More iterations would be necessary if the chains are autocorrelated.

Alternatively, if we compare the MCSE to the posterior standard deviation, this will tell us whether the inaccuracy about estimating the *posterior mean* of $g(\theta)$ is large in the context of the *overall* uncertainty about $g(\theta)$. To elucidate this, with *independent* samples the MCSE would be asymptotically $s/\sqrt{T}$, from the central limit theorem. This suggests that comparing the MCSE to the estimated posterior standard deviation s gives us an estimate of the *effective sample size* $T^* = (s/\text{MCSE})^2$ of an autocorrelated chain, which represents the amount of information about the posterior distribution it contains. Therefore, if MCSE $< ks$, then the effective sample size is $> 1/k^2$. Conventionally (see the WinBUGS manual), chains are run until the MCSE is less than $k = 5\%$ of the posterior standard deviation, giving $T^* = 400$. This is sufficient for

many practical purposes — however, the $T^* = 4000$ suggested by Raftery and Lewis's diagnostic for estimating tail probabilities requires around $k = 0.015$, or 1.5%.

A cruder but pragmatic strategy is to run the chains for increasingly more iterations, repeatedly recalculating whatever posterior summary statistics are of interest, until they do not appear to change within the desired accuracy. Three or four significant figures are usually enough in our experience — excessive precision can impede clarity when presenting results, and given the neglected uncertainties in most real statistical analyses (such as selection effects or measurement error) more precision may be spurious.

Example 4.5.1. *Monte Carlo standard error*

Consider again the two Markov chains shown in Figure 4.7(e) and suppose we wish to accurately estimate the mean of the underlying posterior distribution. Shown below are summary statistics, including MCSE (labelled `MC error`), calculated for various iteration ranges after extending the simulation to 8250 iterations. Note that the "burn-in" values from iterations 1–250 have been discarded and that samples from the two chains have been pooled together.

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
theta	7.333	0.1591	0.007993	7.008	7.339	7.637	251	500
theta	7.314	0.1597	0.006135	6.992	7.321	7.627	251	1000
theta	7.313	0.1604	0.004867	6.985	7.319	7.623	251	2000
theta	7.31	0.1627	0.003654	6.986	7.313	7.629	251	4000
theta	7.312	0.1629	0.002417	6.984	7.313	7.629	251	8000
theta	7.316	0.1625	0.001625	6.993	7.315	7.641	251	16000

The MCSE (fourth column) is less than 5% of the posterior standard deviation (third column) after collecting as few as 1000 samples. Note, however, that if we require MCSE $< 0.01s$, i.e., a standard error that is less than 1% of the posterior standard deviation, then we must collect at least 16,000 samples in this case. This is due to the fact that the estimated MCSE only decreases by a factor of $\sim\sqrt{2}$ as we double the sample size (since MCSE $\approx \sqrt{\hat{\rho}/T}$).

4.6 Beyond MCMC

There are a number of modelling scenarios for which standard MCMC, and Gibbs sampling in particular, are not well suited. One such setting is the analysis of time series data, due to the potentially long chains of serial dependence in the data, although the fact that such models can at least be specified in the BUGS language (and analysed, albeit somewhat inefficiently) is testament to the power of the language — see §11.2. Over the last decade or

so there have been many proposals for alternative formulations of the update mechanism, to develop faster and more efficient techniques for improving mixing and convergence rates. Some techniques use approximations to the true likelihood; others, such as Lagrangian–Hamiltonian updates (Girolami and Calderhead, 2011) generate more efficient proposal distributions. As alluded to in §4.2.4 above, implementation of such methods in OpenBUGS has been, and continues to be, explored. The "Stan" software, under development at `http://code.google.com/p/stan` at the time of writing, implements generic graphical models using Hamiltonian Monte Carlo sampling, and we eagerly await its official release.

Beyond MCMC, there have been advances in *Sequential Monte Carlo* (*SMC*; Doucet et al. (2001); Del Moral et al. (2006)), whereby sets of particles are propagated through sequential *importance samplers* (e.g., Ripley (1987)), rather than constructing a Markov chain under MCMC. There are issues of particle degeneracy, however, whereby the subset of particles (samples) that are consistent with the observed data becomes too small. Hybrid schemes combining MCMC within SMC have been proposed to gain the benefits of both approaches (Andrieu et al., 2010).

Recently, there has been a growing interest in so-called likelihood-free approaches, for example, *approximate Bayesian computation* (*ABC*; Pritchard et al. (1999); Beaumont et al. (2002); Beaumont (2010)), where simulation of the process or model is computationally cheap comparative to evaluating the likelihood in an MCMC approach. There are, however, many unresolved issues, such as model selection (Robert et al., 2011), and the theoretical justification is weaker than for MCMC. Approximation methods that allow fast and precise inference for specific classes of problem have been developed, such as for latent Gaussian models (Rue et al., 2009), but these are not generally applicable outside this class of models.

Variational Bayesian methods are based on approximating an intractable posterior distribution $p(\theta|y)$ by a distribution $q(\theta)$ from a family with a known analytical form and have been applied to machine learning. See, for example, Bishop (2006) or Mackay (2003) for a detailed introduction, and the Infer.NET software (Minka et al., 2011).