
Part III: Advanced Computation

The remainder of this book delves into more sophisticated models. Before we begin this enterprise, however, we detour to describe methods for computing posterior distributions in hierarchical models. Toward the end of Chapter 5, the algebra required for analytic derivation of posterior distributions became less and less attractive, and that was with a model based on the normal distribution! If we try to solve more complicated problems analytically, the algebra starts to overwhelm the statistical science almost entirely, making the full Bayesian analysis of realistic probability models too cumbersome for most practical applications. Fortunately, a battery of powerful methods has been developed over the past few decades for simulating from probability distributions. In the next four chapters, we survey some useful simulation methods that we apply in later chapters in the context of specific models. Some of the simpler simulation methods we present here have already been introduced in examples in earlier chapters.

Because the focus of this book is data analysis rather than computation, we move through the material of Part III briskly, with the intent that it be used as a reference when applying the models discussed in Parts IV and V. We have also attempted to place a variety of useful techniques in the context of a systematic general approach to Bayesian computation. Our general philosophy in computation, as in modeling, is pluralistic, starting with simple approximate methods and gradually working toward more precise computations.



CHAPTER 10

Overview of computation

This chapter provides a general perspective on computation for Bayesian data analysis. Our computational strategy for complex problems begins with crude initial estimates of model parameters and usually continues with Markov chain simulation, sometimes using mode-based approximations of the posterior distribution as an intermediate step. We begin with some notation and terminology that are relevant to all the chapters of this part of the book. Chapters 11, 12, and 13 cover Markov chain simulation, mode-based approximations, and more advanced methods in Bayesian computation.

Joint, conditional, and marginal target distributions

We refer to the (multivariate) distribution to be simulated as the *target distribution* and denote it as $p(\theta|y)$. At various points we consider partitioning a high-dimensional parameter vector as $\theta = (\gamma, \phi)$, where typically γ will include most of the components of θ . (This is not the same as the θ, ϕ notation in Chapter 7.) For example, in a hierarchical model, γ could be the parameters that are exchangeable given hyperparameters ϕ . Later, we will see that the same computational techniques can be used to handle unobserved indicator variables (Chapter 18) and missing data (Chapter 21). Formally, in those cases θ includes all unknown quantities in the joint distribution including unobserved or missing values. For example, in a problem with missing data, γ could be the missing values and ϕ the model parameters. Such factorizations serve as useful computational devices for partitioning high-dimensional distributions into manageable parts. When using the (γ, ϕ) notation, we work with the factorization, $p(\theta|y) = p(\gamma, \phi|y) = p(\gamma|\phi, y)p(\phi|y)$, and compute the conditional and marginal posterior densities in turn.

Normalized and unnormalized densities

Unless otherwise noted (in Section 13.4), we assume that the target density $p(\theta|y)$ can be easily computed for any value of θ , up to a proportionality constant involving only the data y ; that is, we assume there is some easily computable function $q(\theta|y)$, an *unnormalized density*, for which $q(\theta|y)/p(\theta|y)$ is a constant that depends only on y . For example, in the usual use of Bayes' theorem, we work with the product $p(\theta)p(y|\theta)$, which is proportional to the posterior density.

10.1 Crude estimation by ignoring some information

Before developing more elaborate approximations or complicated methods for sampling from the target distribution, it is almost always useful to obtain a rough estimate of the location of the target distribution—that is, a point estimate of the parameters in the model—using some simple, noniterative technique. The method for creating this first estimate will vary from problem to problem but typically will involve discarding parts of the model and data to create a simple problem for which convenient parameter estimates can be found.

In a hierarchical model, one can sometimes roughly estimate the main parameters γ by first estimating the hyperparameters ϕ crudely and then treating the resulting distribution of γ given ϕ as a fixed prior distribution for γ . We applied this approach to the rat tumor example in Section 5.1, where crude estimates of the hyperparameters (α, β) were used to obtain initial estimates of the other parameters, θ_j .

For another example, in the educational testing analysis in Section 5.5, the school effects θ_j can be crudely estimated by the data y_j from the individual experiments, and the between-school standard deviation τ can then be estimated very crudely by the standard deviation of the eight y_j -values or, to be slightly more sophisticated, the estimate (5.22), restricted to be nonnegative.

When some data are missing, a good way to get started is by simplistically imputing the missing values based on available data. Or, even simpler, it may be convenient temporarily to ignore data from all experimental units that have missing observations. (Ultimately, inferences for the missing data should be included as part of the model; see Chapter 21.)

In addition to creating a starting point for a more exact analysis, crude inferences are useful for comparison with later results—if the rough estimate differs greatly from the results of the full analysis, the latter may very well have errors in programming or modeling. Crude estimates are often convenient and reliable because they can be computed using available computer programs.

10.2 Use of posterior simulations in Bayesian data analysis

Bayesian inference are usually most conveniently summarized by random draws from the posterior distribution of the model parameters. Percentiles of the posterior distribution of univariate estimands can be reported to convey the shape of the distribution. For example, reporting the 2.5%, 25%, 50%, 75%, and 97.5% points of the sampled distribution of an estimand provides a 50% and a 95% posterior interval and also conveys skewness in its marginal posterior density. Scatterplots of simulations, contour plots of density functions, or more sophisticated graphical techniques can also be used to examine the posterior distribution in two or three dimensions. Quantities of interest can be defined in terms of the parameters (for example, LD50 in the bioassay example in Section 3.7) or of parameters and data.

We also use posterior simulations to make inferences about predictive quan-

tities. Given each simulation θ^l , we can simulate a posterior draw for any predictive quantity of interest: $\tilde{y}^l \sim p(\tilde{y}|\theta^l)$ or, for a regression model, $\tilde{y}^l \sim p(\tilde{y}|\tilde{X}, \theta^l)$. Posterior inferences and probability calculations can then be performed for each predictive quantity using the L simulations (for example, the predicted probability of Bill Clinton winning each state as displayed in Figure 6.1 on page 159).

Finally, given each simulation θ^l , we can simulate a replicated dataset $y^{\text{rep}l}$. As described in Chapter 6, we can then check the model by comparing the data to these posterior predictive replications.

How many simulation draws are needed?

Our goal in Bayesian computation is to obtain a set of independent draws θ^l , $l = 1, \dots, L$, from the posterior distribution, with enough draws L so that quantities of interest can be estimated with reasonable accuracy. For most examples, $L = 100$ independent samples are enough for reasonable posterior summaries. We can see this by considering a scalar parameter θ with an approximately normal posterior distribution (see Chapter 4) with mean μ_θ and standard deviation σ_θ . We assume these cannot be calculated analytically and instead are estimated from the mean $\bar{\theta}$ and standard deviation s_θ of the L simulation draws. The posterior mean is then estimated to an accuracy of approximately $s_\theta/\sqrt{L}$. The total standard deviation of the computational parameter estimate (including *Monte Carlo error*, the uncertainty contributed by having only a finite number of simulation draws) is then $s_\theta\sqrt{1 + 1/L}$. For $L = 100$, the factor $\sqrt{1 + 1/L} = 1.005$, implying that Monte Carlo error adds almost nothing to the uncertainty coming from actual posterior variance. However, it can be convenient to have more than 100 simulations just so that the numerical summaries are more stable, even if this stability typically confers no important practical advantage.

For some posterior inferences, more simulation draws are needed to obtain desired precisions. For example, posterior probabilities are estimated to a standard deviation of $\sqrt{p(1-p)/L}$, so that $L = 100$ simulations allow estimation of a probability near 0.5 to an accuracy of 5%. $L = 2500$ simulations are needed to estimate to an accuracy of 1%. Even more simulation draws are needed to compute the posterior probability of rare events, unless analytic methods are used to assist the computations.

Example. Educational testing experiments

We illustrate with the hierarchical model fitted to the SAT coaching experiments as described in Section 5.5. First consider inference for a particular parameter, for example θ_1 , the estimated effect of coaching in School A. Table 5.3 shows that from 200 simulation draws, our posterior median estimate was 10, with a 50% interval of [7, 16] and a 95% interval of [-2, 31]. Repeating the computation, another 200 draws gave a posterior median of 9, with a 50% interval of [6, 14] and a 95% interval of [-4, 32]. These intervals differ slightly but convey the same general information about θ_1 . From $L = 10,000$ simulation draws, the median is

10, the 50% interval is [6, 15], and the 95% interval is [−2, 31]. In practice, these are no different from either of the summaries obtained from 200 draws.

We now consider some posterior probability statements. Our original 200 simulations gave us an estimate of 0.73 for the posterior probability $\Pr(\theta_1 > \theta_3|y)$, the probability that the effect is larger in school A than in school C (see the end of Section 5.5). This probability is estimated to an accuracy of $\sqrt{0.73(1 - 0.73)/200} = 0.03$, which is good enough in this example.

How about a rarer event, such as the probability that the effect in School A is greater than 50 points? None of our 200 simulations θ_1^l exceeds 50, so the simple estimate of the probability is that it is zero (or less than $1/200$). When we simulate $L = 10,000$ draws, we find 3 of the draws to have $\theta_1 > 50$, which yields a crude estimated probability of 0.0003.

An alternative way to compute this probability is semi-analytically. Given μ and τ , the effect in school A has a normal posterior distribution, $p(\theta_1|\mu, \tau, y) = N(\hat{\theta}_1, V_1)$, where this mean and variance depend on y_1 , μ , and τ (see (5.17) on page 135). The conditional probability that θ_1 exceeds 50 is then $\Pr(\theta_1 > 50|\mu, \tau, y) = \Phi((\hat{\theta}_1 - 50)/\sqrt{V_1})$, and we can estimate the unconditional posterior probability $\Pr(\theta_1 > 50|y)$ as the average of these normal probabilities as computed for each simulation draw (μ^l, τ^l) . Using this approach, $L = 200$ draws are sufficient for a reasonably accurate estimate.

In general, fewer simulations are needed to estimate posterior medians of parameters, probabilities near 0.5, and low-dimensional summaries than extreme quantiles, posterior means, probabilities of rare events, and higher-dimensional summaries. In most of the examples in this book, we use a moderate number of simulation draws (typically 100 to 2000) as a way of emphasizing that applied inferences do not typically require a high level of simulation accuracy.

10.3 Practical issues

Computational tools

Section 1.10 lists some of the software that is useful for Bayesian data analysis, and Appendix C illustrates with an extended example of several different ways to compute a model using the statistical packages **Bugs** and **R**. More generally, the following computational operations are useful:

- Simulation from standard distributions (see Appendix A) and the ability to sample from arbitrary discrete distributions, including discrete approximations to continuous distributions, as was done for the hyperparameter τ in the hierarchical model in Section 5.4 (see Figure 5.5 on page 141).
- Vector and matrix operations, including Cholesky factorization (matrix square roots), and linear transformations of vectors using matrix multiplication. These operations are useful for regression models, for conditioning in multivariate distributions (see (A.1) and (A.2) on page 579), and for linear transformations to make Gibbs samplers more efficient (see Section 15.4).

- Even more generally, matrices are crucial for organizing the simulations of parameters, missing data, predictions, and replicated data, as indicated in Table 1.1 on page 26.
- Linear regression computations—as discussed in Chapter 14, we like to think of the regression estimate $\hat{\beta}$ and covariance matrix V_{β} as basic operations on data X and y , without worrying about the intermediate matrix computations. Classical regression is also an important component of Bayesian computation for hierarchical regression and generalized linear models.
- All-purpose optimization algorithms (such as the `optim` function in R) can be useful in obtaining crude estimates.
- Other numerical operations that are often performed are transformations (most commonly logarithm, logit, and their inverses), and numerical derivatives (see (12.1) on page 313).
- Finally, graphical tools are needed to display data, posterior inferences, and simulated replicated data. These tools begin with histograms and scatterplots, but software should be flexible enough to display multiple plots on a single page (as in Figure 6.2 on page 160) and to allow the creation of specialized plots to capture complex data structures (as in Figure 6.6 on page 166).

Debugging using fake data

Our usual approach for building confidence in our posterior inferences is to fit different versions of the desired model, noticing when the inferences change unexpectedly. Section 10.1 discusses crude inferences from simplified models that typically ignore some structure in the data.

Within the computation of any particular model, we check convergence by running parallel simulations from different starting points, checking that they mix and converge to the same estimated posterior distribution (see Section 11.6). This can be seen as a form of debugging of the individual simulated sequences.

When a model is particularly complicated, or its inferences are unexpected enough to be not necessarily believable, one can perform more elaborate debugging using fake data. The basic approach is:

1. Pick a reasonable value for the ‘true’ parameter vector θ . Strictly speaking, this value should be a random draw from the prior distribution, but if the prior distribution is noninformative, then any reasonable value of θ should work.
2. If the model is hierarchical (as it generally will be), then perform the above step by picking reasonable values for the hyperparameters, then drawing the other parameters from the prior distribution conditional on the specified hyperparameters.
3. Simulate a large fake dataset y^{fake} from the data distribution $p(y|\theta)$.

4. Perform posterior inference about θ from $p(\theta|y^{\text{fake}})$.
5. Compare the posterior inferences to the ‘true’ θ from step 1 or 2. For instance, for any element of θ , there should be a 50% probability that its 50% posterior interval contains the truth.

Formally, this procedure requires that the model has proper prior distributions and that the frequency evaluations be averaged over many values of the ‘true’ θ , drawn independently from the prior distribution in step 1 above. In practice, however, the debugging procedure can be useful with just a single reasonable choice of θ in the first step. If the model does not produce reasonable inferences with θ set to a reasonable value, then there is probably something wrong, either in the computation or in the model itself.

Inference from a single fake dataset can be revealing for debugging purposes, if the true value of θ is far outside the computed posterior distribution. If the dimensionality of θ is large (as can easily happen with hierarchical models), we can go further and compute debugging checks such as the proportion of the 50% intervals that contain the true value.

To check that inferences are correct on average, one can create a ‘residual plot’ as follows. For each scalar parameter θ_j , define the predicted value as the average of the posterior simulations of θ_j , and the error as the true θ_j (as specified or simulated in step 1 or 2 above) minus the predicted value. If the model is computed correctly, the errors should have zero mean, and we can diagnose problems by plotting errors vs. predicted values, with one dot per parameter.

For models with only a few parameters, one can get the same effect by performing many fake-data simulations, resampling a new ‘true’ vector θ and a new fake dataset y^{fake} each time, and then checking that the errors have zero mean and the correct interval coverage, on average.

Model checking and convergence checking as debugging

Finally, the techniques for model checking and comparison described in Chapter 6, and the techniques for checking for poor convergence of iterative simulations, which we describe in Section 11.6, can also be interpreted as methods for debugging.

In practice, when a model grossly misfits the data, or when a histogram or scatterplot or other display of replicated data looks weird, it is often because of a computing error. These errors can be as simple as forgetting to recode discrete responses (for example, 1 = Yes, 0 = No, -9 = Don’t Know) or misspelling a regression predictor, or as subtle as a miscomputed probability ratio in a Metropolis updating step (see Section 11.4), but typically they show up as predictions that do not make sense or do not fit the data. Similarly, poor convergence of an iterative simulation algorithm can sometimes occur from programming errors or even conceptual errors in the model.

Tips and tricks

Among the practical techniques for making statistical computation more reliable, probably the most important involve programming. We try wherever possible to code procedures as subroutines or functions rather than sequences of commands in the main program. A modular programming style makes it easier to expand a model by adding additional parameters to a Gibbs-Metropolis updating scheme. We illustrate in Section C.4.

When displaying statistical inferences, we recommend routinely putting several graphs on a single page (for example, using the `mfrow` option in the `par` function in R) and labeling all graphs (in R, using the `xlab`, `ylab`, and `main` options in the `plot` function). It is much easier to explore patterns in data and inferences when viewing several graphs at once.

To avoid computational overflows and underflows, one should compute with the logarithms of posterior densities whenever possible. Exponentiation should be performed only when necessary and as late as possible; for example, in the Metropolis algorithm, the required ratio of two densities (11.1) should be computed as the exponential of the difference of the log-densities.

Set up computations to be able to use the L posterior simulation draws rather than a point estimate of the parameters. Thus, any scalar parameter becomes a vector of length L , a vector parameter of length J becomes an $L \times J$ matrix, and so on. Setting everything in a vector-matrix format is convenient for displaying posterior inferences and predictive simulations.

Almost no computer program works the first time. A good way to speed the debugging process is to start with a smaller dataset, perhaps randomly sampled from the original data. This speeds computations and also makes it easier to inspect the data and inferences for possible problems. In many cases, a sample of 10 or 20 data points is enough to take us quickly through the initial stage of debugging. A related piece of advice, when running iterative simulations, is to run for only a few iterations at first, possibly 10 or 100, to check that the algorithm seems to be on the right track. There is no point in waiting an hour for the computer to get highly precise computations for the wrong model.

When the posterior inferences from a fitted model do not make sense, it is sometimes not clear whether there is a bug in the program or a fundamental error in the model itself. At this point, a useful conceptual and computational strategy is to simplify—to remove parameters from the model, or to give them fixed values or highly informative prior distributions, or to separately analyze data from different sources (that is, to un-link a hierarchical model). These computations can be performed in steps, for example first removing a parameter from the model, then setting it equal to a null value (for example, zero) just to check that adding it into the program has no effect, then fixing it at a reasonable nonzero value, then assigning it a precise prior distribution, then allowing it to be estimated more fully from the data. Model building is a gradual process, and we often find ourselves going back and forth between

simpler and more complicated models, both for conceptual and computational reasons.

Finally, almost any statistical computation can be seen as part of a larger problem. As noted already in this chapter, for problems of realistic complexity we usually fit at least one preliminary model to obtain starting points for iterative simulation. To take things one step further, an existing model can be used as an approximation for a future expanded model. Another form of expansion is to add data, for example by taking an analysis that could apply individually to several datasets and linking them into a hierarchical model. This can be straightforward to program using the Gibbs sampler, alternating between updating the small model fit to each dataset and the population model that links the parameters. We illustrate with an example in Section 20.3.

10.4 Exercises

The exercises in Part III focus on computational details. Data analysis exercises using the methods described in this part of the book appear in the appropriate chapters in Parts IV and V.

1. Number of simulation draws: suppose you are interested in inference for the parameter θ_1 in a multivariate posterior distribution, $p(\theta|y)$. You draw 100 independent values θ from the posterior distribution of θ and find that the posterior density for θ_1 is approximately normal with mean of about 8 and standard deviation of about 4.
 - (a) Using the average of the 100 draws of θ_1 to estimate the posterior mean, $E(\theta_1|y)$, what is the approximate standard deviation due to simulation variability?
 - (b) About how many simulation draws would you need to reduce the simulation standard deviation of the posterior mean to 0.1 (thus justifying the presentation of results to one decimal place)?
 - (c) A more usual summary of the posterior distribution of θ_1 is a 95% central posterior interval. Based on the data from 100 draws, what are the approximate simulation standard deviations of the estimated 2.5% and 97.5% quantiles of the posterior distribution? (Recall that the posterior density is approximately normal.)
 - (d) About how many simulation draws would you need to reduce the simulation standard deviations of the 2.5% and 97.5% quantiles to 0.1?
 - (e) In the SAT coaching example of Section 5.5, we simulated 200 independent draws from the posterior distribution. What are the approximate simulation standard deviations of the 2.5% and 97.5% quantiles for school A in Table 5.3?
 - (f) Why was it not necessary, in practice, to simulate more than 200 draws for the SAT coaching example?

CHAPTER 11

Posterior simulation

The bulk of this chapter presents the most widely used Markov chain simulation methods—the Gibbs sampler and the Metropolis-Hastings algorithm—in the context of our general computing approach based on successive approximation. We sketch a proof of the convergence of Markov chain simulation algorithms and present a method for monitoring the convergence in practice. We illustrate these methods in Section 11.7 for a hierarchical normal model. For most of this chapter we consider simple and familiar (even trivial) examples in order to focus on the principles of iterative simulation methods as they are used for posterior simulation. Many examples of these methods appear in the recent statistical literature (see the bibliographic note at the end of this chapter) and also in Parts IV and V of this book. Appendix C shows the details of implementation in the computer languages **R** and **Bugs** for the educational testing example from Chapter 5.

11.1 Direct simulation

In simple nonhierarchical Bayesian models, it is often easy to draw from the posterior distribution directly, especially if conjugate prior distributions have been assumed. For more complicated problems, an often useful strategy is to factor the distribution analytically and simulate it in parts: for example, first obtain draws from the marginal posterior distribution of the hyperparameters, then simulate the other parameters conditional on the data and the simulated hyperparameters. It is sometimes possible to perform direct simulations and analytic integrations for parts of the larger problem, as was done in the examples of Chapter 5.

Frequently, draws from standard distributions or low-dimensional non-standard distributions are required, either as direct draws from the posterior distribution of the estimand in an easy problem, or as an intermediate step in a more complex problem. Appendix A is a relatively detailed source of advice, algorithms, and procedures specifically relating to a variety of commonly-used distributions. In this section, we describe methods of drawing a random sample of size 1, with the understanding that the methods can be repeated to draw larger samples. When obtaining more than one sample, it is often possible to reduce computation time by saving intermediate results such as the Cholesky factor for a fixed multivariate normal distribution.

Direct approximation by calculating at a grid of points

Often as a first approximation or for computational convenience, it is desirable to approximate the distribution of a continuous parameter as a discrete distribution on a grid of points. For the simplest discrete approximation, compute the target density, $p(\theta|y)$, at a set of evenly spaced values $\theta_1, \dots, \theta_N$, that cover a broad range of the parameter space for θ , then approximate the continuous $p(\theta|y)$ by the discrete density at $\theta_1, \dots, \theta_N$, with probabilities $p(\theta_i|y) / \sum_{j=1}^N p(\theta_j|y)$. Because the approximate density must be normalized anyway, this method will work just as well using an unnormalized density function, $q(\theta|y)$, in place of $p(\theta|y)$.

Once the grid of density values is computed, a random draw from $p(\theta|y)$ is obtained by drawing a random sample U from the uniform distribution on $[0, 1]$, then transforming by the inverse cdf method (see Section 1.9) to obtain a sample from the discrete approximation. When the points θ_i are spaced closely enough and miss nothing important beyond their boundaries, this method works well. The discrete approximation is more difficult to use in higher-dimensional multivariate problems, where computing at every point in a dense multidimensional grid becomes prohibitively expensive.

Rejection sampling

Many general techniques are available for simulating draws directly from the target density, $p(\theta|y)$; see the bibliographic note at the end of the chapter. Of the many techniques, we choose rejection sampling for a detailed description here because of its simplicity and generality and because it is often used as part of the more complex approaches described later in the chapter.

Suppose we want to obtain a single random draw from a density $p(\theta|y)$, or perhaps an unnormalized density $q(\theta|y)$ (with $p(\theta|y) = q(\theta|y) / \int q(\theta|y) d\theta$). In the following description we use p to represent the target distribution, but we could just as well work with the unnormalized form q instead. To perform *rejection sampling* we require a positive function $g(\theta)$ defined for all θ for which $p(\theta|y) > 0$ that has the following properties:

- We are able to draw random samples from the probability density proportional to g . It is *not* required that $g(\theta)$ integrate to 1, but $g(\theta)$ must have a finite integral.
- The *importance ratio* $p(\theta|y)/g(\theta)$ must have a known bound; that is, there must be some known constant M for which $p(\theta|y)/g(\theta) \leq M$ for all θ .

The rejection sampling algorithm proceeds in two steps:

1. Sample θ at random from the probability density proportional to $g(\theta)$.
2. With probability $p(\theta|y)/(Mg(\theta))$, *accept* θ as a draw from p . If the drawn θ is rejected, return to step 1.

Figure 11.1 illustrates rejection sampling. An accepted θ has the correct distribution, $p(\theta|y)$; that is, the distribution of drawn θ , conditional on it being

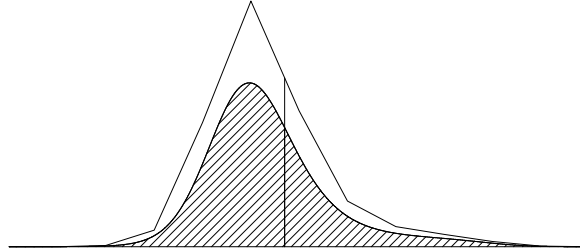


Figure 11.1 *Illustration of rejection sampling. The top curve is an approximation function, $Mg(\theta)$, and the bottom curve is the target density, $p(\theta|y)$. As required, $Mg(\theta) \geq p(\theta|y)$ for all θ . The vertical line indicates a single random draw θ from the density proportional to g . The probability that a sampled draw θ is accepted is the ratio of the height of the lower curve to the height of the higher curve at the value θ .*

accepted, is $p(\theta|y)$ (see Exercise 11.1). The boundedness condition is necessary so that the probability in step 2 is not greater than 1.

A good approximate density $g(\theta)$ for rejection sampling should be roughly proportional to $p(\theta|y)$ (considered as a function of θ). The ideal situation is $g \propto p$, in which case, with a suitable value of M , we can accept every draw with probability 1. When g is not nearly proportional to p , the bound M must be set so large that almost all samples obtained in step 1 will be rejected in step 2. A virtue of rejection sampling is that it is self-monitoring—if the method is not working efficiently, very few simulated draws will be accepted.

The function $g(\theta)$ is chosen to approximately match $p(\theta|y)$ and so in general will depend on y . We do not use the notation $g(\theta, y)$ or $g(\theta|y)$, however, because in practice we will be considering approximations to one posterior distribution at a time, and the functional dependence of g on y is not of interest.

Simulating from predictive distributions

Once simulations have been obtained from the posterior distribution, $p(\theta|y)$, it is typically easy to draw from the predictive distribution of unobserved or future data, $\tilde{y}$. For each draw of θ from the posterior distribution, just draw one value $\tilde{y}$ from the predictive distribution, $p(\tilde{y}|\theta)$. The set of simulated $\tilde{y}$'s from all the θ 's characterizes the posterior predictive distribution. Posterior predictive distributions are crucial to the model-checking approach described in Chapter 6.

11.2 Markov chain simulation

Markov chain simulation (also called *Markov chain Monte Carlo*, or MCMC) is a general method based on drawing values of θ from approximate distributions and then correcting those draws to better approximate the target posterior

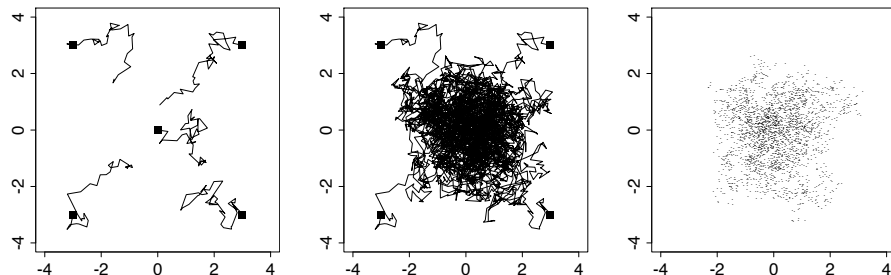


Figure 11.2 *Five independent sequences of a Markov chain simulation for the bivariate unit normal distribution, with overdispersed starting points indicated by solid squares. (a) After 50 iterations, the sequences are still far from convergence. (b) After 1000 iterations, the sequences are nearer to convergence. Figure (c) shows the iterates from the second halves of the sequences. The points in Figure (c) have been jittered so that steps in which the random walks stood still are not hidden. The simulation is a Metropolis algorithm described in the example on page 290.*

distribution, $p(\theta|y)$. The samples are drawn sequentially, with the distribution of the sampled draws depending on the last value drawn; hence, the draws form a Markov chain. (As defined in probability theory, a *Markov chain* is a sequence of random variables $\theta^1, \theta^2, \dots$, for which, for any t , the distribution of θ^t given all previous θ 's depends only on the most recent value, θ^{t-1} .) The key to the method's success, however, is not the Markov property but rather that the approximate distributions are improved at each step in the simulation, in the sense of converging to the target distribution. As we shall see in Section 11.4, the Markov property is helpful in proving this convergence.

Figure 11.2 illustrates a simple example of a Markov chain simulation—in this case, a Metropolis algorithm (see Section 11.4) in which θ is a vector with only two components, with a bivariate unit normal posterior distribution, $\theta \sim N(0, I)$. First consider Figure 11.2a, which portrays the early stages of the simulation. The space of the figure represents the range of possible values of the multivariate parameter, θ , and each of the five jagged lines represents the early path of a random walk starting near the center or the extremes of the target distribution and jumping through the distribution according to an appropriate sequence of random iterations. Figure 11.2b represents the mature stage of the same Markov chain simulation, in which the simulated random walks have each traced a path throughout the space of θ , with a common stationary distribution that is equal to the target distribution. From a simulation such as 11.2b, we can perform inferences about θ using points from the second halves of the Markov chains we have simulated, as displayed in Figure 11.2c.

In our applications of Markov chain simulation, several independent sequences of simulation draws are created; each sequence, θ^t , $t = 1, 2, 3, \dots$, is produced by starting at some point θ^0 and then, for each t , drawing θ^t from a *transition distribution*, $T_t(\theta^t|\theta^{t-1})$ that depends on the previous draw,

θ^{t-1} . As we shall see in the discussion of combining the Gibbs sampler and Metropolis sampling in Section 11.5, it is often convenient to allow the transition distribution to depend on the iteration number t ; hence the notation T_t . The transition probability distributions must be constructed so that the Markov chain converges to a unique stationary distribution that is the posterior distribution, $p(\theta|y)$.

Markov chain simulation is used when it is not possible (or not computationally efficient) to sample θ directly from $p(\theta|y)$; instead we sample *iteratively* in such a way that at each step of the process we expect to draw from a distribution that becomes closer and closer to $p(\theta|y)$. For a wide class of problems (including posterior distributions for many hierarchical models), this approach appears to be the easiest way to get reliable results, at least when used carefully. In addition, Markov chain and other iterative simulation methods have many applications outside Bayesian statistics, such as optimization, that we do not discuss here.

The key to Markov chain simulation is to create a Markov process whose stationary distribution is the specified $p(\theta|y)$ and run the simulation long enough that the distribution of the current draws is close enough to this stationary distribution. For any specific $p(\theta|y)$, or unnormalized density $q(\theta|y)$, a variety of Markov chains with the desired property can be constructed, as we demonstrate in Sections 11.3–11.5.

Once the simulation algorithm has been implemented and the simulations drawn, it is absolutely necessary to check the convergence of the simulated sequences; for example, the simulations of Figure 11.2a are far from convergence and are not close to the target distribution. We discuss how to check convergence in Section 11.6. If convergence is painfully slow, the algorithm should be altered, perhaps using the methods of Sections 11.8 and 11.9.

11.3 The Gibbs sampler

A particular Markov chain algorithm that has been found useful in many multidimensional problems is the *Gibbs sampler*, also called *alternating conditional sampling*, which is defined in terms of subvectors of θ . Suppose the parameter vector θ has been divided into d components or subvectors, $\theta = (\theta_1, \dots, \theta_d)$. Each iteration of the Gibbs sampler cycles through the subvectors of θ , drawing each subset conditional on the value of all the others. There are thus d steps in iteration t . At each iteration t , an ordering of the d subvectors of θ is chosen and, in turn, each θ_j^t is sampled from the conditional distribution given all the other components of θ :

$$p(\theta_j | \theta_{-j}^{t-1}, y),$$

where θ_{-j}^{t-1} represents all the components of θ , except for θ_j , at their current values:

$$\theta_{-j}^{t-1} = (\theta_1^t, \dots, \theta_{j-1}^t, \theta_{j+1}^{t-1}, \dots, \theta_d^{t-1}).$$

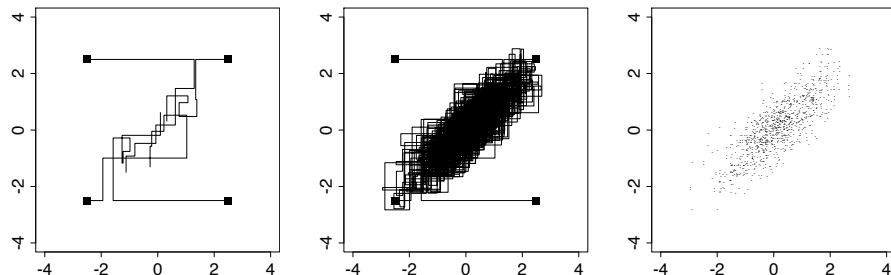


Figure 11.3 *Four independent sequences of the Gibbs sampler for a bivariate normal distribution with correlation $\rho = 0.8$, with overdispersed starting points indicated by solid squares. (a) First 10 iterations, showing the component-by-component updating of the Gibbs iterations. (b) After 500 iterations, the sequences have reached approximate convergence. Figure (c) shows the iterates from the second halves of the sequences.*

Thus, each subvector θ_j is updated conditional on the latest values of the other components of θ , which are the iteration t values for the components already updated and the iteration $t - 1$ values for the others.

For many problems involving standard statistical models, it is possible to sample directly from most or all of the conditional posterior distributions of the parameters. We typically construct models using a sequence of conditional probability distributions, as in the hierarchical models of Chapter 5. It is often the case that the conditional distributions in such models are conjugate distributions that provide for easy simulation. We present an example for the hierarchical normal model at the end of this chapter and another detailed example for a normal-mixture model in Section 18.4. Here, we illustrate the workings of the Gibbs sampler with a very simple example.

Example. Bivariate normal distribution

Consider a single observation (y_1, y_2) from a bivariate normally distributed population with unknown mean $\theta = (\theta_1, \theta_2)$ and known covariance matrix $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. With a uniform prior distribution on θ , the posterior distribution is

$$\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \Big| y \sim N \left(\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right).$$

Although it is simple to draw directly from the joint posterior distribution of (θ_1, θ_2) , we consider the Gibbs sampler for the purpose of exposition. To apply the Gibbs sampler to (θ_1, θ_2) , we need the conditional posterior distributions, which, from the properties of the multivariate normal distribution (either (A.1) or (A.2) on page 579), are

$$\begin{aligned} \theta_1 | \theta_2, y &\sim N(y_1 + \rho(\theta_2 - y_2), 1 - \rho^2) \\ \theta_2 | \theta_1, y &\sim N(y_2 + \rho(\theta_1 - y_1), 1 - \rho^2). \end{aligned}$$

The Gibbs sampler proceeds by alternately sampling from these two normal distributions. In general, we would say that a natural way to start the iterations

would be with random draws from a normal approximation to the posterior distribution; of course, such draws would eliminate the need for iterative simulation in this trivial example. Figure 11.3 illustrates for the case $\rho = 0.8$, data $(y_1, y_2) = (0, 0)$, and four independent sequences started at $(\pm 2.5, \pm 2.5)$.

11.4 The Metropolis and Metropolis-Hastings algorithms

Many clever methods have been devised for constructing and sampling from transition distributions for arbitrary posterior distributions. The *Metropolis-Hastings algorithm* is a general term for a family of Markov chain simulation methods that are useful for drawing samples from Bayesian posterior distributions. We have already seen the Gibbs sampler in the previous section; it can be viewed as a special case of Metropolis-Hastings (as described in Section 11.5). In this section, we present the basic Metropolis algorithm and its generalization to the Metropolis-Hastings algorithm. The next section talks about combining the Gibbs sampler and the Metropolis algorithm, which is often helpful in practical problems. Subsequent sections discuss practical issues in monitoring convergence and improving the efficiency of the simulation algorithms.

The Metropolis algorithm

The Metropolis algorithm is an adaptation of a random walk that uses an acceptance/rejection rule to converge to the specified target distribution. The algorithm proceeds as follows.

1. Draw a starting point θ^0 , for which $p(\theta^0|y) > 0$, from a *starting distribution* $p_0(\theta)$. The starting distribution might, for example, be based on an approximation as described in Section 12.2 in the following chapter. Or we may simply choose starting values dispersed around a crude approximate estimate of the sort discussed in Chapter 10.
2. For $t = 1, 2, \dots$:
 - (a) Sample a *proposal* θ^* from a *jumping distribution* (or *proposal distribution*) at time t , $J_t(\theta^*|\theta^{t-1})$. For the Metropolis algorithm (but not the Metropolis-Hastings algorithm, as discussed later in this section), the jumping distribution must be *symmetric*, satisfying the condition $J_t(\theta_a|\theta_b) = J_t(\theta_b|\theta_a)$ for all θ_a, θ_b , and t .
 - (b) Calculate the ratio of the densities,

$$r = \frac{p(\theta^*|y)}{p(\theta^{t-1}|y)}. \quad (11.1)$$

- (c) Set

$$\theta^t = \begin{cases} \theta^* & \text{with probability } \min(r, 1) \\ \theta^{t-1} & \text{otherwise.} \end{cases}$$

Given the current value θ^{t-1} , the transition distribution $T_t(\theta^t|\theta^{t-1})$ of the Markov chain is thus a mixture of a point mass at $\theta^t = \theta^{t-1}$, and a weighted version of the jumping distribution, $J_t(\theta^t|\theta^{t-1})$, that adjusts for the acceptance rate.

The algorithm requires the ability to calculate the ratio r in (11.1) for all (θ, θ^*) , and to draw θ from the jumping distribution $J_t(\theta^*|\theta)$ for all θ and t . In addition, step (c) above requires the generation of a uniform random number.

Note: if $\theta^t = \theta^{t-1}$ —that is, the jump is not accepted—this counts as an iteration in the algorithm.

Example. Bivariate unit normal density with bivariate normal jumping kernel

For simplicity, we illustrate the Metropolis algorithm with the simple example of the bivariate unit normal distribution. The target density is the bivariate unit normal, $p(\theta|y) = N(\theta|0, I)$, where I is the 2×2 identity matrix. The jumping distribution is also bivariate normal, centered at the current iteration and scaled to 1/5 the size: $J_t(\theta^*|\theta^{t-1}) = N(\theta^*|\theta^{t-1}, 0.2^2 I)$. At each step, it is easy to calculate the density ratio $r = N(\theta^*|0, I)/N(\theta^{t-1}|0, I)$. It is clear from the form of the normal distribution that the jumping rule is symmetric. Figure 11.2 on page 286 displays five simulation runs starting from different points. We have purposely chosen the relatively inefficient jumping rule with scale 1/5 in order to make the random walk aspect of the algorithm obvious in the figure.

Relation to optimization

The acceptance/rejection rule of the Metropolis algorithm can be stated as follows: (a) if the jump increases the posterior density, set $\theta^t = \theta^*$; (b) if the jump decreases the posterior density, set $\theta^t = \theta^*$ with probability equal to the density ratio, r , and set $\theta^t = \theta^{t-1}$ otherwise. The Metropolis algorithm can thus be viewed as a stochastic version of a stepwise mode-finding algorithm, always accepting steps that increase the density but only sometimes accepting downward steps.

Why does the Metropolis algorithm work?

The proof that the sequence of iterations $\theta^1, \theta^2, \dots$ converges to the target distribution has two steps: first, it is shown that the simulated sequence is a Markov chain with a unique stationary distribution, and second, it is shown that the stationary distribution equals this target distribution. The first step of the proof holds if the Markov chain is irreducible, aperiodic, and not transient. Except for trivial exceptions, the latter two conditions hold for a random walk on any proper distribution, and irreducibility holds as long as the random walk has a positive probability of eventually reaching any state from any other state; that is, the jumping distributions J_t must eventually be able to jump to all states with positive probability.

To see that the target distribution is the stationary distribution of the Markov chain generated by the Metropolis algorithm, consider starting the

algorithm at time $t - 1$ with a draw θ^{t-1} from the target distribution $p(\theta|y)$. Now consider any two such points θ_a and θ_b , drawn from $p(\theta|y)$ and labeled so that $p(\theta_b|y) \geq p(\theta_a|y)$. The unconditional probability density of a transition from θ_a to θ_b is

$$p(\theta^{t-1} = \theta_a, \theta^t = \theta_b) = p(\theta_a|y)J_t(\theta_b|\theta_a),$$

where the acceptance probability is 1 because of our labeling of a and b , and the unconditional probability density of a transition from θ_b to θ_a is, from (11.1),

$$\begin{aligned} p(\theta^t = \theta_a, \theta^{t-1} = \theta_b) &= p(\theta_b|y)J_t(\theta_a|\theta_b) \left(\frac{p(\theta_a|y)}{p(\theta_b|y)} \right) \\ &= p(\theta_a|y)J_t(\theta_a|\theta_b), \end{aligned}$$

which is the same as the probability of a transition from θ_a to θ_b , since we have required that $J_t(\cdot|\cdot)$ be symmetric. Since their joint distribution is symmetric, θ^t and θ^{t-1} have the same marginal distributions, and so $p(\theta|y)$ is the stationary distribution of the Markov chain of θ . For more detailed theoretical concerns, see the references at the end of this chapter.

The Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm generalizes the basic Metropolis algorithm presented above in two ways. First, the jumping rules J_t need no longer be symmetric; that is, there is no requirement that $J_t(\theta_a|\theta_b) \equiv J_t(\theta_b|\theta_a)$. Second, to correct for the asymmetry in the jumping rule, the ratio r in (11.1) is replaced by a ratio of ratios:

$$r = \frac{p(\theta^*|y)/J_t(\theta^*|\theta^{t-1})}{p(\theta^{t-1}|y)/J_t(\theta^{t-1}|\theta^*)}. \quad (11.2)$$

(The ratio r is always defined, because a jump from θ^{t-1} to θ^* can only occur if both $p(\theta^{t-1}|y)$ and $J_t(\theta^*|\theta^{t-1})$ are nonzero.)

Allowing asymmetric jumping rules can be useful in increasing the speed of the random walk. Convergence to the target distribution is proved in the same way as for the Metropolis algorithm. The proof of convergence to a unique stationary distribution is identical. To prove that the stationary distribution is the target distribution, $p(\theta|y)$, consider any two points θ_a and θ_b with posterior densities labeled so that $p(\theta_b|y)J_t(\theta_a|\theta_b) \geq p(\theta_a|y)J_t(\theta_b|\theta_a)$. If θ^{t-1} follows the target distribution, then it is easy to show that the unconditional probability density of a transition from θ_a to θ_b is the same as the reverse transition.

Relation between the jumping rule and efficiency of simulations

The ideal Metropolis-Hastings jumping rule is simply to sample the proposal, θ^* , from the target distribution; that is, $J(\theta^*|\theta) \equiv p(\theta^*|y)$ for all θ . Then the ratio r in (11.2) is always exactly 1, and the iterates θ^t are a sequence

of independent draws from $p(\theta|y)$. In general, however, iterative simulation is applied to problems for which direct sampling is not possible.

A good jumping distribution has the following properties:

- For any θ , it is easy to sample from $J(\theta^*|\theta)$.
- It is easy to compute the ratio r .
- Each jump goes a reasonable distance in the parameter space (otherwise the random walk moves too slowly).
- The jumps are not rejected too frequently (otherwise the random walk wastes too much time standing still).

We return to the topic of constructing efficient simulation algorithms in Sections 11.8 and 11.9.

11.5 Building Markov chain algorithms using the Gibbs sampler and Metropolis algorithm

The Gibbs sampler and the Metropolis algorithm can be used as building blocks for simulating from complicated distributions. The Gibbs sampler is the simplest of the Markov chain simulation algorithms, and it is our first choice for conditionally conjugate models, where we can directly sample from each conditional posterior distribution. For example, we could use the Gibbs sampler for the normal-normal hierarchical models in Chapter 5.

The Metropolis algorithm can be used for models that are not conditionally conjugate, for example, the two-parameter logistic regression for the bioassay experiment in Section 3.7. In this example, the Metropolis algorithm could be performed in vector form—jumping in the two-dimensional space of (α, β) —or embedded within a Gibbs sampler structure, by alternately updating α and β using one-dimensional Metropolis jumps. In either case, the Metropolis algorithm will probably have to be tuned to get a good acceptance rate, as discussed in Section 11.9.

If some of the conditional posterior distributions in a model can be sampled from directly and some cannot, then the parameters can be updated one at a time, with the Gibbs sampler used where possible and one-dimensional Metropolis updating used otherwise. More generally, the parameters can be updated in blocks, where each block is altered using the Gibbs sampler or a Metropolis jump of the parameters within the block.

A general problem with conditional sampling algorithms is that they can be slow when parameters are highly correlated in the target distribution (for example, see Figure 11.3 on page 288). This can be fixed in simple problems using reparameterization (see Section 11.8) or more generally using the more advanced algorithms mentioned in Chapter 13.

Interpretation of the Gibbs sampler as a special case of the Metropolis-Hastings algorithm

Gibbs sampling can be viewed as a special case of the Metropolis-Hastings algorithm in the following way. We first define iteration t to consist of a series of d steps, with step j of iteration t corresponding to an update of the subvector θ_j conditional on all the other elements of θ . Then the jumping distribution, $J_{j,t}(\cdot|\cdot)$, at step j of iteration t only jumps along the j th subvector, and does so with the conditional posterior density of θ_j given θ_{-j}^{t-1} :

$$J_{j,t}^{\text{Gibbs}}(\theta^*|\theta^{t-1}) = \begin{cases} p(\theta_j^*|\theta_{-j}^{t-1}, y) & \text{if } \theta_{-j}^* = \theta_{-j}^{t-1} \\ 0 & \text{otherwise.} \end{cases}$$

The only possible jumps are to parameter vectors θ^* that match θ^{t-1} on all components other than the j th. Under this jumping distribution, the ratio (11.2) at the j th step of iteration t is

$$\begin{aligned} r &= \frac{p(\theta^*|y)/J_{j,t}^{\text{Gibbs}}(\theta^*|\theta^{t-1})}{p(\theta^{t-1}|y)/J_{j,t}^{\text{Gibbs}}(\theta^{t-1}|\theta^*)} \\ &= \frac{p(\theta^*|y)/p(\theta_j^*|\theta_{-j}^{t-1}, y)}{p(\theta^{t-1}|y)/p(\theta_j^{t-1}|\theta_{-j}^{t-1}, y)} \\ &= \frac{p(\theta_{-j}^{t-1}|y)}{p(\theta_{-j}^{t-1}|y)} \\ &\equiv 1, \end{aligned}$$

and thus every jump is accepted. The second line above follows from the first because, under this jumping rule, θ^* differs from θ^{t-1} only in the j th component. The third line follows from the second by applying the rules of conditional probability to $\theta = (\theta_j, \theta_{-j})$ and noting that $\theta_{-j}^* = \theta_{-j}^{t-1}$.

Usually, one iteration of the Gibbs sampler is defined as we do, to include all d steps corresponding to the d components of θ , thereby updating all of θ at each iteration. It is possible, however, to define Gibbs sampling without the restriction that each component be updated in each iteration, as long as each component is updated periodically.

Gibbs sampler with approximations

For some problems, sampling from some, or all, of the conditional distributions $p(\theta_j|\theta_{-j}, y)$ is impossible, but one can construct approximations, which we label $g(\theta_j|\theta_{-j})$, from which sampling is possible. The general form of the Metropolis-Hastings algorithm can be used to compensate for the approximation. As in the Gibbs sampler, we choose an order for altering the d elements of θ ; the jumping function at the j th Metropolis step at iteration t is then

$$J_{j,t}(\theta^*|\theta^{t-1}) = \begin{cases} g(\theta_j^*|\theta_{-j}^{t-1}) & \text{if } \theta_{-j}^* = \theta_{-j}^{t-1} \\ 0 & \text{otherwise,} \end{cases}$$

and the ratio r in (11.2) must be computed and the acceptance or rejection of θ^* decided.

11.6 Inference and assessing convergence

The basic method of inference from iterative simulation is the same as for Bayesian simulation in general: use the collection of all the simulated draws from $p(\theta|y)$ to summarize the posterior density and to compute quantiles, moments, and other summaries of interest as needed. Posterior predictive simulations of unobserved outcomes $\tilde{y}$ can be obtained by simulation conditional on the drawn values of θ . Inference using the iterative simulation draws requires some care, however, as we discuss in this section.

Difficulties of inference from iterative simulation

Iterative simulation adds two difficulties to the problem of simulation inference. First, if the iterations have not proceeded long enough, as in Figure 11.2a, the simulations may be grossly unrepresentative of the target distribution. Even when the simulations have reached approximate convergence, the early iterations still are influenced by the starting approximation rather than the target distribution; for example, consider the early iterations of Figures 11.2b and 11.3b.

The second problem with iterative simulation draws is their within-sequence correlation; aside from any convergence issues, simulation inference from correlated draws is generally less precise than from the same number of independent draws. Serial correlation in the simulations is not necessarily a problem because, at convergence, the draws are identically distributed as $p(\theta|y)$, and so when performing inferences, we ignore the order of the simulation draws in any case. But such correlation can cause inefficiencies in simulations. Consider Figure 11.2c, which displays 500 successive iterations from each of five simulated sequences of the Metropolis algorithm: the patchy appearance of the scatterplot would not be likely to appear from 2500 independent draws from the normal distribution but is rather a result of the slow movement of the simulation algorithm. In some sense, the ‘effective’ number of simulation draws here is far fewer than 2500. We formally define this concept in equation (11.4) on page 298.

We handle the special problems of iterative simulation in three ways. First, we attempt to design the simulation runs to allow effective monitoring of convergence, in particular by simulating multiple sequences with starting points dispersed throughout parameter space, as in Figure 11.2a. Second, we monitor the convergence of all quantities of interest by comparing variation between and within simulated sequences until ‘within’ variation roughly equals ‘between’ variation, as in Figure 11.2b. Only when the distribution of each simulated sequence is close to the distribution of all the sequences mixed together can they all be approximating the target distribution. Third, if the simulation

efficiency is unacceptably low (in the sense of requiring too much real time on the computer to obtain approximate convergence of posterior inferences for quantities of interest), the algorithm can be altered, as we discuss in Sections 11.8 and 11.9.

Discarding early iterations of the simulation runs

To diminish the effect of the starting distribution, we generally discard the first half of each sequence and focus attention on the second half. Our ultimate inferences will be based on the assumption that the distributions of the simulated values θ^t , for large enough t , are close to the target distribution, $p(\theta|y)$. The practice of discarding early iterations in Markov chain simulation is referred to as *burn-in*; depending on the context, different burn-in fractions can be appropriate. For example, in the Gibbs sampler displayed in Figure 11.3, it would be necessary to discard only a few initial iterations.

We adopt the general practice of discarding the first half as a conservative choice. For example, we might run 200 iterations and discard the first half. If approximate convergence has not yet been reached, we might then run another 200 iterations, now discarding all of the initial 200 iterations.

Dependence of the iterations in each sequence

Another issue that sometimes arises, once approximate convergence has been reached, is whether to *thin* the sequences by keeping every k th simulation draw from each sequence and discarding the rest. In our applications, we have found it useful to skip iterations in problems with large numbers of parameters where computer storage is a problem, perhaps setting k so that the total number of iterations saved is no more than 1000.

Whether or not the sequences are thinned, if the sequences have reached approximate convergence, they can be directly used for inferences about the parameters θ and any other quantities of interest.

Monitoring convergence based on multiple sequences with overdispersed starting points

Our recommended approach to assessing convergence of iterative simulation is based on comparing different simulated sequences, as illustrated in Figure 11.2, which shows five parallel simulations before and after approximate convergence. In Figure 11.2a, the multiple sequences clearly have not converged; the variance within each sequence is much less than the variance between sequences. Later, in Figure 11.2b, the sequences have mixed, and the two variance components are essentially equal.

To see such disparities, we clearly need more than one independent sequence. Thus our plan is to simulate independently at least two sequences, with starting points drawn from an overdispersed distribution (either from a

crude estimate such as discussed in Section 10.1 or a more elaborate approximation as discussed in the next chapter).

Monitoring scalar estimands

Our approach involves monitoring each scalar estimand or other scalar quantities of interest separately. Estimands include all the parameters of interest in the model and any other quantities of interest (for example, the ratio of two parameters or the value of a predicted future observation). It is often useful also to monitor the value of the logarithm of the posterior density, which has probably already been computed if we are using a version of the Metropolis algorithm. Since our method of assessing convergence is based on means and variances, it is best where possible to transform the scalar estimands to be approximately normal (for example, take logarithms of all-positive quantities and logs of quantities that lie between 0 and 1).

Monitoring convergence of each scalar estimand

Suppose we have simulated m parallel sequences, each of length n (after discarding the first half of the simulations). For each scalar estimand ψ , we label the simulation draws as ψ_{ij} ($i = 1, \dots, n; j = 1, \dots, m$), and we compute B and W , the between- and within-sequence variances:

$$B = \frac{n}{m-1} \sum_{j=1}^m (\bar{\psi}_{\cdot j} - \bar{\psi}_{\cdot\cdot})^2, \quad \text{where} \quad \bar{\psi}_{\cdot j} = \frac{1}{n} \sum_{i=1}^n \psi_{ij}, \quad \bar{\psi}_{\cdot\cdot} = \frac{1}{m} \sum_{j=1}^m \bar{\psi}_{\cdot j}$$

$$W = \frac{1}{m} \sum_{j=1}^m s_j^2, \quad \text{where} \quad s_j^2 = \frac{1}{n-1} \sum_{i=1}^n (\psi_{ij} - \bar{\psi}_{\cdot j})^2.$$

The between-sequence variance, B , contains a factor of n because it is based on the variance of the within-sequence means, $\bar{\psi}_{\cdot j}$, each of which is an average of n values ψ_{ij} . If only one sequence is simulated (that is, if $m = 1$), then B cannot be calculated.

We can estimate $\text{var}(\psi|y)$, the marginal posterior variance of the estimand, by a weighted average of W and B , namely

$$\widehat{\text{var}}^+(\psi|y) = \frac{n-1}{n} W + \frac{1}{n} B. \quad (11.3)$$

This quantity *overestimates* the marginal posterior variance assuming the starting distribution is appropriately overdispersed, but is *unbiased* under stationarity (that is, if the starting distribution equals the target distribution), or in the limit $n \rightarrow \infty$ (see Exercise 11.4). This is analogous to the classical variance estimate with cluster sampling.

Meanwhile, for any finite n , the ‘within’ variance W should be an *underestimate* of $\text{var}(\psi|y)$ because the individual sequences have not had time to range over all of the target distribution and, as a result, will have less variability; in the limit as $n \rightarrow \infty$, the expectation of W approaches $\text{var}(\psi|y)$.

We monitor convergence of the iterative simulation by estimating the factor by which the scale of the current distribution for ψ might be reduced if the simulations were continued in the limit $n \rightarrow \infty$. This potential scale reduction is estimated by*

$$\hat{R} = \sqrt{\frac{\widehat{\text{var}}^+(\psi|y)}{W}},$$

which declines to 1 as $n \rightarrow \infty$. If the potential scale reduction is high, then we have reason to believe that proceeding with further simulations may improve our inference about the target distribution of the associated scalar estimand.

Monitoring convergence for the entire distribution

We recommend computing the potential scale reduction for all scalar estimands of interest; if $\hat{R}$ is not near 1 for all of them, continue the simulation runs (perhaps altering the simulation algorithm itself to make the simulations more efficient, as described in the next section). Once $\hat{R}$ is near 1 for all scalar estimands of interest, just collect the mn simulations from the second halves of all the sequences together and treat them as a sample from the target distribution. The condition of $\hat{R}$ being ‘near’ 1 depends on the problem at hand; for most examples, values below 1.1 are acceptable, but for a final analysis in a critical problem, a higher level of precision may be required.

In addition, even if an iterative simulation appears to converge and has passed all tests of convergence, it still may actually be far from convergence if important areas of the target distribution were not captured by the starting distribution and are not easily reachable by the simulation algorithm

Example. Bivariate unit normal density with bivariate normal jumping kernel (continued)

We illustrate the multiple sequence method using the Metropolis simulations of the bivariate normal distribution illustrated in Figure 11.2. Table 11.1 displays posterior inference for the two parameters of the distribution as well as the log posterior density (relative to the density at the mode). After 50 iterations, the variance between the five sequences is much greater than the variance within, for all three univariate summaries considered. However, the five simulated sequences have converged adequately after 2000 or certainly 5000 iterations for the quantities of interest. The comparison with the true target distribution shows how some variability remains in the posterior inferences even after the Markov chains have converged. (This must be so, considering that even if the simulation draws were independent, so that the Markov chains would converge in a single iteration, it would still require hundreds or thousands of draws to obtain precise estimates of extreme posterior quantiles.)

The method of monitoring convergence presented here has the key advantage of not requiring the user to examine time series graphs of simulated

* In the first edition of this book, $\hat{R}$ was defined as $\widehat{\text{var}}^+(\psi|y)/W$. We switch to the square-root definition for notational convenience.

Number of iterations	95% intervals and $\hat{R}$ for ...		
	θ_1	θ_2	$\log p(\theta_1, \theta_2 y)$
50	$[-2.14, 3.74], 12.3$	$[-1.83, 2.70], 6.1$	$[-8.71, -0.17], 6.1$
500	$[-3.17, 1.74], 1.3$	$[-2.17, 2.09], 1.7$	$[-5.23, -0.07], 1.3$
2000	$[-1.83, 2.24], 1.2$	$[-1.74, 2.09], 1.03$	$[-4.07, -0.03], 1.10$
5000	$[-2.09, 1.98], 1.02$	$[-1.90, 1.95], 1.03$	$[-3.70, -0.03], 1.00$
∞	$[-1.96, 1.96], 1$	$[-1.96, 1.96], 1$	$[-3.69, -0.03], 1$

Table 11.1 95% central intervals and estimated potential scale reduction factors for three scalar summaries of the bivariate normal distribution simulated using a Metropolis algorithm. (For demonstration purposes, the jumping scale of the Metropolis algorithm was purposely set to be inefficient; see Figure 11.2.) Displayed are inferences from the second halves of five parallel sequences, stopping after 50, 500, 2000, and 5000 iterations. The intervals for ∞ are taken from the known normal and $\chi^2_2/2$ marginal distributions for these summaries in the target distribution.

sequences. Inspection of such plots is a notoriously unreliable method of assessing convergence (see references at the end of this chapter) and in addition is unwieldy when monitoring a large number of quantities of interest, such as can arise in complicated hierarchical models. Because it is based on means and variances, the simple method presented here is most effective for quantities whose marginal posterior distributions are approximately normal. When performing inference for extreme quantiles, or for parameters with multimodal marginal posterior distributions, one should monitor also extreme quantiles of the ‘between’ and ‘within’ sequences.

Effective number of independent draws

Once the simulated sequences have mixed, we can compute an approximate ‘effective number of independent simulation draws’ for any estimand of interest ψ by comparing the variances between and within the simulated sequences. We start with the observation that if the n simulation draws within each sequence were truly independent, then the between-sequence variance B would be an unbiased estimate of the posterior variance, $\text{var}(\psi|y)$, and we would have a total of mn independent simulations from the m sequences. In general, however, the simulations of ψ within each sequence will be autocorrelated, and B will be larger than $\text{var}(\psi|y)$, in expectation. We thus define the effective number of independent draws of ψ as

$$n_{\text{eff}} = mn \frac{\widehat{\text{var}}^+(\psi|y)}{B}, \quad (11.4)$$

with $\widehat{\text{var}}^+(\psi|y)$ and B as defined on page 296. If m is small, then B will have a high sampling variability, so that n_{eff} is a fairly crude estimate. We actually report $\min(n_{\text{eff}}, mn)$, to avoid claims that our simulation is more efficient than

Diet	Measurements
A	62, 60, 63, 59
B	63, 67, 71, 64, 65, 66
C	68, 66, 71, 67, 68, 68
D	56, 62, 60, 61, 63, 64, 63, 59

Table 11.2 *Coagulation time in seconds for blood drawn from 24 animals randomly allocated to four different diets. Different treatments have different numbers of observations because the randomization was unrestricted. From Box, Hunter, and Hunter (1978), who adjusted the data so that the averages are integers, a complication we ignore in our analysis.*

random sampling. (Superefficient iterative simulation is possible but in practice highly unlikely.) If desired, more precise measures of simulation accuracy and effective number of draws could be constructed based on autocorrelations within each sequence.

11.7 Example: the hierarchical normal model

We illustrate the simulation algorithms with the hierarchical normal model, extending the problem discussed in Section 5.4 by allowing the data variance, σ^2 , to be unknown. The example is continued in Section 12.5 to illustrate mode-based computation. We demonstrate with the normal model because it is simple enough that the key computational ideas do not get lost in the details.

Data from a small experiment

We demonstrate the computations on a small experimental dataset, displayed in Table 11.2, that has been used previously as an example in the statistical literature. Our purpose here is solely to illustrate computational methods, not to perform a full Bayesian data analysis (which includes model construction and model checking), and so we do not discuss the applied context.

The model

Under the hierarchical normal model (restated here, for convenience), data y_{ij} , $i = 1, \dots, n_j$, $j = 1, \dots, J$, are independently normally distributed within each of J groups, with means θ_j and common variance σ^2 . The total number of observations is $n = \sum_{j=1}^J n_j$. The group means are assumed to follow a normal distribution with unknown mean μ and variance τ^2 , and a uniform prior distribution is assumed for $(\mu, \log \sigma, \tau)$, with $\sigma > 0$ and $\tau > 0$; equivalently, $p(\mu, \log \sigma, \log \tau) \propto \tau$. If we were to assign a uniform prior distribution to $\log \tau$, the posterior distribution would be improper, as discussed in Chapter 5.

The joint posterior density of all the parameters is

$$p(\theta, \mu, \log \sigma, \log \tau | y) \propto \tau \prod_{j=1}^J N(\theta_j | \mu, \tau^2) \prod_{j=1}^J \prod_{i=1}^{n_j} N(y_{ij} | \theta_j, \sigma^2).$$

Starting points

In this example, we can choose overdispersed starting points for each parameter θ_j by simply taking random points from the data y_{ij} from group j . We obtain 10 starting points for the simulations by drawing θ_j independently in this way for each group. We also need starting points for μ , which can be taken as the average of the starting θ_j values. No starting values are needed for τ or σ as they can be drawn as the first steps in the Gibbs sampler.

Section 12.5 presents a more elaborate procedure for constructing a starting distribution for the iterative simulations using the posterior mode and a normal approximation.

Gibbs sampler

The conditional distributions for this model all have simple conjugate forms:

1. *Conditional posterior distribution of each θ_j .* The factors in the joint posterior density that involve θ_j are the $N(\mu, \tau^2)$ prior distribution and the normal likelihood from the data in the j th group, $y_{ij}, i = 1, \dots, n_j$. The conditional posterior distribution of each θ_j given the other parameters in the model is

$$\theta_j | \mu, \sigma, \tau, y \sim N(\hat{\theta}_j, V_{\theta_j}), \quad (11.5)$$

where the parameters of the conditional posterior distribution depend on μ , σ , and τ as well as y :

$$\hat{\theta}_j = \frac{\frac{1}{\tau^2} \mu + \frac{n_j}{\sigma^2} \bar{y}_{.j}}{\frac{1}{\tau^2} + \frac{n_j}{\sigma^2}} \quad (11.6)$$

$$V_{\theta_j} = \frac{1}{\frac{1}{\tau^2} + \frac{n_j}{\sigma^2}}. \quad (11.7)$$

These conditional distributions are independent; thus drawing the θ_j 's one at a time is equivalent to drawing the vector θ all at once from its conditional posterior distribution.

2. *Conditional posterior distribution of μ .* Conditional on y and the other parameters in the model, μ has a normal distribution determined by the J values θ_j :

$$\mu | \theta, \sigma, \tau, y \sim N(\hat{\mu}, \tau^2/J), \quad (11.8)$$

where

$$\hat{\mu} = \frac{1}{J} \sum_{j=1}^J \theta_j. \quad (11.9)$$

3. *Conditional posterior distribution of σ^2 .* The conditional posterior density for σ^2 has the form corresponding to a normal variance with known mean; there are n observations y_{ij} with means θ_j . The conditional posterior distribution is

$$\sigma^2 | \theta, \mu, \tau, y \sim \text{Inv-}\chi^2(n, \hat{\sigma}^2), \quad (11.10)$$

where

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{j=1}^J \sum_{i=1}^{n_j} (y_{ij} - \theta_j)^2. \quad (11.11)$$

4. *Conditional posterior distribution of τ^2 .* Conditional on the data and the other parameters in the model, τ^2 has a scaled inverse- χ^2 distribution, with parameters depending only on μ and θ (as can be seen by examining the joint posterior density):

$$\tau^2 | \theta, \mu, \sigma, y \sim \text{Inv-}\chi^2(J-1, \hat{\tau}^2), \quad (11.12)$$

with

$$\hat{\tau}^2 = \frac{1}{J-1} \sum_{j=1}^J (\theta_j - \mu)^2. \quad (11.13)$$

The expressions for τ^2 have $(J-1)$ degrees of freedom instead of J because $p(\tau) \propto 1$ rather than τ^{-1} .

Numerical results with the coagulation data

We illustrate the Gibbs sampler with the coagulation data of Table 11.2. Inference from ten parallel Gibbs sampler sequences appears in Table 11.3; 100 iterations were sufficient for approximate convergence.

The Metropolis algorithm

We also describe how the Metropolis algorithm can be used for this problem. It would be possible to apply the algorithm to the entire joint distribution, $p(\theta, \mu, \sigma, \tau | y)$, but we can work more efficiently in a lower-dimensional space by taking advantage of the conjugacy of the problem that allows us to compute the function $p(\mu, \log \sigma, \log \tau | y)$, as we discuss in Section 12.5. We use the Metropolis algorithm to jump through the marginal posterior distribution of $(\mu, \log \sigma, \log \tau)$ and then draw simulations of the vector θ from its normal conditional posterior distribution (11.5). Following a principle of efficient Metropolis jumping that we shall discuss in Section 11.9, we jump through the space of $(\mu, \log \sigma, \log \tau)$ using a multivariate normal jumping kernel with mean equal to the current value of the parameters and variance matrix equal to that of a normal approximation (see Section 12.5), multiplied by $2.4^2/d$, where d is the dimension of the Metropolis jumping distribution. In this case, $d = 3$.

Estimand	Posterior quantiles					$\hat{R}$
	2.5%	25%	median	75%	97.5%	
θ_1	58.9	60.6	61.3	62.1	63.5	1.01
θ_2	63.9	65.3	65.9	66.6	67.7	1.01
θ_3	66.0	67.1	67.8	68.5	69.5	1.01
θ_4	59.5	60.6	61.1	61.7	62.8	1.01
μ	56.9	62.2	63.9	65.5	73.4	1.04
σ	1.8	2.2	2.4	2.6	3.3	1.00
τ	2.1	3.6	4.9	7.6	26.6	1.05
$\log p(\mu, \log \sigma, \log \tau y)$	-67.6	-64.3	-63.4	-62.6	-62.0	1.02
$\log p(\theta, \mu, \log \sigma, \log \tau y)$	-70.6	-66.5	-65.1	-64.0	-62.4	1.01

Table 11.3 *Summary of posterior inference for the individual-level parameters and hyperparameters for the coagulation example. Posterior quantiles and estimated potential scale reductions computed from the second halves of ten Gibbs sampler sequences, each of length 100. Potential scale reductions for σ and τ were computed on the log scale. The hierarchical standard deviation, τ , is estimated less precisely than the unit-level standard deviation, σ , as is typical in hierarchical modeling with a small number of batches.*

Metropolis results with the coagulation data

We ran ten parallel sequences of Metropolis algorithm simulations. In this case 500 iterations were sufficient for approximate convergence ($\hat{R} < 1.1$ for all parameters); at that point we obtained similar results to those obtained using Gibbs sampling. The acceptance rate for the Metropolis simulations was 0.35, which is close to the expected result for the normal distribution with $d = 3$ using a jumping distribution scaled by $2.4/\sqrt{d}$ (see Section 11.8).

11.8 Efficient Gibbs samplers

Various theoretical arguments suggest methods for constructing efficient simulation algorithms and improving the efficiency of existing algorithms. This is an area of much current research (see the bibliographic notes at the end of this chapter and the next); in this section and the next we discuss two of the simplest and most general approaches: choice of parameterization and scaling of Metropolis jumping rules.

Transformations and reparameterization

The Gibbs sampler is most efficient when parameterized in terms of independent components; Figure 11.3 shows an example with highly dependent components that create slow convergence. The simplest way to reparameterize is by a linear transformation of the parameters, but posterior distributions that are not approximately normal may require special methods.

The same arguments apply to Metropolis jumps. In a normal or approximately normal setting, the jumping kernel should ideally have the same covariance structure as the target distribution, which can be approximately estimated based on the normal approximation at the mode. Markov chain simulation of a distribution with multiple modes can be greatly improved by allowing jumps between modes. Section 13.1 describes an approach for dealing with multiple modes.

Auxiliary variables

Gibbs sampler computations can often be simplified or convergence accelerated by adding auxiliary variables, for example indicator variables for mixture distributions, as described in Chapter 18. The idea of adding variables is also called *data augmentation* and is a useful conceptual and computational tool for many problems, both for the Gibbs sampler and for the EM algorithm (as discussed in Section 12.3).

Example. Student- t model

A simple but important example of auxiliary variables arises with the t distribution, which can be expressed as a mixture of normal distributions, as noted in Chapter 3 and discussed in more detail in Chapter 17. We illustrate with the example of inference for the parameters μ, σ^2 given n iid data points from the $t_\nu(\mu, \sigma^2)$ distribution, where for simplicity we assume ν is known. We also assume a uniform prior distribution on $\mu, \log \sigma$. The t likelihood for each data point is equivalent to the model,

$$\begin{aligned} y_i &\sim N(\mu, V_i) \\ V_i &\sim \text{Inv-}\chi^2(\nu, \sigma^2), \end{aligned} \quad (11.14)$$

where the V_i 's are auxiliary variables that cannot be directly observed. If we perform inference using the joint posterior distribution, $p(\mu, \sigma^2, V|y)$, and then just consider the simulations for μ, σ , these will represent the posterior distribution under the original t model.

There is no direct way to sample from the parameters μ, σ^2 in the t model, but it is straightforward to perform the Gibbs sampler on V, μ, σ^2 in the augmented model:

1. *Conditional posterior distribution of each V_i .* Conditional on the data y and the other parameters of the model, each V_i is a normal variance parameter with a scaled inverse- χ^2 prior distribution, and so its posterior distribution is also inverse- χ^2 (see Section 2.7):

$$V_i | \mu, \sigma^2, \nu, y \sim \text{Inv-}\chi^2 \left(\nu + 1, \frac{\nu\sigma^2 + (y_i - \mu)^2}{\nu + 1} \right).$$

The n parameters V_i are independent in their conditional posterior distribution, and we can directly apply the Gibbs sampler by sampling from their scaled inverse- χ^2 distributions.

2. *Conditional posterior distribution of μ .* Conditional on the data y and the other parameters of the model, information about μ is supplied by the n data points

y_i , each with its own variance. Combining with the uniform prior distribution on μ yields,

$$\mu|\sigma^2, V, \nu, y \sim N\left(\frac{\sum_{i=1}^n \frac{1}{V_i} y_i}{\sum_{i=1}^n \frac{1}{V_i}}, \frac{1}{\sum_{i=1}^n \frac{1}{V_i}}\right).$$

3. *Conditional posterior distribution of σ^2 .* Conditional on the data y and the other parameters of the model, all the information about σ comes from the variances V_i . The conditional posterior distribution is,

$$\begin{aligned} p(\sigma^2|\mu, V, \nu, y) &\propto \sigma^{-2} \prod_{i=1}^n \sigma^\nu e^{-\nu\sigma^2/(2V_i)} \\ &= (\sigma^2)^{n\nu/2-1} \exp\left(-\frac{\nu}{2} \sum_{i=1}^n \frac{1}{V_i} \sigma^2\right) \\ &\propto \text{Gamma}\left(\sigma^2 \left| \frac{n\nu}{2}, \frac{\nu}{2} \sum_{i=1}^n \frac{1}{V_i} \right.\right), \end{aligned}$$

from which we can sample directly.

Parameter expansion

For some problems, the Gibbs sampler can be slow to converge because of posterior dependence among parameters that cannot simply be resolved with a linear transformation. Paradoxically, adding an additional parameter—thus performing the random walk in a larger space—can improve the convergence of the Markov chain simulation. We illustrate with the Student- t example above.

Example. Student- t model (continued)

In the latent-parameter form (11.14) of the t model, convergence will be slow if a simulation draw of σ is close to zero, because the conditional distributions will then cause the V_i 's to be sampled with values near zero, and then the conditional distribution of σ will be near zero, and so on. Eventually the simulations will get unstuck but it can be slow for some problems. We can fix things by adding a new parameter whose only role is to allow the Gibbs sampler to move in more directions and thus avoid getting stuck.

The expanded model is,

$$\begin{aligned} y_i &\sim N(\mu, \alpha^2 U_i) \\ U_i &\sim \text{Inv-}\chi^2(\nu, \tau^2), \end{aligned}$$

where $\alpha > 0$ can be viewed as an additional scale parameter. In this new model, $\alpha^2 U_i$ plays the role of V_i in (11.14) and $\alpha\tau$ plays the role of σ . The parameter α has no meaning on its own and we can assign it a noninformative uniform prior distribution on the logarithmic scale.

The Gibbs sampler on this expanded model now has four steps:

1. For each i , U_i is updated much as V_i was before:

$$U_i | \alpha, \mu, \tau^2, \nu, y \sim \text{Inv-}\chi^2 \left(\nu + 1, \frac{\nu \tau^2 + ((y_i - \mu)/\alpha)^2}{\nu + 1} \right).$$

2. The mean, μ , is updated as before:

$$\mu | \alpha, \tau^2, U, \nu, y \sim N \left(\frac{\sum_{i=1}^n \frac{1}{\alpha^2 U_i} y_i}{\sum_{i=1}^n \frac{1}{\alpha^2 U_i}}, \frac{1}{\sum_{i=1}^n \frac{1}{\alpha^2 U_i}} \right).$$

3. The variance parameter τ^2 , is updated much as σ^2 was before:

$$\tau^2 | \alpha, \mu, U, \nu, y \sim \text{Gamma} \left(\frac{n\nu}{2}, \frac{\nu}{2} \sum_{i=1}^n \frac{1}{U_i} \right).$$

4. Finally, we must update α^2 , which is easy since conditional on all the other parameters in the model it is simply a normal variance parameter:

$$\alpha^2 | \mu, \tau^2, U, \nu, y \sim \text{Inv-}\chi^2 \left(n, \frac{1}{n} \sum_{i=1}^n \frac{(y_i - \mu)^2}{U_i} \right).$$

The parameters α^2, U, τ in this expanded model are not identified in that the data do not supply enough information to estimate each of these quantities. However, the model as a whole is identified as long as we monitor convergence of the summaries μ , $\sigma = \alpha\tau$, and $V_i = \alpha^2 U_i$ for $i = 1, \dots, n$. (Or, if the only goal is inference for the original t model, we can simply save μ and σ from the simulations.)

The Gibbs sampler under the expanded parameterizations converges more reliably because the new parameter α breaks the dependence between τ and the V_i 's.

Parameter expansion is particularly useful for hierarchical linear models such as in Chapter 15, as we discuss in Section 15.4 and illustrate in Appendix C.

11.9 Efficient Metropolis jumping rules

For any given posterior distribution, the Metropolis-Hastings algorithm can be implemented in an infinite number of ways. Even after reparameterizing, there are still endless choices in the jumping rules, J_t . In many situations with conjugate families, the posterior simulation can be performed entirely or in part using the Gibbs sampler, which is not always efficient but generally is easy to program, as we illustrated with the hierarchical normal model in Section 11.7. For nonconjugate models we must rely on Metropolis-Hastings algorithms (either within a Gibbs sampler or directly on the multivariate posterior distribution). The choice of jumping rule then arises.

There are two main classes of simple jumping rules. The first are essentially random walks around the parameter space. These jumping rules are often normal jumping kernels with mean equal to the current value of the parameter and variance set to obtain efficient algorithms. The second approach uses

proposal distributions that are constructed to closely approximate the target distribution (either the conditional distribution of a subset in a Gibbs sampler or the joint posterior distribution). In the second case the goal is to accept as many draws as possible with the Metropolis-Hastings acceptance step being used primarily to correct the approximation. There is no natural advantage to altering one parameter at a time except for potential computational savings in evaluating only part of the posterior density at each step.

It is hard to give general advice on efficient jumping rules, but some results have been obtained for normal random walk jumping distributions that seem to be useful in many problems. Suppose there are d parameters, and the posterior distribution of $\theta = (\theta_1, \dots, \theta_d)$, after appropriate transformation, is multivariate normal with known variance matrix Σ . Further suppose that we will take draws using the Metropolis algorithm with a normal jumping kernel centered on the current point and with the same shape as the target distribution: that is, $J(\theta^*|\theta^{t-1}) = N(\theta^*|\theta^{t-1}, c^2\Sigma)$. Among this class of jumping rules, the most efficient has scale $c \approx 2.4/\sqrt{d}$, where efficiency is defined relative to independent sampling from the posterior distribution. The efficiency of this optimal Metropolis jumping rule for the d -dimensional normal distribution can be shown to be about $0.3/d$ (by comparison, if the d parameters were independent in their posterior distribution, the Gibbs sampler would have efficiency $1/d$, because after every d iterations, a new independent draw of θ would be created). Which algorithm is best for any particular problem also depends on the computation time for each iteration, which in turn depends on the conditional independence and conjugacy properties of the posterior density.

A Metropolis algorithm can also be characterized by the proportion of jumps that are accepted. For the multivariate normal random walk jumping distribution with jumping kernel the same shape as the target distribution, the optimal jumping rule has acceptance rate around 0.44 in one dimension, declining to about 0.23 in high dimensions (roughly $d > 5$). This result suggests an *adaptive* simulation algorithm:

1. Start the parallel simulations with a fixed algorithm, such as a version of the Gibbs sampler, or the Metropolis algorithm with a normal random walk jumping rule shaped like an estimate of the target distribution (using the covariance matrix computed at the joint or marginal posterior mode scaled by the factor $2.4/\sqrt{d}$).
2. After some number of simulations, update the Metropolis jumping rule as follows.
 - (a) Adjust the covariance of the jumping distribution to be proportional to the posterior covariance matrix estimated from the simulations.
 - (b) Increase or decrease the scale of the jumping distribution if the acceptance rate of the simulations is much too high or low, respectively. The goal is to bring the jumping rule toward the approximate optimal value

of 0.44 (in one dimension) or 0.23 (when many parameters are being updated at once using vector jumping).

This algorithm can be improved in various ways, but even in its simple form, we have found it useful for drawing posterior simulations for some problems with d ranging from 1 to 50.

Adaptive algorithms

When an iterative simulation algorithm is ‘tuned’—that is, modified while it is running—care must be taken to avoid converging to the wrong distribution. (If the updating rule depends on previous simulation steps, then the transition probabilities are more complicated than as stated in the Metropolis-Hastings algorithm, and the iterations will not necessarily converge to the target distribution.) To be safe, we usually run any adaptive algorithm in two phases: first, the adaptive phase, where the parameters of the algorithm can be tuned as often as desired to increase the simulation efficiency, and second, a fixed phase, where the adapted algorithm is run long enough for approximate convergence. Only simulations from the fixed phase are used in the final inferences.

11.10 Recommended strategy for posterior simulation

We summarize our recommended basic approach to Bayesian computation:

1. Start off with crude estimates and possibly a mode-based approximation to the posterior distribution (see Section 10.1 and Chapter 12).
2. If possible, simulate from the posterior distribution directly or sequentially, starting with hyperparameters and then moving to the main parameters (as in the simple hierarchical models in Chapter 5).
3. Most likely, the best approach is to set up a Markov chain simulation algorithm. The updating can be done one parameter at a time or with parameters in batches (as is often convenient in regressions and hierarchical models; see Chapter 15).
4. For parameters (or batches of parameters) whose conditional posterior distributions have standard forms, use the Gibbs sampler.
5. For parameters whose conditional distributions do not have standard forms, use Metropolis jumps. Tune the scale of each jumping distribution so that acceptance rates are near 20% (when altering a vector of parameters) or 40% (when altering one parameter at a time).
6. Construct a transformation so that the parameters are approximately independent—this will speed the convergence of the Gibbs sampler.
7. Start the Markov chain simulations with parameter values taken from the crude estimates or mode-based approximations, with noise added so they are overdispersed with respect to the target distribution. Mode-based approximations to the posterior density are discussed in Chapter 12.

8. Run multiple Markov chains and monitor the mixing of the sequences. Run until approximate convergence appears to have been reached, in the sense that the statistic $\hat{R}$, defined in Section 11.6, is near 1 (below 1.1, say) for each scalar estimand of interest. This will typically take hundreds of iterations, at least. If approximate convergence has not been reached after a long time, consider making the simulations more efficient as discussed in Sections 11.8 and 11.9.
9. If $\hat{R}$ is near 1 for all scalar estimands of interest, summarize inference about the posterior distribution by treating the set of all iterates from the second half of the simulated sequences as an identically distributed sample from the target distribution. These can be stored as a matrix (as in Figure 1.1 on page 26). At this point, simulations from the different sequences can be mixed.
10. Compare the posterior inferences from the Markov chain simulation to the approximate distribution used to start the simulations. If they are not close with respect to locations and approximate distributional shape, check for errors before believing that the Markov chain simulation has produced a better answer.

11.11 Bibliographic note

An excellent general book on simulation from a statistical perspective is Ripley (1987), which covers two topics that we do not address in this chapter: creating uniformly distributed (pseudo)random numbers and simulating from standard distributions (on the latter, see our Appendix A for more details). Hammersley and Handscomb (1964) is a classic reference on simulation. Thisted (1988) is a general book on statistical computation that discusses many optimization and simulation techniques. Robert and Casella (1999) covers simulation algorithms from a variety of statistical perspectives. The book by Liu (2002) reviews more advanced simulation algorithms, some of which we discuss in Chapter 13.

Kass et al. (1998) discuss many practical issues in Bayesian simulation. Gilks, Richardson, and Spiegelhalter (1996) is a book full of examples and applications of Markov chain simulation methods. Further references on Bayesian computation appear in the books by Tanner (1993) and Chen, Shao, and Ibrahim (2000). Many other applications of Markov chain simulation appear in the recent applied statistical literature.

Metropolis and Ulam (1949) and Metropolis et al. (1953) apparently were the first to describe Markov chain simulation of probability distributions (that is, the ‘Metropolis algorithm’). Their algorithm was generalized by Hastings (1970); see Chib and Greenberg (1995) for an elementary introduction and Tierney (1998) for a theoretical perspective. The conditions for Markov chain convergence appear in probability texts such as Feller (1968), and more recent work such as Rosenthal (1995) has evaluated the rates of convergence of Markov chain algorithms for statistical models. The Gibbs sampler was first so named by Geman and Geman (1984) in a discussion of applications to image

processing. Tanner and Wong (1987) introduced the idea of iterative simulation to many statisticians, using the special case of ‘data augmentation’ to emphasize the analogy to the EM algorithm (see Section 12.3). Gelfand and Smith (1990) showed how the Gibbs sampler could be used for Bayesian inference for a variety of important statistical models. The Metropolis-approximate Gibbs algorithm introduced at the end of Section 11.5 appears in Gelman (1992b) and is used by Gilks, Best, and Tan (1995).

Gelfand et al. (1990) applied Gibbs sampling to a variety of statistical problems, and many other applications of the Gibbs sampler algorithms have appeared since; for example, Clayton (1991) and Carlin and Polson (1991). Besag and Green (1993), Gilks et al. (1993), and Smith and Roberts (1993) discuss Markov simulation algorithms for Bayesian computation. **Bugs** (Spiegelhalter et al., 1994, 2003) is a general-purpose computer program for Bayesian inference using the Gibbs sampler; see Appendix C for details.

Inference and monitoring convergence from iterative simulation is reviewed by Gelman and Rubin (1992b), who provide a theoretical justification of the method presented in Section 11.6 and discuss a more elaborate version of the method; Brooks and Gelman (1998) and Brooks and Giudici (2000) present more recent work along these lines. Other views on assessing convergence appear in the ensuing discussion of Gelman and Rubin (1992b) and in Cowles and Carlin (1996) and Brooks and Roberts (1998). Gelman and Rubin (1992a, b) and Glickman (1993) present examples of iterative simulation in which lack of convergence is impossible to detect from single sequences but is obvious from multiple sequences.

The data on coagulation times used to illustrate the computations for the hierarchical normal model were analyzed by Box, Hunter, and Hunter (1978) using non-Bayesian methods based on the analysis of variance.

For the relatively simple ways of improving simulation algorithms mentioned in Sections 11.8 and 11.9, Tanner and Wong (1987) discuss data augmentation and auxiliary variables, and Hills and Smith (1992) and Roberts and Sahu (1997) discuss parameterization for the Gibbs sampler. Higdon (1998) discusses some more complicated auxiliary variable methods, and Liu and Wu (1999), van Dyk and Meng (2001), and Liu (2003) present different approaches to parameter expansion. The results on acceptance rates for efficient Metropolis jumping rules appear in Gelman, Roberts, and Gilks (1995); more general results for Metropolis-Hastings algorithms appear in Roberts and Rosenthal (2001) and Brooks, Giudici, and Roberts (2003).

Gelfand and Sahu (1994) discuss the difficulties of maintaining convergence to the target distribution when adapting Markov chain simulations, as discussed at the end of Section 11.9. Gilks and Berzuini (2001) and Andrieu and Robert (2001) present further work on adaptive Markov chain simulation algorithms.

11.12 Exercises

1. Rejection sampling:
 - (a) Prove that rejection sampling gives draws from $p(\theta|y)$.
 - (b) Why is the boundedness condition on $p(\theta|y)/q(\theta)$ necessary for rejection sampling?
2. Metropolis-Hastings algorithm: Show that the stationary distribution for the Metropolis-Hastings algorithm is, in fact, the target distribution, $p(\theta|y)$.
3. Metropolis algorithm: Replicate the computations for the bioassay example of Section 3.7 using the Metropolis algorithm. Be sure to define your starting points and your jumping rule. Run the simulations long enough for approximate convergence.
4. Monitoring convergence:
 - (a) Prove that $\widehat{\text{var}}^+(\psi|y)$ as defined in (11.3) is an unbiased estimate of the marginal posterior variance of ϕ , if the starting distribution for the Markov chain simulation algorithm is the same as the target distribution, and if the m parallel sequences are computed independently. (Hint: show that $\widehat{\text{var}}^+(\psi|y)$ can be expressed as the average of the halved squared differences between simulations ϕ from different sequences, and that each of these has expectation equal to the posterior variance.)
 - (b) Determine the conditions under which $\widehat{\text{var}}^+(\psi|y)$ approaches the marginal posterior variance of ϕ in the limit as the lengths n of the simulated chains approach ∞ .
5. Analysis of a stratified sample survey: Section 7.4 presents an analysis of a stratified sample survey using a hierarchical model on the stratum probabilities.
 - (a) Perform the computations for the simple nonhierarchical model described in the example.
 - (b) Using the Metropolis algorithm, perform the computations for the hierarchical model, using the results from part (a) as a starting distribution. Check your answer by comparing your simulations to the results in Figure 7.1b.