AMS 207: Intermediate Bayesian Modeling

8: Comparative MCMC and Monte Carlo Efficiency

David Draper

Department of Applied Mathematics and Statistics
University of California, Santa Cruz

draper@ams.ucsc.edu
http://www.ams.ucsc.edu/~draper

© 2010 David Draper (all rights reserved)
Simulation-Based Computation

Computation via conjugate analysis produces closed-form results (good) but is limited in scope to a fairly small set of models for which straightforward conjugate results are possible (bad).

This was a severe limitation for Bayesians for almost 250 years (from the 1750s to the 1980s).

Over the past 20 years the Bayesian community has “discovered” and developed an entirely new computing method, Markov chain Monte Carlo (MCMC) (“discovered” because the physicists first figured it out about 50 years ago: Metropolis and Ulam, 1949; Metropolis et al., 1953).

We’ve seen that the central Bayesian practical challenge is the computation of high-dimensional integrals.

People working on the first atom bomb in World War II faced a similar challenge, and noticed that digital computers (which were then passing from theory (Turing, 1943) to reality) offered an entirely new approach to solving the problem.

The idea (Metropolis and Ulam, 1949) was based on the observation that anything you want to know about a probability distribution (of arbitrary finite dimensionality) can be learned to arbitrary accuracy by sampling from it.

Suppose, for example, that you’re interested in a posterior distribution $p(\theta|y)$ which can’t be worked with (easily) in closed form, and initially (to keep things simple) think of $\theta$ as a scalar (real number) rather than vector.
Simulation-Based Computation

Four things of direct interest to you about \( p(\theta|y) \) would be

- its **mean** \( \mu = E(\theta|y) \) and **standard deviation** \( \sigma = \sqrt{V(\theta|y)} \),

- its **shape** (basically you’d like to be able to trace out (an estimate of) the entire **density curve**), and

- one or more of its **quantiles** (e.g., to construct a 95% central posterior interval for \( \theta \) you need to know the 2.5% and 97.5% **quantiles**), and sometimes the **posterior median** (the 50th **percentile**) is of interest too.

Suppose you could take an **arbitrarily large random sample** from \( p(\theta|y) \), say \( \theta^*_1, \ldots, \theta^*_m \).

Then each of the above four aspects of \( p(\theta|y) \) can be **estimated** from the \( \theta^* \) sample:

- \( \tilde{E}(\theta|y) = \overline{\theta}^* = \frac{1}{m} \sum_{j=1}^{m} \theta^*_j \),

- \( \sqrt{\tilde{V}(\theta|y)} = \sqrt{\frac{1}{m-1} \sum_{j=1}^{m} (\theta^*_j - \overline{\theta}^*)^2} \),

- the density curve can be estimated by a **histogram** or **kernel density estimate**, and

- **percentiles** can be estimated by **counting** how many of the \( \theta^* \) values fall below a series of specified points — e.g., to find an estimate of the 2.5% quantile you solve the equation

\[
\tilde{F}_\theta(t) = \frac{1}{m} \sum_{j=1}^{m} I(\theta^*_j \leq t) = 0.025
\]  

for \( t \), where \( I(A) \) is the **indicator function** (1 if \( A \) is true, otherwise 0).
IID Sampling; Rejection Sampling

These are called **Monte Carlo** estimates of the true summaries of $p(\theta | y)$ because they’re based on the **controlled use of chance**.

Theory (the **Law of Large Numbers**) shows that with large enough $m$, each of the Monte Carlo (or **simulation-based**) estimates can be made arbitrarily close to the truth with arbitrarily high probability, under some reasonable assumptions about the **nature of the random sampling**.

One way to achieve this, of course, is to make the sampling **IID** (this turns out to be **sufficient** but **not necessary** — see below).

If, for example, $\bar{\theta}^* = \frac{1}{m} \sum_{j=1}^{m} \theta_j^*$ is based on an IID sample of size $m$ from $p(\theta | y)$, we can use the **frequentist fact** that in repeated sampling $V(\bar{\theta}^*) = \frac{\sigma^2}{m}$, where (as above) $\sigma^2$ is the variance of $p(\theta | y)$, to construct a **Monte Carlo standard error** (MCSE) for $\bar{\theta}^*$:

$$\hat{SE}(\bar{\theta}^*) = \frac{\hat{\sigma}}{\sqrt{m}},$$

where $\hat{\sigma}$ is the **sample SD** of the $\theta^*$ values.

This can be used, possibly after some **preliminary experimentation**, to decide on $m$, the Monte Carlo sample size, which later we’ll call the length of the **monitoring run**.

**An IID example.** Consider the posterior distribution $p(\lambda | y) = \Gamma(29.001, 14.001)$ in the **length-of-stay example** in part 6 ($n = 14$ observations from a **Poisson distribution**).

We already know that the **posterior mean** of $\lambda$ in this example is $\frac{29.001}{14.001} = 2.071$; let’s see how well the Monte Carlo method does in estimating this **known truth**.
IID Example (continued)

Here's an R function to construct Monte Carlo estimates of the posterior mean and MCSE values for these estimates.

```r
gamma.sim <- function( m, alpha, beta, n.sim, seed ) {
  set.seed( seed )

  theta.out <- matrix( 0, n.sim, 2 )

  for ( i in 1:n.sim ) {
    theta.sample <- rgamma( m, alpha, 1 / beta )
    theta.out[ i, 1 ] <- mean( theta.sample )
    theta.out[ i, 2 ] <- sqrt( var( theta.sample ) / m )
  }

  return( theta.out )
}
```

This function simulates, n.sim times, the process of taking an IID sample of size m from the $\Gamma(\alpha,\beta)$ distribution and calculating $\bar{\theta}^*$ and $SE(\bar{\theta}^*)$.

sauternes 296> R

R : Copyright 2008, The R Development Core Team
Version 2.7.1 (2008-06-23)

> m <- 1000

> alpha <- 29.001

> beta <- 14.001

> n.sim <- 500

> seed <- c( 6425451, 9626954 )
IID Example (continued)

> theta.out <- gamma.sim( m, alpha, beta, n.sim, seed )

# This took about 1 second at 550 Unix MHz.

> theta.out[1:10,]

[,1]       [,2]
[1,] 2.082105  0.01166379
[2,] 2.072183  0.01200723
[3,] 2.066756  0.01247277
[4,] 2.060785  0.01200449
[5,] 2.078591  0.01212440
[6,] 2.050640  0.01228875
[7,] 2.071706  0.01182679
[8,] 2.063158  0.01176577
[9,] 2.058440  0.01186379
[10,] 2.068976  0.01220723

The $\bar{\theta}^*$ values fluctuate around the truth with a give-or-take of about 0.012, which agrees well with the theoretical SE $\frac{\sigma}{\sqrt{m}} = \frac{\sqrt{\alpha}}{\beta \sqrt{m}} \approx 0.01216$ (recall that the variance of a Gamma distribution is $\frac{\alpha}{\beta^2}$).

> postscript( "gamma-sim1.ps" )

> theta.bar <- theta.out[, 1]

> qqnorm( (theta.bar - mean(theta.bar)) / sqrt(var(theta.bar)) )

> abline(0, 1)

> dev.off()

null device

1

Each of the $\bar{\theta}^*$ values is the mean of $m = 1,000$ IID draws, so (by the CLT) the distribution of the random variable $\bar{\theta}^*$ should be closely approximated by a Gaussian.
Thus we can use frequentist ideas to work out how big \( m \) needs to be to have any desired Monte Carlo accuracy for \( \bar{\theta}^* \) as an estimate of the posterior mean \( E(\theta|y) \).

In practice, with \( p(\theta|y) \) unknown, you would probably take an initial sample (of size \( m = 1,000 \), say) and look at the MCSE to decide how big \( m \) really needs to be.
IID Example (continued)

> theta.bar <- gamma.sim( m, alpha, beta, 1, seed )

> theta.bar

[,1]    [,2]
[1,]  2.082105 0.01166379

(1) Suppose you wanted the MCSE of $\bar{\theta}^*$ to be (say) $\varepsilon = 0.001$. Then you could solve the equation

$$\frac{\hat{\sigma}}{\sqrt{m}} = \varepsilon \quad \leftrightarrow \quad m = \frac{\sigma^2}{\varepsilon^2},$$

which says (unhappily) that the required $m$ goes up as the square of the posterior SD and as the inverse square of $\varepsilon$.

The previous calculation shows that $\frac{\hat{\sigma}}{\sqrt{1000}} = 0.01166379$,
from which $\hat{\sigma} = 0.3688414$, meaning that to get $\varepsilon = 0.001$ you need a sample of size $\frac{0.3688414^2}{0.001^2} = 136,044 = 136k$ (!).

(2) Suppose instead that you wanted $\bar{\theta}^*$ to differ from the true posterior mean $\mu$ by no more than $\varepsilon_1$ with Monte Carlo probability at least $(1 - \varepsilon_2)$:

$$P(|\bar{\theta}^* - \mu| \leq \varepsilon_1) \geq 1 - \varepsilon_2,$$

where $P(\cdot)$ here is based on the (frequentist) Monte Carlo randomness inherent in $\bar{\theta}^*$.

We know from the CLT and the calculations above that in repeated sampling $\bar{\theta}^*$ is approximately normal with mean $\mu$ and variance $\sigma^2/m$; this leads to the inequality

$$m \geq \frac{\sigma^2 \left[ \Phi^{-1}(1 - \frac{\varepsilon_2}{2}) \right]^2}{\varepsilon_1^2},$$

where $\Phi^{-1}(q)$ is the place on the standard normal curve where $100q\%$ of the area is to the left of that place (the $q$th quantile of the standard normal distribution).
A Closer Look at IID Sampling

(5) is like (3) except that the value of \( m \) from (3) has to be multiplied by \( [\Phi^{-1}(1 - \frac{\epsilon_2}{2})]^2 \), which typically makes the required sample sizes even bigger.

For example, with \( \epsilon_1 = 0.001 \) and \( \epsilon_2 = 0.05 \) — i.e., to have at least 95% Monte Carlo confidence that reporting the posterior mean as 2.071 will be correct to about four significant figures — (5) says that you would need a monitoring run of at least 136,044(1.959964)^2 \approx 522,608 \div 523k \).

(On the other hand, this sounds like a long monitoring run but only takes about \textit{0.8 seconds} at 1.6 Unix GHz on a Sun Ultra 45, yielding \([\hat{\theta}^*, \hat{SE}(\hat{\theta}^*)] = (2.0709, 0.00053)\).

It’s evident from calculations like these that people often report simulation-based answers with numbers of significant figures far in excess of what’s justified by the actual accuracy of the Monte Carlo estimates.

\textbf{A Closer Look at IID Sampling.} I was able to easily perform the above simulation study because \texttt{R} has a large variety of built-in functions like \texttt{rgamma} for pseudo-random-number generation.

How would you go about writing such functions yourself?

There are a number of \textit{general-purpose} methods for generating random numbers (I won’t attempt a survey here); the one we need to look closely at, to understand the algorithms that arise later in this section, is rejection sampling (von Neumann, 1951), which is often one of the most \textit{computationally efficient} ways to make IID draws from a distribution.
Rejection Sampling

**Example.** In the spring of 1993 a survey was taken of bicycle and other traffic in the vicinity of the University of California, Berkeley, campus (Gelman, Carlin et al. 2003).

As part of this survey 10 city blocks on residential streets with bike routes were chosen at random from all such blocks at Berkeley; on one of those blocks $n$ vehicles were observed on a randomly chosen Tuesday afternoon from 3 to 4pm, and $s$ of them were bicycles.

To draw inferences about the underlying proportion $\theta$ of bicycle traffic (PBT) on blocks similar to this one at times similar to Tuesday afternoons from 3 to 4pm, it’s natural (as in the AMI mortality case study) to employ the model

$$
\begin{cases}
\theta \sim \text{Beta}(\alpha_0, \beta_0) \\
(S|\theta) \sim \text{Binomial}(n, \theta)
\end{cases} \rightarrow (\theta|s) \sim \text{Beta}(\alpha_0 + s, \beta_0 + n - s),
$$

(6)

provided that whatever prior information I have about $\theta$ can be meaningfully captured in the Beta family.

After reflection I realize that I’d be quite surprised if the PBT in residential city blocks with bike routes in Berkeley on Tuesday afternoons from 3 to 4pm was less than 5% or greater than 50%.

Making this operational by assuming that in the prior $p(0.05 \leq \theta \leq 0.5) = 0.9$, and putting half of the remaining prior probability in each of the left and right tails of the Beta distributions, yields $(\alpha_0, \beta_0) = (2.0, 6.4)$ (this Beta distribution has prior mean and SD 0.24 and 0.14, respectively).

In the city block in question the data came out $(n, s) = (74, 16)$, so that the data mean was 0.216, and the posterior is then $\text{Beta}(\alpha_0 + s, \beta_0 + n - s) = \text{Beta}(18.0, 64.4)$. 

Rejection Sampling (continued)

Pretend for the sake of illustration of rejection sampling that you didn’t know the formulas for the mean and SD of a Beta distribution, and suppose that you wanted to use IID Monte Carlo sampling from the Beta($\alpha_0 + s, \beta_0 + n - s$) posterior to estimate the posterior mean.

Here’s von Neumann’s basic idea: suppose the target density $p(\theta|y)$ is difficult to sample from, but you can find an integrable envelope function $G(\theta|y)$ such that (a) $G$ dominates $p$ in the sense that $G(\theta|y) \geq p(\theta|y) \geq 0$ for all $\theta$ and (b) the density $g$ obtained by normalizing $G$ — later to be called the proposal distribution — is easy and fast to sample from.

Then to get a random draw from $p$, make a draw $\theta^*$ from $g$ instead and accept or reject it according to an acceptance probability $\alpha_R(\theta^*|y)$; if you reject the draw, repeat this process until you accept.

von Neumann showed that the choice

$$\alpha_R(\theta^*|y) = \frac{p(\theta^*|y)}{G(\theta^*|y)}$$  \hspace{1cm} (7)

correctly produces IID draws from $p$, and you can intuitively see that he’s right by the following argument.
Rejection Sampling (continued)

Making a draw from the posterior distribution of interest is like choosing a point at random (in two dimensions) under the density curve \( p(\theta|y) \) in such a way that all possible points are equally likely, and then writing down its \( \theta \) value.

If you instead draw from \( G \) so that all points under \( G \) are equally likely, to get correct draws from \( p \) you'll need to throw away any point that falls between \( p \) and \( G \), and this can be accomplished by accepting each sampled point \( \theta^* \) with probability \( \frac{p(\theta^*|y)}{G(\theta^*|y)} \), as von Neumann said.

**A summary** of this method is as follows.

**Algorithm** (rejection sampling). To make \( m \) IID draws at random from the density \( p(\theta|y) \) for real-valued \( \theta \), select an integrable **envelope function** \( G \) — which when normalized to integrate to 1 is the proposal **distribution** \( g \) — such that \( G(\theta|y) \geq p(\theta|y) \geq 0 \) for all \( \theta \); define the acceptance probability \( \alpha_R(\theta^*|y) = \frac{p(\theta^*|y)}{G(\theta^*|y)} \); and

\[
\text{Initialize } t \leftarrow 0
\]

Repeat
\[
\begin{align*}
\text{Sample } \theta^* & \sim g(\theta|y) \\
\text{Sample } u & \sim \text{Uniform}(0,1) \\
\text{If } u & \leq \alpha_R(\theta^*|y) \text{ then} \\
\{} & \theta_{t+1} \leftarrow \theta^*; \ t \leftarrow (t + 1) \ \\
\end{align*}
\]

until \( t = m \).

(Choosing a \( \theta^* \) from \( g \) locates the horizontal coordinate according to \( G \); choosing a \( u \) as above is equivalent to **picking a point at random vertically** on the line segment from \((\theta^*,0)\) to \((\theta^*,G(\theta^*))\) and seeing whether it's **below** \( p \) or not.)
Rejection Sampling (continued)

The figure below demonstrates this method on the Beta(18.0, 64.4) density arising in the Beta-Bernoulli example above.

Rejection sampling permits considerable flexibility in the choice of envelope function; here, borrowing an idea from Gilks and Wild (1992), I’ve noted that the relevant Beta density is log concave (a real-valued function is log concave if its second derivative on the log scale is everywhere non-positive), meaning that it’s easy to construct an envelope on that scale in a piecewise linear fashion, by choosing points on the log density and constructing tangents to the curve at those points.

The simplest possible such envelope involves two line segments, one on either side of the mode.

The optimal choice of the tangent points would maximize the marginal probability of acceptance of a draw in the rejection algorithm, which can be shown to be

$$\left[ \int G(\theta) \, d\theta \right]^{-1};$$

(9)
Rejection Sampling (continued)

in other words, you should **minimize** the area under the (un-normalized) envelope function subject to the constraint that it **dominates** the target density \( p(\theta|y) \) (which makes eminently good sense).

Here this optimum turns out to be attained by locating the two tangent points at about 0.17 and 0.26, as in the figure above; the resulting acceptance probability of about 0.75 could clearly be **improved** by adding more tangents.

**Piecewise linear** envelope functions on the log scale are a **good choice** because the resulting envelope density on the raw scale is a piecewise set of **scaled exponential distributions** (see the bottom panel in the figure above), from which random samples can be taken **quickly**.

A **preliminary** sample of \( m_0 = 500 \) IID draws from the Beta(18.0, 64.4) distribution using the above rejection sampling method yields \( \bar{\theta}^* = 0.2197 \) and \( \hat{\sigma} = 0.04505 \), meaning that the posterior mean has already been estimated with an **MCSE** of only \( \frac{\hat{\sigma}}{\sqrt{m_0}} = 0.002 \) even with just 500 draws.

Suppose, however, that — as in equation (4) above — I want \( \bar{\theta}^* \) to **differ** from the true posterior mean \( \mu \) by no more than some (perhaps even smaller) **tolerance** \( \epsilon_1 \) with Monte Carlo probability at least \( (1 - \epsilon_2) \); then equation (5) tells me how long to **monitor** the simulation output.

For instance, to pin down **three significant figures** (sigfigs) in the posterior mean in this example with high Monte Carlo accuracy I might take \( \epsilon_1 = 0.0005 \) and \( \epsilon_2 = 0.05 \), which yields a **recommended IID sample size** of \( \frac{(0.04505^2)(1.96)^2}{0.0005^2} \approx 31,200 \).
Rejection Sampling (continued)

So I take another sample of 30,700 (which is virtually instantaneous at 1.6 Unix GHz) and merge it with the 500 draws I already have; this yields $\tilde{\theta}^* = 0.21827$ and $\tilde{\sigma} = 0.04528$, meaning that the MCSE of this estimate of $\mu$ is $\frac{0.04528}{\sqrt{31200}} \approx 0.00026$.

I might announce that I think $E(\theta|y)$ is about 0.2183, give or take about 0.0003, which accords well with the true value 0.2184.

Of course, other aspects of $p(\theta|y)$ are equally easy to monitor; for example, if I want a Monte Carlo estimate of $p(\theta \leq q|y)$ for some $q$, as noted above I just work out the proportion of the sampled $\theta^*$ values that are no larger than $q$.

Or, even better, I recall that $P(A) = E[I(A)]$ for any event or proposition $A$, so to the Monte Carlo dataset (see p. 26 below) consisting of 31,200 rows and one column (the $\theta^*_t$) I add a column monitoring the values of the derived variable which is 1 whenever $\theta^*_t \leq q$ and 0 otherwise; the mean of this derived variable is the Monte Carlo estimate of $p(\theta \leq q|y)$, and I can attach an MCSE to it in the same way I did with $\tilde{\theta}^*$.

By this approach, for instance, the Monte Carlo estimate of $p(\theta \leq 0.15|y)$ based on the 31,200 draws examined above comes out $\hat{p} = 0.0556$ with an MCSE of 0.0013.

Percentiles are typically harder to pin down with equal Monte Carlo accuracy (in terms of sigfigs) than means or SDs, because the 0/1 scale on which they’re based is less information-rich than the $\theta^*$ scale itself; if I wanted an MCSE for $\hat{p}$ of 0.0001 I would need an IID sample of more than 5 million draws (which would still only take a few seconds at contemporary workstation speeds).
Beyond Rejection Sampling

**IID sampling is not necessary.** Nothing in the Metropolis-Ulam idea of Monte Carlo estimates of posterior summaries requires that these estimates be based on IID samples from the posterior.

This is lucky, because in practice it's often difficult, particularly when \( \theta \) is a **vector of high dimension** (say \( k \)), to figure out how to make such an IID sample, via rejection sampling or other methods (e.g., imagine trying to find an envelope function for \( p(\theta|y) \) when \( k \) is 10 or 100 or **1,000**).

Thus it's necessary to relax the assumption that \( \theta_j^* \overset{\text{IID}}{\sim} p(\theta|y) \), and to consider samples \( \theta_1^*, \ldots, \theta_m^* \) that form a **time series**: a series of draws from \( p(\theta|y) \) in which \( \theta_j^* \) may depend on \( \theta_j^* \) for \( j' < j \).

In their pioneering paper Metropolis et al. (1953) allowed for **serial dependence** of the \( \theta_j^* \) by combining von Neumann's idea of rejection sampling (which had itself only been published a few years earlier in 1951) with concepts from **Markov chains**, a subject in the theory of stochastic processes.

Combining **Monte Carlo sampling** with **Markov chains** gives rise to the name now used for this technique for solving the Bayesian high-dimensional integration problem: **Markov chain Monte Carlo** (MCMC).
Metropolis Sampling

Given a parameter vector $\theta$ and a data vector $y$, the Metropolis et al. (1953) idea is to simulate random draws from the posterior distribution $p(\theta|y)$, by constructing a Markov chain with the following three properties:

- It should have the same state space as $\theta$,
- It should be easy to simulate from, and
- Its equilibrium distribution should be $p(\theta|y)$.

If you can do this, you can run the Markov chain for a long time, generating a huge sample from the posterior, and then use simple descriptive summaries (means, SDs, correlations, histograms or kernel density estimates) to extract any features of the posterior you want.

There is a fourth desirable condition as well:

- It should not be necessary to work out the normalizing constant for $p(\theta|y)$ to implement the algorithm, which is equivalent to saying that $p(\theta|y)$ should appear in the calculations only through ratios of the form $\frac{p(\theta|y)}{p(\theta'|y)}$. 
The Ergodic Theorem

The mathematical fact that underpins this strategy is the **ergodic theorem**: if the Markov chain \( \{\theta^*_i\} \) is ergodic and \( f \) is any real-valued function for which \( E_\pi[f(\theta)] \) is finite, then with probability 1 as \( m \to \infty \)

\[
\frac{1}{m} \sum_{i=1}^{m} f(\theta^*_i) \to E_\pi[f(\theta)] = \sum_i f(i) \pi(i),
\]

(10)

in which the right side is just the **expectation** of \( f(\theta) \) under the stationary distribution \( \pi \).

In plain English this means that—as long as the stationary distribution is \( p(\theta|y) \)—you can learn (to arbitrary accuracy) about things like posterior means, SDs, and so on just by **waiting for stationarity to kick in** and monitoring thereafter for a long enough period.

Of course, as Roberts (1996) notes, the theorem is **silent** on the two key practical questions it raises: **how long you have to wait** for stationarity, and **how long to monitor** after that.

A third practical issue is what to use for the **initial value** \( \theta_0^* \): intuitively the **closer** \( \theta_0^* \) is to the **center** of \( p(\theta|y) \) the **less time** you should have to wait for stationarity.

The standard way to deal with **waiting for stationarity** is to (a) run the chain from a **good starting value** \( \theta_0^* \) for \( b \) iterations, until **equilibrium** has been reached, and (b) **discard** this initial **burn-in** period.

All of this motivates the topic of **MCMC diagnostics**, which are intended to answer the following questions:

- **What should I use for the initial value** \( \theta_0^* \)?

- **How do I know when I’ve reached equilibrium?** (This is equivalent to asking **how big** \( b \) should be.)

- **Once I’ve reached equilibrium, how big should** \( m \) **be**, i.e., **how long should I monitor the chain** to get posterior summaries with **decent accuracy**?
The Monte Carlo and MCMC Datasets

The basis of the Monte Carlo approach to obtaining numerical approximations to posterior summaries like means and SDs is the (weak) Law of Large Numbers: with IID sampling the Monte Carlo estimates of the true summaries of $p(\theta|y)$ are consistent, meaning that they can be made arbitrarily close to the truth with arbitrarily high probability as the number of monitoring iterations $m \to \infty$.

Before we look at how Metropolis et al. attempted to achieve the same goal with a non-IID Monte Carlo approach, let's look at the practical consequences of switching from IID to Markovian sampling.

Running the IID rejection sampler on the Berkeley PBT example above for a total of $m$ monitoring iterations would produce something that might be called the Monte Carlo dataset, with one row for each iteration and one column for each monitored quantity; in that example it might look like this (MCSEs in parenthesis):

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$\theta$</th>
<th>$I(\theta \leq 0.15)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\theta_1^* = 0.244$</td>
<td>$I_1^* = 0$</td>
</tr>
<tr>
<td>2</td>
<td>$\theta_2^* = 0.137$</td>
<td>$I_2^* = 1$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$m = 31,200$</td>
<td>$\theta_m^* = 0.320$</td>
<td>$I_m^* = 0$</td>
</tr>
</tbody>
</table>

| Mean      | 0.2183 (0.003) | 0.0556 (0.0013) |
| SD        | 0.04528      | —                |
| Density   | (like the bottom plot on p. 13) | — |

Running the Metropolis sampler on the same example would produce something that might be called the MCMC dataset.

It would have a similar structure as far as the columns are concerned, but the rows would be divided into three phases:
The MCMC Dataset (continued)

- Iteration 0 would be the value(s) used to **initialize** the Markov chain;

- Iterations 1 through \( b \) would be the **burn-in** period, during which the chain reaches its **equilibrium** or **stationary** distribution (as mentioned above, iterations 0 through \( b \) are generally **discarded**); and

- Iterations \((b + 1)\) through \((b + m)\) would be the **monitoring** run, on which **summaries** of the posterior (means, SDs, density traces, ...) will be based.

In the Berkeley PBT example the **MCMC dataset** might look like this:

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Phase</th>
<th>( \theta )</th>
<th>( I(\theta \leq 0.15) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Initialization</td>
<td>( \theta_0^* = 0.200 )</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>Burn-in</td>
<td>( \theta_1^* = 0.244 )</td>
<td>—</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( b = 500 )</td>
<td>Burn-in</td>
<td>( \theta_b^* = 0.098 )</td>
<td>—</td>
</tr>
<tr>
<td>( (b + 1) = 501 )</td>
<td>Monitoring</td>
<td>( \theta_{b+1}^* = 0.275 )</td>
<td>( I_{b+1}^* = 0 )</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( (b + m) = 31,700 )</td>
<td>Monitoring</td>
<td>( \theta_{b+m}^* = 0.120 )</td>
<td>( I_{b+m}^* = 1 )</td>
</tr>
</tbody>
</table>

| Mean      | (Monitoring | 0.2177 (0.009) | 0.0538 (0.004) |
| SD        | Phase      | 0.04615        | —                |
| Density   | Only)      | (like the bottom plot on p. 14) | —                |

Think of iteration number \( i \) in the Monte Carlo sampling process as a discrete **index of time** \( t \), so that the columns of the MC and MCMC datasets can be viewed as **time series**.

An important concept from time series analysis is **autocorrelation**: the autocorrelation \( \rho_k \) of a **stationary** time series \( \theta_t^* \) at lag \( k \) (see, e.g., Chatfield (1996)) is \( \gamma_k \), where \( \gamma_k \) is \( C(\theta_t^*, \theta_{t-k}^*) \), the covariance of the series with itself \( k \) iterations in the past—this measures the degree to which the time series at any given moment **depends on its past history**.
The MCMC Dataset (continued)

IID draws from $p(\theta|y)$ correspond to white noise: a time series with zero autocorrelations at all lags.

This is the behavior of the columns in the MC data set on p. 19, produced by ordinary rejection sampling.

Because of the Markov character of the columns of the MCMC data set on p. 20, each column, when considered as a time series, will typically have non-zero autocorrelations, and because Markov chains use their present values to decide where to go next it shouldn’t surprise you to hear that the typical behavior will be (substantial) positive autocorrelations—in other words, every time you get another draw from the Markov chain you get some new information about the posterior and a rehash of old information mixed in.

It’s a marvelous result from time series analysis (the Ergodic Theorem for Markov chains on p. 18 is an example of this fact) that all of the usual descriptive summaries of the posterior are still consistent as long as the columns of the MCMC data set form stationary time series.

In other words, provided that you can achieve the four goals back on p. 17 which Metropolis et al. set for themselves, and provided that you only do your monitoring after the Markov chain has reached equilibrium, the MCMC approach and the IID Monte Carlo approach are equally valid (they both get the right answers), but they may well differ on their efficiency (the rate per iteration, or per CPU second, at which they learn about the posterior may not be the same); and if, as is typically true, the columns of the MCMC dataset have positive autocorrelations, this will translate into slower learning (larger MCSEs) than with IID sampling (compare the MCSEs on pages 26 and 27).
The Metropolis Algorithm

Metropolis et al. were able to create what people would now call a **successful MCMC algorithm** by the following means (see the excellent book edited by Gilks et al. (1996) for many more details about the MCMC approach).

Consider the **rejection sampling method** given above in (8) as a mechanism for generating realizations of a time series (where as above time indexes iteration number).

At any time $t$ in this process you make a draw $\theta^*$ from the **proposal distribution** $g(\theta|y)$ (the normalized version of the envelope function $G$) and either accept a **move** to $\theta^*$ or reject it, according to the **acceptance probability** $\frac{p(\theta^*|y)}{G(\theta^*|y)}$; if accepted the process moves to $\theta^*$, if not you **draw again** and discard the rejected draws until you do make a successful move.

As noted above, the stochastic process thus generated is an **IID (white noise)** series of draws from the **target distribution** $p(\theta|y)$.

Metropolis et al. had the following **beautifully simple idea** for how this may be generalized to situations where IID sampling is difficult: **they allowed the proposal distribution at time $t$ to depend on the current value $\theta_t$ of the process**, and then—to get the right stationary distribution—if a proposed move is rejected, instead of discarding it **the process is forced to stay where it is for one iteration before trying again**.

The resulting process is a **Markov chain**, because (a) the draws are now dependent but (b) all you need to know in determining where to go next is **where you are now**.
Metropolis-Hastings

Letting $\theta_t$ stand for where you are now and $\theta^*$ for where you’re thinking of going, in this approach there is enormous flexibility in the choice of the proposal distribution $g(\theta^*|\theta_t, y)$, even more so than in ordinary rejection sampling.

The original Metropolis et al. idea was to work with symmetric proposal distributions, in the sense that $g(\theta^*|\theta_t, y) = g(\theta_t|\theta^*, y)$, but Hastings (1970) pointed out that this could easily be generalized; the resulting method is the Metropolis-Hastings (MH) algorithm.

Building on the Metropolis et al. results, Hastings showed that you’ll get the correct stationary distribution $p(\theta|y)$ for your Markov chain by making the following choice for the acceptance probability:

$$\alpha_{MH}(\theta^*|\theta_t, y) = \min \left\{ 1, \frac{p(\theta^*|y)}{g(\theta^*|\theta_t, y)} \frac{g(\theta_t|y)}{g(\theta_t|\theta^*, y)} \right\}. \quad (11)$$

It turns out that the proposal distribution $g(\theta^*|\theta_t, y)$ can be virtually anything and you’ll get the right equilibrium distribution using the acceptance probability (11); see, e.g., Roberts (1996) and Tierney (1996) for the mild regularity conditions necessary to support this statement.

A summary of the method is on the next page.

It’s instructive to compare (12) with (8) to see how heavily the MH algorithm borrows from ordinary rejection sampling, with the key difference that the proposal distribution is allowed to change over time.

Notice how (11) generalizes von Neumann’s acceptance probability ratio $\frac{p(\theta^*|y)}{G(\theta^*|y)}$ for ordinary rejection sampling: the crucial part of the new MH acceptance probability becomes the ratio of two von-Neumann-like ratios, one for where you are now and one for where you’re thinking of going (it’s equivalent to work with $g$ or $G$ since the normalizing constant cancels in the ratio).
**Algorithm** (Metropolis-Hastings sampling). To construct a Markov chain whose equilibrium distribution is \( p(\theta | y) \), choose a proposal distribution \( g(\theta^* | \theta_t, y) \), define the acceptance probability \( \alpha_{MH} (\theta^* | \theta_t, y) \) by (11), and

\[
\text{Initialize } \theta_0; \ t \leftarrow 0 \\
\text{Repeat } \{ \\
\quad \text{Sample } \theta^* \sim g(\theta | \theta_t, y) \\
\quad \text{Sample } u \sim \text{Uniform}(0, 1) \\
\quad \text{If } u \leq \alpha_{MH} (\theta^* | \theta_t, y) \text{ then } \theta_{t+1} \leftarrow \theta^* \\
\quad \text{else } \theta_{t+1} \leftarrow \theta_t \\
\quad t \leftarrow (t + 1)
\}
\tag{12}
\]

When the proposal distribution is symmetric in the Metropolis et al. sense, the acceptance probability ratio reduces to \( \frac{p(\theta^* | y)}{p(\theta_t | y)} \), which is easy to motivate intuitively: whatever the target density is at the current point \( \theta_t \), you want to visit points of higher density more often and points of lower density less often, and it turns out that (11) does this for you in the natural and appropriate way.

As an example of the MH algorithm in action, consider a Gaussian model with known mean \( \mu \) and unknown variance \( \sigma^2 \) applied to the NB10 data in part 6 of the lecture notes.

The likelihood function for \( \sigma^2 \), derived from the sampling model \( (Y_i | \sigma^2) \overset{\text{IID}}{\sim} N(\mu, \sigma^2) \) for \( i = 1, \ldots, n \), is

\[
l(\sigma^2 | y) = c \prod_{i=1}^{n} (\sigma^2)^{-\frac{n}{2}} \exp \left[ -\frac{(y_i - \mu)^2}{2\sigma^2} \right] \\
= c (\sigma^2)^{-\frac{n}{2}} \exp \left[ -\frac{\sum_{i=1}^{n} (y_i - \mu)^2}{2\sigma^2} \right]. \tag{13}
\]
MH Sampling (continued)

This is recognizable as a member of the **Scaled Inverse $\chi^2$ family $\chi^{-2}(\nu, s^2)$** (e.g., Gelman, Carlin et al. (2003)) of distributions, which (as we saw in part 2 of the lecture notes) is a **rescaled version** of the Inverse Gamma family chosen so that $s^2$ is an estimate of $\sigma^2$ based upon $\nu$ “observations.”

You can now convince yourself that if the **prior** for $\sigma^2$ in this model is taken to be $\chi^{-2}(\nu, s^2)$, then the **posterior** for $\sigma^2$ will also be Scaled Inverse $\chi^2$: with this choice of prior

$$p(\sigma^2|y) = \chi^{-2}\left[\nu + n, \frac{\nu s^2 + \sum_{i=1}^{n} (y_i - \mu)^2}{\nu + n}\right]. \quad (14)$$

This makes **good intuitive sense**: the prior estimate $s^2$ of $\sigma^2$ receives $\nu$ votes and the sample estimate $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} (y_i - \mu)^2$ receives $n$ votes in the posterior weighted average estimate $\frac{\nu s^2 + n \hat{\sigma}^2}{\nu + n}$.

Equation (14) provides a satisfying **closed-form solution** to the Bayesian updating problem in this model (e.g., it’s easy to compute posterior moments **analytically**, and you can use numerical integration or well-known approximations to the CDF of the Gamma distribution to compute percentiles).

For **illustration purposes** suppose instead that you want to use **MH sampling** to summarize this posterior.

Then your main **choice** as a user of the algorithm is the specification of the **proposal distribution** (PD) $g(\sigma^2|\sigma_i^2, y)$.

The goal in choosing the PD is getting a chain that **mixes well** (moves freely and fluidly among all of the possible values of $\theta = \sigma^2$), and nobody has (yet) come up with a **sure-fire strategy** for always succeeding at this task.

Having said that, here are **two basic ideas** that often tend to **promote good mixing**:
MH Sampling (continued)

(1) Pick a PD that looks like a somewhat overdispersed version of the posterior you’re trying to sample from (e.g., Tierney (1996)).

Some work is naturally required to overcome the circularity inherent in this choice (if I fully knew \( p(\theta|y) \) and all of its properties, why would I be using this algorithm in the first place?).

(2) Set up the PD so that the expected value of where you’re going to move to \((\theta^*)\), given that you accept a move away from where you are now \((\theta_t)\), is to stay where you are now: \( E_g(\theta^*|\theta_t, y) = \theta_t \).

That way, when you do make a move, there will be an approximate left-right balance, so to speak, in the direction you move away from \( \theta_t \), which will encourage rapid exploration of the whole space.

Using idea (1), a decent choice for the PD in the Gaussian model with unknown variance might well be the Scaled Inverse \( \chi^2 \) distribution: \( g(\sigma^2|\sigma_t^2, y) = \chi^{-2}(\nu_*, \sigma_*^2) \).

This distribution has mean \( \frac{\nu_*}{\nu_*-2} \sigma_*^2 \) for \( \nu_* > 2 \).

To use idea (2), then, I can choose any \( \nu_* \) greater than 2 that I want, and as long as I take \( \sigma_*^2 = \frac{\nu_*-2}{\nu_*} \sigma_t^2 \) that will center the PD at \( \sigma_t^2 \) as desired.

So I’ll use

\[
g(\sigma^2|\sigma_t^2, y) = \chi^{-2}(\nu_*, \frac{\nu_*-2}{\nu_*} \sigma_t^2). \tag{15}
\]

This leaves \( \nu_* \) as a kind of potential tuning constant—the hope is that I can vary \( \nu_* \) to improve the mixing of the chain.
The above figure (motivated by an analogous plot in Gilks et al. (1996)) presents time series traces of some typical output of the MH sampler with $\nu_* = (2.5, 20, 500)$.

The acceptance probabilities with these values of $\nu_*$ are $(0.07, 0.44, 0.86)$, respectively.

The SD of the $\chi^{-2} \left( \nu_*, \frac{\nu_* - 2}{\nu_*} \sigma_t^2 \right)$ distribution is proportional to $\frac{\nu_*^2}{(\nu_*^2 - 2)^{\frac{3}{2}} \sqrt{\nu_* - 4}}$, which decreases as $\nu_*$ increases, and this turns out to be crucial: when the proposal distribution SD is too large (small $\nu_*$, as in the top panel in the figure), the algorithm tries to make big jumps around $\theta$ space (good), but almost all of them get rejected (bad), so there are long periods of no movement at all, whereas when the PD SD is too small (large $\nu_*$; see the bottom panel of the figure), the algorithm accepts most of its proposed moves (good), but they’re so tiny that it takes a long time to fully explore the space (bad).
MH Sampling (continued)

Gelman, Roberts, et al. (1995) have shown that in simple problems with approximately normal target distributions, the optimal acceptance rate for MH samplers like the one illustrated here is about 44% when the vector of unknowns is one-dimensional, and this can serve as a rough guide: you can modify the proposal distribution SD until the acceptance rate is around the Gelman et al. target figure.

The central panel of the figure displays the best possible MH behavior in this problem in the family of PDs chosen.

Even with this optimization you can see that the mixing is not wonderful, but contemporary computing speeds enable huge numbers of draws to be collected in a short period of time, compensating for the comparatively slow rate at which the MH algorithm learns about the posterior distribution of interest.

In this example the unknown quantity \( \theta = \sigma^2 \) was real-valued, but there's nothing in the MH method that requires this; in principle it works equally well when \( \theta \) is a vector of any finite dimension (look back at the algorithm in (12) to verify this).

Notice, crucially, that to implement this algorithm you only need to know how to calculate \( p(\theta | y) \) up to a constant multiple, since any such constant will cancel in computing the acceptance probability (11) — thus you’re free to work with unnormalized versions of \( p(\theta | y) \), which is a great advantage in practice.
MH Sampling (continued)

There’s even more flexibility in this algorithm than might first appear: it’s often possible to identify a set $A$ of auxiliary variables—typically these are latent (unobserved) quantities—to be sampled along with the parameters, which have the property that they improve the mixing of the MCMC output (even though extra time is spent in sampling them).

When the set $(\theta, A)$ of quantities to be sampled is a vector of length $k$, there is additional flexibility: you can block update all of $(\theta, A)$ at once, or with appropriate modifications of the acceptance probability you can divide $(\theta, A)$ up into components, say $(\theta, A) = (\lambda_1, \ldots, \lambda_i)$, and update the components one at a time (as Metropolis et al. originally proposed in 1953).

The idea in this component-by-component version of the algorithm, which Gilks et al. (1996) call single-component MH sampling, is to have $k$ different proposal distributions, one for each component of $\theta$.

Each iteration of the algorithm (indexed as usual by $t$) has $k$ steps, indexed by $i$; at the beginning of iteration $t$ you scan along, updating $\lambda_1$ first, then $\lambda_2$, and so on until you’ve updated $\lambda_k$, which concludes iteration $t$.

Let $\lambda_{t,i}$ stand for the current state of component $i$ at the end of iteration $t$, and let $\lambda_{-i}$ stand for the $(\theta, A)$ vector with component $i$ omitted (the notation gets awkward here; it can’t be helped).

The proposal distribution $g_i(\lambda^*_i | \lambda_{t,i}, \lambda_{t,-i}, y)$ for component $i$ is allowed to depend on the most recent versions of all components of $(\theta, A)$; here $\lambda_{t,-i}$ is the current state of $\lambda_{-i}$ after step $(i-1)$ of iteration $t$ is finished, so that components 1 through $(i-1)$ have been updated but not the rest.
Gibbs Sampling

The acceptance probability for the proposed move to $\lambda_i^*$ that creates the correct equilibrium distribution turns out to be

$$\alpha_{MH}(\lambda_i^*|\lambda_{t,-i}, \lambda_{t,i}, y) = \min \left[ 1, \frac{p(\lambda_i^*|\lambda_{t,-i}, y) g_i(\lambda_{t,i}|\lambda_i^*, \lambda_{t,-i}, y)}{p(\lambda_{t,i}|\lambda_{t,-i}, y) g_i(\lambda_i^*|\lambda_{t,i}, \lambda_{t,-i}, y)} \right].$$

(16)

The distribution $p(\lambda_i|\lambda_{-i}, y)$ appearing in (16), which is called the full conditional distribution for $\lambda_i$, has a natural interpretation: it represents the posterior distribution for the relevant portion of $(\theta, A)$ given $y$ and the rest of $(\theta, A)$.

The full conditional distributions act like building blocks in constructing the complete posterior distribution $p(\theta|y)$, in the sense that any multivariate distribution is uniquely determined by its set of full conditionals (Besag (1974)).

An important special case of single-component MH sampling arises when the proposal distribution $g_i(\lambda_i^*|\lambda_{t,i}, \lambda_{t,-i}, y)$ for component $i$ is chosen to be the full conditional $p(\lambda_i^*|\lambda_{t,-i}, y)$ for $\lambda_i$: you can see from (16) that when this choice is made a glorious cancellation occurs and the acceptance probability is 1.

This is Gibbs sampling, independently (re)discovered by Geman and Geman (1984): the Gibbs recipe is to sample from the full conditionals and accept all proposed moves.

Even though it's just a version of MH, Gibbs sampling is important enough to merit a summary of its own.

**Single-element** Gibbs sampling, in which each real-valued coordinate $(\theta_1, \ldots, \theta_k)$ gets updated in turn, is probably the most frequent way Gibbs sampling gets used, so that's what I'll summarize ((17) details Gibbs sampling in the case with no auxiliary variables $A$, but the algorithm works equally well when $\theta$ is replaced by $(\theta, A)$ in the summary).
Gibbs Sampling (continued)

**Algorithm** (Single-element Gibbs sampling). To construct a Markov chain whose equilibrium distribution is \( p(\theta | y) \) with \( \theta = (\theta_1, \ldots, \theta_k) \),

\[
\text{Initialize } \theta_{0,1}^*, \ldots, \theta_{0,k}^*; \ t \leftarrow 0 \\
\text{Repeat } \{ \\
\quad \text{Sample } \theta_{t+1,1}^* \sim p(\theta_1 | y, \theta_{t,2}^*, \theta_{t,3}^*, \theta_{t,4}^*, \ldots, \theta_{t,k}^*) \\
\quad \text{Sample } \theta_{t+1,2}^* \sim p(\theta_2 | y, \theta_{t+1,1}^*, \theta_{t,3}^*, \theta_{t,4}^*, \ldots, \theta_{t,k}^*) \\
\quad \text{Sample } \theta_{t+1,3}^* \sim p(\theta_3 | y, \theta_{t+1,1}^*, \theta_{t+1,2}^*, \theta_{t,4}^*, \ldots, \theta_{t,k}^*) \\
\quad \vdots \\
\quad \text{Sample } \theta_{t+1,k}^* \sim p(\theta_k | y, \theta_{t+1,1}^*, \theta_{t+1,2}^*, \theta_{t,4}^*, \ldots, \theta_{t+1,k-1}^*) \\
\quad t \leftarrow (t + 1) \\
\}
\]

(17)

**Example:** the NB10 Data. Recall from the posterior predictive plot toward the end of part 2 of the lecture notes that the Gaussian model for the NB10 data was inadequate: the tails of the data distribution are too heavy for the Gaussian.

It was also clear from the normal qqplot that the data are symmetric.

This suggests thinking of the NB10 data values \( y_i \) as like draws from a \( t \) distribution with fairly small degrees of freedom \( \nu \).

**One way** to write this model is

\[
(\mu, \sigma^2, \nu) \sim p(\mu, \sigma^2, \nu) \\
(y_i | \mu, \sigma^2, \nu) \overset{\text{IID}}{\sim} t_\nu(\mu, \sigma^2),
\]

(18)

where \( t_\nu(\mu, \sigma^2) \) denotes the scaled \( t \)-distribution with mean \( \mu \), scale parameter \( \sigma^2 \), and shape parameter \( \nu \).
Model Expansion

This distribution has variance \( \sigma^2 \left( \frac{\nu}{\nu - 2} \right) \) for \( \nu > 2 \) (so that shape and scale are mixed up, or **confounded** in \( t_\nu(\mu, \sigma^2) \)) and may be thought of as the distribution of the quantity \( \mu + \sigma e \), where \( e \) is a draw from the **standard** \( t \) distribution that is tabled at the back of all introductory statistics books.

However, a **better way** to think about model (18) is as follows.

It’s a fact from **basic distribution theory**, probably of more interest to Bayesians than frequentists, that the \( t \) distribution is an **Inverse Gamma mixture of Gaussians**.

This just means that to generate a \( t \) random quantity you can first draw from an Inverse Gamma distribution and then draw from a Gaussian **conditional** on what you got from the Inverse Gamma.

(As you’ll recall, \( \lambda \sim \Gamma^{-1}(\alpha, \beta) \) just means that \( \lambda^{-1} = \frac{1}{\lambda} \sim \Gamma(\alpha, \beta) \)).

In more detail, \( (y|\mu, \sigma^2, \nu) \sim t_\nu(\mu, \sigma^2) \) is the same as the **hierarchical model**

\[
\begin{align*}
(\lambda|\nu) & \sim \Gamma^{-1}\left(\frac{\nu}{2}, \frac{\nu}{2}\right) \\
(y|\mu, \sigma^2, \lambda) & \sim N\left(\mu, \lambda \sigma^2\right).
\end{align*}
\]  

(19)

Putting this together with the **conjugate prior** for \( \mu \) and \( \sigma^2 \) we looked at earlier in the Gaussian model gives the following HM for the NB10 data:

\[
\begin{align*}
\nu & \sim p(\nu) \\
\sigma^2 & \sim \text{SI}-\chi^2(\nu_0, \sigma_0^2) \\
(\mu|\sigma^2) & \sim N\left(\mu_0, \frac{\sigma^2}{\kappa_0}\right) \\
(\lambda_i|\nu) & \sim \Gamma^{-1}\left(\frac{\nu}{2}, \frac{\nu}{2}\right) \\
(y_i|\mu, \sigma^2, \lambda_i) & \sim \text{indep} N\left(\mu, \lambda_i \sigma^2\right).
\end{align*}
\]  

(20)

Remembering also from introductory statistics that the Gaussian distribution is the **limit** of the \( t \) family as \( \nu \to \infty \), you can see that the idea here has been to **expand** the Gaussian model by embedding it in the richer \( t \) family, of which it’s a special case with \( \nu = \infty \).
Implementing Gibbs

Model expansion is often the best way to deal with uncertainty in the modeling process: when you find deficiencies of the current model, embed it in a richer class, with the model expansion in directions suggested by the deficiencies (we'll also see this method in action again later).

**The MCMC Dataset.** Imagine trying to do Gibbs sampling on model (18), with the parameter vector \( \theta = (\mu, \sigma^2, \nu) \).

Carrying out the iterative program described in (17) above would produce the following MCMC Dataset:

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Phase</th>
<th>( \mu )</th>
<th>( \sigma^2 )</th>
<th>( \nu )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Initializing</td>
<td>( \mu_0 )</td>
<td>( \sigma_0^2 )</td>
<td>( \nu_0 )</td>
</tr>
<tr>
<td>1</td>
<td>Burn-In</td>
<td>( \mu_1(y, \sigma_1^2, \nu_0) )</td>
<td>( \sigma_1^2(y, \mu_1, \nu_0) )</td>
<td>( \nu_1(y, \mu_1, \sigma_1^2) )</td>
</tr>
<tr>
<td>2</td>
<td>Burn-In</td>
<td>( \mu_2(y, \sigma_1^2, \nu_1) )</td>
<td>( \sigma_2^2(y, \mu_2, \nu_1) )</td>
<td>( \nu_1(y, \mu_2, \sigma_2^2) )</td>
</tr>
<tr>
<td>( b )</td>
<td>Burn-In</td>
<td>( \mu_b )</td>
<td>( \sigma_b^2 )</td>
<td>( \nu_b )</td>
</tr>
<tr>
<td>( b+1 )</td>
<td>Monitoring</td>
<td>( \mu_{b+1} )</td>
<td>( \sigma_{b+1}^2 )</td>
<td>( \nu_{b+1} )</td>
</tr>
<tr>
<td>( b+2 )</td>
<td>Monitoring</td>
<td>( \mu_{b+2} )</td>
<td>( \sigma_{b+2}^2 )</td>
<td>( \nu_{b+2} )</td>
</tr>
<tr>
<td>( b+m )</td>
<td>Monitoring</td>
<td>( \mu_{b+m} )</td>
<td>( \sigma_{b+m}^2 )</td>
<td>( \nu_{b+m} )</td>
</tr>
</tbody>
</table>

Looking at iterations 1 and 2 you can see that, in addition to \( y \), the sampler makes use only of parameter values in the current row and the previous row (this illustrates the Markov character of the samples).

As we've seen above, at the end of the \( (b+m) \) iterations, if you want (say) the marginal posterior for \( \mu \), \( p(\mu|y) \), all you have to do is take the \( m \) values \( \mu_{b+1}, \ldots, \mu_{b+m} \) and summarize them in any ways that interest you: their sample mean is your simulation estimate of the posterior mean of \( \mu \), their sample histogram (or, better, their kernel density trace) is your simulation estimate of \( p(\mu|y) \), and so on.
Practical Issues

Implementation Details. (1) How do you figure out the full conditionals, and how do you sample from them?

(2) What should you use for initial values?

(3) How large should $b$ and $m$ be?

(4) More generally, how do you know when the chain has reached equilibrium?

Questions (3–4) fall under the heading of MCMC diagnostics, which I’ll cover a bit later, and I’ll address question (2) in the case studies below.

Computing the full conditionals. For a simple example of working out the full conditional distributions, consider the conjugate Gaussian model we looked at earlier:

$$
\sigma^2 \sim \text{SI-}\chi^2(\nu_0, \sigma_0^2)
$$

$$
(\mu|\sigma^2) \sim N\left(\mu_0, \frac{\sigma^2}{\kappa_0}\right)
$$

$$
(Y_i|\mu, \sigma^2) \overset{\text{IID}}{\sim} N(\mu, \sigma^2).
$$

The full conditional distribution for $\mu$ in this model is $p(\mu|\sigma^2, y)$, considered as a function of $\mu$ for fixed $\sigma^2$ and $y$—but this is just

$$
p(\mu|\sigma^2, y) = \frac{p(\mu, \sigma^2, y)}{p(\sigma^2, y)}
$$

$$
= c p(\mu, \sigma^2, y)
$$

$$
= c p(\sigma^2) p(\mu|\sigma^2) p(y|\mu, \sigma^2)
$$

$$
= c \exp\left[-\frac{\kappa_0}{2\sigma^2}(\mu - \mu_0)^2\right] \prod_{i=1}^n \exp\left[-\frac{1}{2\sigma^2}(y_i - \mu)^2\right].
$$
Full Conditionals

From this

\[ p(\mu|\sigma^2, y) = c \exp\left[-\frac{\kappa_0}{2\sigma^2} (\mu - \mu_0)^2\right] \exp\left[-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu)^2\right]. \]

Expanding out the squares, collecting powers of \( \mu \), and completing the square in \( \mu \) gives

\[ p(\mu|\sigma^2, y) = c \exp\left[-\frac{\kappa_0 + n}{2\sigma^2} \left(\mu - \frac{\kappa_0 \mu_0 + n \bar{y}}{\kappa_0 + n}\right)^2\right], \tag{23} \]

from which it's clear that the full conditional for \( \mu \) in model (21) is

\[ (\mu|\sigma^2, y) \sim N\left(\frac{\kappa_0 \mu_0 + n \bar{y}}{\kappa_0 + n}, \frac{\sigma^2}{\kappa_0 + n}\right). \tag{24} \]

Similarly, the full conditional for \( \sigma^2 \) in this model, \( p(\sigma^2|\mu, y) \), considered as a function of \( \sigma^2 \) for fixed \( \mu \) and \( y \), is just

\[
\begin{align*}
 p(\sigma^2|\mu, y) & = \frac{p(\sigma^2, \mu, y)}{p(\mu, y)} \\
 & = \frac{c p(\sigma^2, \mu, y)}{c p(\sigma^2, \mu, y)} \\
 & = c p(\sigma^2) p(\mu|\sigma^2) p(y|\mu, \sigma^2) \\
 & = c (\sigma^2)^{-(1 + \frac{1}{2} \nu_0)} \exp\left(-\frac{\nu_0 \sigma_0^2}{2\sigma^2}\right). \\
 & \quad (\sigma^2)^{-1/2} \exp\left[-\frac{\kappa_0}{2\sigma^2} (\mu - \mu_0)^2\right]. \\
 & \quad (\sigma^2)^{-n/2} \exp\left[-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu)^2\right].
\end{align*}
\]

When this is simplified you get
Full Conditionals (continued)

\[ p(\sigma^2|\mu, y) = c (\sigma^2)^{-\left(1+\frac{n_0+1+n}{2}\right)} \exp \left[ -\frac{\nu_0\sigma_0^2 + \kappa_0(\mu - \mu_0)^2 + ns_\mu^2}{2\sigma^2} \right], \]

where \( s_\mu^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mu)^2. \)

From the form of this distribution it becomes clear that

\[ (\sigma^2|\mu, y) \sim \text{SI-}\chi^2\left(\nu_0 + 1 + n, \frac{\nu_0\sigma_0^2 + \kappa_0(\mu - \mu_0)^2 + ns_\mu^2}{\nu_0 + 1 + n} \right). \]

(26)

Thus in conjugate situations the full conditional distributions have conjugate forms, which are tedious but straightforward to compute.

Both the directness and the tedium of this calculation suggest that it should be possible to write a computer program to work out the full conditionals for you; this is where WinBUGS, MLwiN (a program that does both maximum-likelihood and Bayesian calculations in hierarchical (multilevel) models (Rasbash et al. 2000): multilevel.ioe.ac.uk) and other programs come in.