Bayesian Hierarchical Modeling

David Draper
Department of Applied Mathematics and Statistics
University of California, Santa Cruz
draper@ams.ucsc.edu
http://www.cse.ucsc.edu/~draper


© 2001 David Draper

All rights reserved. No part of this book may be reprinted, reproduced, or utilized in any form or by any electronic, mechanical or other means, now known or hereafter invented, including photocopying and recording, or by an information storage or retrieval system, without permission in writing from the author.

This book was typeset by the author using a PostScript (Adobe Systems, Inc.) based phototypesetter. The figures were generated in PostScript using the S+ data analysis language (Becker et al., 1988), and were directly incorporated into the typeset document. The text was formatted using the \LaTeX\ language (Lamport, 1994), a version of \TeX\ (Knuth, 1984).
TO ANDREA
Contents

Preface ix

1 Introduction to Bayesian modeling 1
   1.1 Quantification of uncertainty about observables 1
   1.2 Discrete outcomes: Exchangeability 7
   1.3 Prior, posterior, and predictive distributions 9
   1.4 Inference and prediction 12
   1.5 Coherence and calibration 13
   1.6 Conjugate analysis 15
   1.7 Comparison with frequentist modeling 18
   1.8 Continuous outcomes 20
   1.9 Additional reading 33
   1.10 Problems 34
   1.11 Notes 38

2 Simulation-based computation 47
   2.1 The need for Markov Chain Monte Carlo methods 47
   2.2 Hastings and Metropolis sampling 53
   2.3 Practical implementation issues 55
   2.4 MCMC monitoring and convergence diagnostics 73
   2.5 Gibbs sampling 83
   2.6 Case study: Measurement of physical constants 93
   2.7 Additional reading 109
   2.8 Problems 110
   2.9 Notes 111

3 Hierarchical models for combining information 123
   3.1 Meta-analysis 123
   3.2 Case study: Can aspirin prevent heart attack mortality? 123
3.3 Approximate fitting of Gaussian HMs: Maximum likelihood and empirical Bayes 123
3.4 Incorporating study-level covariates 123
3.5 Case study: Effects of teacher expectancy on pupil IQ 123
3.6 Additional reading 123
3.7 Problems 123
3.8 Notes 123

4 Hierarchical model diagnostics 125
4.1 Frequentist-inspired diagnostics 125
4.2 Predictive validation 125
4.3 Case study: Dose-response relationships in carcinogenicity assessment of exposure to diesel fumes 125
4.4 Additional reading 125
4.5 Problems 125
4.6 Notes 125

5 Random-effects and mixed models 127
5.1 Model-based analysis of cluster samples 127
5.2 Predictor variables at all levels of the hierarchy 127
5.3 Comparison between Bayesian and frequentist methods for random-effects and mixed models 127
5.4 Case study: Quality of care measurement for elderly hospitalized Americans 127
5.5 Additional reading 127
5.6 Problems 127
5.7 Notes 127

6 Longitudinal data analysis 129
6.1 Repeated-measures designs 129
6.2 Growth-curve analysis 129
6.3 Case study: Effects of maternal speech patterns on infant speech development 129
6.4 Additional reading 129
6.5 Problems 129
6.6 Notes 129

7 Mixture modeling 131
7.1 Density estimation 131
Preface

This book provides an introduction to the formulation, fitting, and checking of [hierarchical] or multi-level models, from the Bayesian point of view. Hierarchical models (HMs) arise frequently in five main kinds of applications:

- HMs are common in fields such as health and education, in which data—both outcomes and predictors—are often gathered in a nested or hierarchical fashion: for example, patients within hospitals, or students within classrooms within schools. HMs are thus also ideally suited to the wide range of applications in government and business in which single- or multi-stage cluster samples are routinely drawn, and offer a unified approach to the analysis of random-effects (variance-components) and mixed models.

- A different kind of nested data arises in meta-analysis in, e.g., medicine and the social sciences. In this setting the goal is combining information from a number of studies of essentially the same phenomenon, to produce more accurate inferences and predictions than those available from any single study. Here the data structure is subjects within studies, and as in the clustered case above there will generally be predictors available at both the subject and study levels.

- When individuals—in medicine, for instance—are sampled cross-sectionally but then studied longitudinally, with outcomes observed at multiple time points for each person, a hierarchical data structure of the type studied in repeated-measures or growth curve analyses arises, with the readings at different time points nested within person.

- For simplicity people often try to model data as (conditionally) IID at a fairly high level of aggregation—for instance, by pretending that all the subjects in a sampling experiment are drawn homogeneously from a single population. In fact, heterogene-
ity is often the rule rather than the exception, and frequently
the available predictor variables do not "explain" this hetero-
geneity sufficiently. With recent computational advances it is
becoming increasingly straightforward to at least describe such
heterogeneity with mixture models that employ latent vari-
ables (unobserved predictors) in a hierarchical structure. Exam-
examples include density estimation with an unknown number of
sub-populations mixed together and Bayesian nonparameti-
cric modeling, in which people work with distributions whose
sample spaces are themselves sets of distributions instead of
(say) real numbers.

- Hierarchical modeling also provides a natural way to treat issues
  of model selection and model uncertainty with all types of
data, not just cluster samples or repeated measures outcomes.
For example, in regression, if the data appear to exhibit residual
variation that changes with the predictors, you can expand the
model that assumes constant variation, by embedding it hierar-
chically in a family of models that span a variety of assumptions
about residual variation. In this way, instead of having to choose
one of these models and risk making the wrong choice, you can
work with several models at once, weighting them in proportion
to their plausibility given the data.

In studying HMs there are two kinds of technical issues that
also arise: fully Bayesian computation in HMs requires the use
of simulation methods such as those based on Markov Chain
Monte Carlo (MCMC) ideas, and—as usual with any class of
statistical models—there are questions of model diagnostics.

Plan of the book. In the chapters below I describe the prin-
ciples of Bayesian hierarchical modeling, with emphasis on practi-
cal rather than theoretical issues, and I illustrate these principles
with analyses of real data drawn from case studies. The material is
intended for applied statisticians with an interest in learning more
about hierarchical models in general, and the Bayesian analysis
of such models in particular. The field of study examined here is
surprisingly wide, touching on topics in numerical analysis, high-
dimensional integration, and measures on function space (on the
mathematical side), the meaning of uncertainty and probability (in
philosophy and statistics), and practical issues in Markov chains,
time series, and modern nonparametric analysis.
The nine chapters cover the five application areas mentioned above, together with an introductory chapter on Bayesian modeling, one chapter each on MCMC and model diagnostics, and a concluding chapter with discussion and suggestions for future research. An appendix reviews standard probability distributions useful in Bayesian work, and another provides computing details in the environments I used to write the book: the statistical computing and graphics package S+, the Gibbs sampling package BUGS, the multi-level modeling package MLwiN, the symbolic computing package Maple, and the high-level programming language C.

An understanding of probability at the level typically required for a master's degree in statistics provides ample mathematical background. I have taught subsets of this material successfully to groups including British final-year undergraduates, American PhD students, and PhD-level researchers enrolled in short courses, and the book has also proven useful for self-study by researchers and graduate students in a variety of disciplines (including statistics).

No previous experience with Bayesian methods is needed—all relevant ideas are covered in a self-contained fashion. If you already know a fair bit about Bayes you can move through Chapter 1 briskly, although there are philosophical and practical issues of potential interest even to seasoned Bayesians there. If you are new to Bayes, a good way to read this book is in conjunction with one or both of the following excellent publications: the Bayesian text by Gelman et al. (1995), and the monograph on MCMC by Gilks et al. (1996) (although the latter is at a more advanced level than the former). A supplementary and complementary perspective on many of the issues covered here can also be obtained by doing some reading in parallel in the excellent book by Carlin and Louis (1996).

Some style and layout conventions to be aware of in the chapters that follow:

- I like to teach and talk about research ideas informally, and the book reflects this. I have tried to write as if you and I were having an extended conversation on the topics covered here. This is natural in a book on applications of the Bayesian approach to probability, and has various advantages, but one possible disadvantage is that the scope of agreement in the statistics community with statements I make may not be immediately clear. So here is a dictionary: sentences including phrases like “You
can show that" and "Evidently" are meant to be expressions of mathematical fact; phrases like "Most people believe that" signal general unanimity (in my view) among (Bayesian) statisticians on the point I'm covering; and phrases like "It seems to me that" precede a personal opinion of mine, which may or may not be shared by other statisticians.

- I am writing in \LaTeX, and I don't like \LaTeX's subsection layout, so \textbf{one-line text boxes} act as subsection headings. Multi-line text boxes, in contrast, bring emphasis to definitions, theorems, and summaries of important points.

- The book is dotted with blocks of text that begin \textbf{NB}—these highlight things like general notational conventions and pitfalls to be avoided in implementing the ideas I'm discussing at that point.

- \textbf{Bold font} is generally reserved for the first appearance of important technical terms, and \textit{italics} signal items of particular emphasis.

- I have tried to write for a fairly diverse audience in terms of mathematical and statistical background. One of the main devices for (I hope) achieving this fairly smoothly is \textit{footnotes}⁰, which are often too long to be at the bottom of the page where they belong, so I have collected them at the end of each chapter. The naming convention is that, for instance, note⁶ in Chapter 3 will be found as item 3.6 in the Notes section of that chapter. In general, the footnotes supplement the main text by adding historical details, additional mathematical formalism, notices of nonstandard terminology, and the like. The intent is that if you are new to much of this material, you can skip (many or all of) the notes on first reading if you want; whereas if you are fairly experienced in the topics covered here, or you want to dig a bit deeper, you may find that the notes enrich the material and suggest directions for further reading.

- I also offer a somewhat eclectic variety of problems in each chapter: some are data-analytic, others somewhat more theoretical, and they vary widely in difficulty. Problems that use material in the notes begin with the symbol \textit{\(\mathcal{N}n\)}, where \(n\) refers to the chapter in which the relevant notes may be found. To get the most out of the material, I recommend not only working many or all of the problems but also programming up most or all of
the examples and case studies to see if you get results similar to mine.

I am grateful to Bill Browne, Ryan Cheal, Dimitris Fouskakis, David Freedman, Andrew Gelman, Sander Greenland, Merilee Hurn, Dennis Lindley, Nick Longford, David Madigan, Colin Mallows, Michael Seltzer, and David Williams for comments on earlier versions of this material, and to the UK Engineering and Physical Sciences Research Council, the European Commission, the University of Bath (UK), and the University of California, Santa Cruz for support. Membership on this list does not imply agreement with the ideas expressed here, nor are any of these people or institutions responsible for any errors that may be present.

Santa Cruz, California  David Draper
April 2001
CHAPTER 2

Simulation-based computation

2.1 The need for Markov Chain Monte Carlo methods

**Case study 2.1:** *Estimation of species life-span from the stratigraphic fossil record.* One class of questions of particular interest to people like geologists and paleobotanists concerns the points in the remote past (a) when a given species first appeared on Earth and (b) when it became extinct. The standard way to estimate these points is to dig below the surface—for instance by taking a vertical, cylindrical core sample—and look for the first and last occurrences of the species in the fossil record, measured in (say) meters below ground level. By means of carbon dating and cross-referencing against “known” times of major past events, an approximate one-to-one correspondence can be established between distance below ground and the time scale of interest.

| Table 2.1. Observed locations, in meters below ground, of finds of 6 taxa of ammonites, from Macellari (1986) by way of Strauss and Sadler (1989). |
|---|---|
| Name | $n$ | Locations $y_i$ |
| D. lambi | 14 | 484, 517, 533, 550, 690, 780, 850, 995, 1055, 1083, 1100, 1115, 1130, 1157 |
| (0) | | 617, 634, 645, 667, 692, 707, 730, 748, 755, 772, 779, 793, 822 |
| M. seymour. | 13 | 608, 622, 650, 685, 693, 704 |
| (1) | | 725, 742, 757, 771, 780, 800, 820 |
| K. darwini | 13 | 650, 700, 757, 785, 793, 800, 892, 911, 934, 961, 994, 1005, 1025, 1032, 1048, 1067, 1077, 1091, 1100, 1115, 1124, 1140, 1157, 1166, 1171 |
| (2) | | 668, 700, 767 |
| G. gemmatus | 25 | 815, 900, 950, 967, 982, 1000, 1015, 1033, 1050, 1070, 1098, 1115, 1140, 1150, 1158, 1175 |
| (3) | | 1091, 1100, 1115, 1124, 1140, 1157, 1166, 1171 |
| M. waddel. (4) | 3 | 668, 700, 767 |
| M. dens. α | 16 | 815, 900, 950, 967, 982, 1000, 1015, 1033, 1050, 1070, 1098, 1115, 1140, 1150, 1158, 1175 |
| (5) | | 1091, 1100, 1115, 1124, 1140, 1157, 1166, 1171 |

As an example of this sort of work, Tables 2.1 and 2.2 give data on the observed range of late Cretaceous ammonites—a kind of
mollusk that left behind flat, spiral fossil shells—from samples
gathered by Macellari (1986) on Seymour Island in the Antar-
cctic Peninsula (the data were digitized from Figure 1 in Strauss
and Sadler, 1989). Range information on 13 taxa of ammonite are
given, and the data are in meters below the surface (I can’t trans-
form to time because Macellari and Strauss-Sadler don’t say how
to, but the late Cretaceous period ended about 70 million years
ago). Most of the taxa have rather grand names—Anagaudryceras
seymouriense, for instance—that are too big to fit into the tables
except as abbreviations, and for “ease of subsequent reference”
Strauss and Sadler number them, slightly curiously, from 0 to 12.
Let’s concentrate at first on one of the taxa, say M. dens. α, and
denote the observed locations of fossil finds by \( y_1, \ldots, y_n \). What
sort of model would be appropriate for the \( y_i \)?

<table>
<thead>
<tr>
<th>Names</th>
<th>( n )</th>
<th>Locations ( y_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>K. laurae</td>
<td>8</td>
<td>900, 928, 950, 973, 992, 1008, 1024, 1160</td>
</tr>
<tr>
<td>A. seymour</td>
<td>10</td>
<td>908, 985, 1000, 1025, 1035, 1042, 1060, 1082, 1115, 1137</td>
</tr>
<tr>
<td>M. dens. γ</td>
<td>11</td>
<td>935, 947, 1000, 1015, 1024, 1040, 1050, 1084, 1100, 1110, 1120</td>
</tr>
<tr>
<td>P. ricardi</td>
<td>9</td>
<td>960, 990, 1000, 1016, 1033, 1050, 1074, 1115, 1132</td>
</tr>
<tr>
<td>M. dens. β</td>
<td>10</td>
<td>967, 977, 990, 1000, 1030, 1048, 1066, 1080</td>
</tr>
<tr>
<td>P. lori</td>
<td>11</td>
<td>988, 1032, 1115</td>
</tr>
<tr>
<td>P. ultimus</td>
<td>12</td>
<td>1100, 1110, 1127, 1150</td>
</tr>
</tbody>
</table>

**A model for species life-span data.** The observations are
conceptually continuous, so if my predictive uncertainty is ex-
changeable I must be in the realm of model (1.22): \( F \sim p(F), (y_i|F) \)
\( \sim F \) for continuous \( F \). Strauss (a statistician) and Sadler (a geologist)
say that there is lots of evidence to support the assumption of
randomness of fossil distribution, locally in space and time, for
ammonites and many other species, and further they say this evidence
indicates that a Poisson process \( ^2 \) is a reasonable approximation.
This makes the number \( n \) of finds of a given taxon at a particular
location in a fixed interval in time follow a Poisson distribution,
and it is a basic fact about the Poisson process (e.g., Ross, 1970) that
conditional on \( n \) the finds are uniformly distributed through-
out the interval. So it would seem from the physical situation that, if this story holds, we don't have any model uncertainty about $F$: denoting the left and right endpoints of the true underlying range for the taxon in question by $A$ and $B$, the model is evidently

$$(A, B) \sim p(A, B)$$

$$\left(y_i | A, B \right) \overset{\text{IID}}{\sim} U(A, B).$$

(2.1)

Of course, this model needs to be checked before it is applied: for instance, under the Poisson process the gap lengths $l_i = (y_{i+1} - y_i)$ between successive finds should be IID exponential, and the $l_b$ should exhibit no serial correlation. Strauss and Sadler, who examined these data, report little serial correlation and say that "gap lengths for ammonite data roughly follow an exponential distribution truncated below 8.5m, [which is] approximately the limit of resolution of Macellari's published [core sample] illustration."

Figure 2.1. Uniform qqplots for the ammonite data, showing the 6 taxa with the largest sample sizes.

An even more direct way to check the distributional assumptions in model (2.1) is with uniform probability plots. Figure 2.1 presents such plots for the 6 taxa in Tables 2.1 and 2.2 with the largest sample sizes (it is hard to make much of a qqplot based on 3 or 4 points). Apart from a bit of wobble in the upper-left panel and a few left-tail outliers in several of the taxa (see Problem 2.1 for
a sensitivity analysis, exploring the effects of these points on the subsequent inferences), these plots are not desperately far from uniformity. I will go with model (2.1) in what follows, but I will differ a bit from Strauss and Sadler in that I am also interested in two other parameters: $\mu = \frac{B + A}{2}$, the center of the true range, and $\sigma = \frac{B - A}{2}$, a measure of the scale of this range. Reparameterized in this way the model is

\[
(\mu, \sigma) \sim p(\mu, \sigma) \\
(y_i | \mu, \sigma) \overset{\text{IID}}{\sim} U(\mu - \sigma, \mu + \sigma).
\] (2.2)

**Computational strategies.** Now it turns out that conjugate analysis of this model is possible if one of the two parameters is known but not if both are unknown (Problem 2.2). So, as with the NB10 $t$ model mentioned in Section 1.8, conjugate analysis can only take us partway to the goal of a fully Bayesian treatment of the broadest possible class of practically useful models: a more flexible computing strategy is needed. In describing such a strategy I will digress for quite awhile, and then I'll return to the ammonite data in Section 2.5.

In this century people have known about the need for a better approach to computing for decades, of course, and (as I mentioned in Chapter 1) it was clear to Laplace way back in the 1770s that the integrals (1.30–1.32) in problems with multiple parameters could be immensely troublesome in general. With IID data $y = (y_1, \ldots, y_n | \theta)$ from a sampling distribution driven by a parameter $\theta = (\theta_1, \ldots, \theta_k)$ that is in most cases multivariate, three main strategies, all of them with the goal of accurate approximate (rather than exact) results, have so far been developed to cope with this problem:

- **Asymptotic analysis** (e.g., Bernardo and Smith, 1994) relies on Central-Limit-Theorem-style results to approximate posterior distributions such as $p(\theta | y)$, $p(\theta_1 | y)$, and $p(y_{n+1} | y)$ by suitable univariate and multivariate normal distributions. With a few notable exceptions, for most Bayesians this was the leading supplement to conjugate analysis until the early 1980s.

Asymptotic approximations work just about the way you might think they would, given what you know about the normal distribution: even with fairly small $n$ this approach can produce reasonably accurate posterior summaries for parameters with
symmetric distributions on the whole real line, like $\mu$ in the Gaussian NB10 model (1.24). However, for parameters with skewed distributions and restricted ranges—such as $\theta$ in the Bernoulli/binomial model (1.11), which lives on $(0,1)$, and $\sigma^2$ in (1.24), which lives on $(0, \infty)$—it is generally necessary to (a) **transform** the parameters to have support on all of $\mathbb{R}$ (for instance, by working with $\text{logit}(\theta) = \log \left( \frac{\theta}{1-\theta} \right)$ and $\text{log}(\sigma^2)$, respectively), (b) do the normal approximation on the new scale, and (c) back-transform. Unfortunately, even after all of this fiddling about, in practice you may not get highly accurate results with small $n$.

- **Closed-form approximations** try to use higher-order asymptotic expansions than those on which standard asymptotic analysis are based to produce extremely accurate small-sample asymptotics (as some people put it). The leading example of this approach is the class of Laplace approximations (e.g., Tierney and Kadane. 1986), which I mentioned briefly in Chapter 1 and to which I will return in Chapter 8. Although invented by Laplace more than 200 years ago, the method seems to have languished with few practical applications until the 1980s. This approach works well with small $k$, particularly in conjunction with the kind of parameter transformation, or **reparameterization**, discussed above, but can lead to substantial numerical difficulties when $k$ is large or the posterior distribution is multimodal.

- **Sampling-based approximations** try to take advantage of modern computing power, either (i) to approximate the integrals that arise in computing $p(\theta|y)$, $p(\theta_j|y)$, and $p(y_{n+1}|y)$ by Monte Carlo sampling experiments or (ii) to directly draw random samples from these distributions.

- **Importance sampling** (e.g., Geweke, 1989) is an example of approach (i): for instance, if you want to calculate a posterior mean

$$E(\theta|y) = \int \theta \, p(\theta|y) \, d\theta$$

and the integral in (2.3) is intractable, you could choose a density $g(\theta)$ that is everywhere positive, re-express (2.3) as

$$\int \theta \, p(\theta|y) \, d\theta = \int \left[ \frac{\theta \, p(\theta|y)}{g(\theta)} \right] g(\theta) \, d\theta$$
\[ E_{\theta} \left[ \frac{\phi p(\theta|y)}{g(\theta)} \right] \]  

(2.4)

take a large IID sample \((\theta_i, i = 1, \ldots, N)\) of points from \(g(\theta)\), calculate \(W_i = \frac{\phi p(\theta|y)}{g(\theta)}\), and approximate \(E(\theta|y)\) by \(\frac{1}{N} \sum_{i=1}^{N} W_i\). This method, which was extensively used in econometrics in the 1980s, requires considerable skill in the choice of the importance sampling density \(g\), and (like many other approaches) runs into implementational and accuracy difficulties with large \(k\).

- Markov Chain Monte Carlo (MCMC) methods (e.g., Gilks et al., 1996a) are the leading current example of approach (ii), and have been used extensively in statistics since the early 1990s with increasing success. Forerunners to this approach appeared in the statistics literature in the 1980s in the form of data augmentation (Tanner and Wong, 1987) and sampling-importance-resampling (Rubin, 1987), and MCMC methods were first widely popularized by Gelfand and Smith (1990), all of which makes it seem as though MCMC methods were not developed until the 1990s, but in fact the leading special cases of MCMC were introduced (a) in the 1950s (!) by physicists (Metropolis et al., 1953) whose work was unknown to the statistics community; (b) in the 1970s by a statistician (Hastings, 1970), whose efforts in generalizing Metropolis et al. went almost completely unnoticed; and (c) in the early 1980s by applied mathematicians (Geman and Geman, 1984) working in cognitive neuroscience, the generality of whose methods was not at first appreciated. (So much for cross-disciplinary collaboration and smooth historical sailing of important ideas.)

I am going to focus in this chapter—and in the rest of the book—on MCMC methods, because they appear to me (and to many others) to be the most promising general approach to Bayesian computation available at present. They can be highly computationally intensive (in other words, it can take minutes or even hours on your computer to get accurate answers), but I think it is fair to say that they have opened up the floodgates on applied Bayesian work since the early 1990s like no approach before them.

[MCMC methods.] The idea behind MCMC methods is simple and intuitive: I start out wanting to compute a probability den-
sity, for example $p(\theta | y)$, but then I notice after thinking about it for awhile that for many purposes I would be just as happy to have a large random sample from $p(\theta | y)$ as to know its precise form, because if I had the sample and it was big enough I could approximate its form, to a high degree of accuracy, with a histogram or kernel density estimator, and if I wanted to know its mean (say) I could approximate it by the sample mean. So the question becomes: can I figure out how to efficiently simulate a large number of random draws from $p(\theta | y)$?

This stage—the implementation of MCMC—is not so straightforward. In fact, it required a substantial bit of lateral thinking on the part of Metropolis et al. (the 1950s physicists). They said, in effect: suppose you could construct a Markov chain—a stochastic process $\{\theta_t, t \geq 0\}$ of values unfolding in time $t$—with three properties:

- It should have the same state space (set of possible values) as $\theta$;
- It should be easy to simulate from; and
- Its equilibrium (or stationary) distribution—the distribution from which samples from the chain eventually will be drawn, after it has been run for a long time—is $p(\theta | y)$.

If you could do this, you could run the Markov chain for a very large number of iterations, generating a huge sample ($\theta_t, t = 0, 1, \ldots$) from the posterior, and then use simple descriptive summaries (means, SDs, correlations, histograms or density estimates) to extract any features of the posterior you want.

How to construct such a Markov chain—and make sure it does what you want it to—is the subject of the next several sections.

### 2.2 Hastings and Metropolis sampling

The most general MCMC method in wide use today is due to Hastings (1970), and I will look at it first (the methods due to Metropolis et al. and Geman-Geman are special cases, to be covered later). In effect Hastings said, OK, I am trying to build a Markov chain on $\theta$ starting at some initial value $\theta_0$. Given that the chain has found its way to state $\theta_t$ at time $t$, all you need to know to characterize the chain (since it is Markov) is the probability distribution for where it will go at time $(t + 1)$. Hastings, following Metropolis
et al. but adding a new wrinkle (to be explained below), suggested the following way to generate \( \theta_{t+1} \):

- Choose something called a proposal distribution (PD) \( f(\theta | \theta_t) \) for where to consider going next, given that you are at \( \theta_t \) now, and sample a candidate point \( \theta^* \) from this distribution.

- **Accept the move to \( \theta^* \) at time \( t + 1 \) with probability**

\[
\alpha_H(\theta_t, \theta^*) = \min \left[ 1, \frac{p(\theta^* | y) f(\theta_t | \theta^*)}{p(\theta_t | y) f(\theta^* | \theta_t)} \right]; \tag{2.5}
\]

otherwise stay where you are. In other words, toss a Bernoulli coin with probability \( \alpha_H \) of coming up heads—if you get heads, set \( \theta_{t+1} = \theta^* \), otherwise set \( \theta_{t+1} = \theta_t \).

He then proved the remarkable fact that with just about any PD \( f \), the equilibrium distribution \( p(\theta | y) \) for the Markov chain is \( p(\theta | y) \), as desired.

Gilks et al. (1996b) note that the resulting algorithm is extremely easy to code:

**Algorithm** (Hastings, 1970, generalizing Metropolis et al., 1953). To construct a Markov chain whose equilibrium distribution is \( p(\theta | y) \), choose a proposal distribution (PD) \( f(\theta | \theta_t) \), define \( \alpha_H(\theta_t, \theta^*) \) as in (2.5), and

\[
\text{Initialize } \theta_0; \ t \leftarrow 0
\]

Repeat {\allowdisplaybreaks
\begin{align*}
\text{Sample } \theta^* & \sim f(\theta | \theta_t) \\
\text{Sample } u & \sim U(0,1) \\
\text{If } u & \leq \alpha_H(\theta_t, \theta^*) \text{ then } \theta_{t+1} \leftarrow \theta^* \\
\text{else } \theta_{t+1} & \leftarrow \theta_t \\
& t \rightarrow (t + 1)
\end{align*}
\}

**Example: Gaussian with unknown \( \sigma^2 \) and known \( \mu \).**

To see the Hastings algorithm in action, consider the simple model whose likelihood is specified by Gaussian draws with known mean but unknown variance, for instance applied to the NB10 data of Chapter 1 by pretending that we know \( \mu \):

\[
\sigma^2 \overset{\text{SI}}{\sim} \chi^2(\nu_p, \sigma_b^2) \\
(y_i | \sigma^2) \overset{\text{IID}}{\sim} N(\mu, \sigma^2), \quad i = 1, \ldots, n. \tag{2.7}
\]
You can use the standard conjugate machinery to work out the right answer for $p(\sigma^2|y)$ in this model,

$$
(\sigma^2|y) \sim SI-\chi^2 \left( \nu_p + n, \frac{\nu_p \sigma^2 + ns^2}{\nu_p + n} \right),
$$

where $s^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mu)^2$, so I'm not doing Hastings here because it's the only way to compute the posterior, but it's convenient to know the right answer so that I can compare the Hastings results with it. In this example $\theta = \sigma^2$, and I will take $\mu = 404.59$ (the NB10 sample mean) for illustration.

### 2.3 Practical implementation issues

If you look at the Hastings² algorithm (2.6) for awhile, you will see that three practical issues still need to be addressed in implementing it: picking the initial value $\theta_0$; deciding how long to run the Markov chain and how to use the output to approximate $p(\theta|y)$; and choosing a PD $f(\theta|\theta_t)$. I will take a first crack at addressing all three of these issues in this section, in each case first by making some general comments and then coming back each time to the simple Gaussian model (2.7) above.

**Choosing initial value(s) $\theta_0$.** The Hastings algorithm only mentions the need to choose a single initial value $\theta_0$, and indeed in many problems one judicious choice of $\theta_0$ is enough. If you are only going to pick one $\theta_0$, the idea is to choose a value that is close to the center of the posterior distribution you are trying to simulate from—this will increase the chance that the Markov chain settles down into its equilibrium distribution quickly. A good $\theta_0$ of this kind can come from anything simple that you may know about the posterior, for instance a decent frequentist estimate of $\theta$ like the **maximum likelihood estimate (MLE).**

There is a potential danger in only choosing one $\theta_0$, however. When the Markov chain is run it will wander around in $\theta$-space over time $t$, and you would like to be sure that it has fully explored all regions of high posterior probability by the time you decide that the number of iterations you have looked at is enough. If the chain moves around freely, happily jumping all over the place, people say that it is **mixing well** (I will give some examples later in this section of poor mixing and good mixing). If (1) the posterior is multi-modal; (2) the particular MCMC implementation you are
currently using is mixing poorly; (3) you start the chain off near only one of the modes; and (4) you don't run it for very long, then you can see there is a real possibility you will never find the other mode(s).

There are two leading strategies for dealing with this problem: simulated annealing (SA; e.g., Geman and Geman, 1984) and multiple highly dispersed initial values (Gelman and Rubin, 1992). SA works by (a) thinking of mode-finding as like hill-climbing, with the hill(s) defined by \( p(\theta|y) \), and (b) using a clever "non-greedy" strategy for iterative hill-climbing that sometimes is willing to go downhill to increase the chance of not getting stuck at a local maximum. The Gelman-Rubin plan is (a) to start up the chain at a number of wildly different \( \theta_0 \) values and then (b) to see if it always converges to the same mode.

I like SA better than Gelman-Rubin because it turns out\(^a\) that you can think of the Metropolis algorithm (see (2.21) below) as a special case of SA—so that you really only have to write one computer program to solve both the mode-finding and the posterior-sampling problems—but the Gelman-Rubin approach also has many fans (and I will give an example of it in Section x.x). Fortunately the problem of multiple modes generally only arises with HMIs when you have used a highly informative prior that conflicts sharply with the likelihood, a situation you generally want to avoid in any case, so in what follows I will mostly deal with the question of initial values by choosing a single good \( \theta_0 \).

Application to (2.7). The MLE for \( \sigma^2 \) in the simple Gaussian model (2.6) is \( s^2 \), the sample variance centered at the known \( \mu = 404.59 \), which in the NB10 data comes out 41.402, so that's what I'll use for \( \sigma^2 \) when I want to illustrate a good initial value below.

Choosing a convergence-monitoring strategy. This task in turn divides into two sub-tasks: how to decide when you've reached equilibrium, and how to monitor the output of the chain from that point onward to get posterior summaries of whatever accuracy you want.

- Reaching equilibrium. Think of the output of the chain as a time series indexed by the iteration counter \( t \) in (2.6). Equilibrium in this context is like stationarity of the time series, for which there are a variety of standard tests. I will cover this topic much more fully in Section 2.4 below; for now, pretend we have already solved the PD problem and consider Figure 2.2, which
was obtained using a particular PD I'll motivate below and an initial value that is far from the correct posterior mean in model (2.7) with the NB10 data.

\begin{center}
\includegraphics[width=0.5\textwidth]{hastings_output}
\end{center}

\textbf{Figure 2.2. Hastings output in model (2.7) with the NB10 data, using a PD of the form (2.15) and an initial value far from equilibrium.}

The output in this figure exhibits two undesirable features in MCMC sampling: it is not mixing very well—notice that there are fairly substantial periods (for instance, from about iterations 120 to 160) during which it does not move at all—and the beginning of the series was spent looking for equilibrium, which the sampler seems to have found at about iteration 50.

The first of these undesirable behaviors can be diagnosed by computing the \textbf{first-order autocorrelation} (or \textbf{serial correlation}) of the series, which is about $\hat{\rho} \approx +0.94$ in this case; a better-mixing chain would have a value of $\hat{\rho}$ much closer to 0. I will talk below about how to reduce serial correlation in Hastings samplers.

The simplest way to get around the second problem in Figure 2.2 is to \textbf{throw away the first 50 iterations} and then start monitoring the chain from that point on. People refer to an initial period which is discarded in this way as the \textbf{burn-in} period $n_B$ for
the MCMC sampler. If the iterations are quick to compute and a good initial value is available, people often use a fairly standard value of $n_B$ like 1,000 (or 5,000, just to be safe), and then increase $n_B$ if the time series plot of the output shows that a larger burn-in period is needed.

- Monitoring the chain to summarize the posterior accurately. Suppose you are convinced that the sampler is in equilibrium after $n_B$ burn-in iterations, which you have discarded. Then the basic idea for what to do next is (a) to run the chain for a further $n_M$ monitoring iterations, creating what I will call the MCMC data set, and (b) to approximate interesting features of the posterior distribution just by using simple descriptive summaries of this data set.

Table 2.3. Hypothetical MCMC data set in a model with parameter vector $\theta = (\gamma, \beta, \Delta)$, using $n_B = 5,000$ and $n_M = 20,000$. Here $\eta$ is the derived quantity $\frac{\beta_0\Delta}{\tau_1}$ and the $y_i^*$ are draws from the predictive distribution for a new $y$.

<table>
<thead>
<tr>
<th>Iteration Number ($t$)</th>
<th>MCMC Phase</th>
<th>$\gamma_t$</th>
<th>$\beta_t$</th>
<th>$\Delta_t$</th>
<th>$\eta_t = \frac{\beta_0 \Delta_t}{\tau_t}$</th>
<th>$y_i^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Initial Value</td>
<td>10.4</td>
<td>0.0762</td>
<td>-328</td>
<td>-2.40</td>
<td>54.2</td>
</tr>
<tr>
<td>1</td>
<td>Burn-in</td>
<td>11.7</td>
<td>0.0556</td>
<td>-359</td>
<td>-1.71</td>
<td>60.0</td>
</tr>
<tr>
<td>$n_B = 5000$</td>
<td>Burn-in</td>
<td>9.26</td>
<td>0.0610</td>
<td>-274</td>
<td>-1.80</td>
<td>63.7</td>
</tr>
<tr>
<td>5001</td>
<td>Monitor</td>
<td>10.6</td>
<td>0.0804</td>
<td>-355</td>
<td>-2.69</td>
<td>49.9</td>
</tr>
<tr>
<td>$n_B + n_M = 25000$</td>
<td>Monitor</td>
<td>10.9</td>
<td>0.0793</td>
<td>-338</td>
<td>-2.45</td>
<td>58.1</td>
</tr>
</tbody>
</table>

Table 2.3 presents a hypothetical MCMC data set in a problem with data vector $y$ and parameter vector $\theta = (\gamma, \beta, \Delta)$. (I will talk about the last two columns of this table in a few paragraphs.) Here let's suppose that 5,000 burn-in and 20,000 monitoring iterations are adequate to attain equilibrium and produce posterior summaries of sufficient accuracy for whatever you're doing. Then (a) if you'd like an estimate of the posterior mean of $\Delta$, you can just calculate the sample mean of the
20,000 values \((-355, \ldots , -338)\) in rows 5,001 to 25,000 in the 
\(\Delta\) column of the MCMC data set; (b) if you want a plot of the 
marginal posterior for \(\gamma\), all you have to do is pass the 20,000 
values \((10.6, \ldots , 10.9)\) in those same rows in the \(\gamma\) column 
of the MCMC data set to the histogram or kernel density trace 
function in S+ (or whatever your favorite data analysis package 
is); (c) if you'd like an estimate of the posterior correlation 
between \(\beta\) and \(\Delta\), you just calculate the sample correlation of the 
\(\beta\) and \(\Delta\) columns in the monitoring part of Table 2.3; and so 
on. Estimates of (just about) anything you'd like to know about 
the posterior for \(\theta\)—univariate, multivariate, whatever—are 
obtainable from the MCMC data set.

Notice that two of the difficult integration problems in Bayesian 
calculations with multivariate \(\theta\) I mentioned in Chapter 1— 
normalizing constants (1.30) and marginal posteriors (1.31)— 
have disappeared with the MCMC approach: the normalizing 
constants cancel in the acceptance probabilities (look at the 
form of (2.5)), and sampling from the posterior instead of 
approximating the actual density makes marginalization trivial 
(if you want to know something that pertains only to \(\beta\), say, then 
you simply ignore all the other columns in the MCMC data set).

It also turns out that the other two difficult integration problems 
from Chapter 1—computing predictive distributions (1.32) and 
posteriors for functions of parameters (1.33)—evaporate as well 
with MCMC. Concerning functions of parameters, you can 
convince yourself (Problem 2.3) that if \(\{\theta_t, t = n_B + 1, \ldots , n_B + n_M\}\) 
is a valid sample from the posterior for \(\theta\), then \(\{f(\theta_t), t = n_B + 1, \ldots , n_B + n_M\}\) is a valid sample from the posterior for 
f\(\theta\) for all reasonable \(f\). This means that all you need to do is calculate 
f\(\theta\) in each row of the MCMC data set and summarize as usual. 
The sixth column (counting from the left) of Table 2.3 illustrates 
this with the derived quantity \(\eta = \frac{\beta \Delta}{\gamma}\) in the hypothetical model 
examined in that table.

As for prediction, recall from (1.32) that the predictive distribu-
tion for a new \(y\)—call it \(y^*\), say—has the form

\[
p(y^*|y) = \int p(y^*|\theta) p(\theta|y) \, d\theta. \tag{2.9}
\]

The argument suggesting how to sample from this distribution 
with MCMC is in two parts.
(1) If you temporarily pretend the integral in (2.9) is a sum, I can probably convince you that summing the right-hand-side quantity \( p(y^*|\theta) \) with index of summation \( \theta \) and with respect to the density \( p(\theta|y) \)—in other words, computing \( \sum_\theta p(y^*|\theta) p(\theta|y) \)—is like taking a weighted average of the \( p(y^*|\theta) \) values with weights (adding up to 1) given by \( p(\theta|y) \), and if you let me pass from discrete to continuous \( \theta \) in the right way and wave my hands a bit I can then probably convince you that in words (2.9) says that the predictive distribution for \( y^* \) given \( y \) is a weighted average (or mixture) of the sampling distributions \( p(y^*|\theta) \) for \( y^* \) given \( \theta \), weighted by the posterior distribution \( p(\theta|y) \) for \( \theta \) given \( y \).

(2) I am getting considerably ahead of myself to bring it up here, since mixtures are the topic of Chapter 8, but it turns out (as we will see in that chapter) that mixtures correspond directly to hierarchical models. The HM suggested by the right-hand side of (2.9), in fact, has the simple form

\[
\begin{align*}
\theta & \sim p(\theta|y) \\
(y^*|\theta) & \sim p(y^*|\theta).
\end{align*}
\]  

What (2.10) means in sampling terms is that

To use MCMC to sample a \( y^* \) from \( p(y^*|y) \), draw a \( \theta \) from \( p(\theta|y) \), say \( \hat{\theta} \), and then sample \( y^* \) from \( p(y^*|\hat{\theta}) \).

Thus to draw from the predictive distribution of a new \( y \) in Table 2.3, for instance, you fill in each row from left to right, sampling \( \hat{\gamma}_t, \hat{\beta}_t, \) and \( \hat{\Delta}_t \) (say), and then \( y^*_t \) is just a draw from the sampling distribution \( p(y|\hat{\gamma}_t, \hat{\beta}_t, \hat{\Delta}_t) \) specified by the model featured in that table.

This all may sound too good to be true—all four major integration problems vanishing in one stroke—and in fact you may have developed a variety of something-for-nothing-style questions in reading the last few pages. The main question that occurs to me is

**Q:** Doesn’t it say earlier this section that columns of the MCMC data set, when thought of as time series in \( t \), often exhibit strong positive serial correlation, and doesn’t that invalidate the MCMC data set as a summary of the posterior?

**A:** It’s an interesting fact from time series (e.g., Anderson, 1971)
that if \((\theta_t, t = n_B + 1, \ldots)\) is a stationary (correlated) process, then 
\[ \hat{\theta} = \frac{1}{n_M} \sum_{t=n_B+1}^{n_M} \theta_t \]
is a consistent (as \(n_M \to \infty\)) estimate of \(E(\theta)\), and similar results apply for SDs, correlations, 
and so on. So it's OK to simulate correlated draws from a 
distribution in summarizing features of that distribution, as 
long as you get enough of them, and that's where the amount 
of serial correlation comes in. You can show, for instance, that 
if \(\theta_t\) is a (stationary) autoregressive process\(^{11}\) of order 1 
with mean \(\mu\), SD \(\sigma\), and first-order serial correlation \(\rho\) (people 
abbreviate this \(\theta_t \sim AR_1(\rho)\)), then \(\hat{\theta}\), as an estimate of 
\(\mu\), has standard error 
\[ SE(\hat{\theta}) = \frac{\sigma}{\sqrt{n_M}} \sqrt{\frac{1 + \rho}{1 - \rho}}. \tag{2.11} \]

If \(\rho = 0\), which corresponds to an IID or white noise series, 
the SE has the usual \(\frac{\sigma}{\sqrt{n_M}}\) form familiar to you from working 
with sample means of IID draws, but you can see that if \(\rho\) 
is close to +1 then the SE can become prohibitively large. 
For example, if \(\rho = 0.9\), you would have to run the chain 
\(\sqrt{\frac{1.2}{0.1}} \approx 4.4\) times longer than if it had been white noise to 
get the same accuracy in estimating \(\mu\), and with \(\rho = 0.995\) 
(which can happen) this multiplier is almost 20!

It turns out that, when considered as time series, MCMC 
samples for many quantities that you would want to monitor 
do behave a lot like \(AR_1\) processes, so (2.11) is a useful 
formula in figuring out how long the chain should be run to 
achieve your accuracy goals. I will have more to say on this 
matter in Section 2.4.

**Thinning the output.** One more practical point: consider 
a situation in which you'd like your estimate \(\hat{\theta}\) to have a high 
probability (95%, say) of being no more than (say) 0.1 from 
the correct posterior mean \(\mu = E(\theta|y)\)—in other words, you want 
\[ P(|\hat{\theta} - \mu| \leq d) = 1 - \epsilon \tag{2.12} \]
for \(d = 0.1\) and \(\epsilon = 0.05\). This is a sample size calculation, 
of the type that arises frequently when people design surveys, 
and the usual thing to do is to appeal to the Central 
Limit Theorem (CLT)—\(\hat{\theta}\) is, after all, just a sample mean. 
Now it is another interesting fact from time series (e.g., An-
derson, 1971) that $AR_1(\rho)$ processes do obey the CLT (even though they are not IID unless $\rho = 0$), so—provided that your series looks like an $AR_1$ (I'll cover how to check this in Section 2.4)—the standard result from the sampling literature (e.g., Cochran, 1977), using the SE calculation in (2.11), yields the requirement that

$$n_M = \frac{\sigma^2 (1 + \rho) \left[ \Phi^{-1}(1 - \frac{5}{2}) \right]^2}{d^2 (1 - \rho)},$$

(2.13)

where $\sigma$ is the SD of the $\theta_i$ and $\Phi$ is the usual standard normal CDF.

As is often the case with sample size calculations, the right side of (2.13) involves things you don't know, in this case $\rho$ and $\sigma$. The natural thing to do here is to make a trial run of the sampler to estimate these quantities. Suppose you get $\hat{\rho} = 0.89$ (in other words, your chain is not mixing very well) and $\hat{\sigma} = 3.3$. Then (2.13) produces the rather sobering estimate $\hat{n}_M = 79,500$, which we may as well round up to 80,000. Add a burn-in of (say) 5,000 iterations and we are up to 85K.

With the speed of today's machines (and the fact that next year's CPUs will probably be about twice as fast as today's), actually doing the 85,000 iterations may not be so bad, as far as clock time is concerned: if the trial run has shown that your computer can do about 50 iterations a second, for instance, 85K iterations works out to about 28 minutes, which might motivate a pleasant coffee break. But suppose $\theta$ has $k = 10$ components, and your worst-case $\hat{n}_M$ across all 10 parameters is 80,000. Then disk space starts to become an issue, as follows.

The MCMC data set will have 80,000 rows and $k + 1 = 11$ columns (including one for the iteration number). If you write it out to a character file for future data analysis, to obtain the posterior summaries of interest to you, in each row you'll need to allow 5 characters for the iteration counter and 7 characters for each parameter (given, say, 5 significant digits, a decimal point, and a space between each value). With $k = 10$ this comes out to $80,000 \cdot (5 + 7 \cdot 10) = 6$ megabytes of storage, and five or 10 runs like that can clog up your hard disk in no time.
PRACTICAL IMPLEMENTATION ISSUES

So what most people would do in this situation is to make a long run of 80,000 but to only store every \(n_T\)-th row of the MCMC data set—this is called thinning the output of the chain. Here to hold the stored data set down to (say) 5,000 rows (which would only take up about 375K on disk), you would take \(n_T = \frac{80000}{5000} = 16\).

In situations with extremely high serial correlation, I have sometimes needed to make monitoring runs of 1,000,000 or so iterations, storing every 200th, and I bet some readers of this book have made substantially longer runs than that, so thinning can be quite handy. It also acts to (greatly) reduce the serial correlation exhibited by the rows of the stored MCMC data set, although of course you still have to compute all \(n_M\) rows even if you store far less than \(n_M\) of them.

**Choosing a PD \(f(\theta|\theta_t)\).** This is the hardest of the three practical tasks to pin down with any generality. Since the Hastings algorithm works for (just about) any PD, in fact there isn’t just one Hastings solution to a given problem, there’s an infinity of such solutions. The main goal in choose \(f(\theta|\theta_t)\) is getting a chain that mixes well, 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:

1. Pick a PD that looks like a somewhat overdispersed version of the posterior you are trying to sample from (e.g., Tierney, 1994). Some work is naturally required to overcome the circularity inherent in this choice (if I knew \(p(\theta|y)\), why would I be using this algorithm in the first place?).

2. Set the PD up so that the expected value of where you are going to move to \((\theta^*)\), given that you accept a move away from where you are now \((\theta_t)\), is to stay\(^\text{12}\) where you are now: \(E_f(\theta^*|\theta_t) = \theta_t\). 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.

The first chapter in Gilks et al. (1996a) has lots of good general ideas for choosing PDs. I will deal with this issue in the main body of this book principally by example, although in Section 2.6 and Appendix 2 Section 6 I will describe a fairly general strategy for
Application to (2.7). Even if I didn’t know the right answer (2.8) in this problem, a good place to begin in choosing the PD using idea (1) above—based on the form of the prior, and therefore the possible form of the posterior (given at least approximate conjugacy)—would be a scaled inverse $\chi^2$ distribution: 

$$f(\sigma^2|\sigma_0^2) = SI-\chi^2(\nu^*,\sigma_0^2)$$

for some $\nu^*$ and $\sigma_0^2$. This distribution (Appendix 1) has density

$$p(\sigma^2|\nu^*,\sigma_0^2) = c(\sigma_0^2)^{\nu^*-1} (\sigma^2)^{\nu^*-1} \exp\left(-\frac{\nu^*\sigma_0^2}{2\sigma^2}\right)$$

(2.14)

and mean $\nu^*^\frac{-2}{\nu^*-2} \sigma_0^2$ for $\nu^* > 2$. To use idea (2) above, then, I can choose any $\nu^*$ greater than 2 that I want, and as long as I take $\sigma_0^2 = \nu^* \sigma_0^2$ I will have centered the PD at $\sigma_0^2$ as desired. So I will use

$$f(\sigma^2|\sigma_0^2) = SI-\chi^2\left(\nu^*, \frac{\nu^* - 2}{\nu^*} \sigma_0^2\right).$$

(2.15)

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.

Section 1 of Appendix 2 contains some S+ functions to do Hastings sampling in this model with PD (2.15). Various details need filling in, as follows.

- **PD simulation.** S+ doesn’t have a built-in function to sample from the scaled inverse $\chi^2$ distribution, but it does generate random $\chi^2$ draws nicely. As in Section 1.8, a bit of distributional manipulation bridges the gap: it turns out (Appendix 1) that

$$\sigma^2 \sim SI-\chi^2(\nu, \sigma^2) \iff \frac{1}{\sigma^2} \sim \Gamma\left(\frac{\nu}{2}, \frac{\nu}{2}\sigma^2\right).$$

(2.16)

Now $\chi^2$ distributions are just special gamma distributions—

$$\chi^2_{\nu} = \Gamma\left(\frac{\nu}{2}, \frac{1}{2}\right)$$

—so if I could get the second parameter in (2.16) to be $\frac{1}{2}$ I’d be home. But the second parameter in gamma distributions is an inverse scale parameter, by which I mean that multiplying a $\Gamma(\alpha, \beta)$ random draw by $c$ turns it into a $\Gamma(\alpha, \frac{\beta}{c})$ draw. So evidently

$$\frac{\nu s^2}{\sigma^2} \sim \Gamma\left(\frac{\nu}{2}, \frac{1}{2}\right) = \chi^2_{\nu},$$

(2.17)

meaning that to generate a random draw $\sigma^2$ from $SI-\chi^2(\nu, \sigma^2)$
you just generate a draw \( d \) from \( \chi_d^2 \) and compute \( \sigma^2 = \frac{\nu_p^2}{d} \). This explains the function `PD_sim`.

- **Log prior and log likelihood.** In the function `alpha` I compute the acceptance probability \( \alpha_H \) in (2.5) by calculating \( \exp(\log(\alpha_H)) \), so I need to compute the log posterior and log PD densities at various points. The log posterior is in turn just the sum \( \log(\text{prior}) + \log(\text{likelihood}) \). The \( SI-\chi^2 \) prior density was given in (2.14), except that here I am using \( \nu_p \) and \( \sigma_p^2 \) in place of \( \nu^* \) and \( \sigma^2 \); its logarithm is

\[
\log [p(\sigma^2|\nu_p, \sigma_p^2)] = c - \left( \frac{\nu_p}{2} + 1 \right) \log(\sigma^2) - \frac{\nu_p \sigma_p^2}{2\sigma^2}. \tag{2.18}
\]

Note from the form of \( \alpha_H \) that the constant \( c \) in (2.18) doesn’t need to be computed—\( e^c \) cancels in the acceptance ratio—so I have used \( c = 0 \) in the function `log_prior`.

The likelihood function in model (2.7) is a simple Gaussian:

\[
l(\sigma^2|y) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[ -\frac{1}{2\sigma^2} (y_i - \mu)^2 \right], \quad \text{so}
\]

\[
\log [l(\sigma^2|y)] = c - \frac{n}{2} \log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu)^2, \tag{2.19}
\]

and again for the same reason I have taken \( c = 0 \) in the function `log_lik`.

- **Log PD calculation.** I am using the PD (2.15), and in view of (2.14) the log of this density with \( \sigma^2 = \frac{\nu^*}{\nu^* - 2} \sigma_t^2 \) can, after a bit of simplification, be written

\[
\log [p(\sigma^2|\sigma_t^2)] = c + \frac{\nu^*}{2} \log(\sigma_t^2) - \left( \frac{\nu^*}{2} + 1 \right) \log(\sigma^2) - \frac{(\nu^* - 2) \sigma_t^2}{2\sigma^2}, \tag{2.20}
\]

and once again I used \( c = 0 \) in the function `log_PD`.

This may all seem like a lot of work, but in fact much of the process of creating a Hastings sampler is generic: for instance, the driver, acceptance probability, and log posterior functions require little change from problem to problem. You will probably find that once you have written one Hastings sampler from scratch, it doesn’t take much effort to do another one. The same is true for the other
kinds of MCMC samplers as well (although there are various tricks to learn to get decent mixing in high dimensions).

![Graphs](image)

**Figure 2.3. Output of a Hastings sampler in the Gaussian model (2.7):**

Time series trace (left panel) and density trace (right panel) for $\sigma^2$. The solid curve on the right is based on a kernel density estimate from the 5,000 stored iterations; the dotted curve is the theoretical density.

**Hastings results for model (2.7).** Figures 2.3 and 2.4 present results from applying the sampling strategy outlined above to the Gaussian model (2.7) with known mean and unknown variance, using the NB10 data for illustration. I chose $\nu^* = 20$ in specifying my PD (in Section 2.4 I'll justify this choice), and I used a burn-in of 1,000 starting from $\sigma_0^2 = s_0^2 = 41.402$, followed by a monitoring run of 40,000, storing every 8th iteration. For illustration, I set $\mu$ to $\bar{y} = 404.59$, and took $(\nu_p, \sigma_p^2) = (0.001, 41.402)$ (I will describe a more scientifically relevant prior in Section 2.6). This run took about 5.5 minutes using $S^*$ on a 333Mhz machine, and yielded an acceptance rate of about 44%, which (as we will see in Section 2.4) leads to pretty good mixing (not far from best possible with a $SI-\chi^2$ PD in this problem, in fact). Figure 2.3 plots the monitored iterations for $\sigma^2$ in two ways, and Figure 2.4 shows draws from the predictive distribution for a future $y^*$; in both cases the left panel is a time series trace of the 5,000 stored iterations, and the right
panel compares a kernel density trace based on the 5,000 draws with the theoretical density.

In both figures the time series traces look a lot more like white noise than Figure 2.2, which was produced by choosing $\nu^* = 5$ (leading to an acceptance rate of only about 20%); here the serial correlations for $\sigma^2$ and $y^*$ were 0.03 and 0.00, respectively, compared with $\rho = 0.94$ back in Figure 2.2. And you can see that (apart from the vagaries of slightly undersmoothed kernel density traces) the MCMC distributions match their theoretical counterparts well. The posterior means, SDs, and 95% central intervals for $\sigma^2$ and $y^*$ from the MCMC output are (42.2; 6.09; (32.0, 55.3)) and (404, 6.43, (392, 417)), respectively, which pretty closely match their theoretical values (42.2; 6.10; (32.0, 55.6)) and (405, 6.50, (392, 417)).

Figure 2.4. Similar to Figure 2.3, but the quantity being monitored here is a future value $y^*$ in model (2.7).

**S+ versus C.** By virtue of its relatively friendly syntax, graphics capabilities, and interactive nature, S+ is an excellent environment in which to develop statistical software. Some of these features, however, act to hobble it sufficiently for MCMC-style calculations that it may not be the best environment in which to run such software. The reasons are as follows:

- The S+ people have implemented a design philosophy that in-
cludes a desire for their program to recover gracefully if interrupted in the middle of (essentially) any calculation. This is good in many ways but comes at a price: relatively poor dynamic memory management, particularly with explicit looping (of the kind that is unavoidable in MCMC: how can you avoid writing something like for (i in 1:(n.burnin + n.monitor)))?). Naive versions of the S+ programs in Appendix 2 routinely crash with the message Unable to allocate dynamic memory, even with only 2 or 3 parameters and (say) 20,000 monitoring iterations. In Appendix 2 I have implemented a trick I learned from Brian Ripley to overcome this problem; using this idea, the code can be run with far larger values of n.monitor without crashing.

- However, even with this memory-allocation trick S+ MCMC code tends to run fairly slowly, because S+ is an interpreted—rather than a compiled—language.

What is needed is to be able to throw a switch: to program in interpretive (interactive) mode while developing the code, and then switch over to compiled (more like old-fashioned batch) mode to get results. Some readers of this book are probably using other software environments, such as GAUSS and MATLAB, in which (I think; manuscript readers: is this correct?) such a switch may be thrown; I have stuck here with S+—which has no such switch—because it is the most widely used academic statistical software environment worldwide and because I’m familiar with it. But I evidently need a way to toggle between S+ and a faster run-time environment.

One reasonably simple option is to convert your S+ code into C once you have debugged it. Section 2 of Appendix 2, for instance, contains a C version of the S+ Hastings sampler used above in model (2.7). When I began writing this book I was a complete C novice, but with the help of a few able graduate students it only took about an hour to translate the S+ code for this example into a working C program (and experience—plus the generic nature of MCMC sampler coding—have cut this time substantially on subsequent problems. The C code takes a lot more lines, mainly because you have to supply your own random number generators, but these only need to be written once and stored in a library).

The point of spending this hour is run-time efficiency: the same code that took 5.5 minutes in S+ on a 333Mhz machine only took 6.5 seconds in C. I’m not claiming that the programs in Appendix 2
are best possible; I'm just noting that in this problem, a reasonably straightforward C implementation was 51 times faster than a reasonably straightforward S+ implementation (and I have seen other MCMC applications where the advantage for C is more like 350 to 1). If I am only going to run my sampler a few times and I can get decent results with 40,000 iterations, then I'd rather wait 5 extra minutes for the S+ code to finish than translate it into C, but what if (a) I want to write a simulation program in which random data sets are analyzed with MCMC, or (b) it takes more like 800,000 iterations to get accurate findings?

The MCMC moral seems to be: either find a statistical programming environment you're happy with—in which you can toggle back and forth between interpretive and compiled mode—or get somebody (and it may be turn out to be you) to convert your slow code into fast code.  

**Metropolis sampling.** All of this section so far has been about implementing the Hastings (1970) sampler, which often requires a bit of ingenuity in the choice of proposal distribution (PD). There is, however, a simpler MCMC approach, due to Metropolis et al. (1953), as follows.

If you look at the form of the acceptance probability (2.5) in Hastings sampling, you will see that one particular assumption about the proposal distribution would make things easier: if the PD were *symmetric* in its two arguments, $\theta_t$ (which you will recall is where the chain is now) and $\theta^*$ (where the chain is thinking of going)—in other words, if $f(\theta^* | \theta_t) = f(\theta_t | \theta^*)$—then the ratio $f(\theta_t | \theta^*) / f(\theta^* | \theta_t)$ in (2.5) would cancel. This was the original idea Metropolis and his co-authors had almost 20 years before Hastings generalized it: Metropolis et al. suggested the use of symmetric PDs, and Hastings pointed out that PDs didn't have to be symmetric. Thus

**Algorithm** (Metropolis et al., 1953). Same as Hastings (2.5, 2.6), except that the proposal distribution (PD) $f(\theta | \theta_t)$ must be symmetric: in other words, it must satisfy $f(\theta^* | \theta_t) = f(\theta_t | \theta^*)$. In this case the acceptance probability simplifies to

$$\alpha_M(\theta_t, \theta^*) = \min \left[ 1, \frac{p(\theta^* | y)}{p(\theta_t | y)} \right].$$

Notice that this automatically satisfies heuristic idea (2) in the
section earlier on choosing a good PD: symmetric proposals make unbiased moves (in a left-right sense along the number line).

Now why is (2.21) easier? Well, it often makes choosing the PD more straightforward: people just tend to implement Metropolis with their favorite symmetric distribution. One possibility, for instance, is to propose a $U(-c, c)$ move from where you are now (this was in fact what Metropolis et al. suggested); another possibility is to make a $N(0, \sigma^2_{PD})$ move, where $c$ and $\sigma^2_{PD}$ play the same tuning constant role that $\nu^*$ did in the Hastings example above. Notice that $c$ and $\sigma^2_{PD}$ are both scale factors in their respective PDs: presumably you can tune them to get an acceptance rate that leads to good mixing.

If I propose $U(-c, c)$ moves from where I am now and you use $N(0, \sigma^2_{PD})$ moves, our PDs would both have the feature that the probability of generating a move to $\theta^*$ from $\theta_i$ depends only on the distance $|\theta^* - \theta_i|$ between the target and current locations—in other words, in both cases there is a univariate density $h$ such that $f(\theta^*|\theta_i) = h(|\theta^* - \theta_i|)$. People call an MCMC sampler based on such a proposal a random-walk Metropolis (or Hastings) algorithm (because the output of the sampler, examined only at the times when you actually do make a move, forms a random walk (e.g., Feller, 1968) in $\mathbb{R}^k$). These samplers are an important special case of the general Metropolis idea, since it turns out both that they are easy to program and they tend to have good MCMC convergence properties (e.g., Roberts 1996).

None of this sounds particularly applicable to the normal variance problem (2.7) I tackled above with Hastings, however: after all, it would look funny to propose a symmetric $U(-2, 2)$ (say) move from $\sigma^2 = 0.5$ (say), with a big chance of going negative, when everybody knows that $\sigma^2$ has to be positive. A moment’s reflection indicates the way out of this problem, though: since $U(-c, c)$ and $N(0, \sigma^2_{PD})$ moves would cause you to travel (in principle) all over the entire real line, the parameter you’re sampling had better live on the whole line, too. That’s easily enough accomplished with a parameter like $\sigma^2$: just work instead with $\lambda \equiv \log(\sigma^2)$, and monitor the function $\sigma^2 = e^\lambda$ in your MCMC draws.

Parameter transformation. On this line of reasoning I want to work with model (2.7) except re-expressed in terms of $\lambda$. If you look at the log likelihood (2.19) for this model, you will see that the only change needed there is to stick in $\lambda$ every time you see $\log(\sigma^2)$, and $e^\lambda$ everywhere $\sigma^2$ appears, so the new log likelihood
The log prior, which will be based on (2.18), requires a bit more work, though: I have to put in the Jacobian for going from one parameterization to the other. With \( \lambda = g(\sigma^2) = \log(\sigma^2) \), the usual result from probability, written on the log scale, is

\[
\log[p(\lambda|\nu_p, \sigma_p^2)] = \log\{p_{\nu_2}[g^{-1}(\lambda)|\nu_p, \sigma_p^2]\}
\]

\[
+ \log \left( \left| \frac{\partial g^{-1}(\lambda)}{\partial \lambda} \right| \right). \tag{2.23}
\]

Here \( g^{-1}(\lambda) = e^\lambda \), \( \left| \frac{\partial g^{-1}(\lambda)}{\partial \lambda} \right| = e^\lambda \), and from (2.18) \( \log\{p_{\nu_2}[g^{-1}(\lambda)|\nu_p, \sigma_p^2]\} = c - \left( \frac{\nu_p}{2} + 1 \right) \lambda - \frac{\nu_p \sigma_p^2}{2e^\lambda} \), so when all the dust settles

\[
\log[p(\lambda|\nu_p, \sigma_p^2)] = c - \frac{\nu_p}{2} \lambda - \frac{\nu_p \sigma_p^2}{2e^\lambda}. \tag{2.24}
\]

By way of proposal distribution I will use a Gaussian\(^{17} \) centered at where I am now and with SD \( \sigma_{PD} \), but what should this SD be? Heuristic idea (1) in the earlier section on choosing a PD suggested making the PD look like a somewhat overdispersed version of the posterior distribution, so maybe that would work here. You may recall from earlier study of maximum likelihood estimates (e.g., Lehmann, 1983) that when \( n \) is fairly large the MLE \( \hat{\lambda} \) for \( \lambda \) should have approximate sampling distribution

\[ \hat{\lambda} \sim N(\lambda, \sigma_\lambda^2), \tag{2.25} \]

where \( \sigma_\lambda^2 = \hat{I}_\lambda^{-1} \) is the reciprocal of the observed Fisher information evaluated at the MLE,

\[ \hat{I}_\lambda = -\left( \frac{\partial^2}{\partial \lambda^2} \log [I(\lambda|y)] \right)_{\lambda=\hat{\lambda}}. \tag{2.26} \]

As long as the amount of prior information is small in relation to the data information, (2.25) implies that the posterior for \( \lambda \) will be approximately

\[ (\lambda|y) \sim N(\hat{\lambda}, \sigma_\lambda^2), \tag{2.27} \]

so heuristic idea (1) suggests in this case a PD of the form

\[ f(\lambda|\lambda_t) = N(\lambda_t, \kappa \sigma_\lambda^2) \tag{2.28} \]

for some scaling factor \( \kappa > 1 \).
To implement this I still have to compute $\sigma_\lambda^2$. Differentiating the log likelihood (2.22) yields
\begin{equation}
-\frac{\partial^2}{\partial \lambda^2} \log L(\lambda|y) = \frac{\sum_{i=1}^{n} (y_i - \mu)^2}{2\sigma^2},
\end{equation}
which simplifies considerably when evaluated at the MLE: in this model $\hat{\sigma}^2_{\text{MLE}} = \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mu)^2$, so $\sigma_\lambda^2$ just reduces to $\frac{2}{n}$. Evidently I should use $\sqrt{\frac{2\sigma^2}{n}}$ as my proposal distribution SD, varying $\kappa$ to get a decent acceptance rate.

I will talk more about how to choose the optimal $\kappa$ in the next section; for now it is worth noting that intuitively there ought to be a best $\kappa$ somewhere in the middle of its possible range, because

- If the proposal distribution SD is too big, when you do move you will make big moves, which is good, but you won’t accept such a move very often (look at the form of the acceptance probability $\alpha_M$ in (2.21)), which is bad, and in the limit as the SD gets huge you will hardly ever move, leading to high autocorrelation and terrible mixing; and

- If the proposal distribution SD is too small, you’ll accept the resulting moves frequently (which is good), but when you move you won’t move very far (which is bad), and if you mentally let the SD go to 0 you’ll see that again you have high autocorrelation and terrible mixing, because it will take the chain a very long time to flesh out the whole posterior.

Two qualitative conclusions emerge from this: proposal distribution SDs—and acceptance probabilities—somewhere in the middle of their possible ranges are best; and

To increase the Metropolis or Hastings acceptance probability, you should decrease the proposal distribution SD.

This is why $\nu^* = 20$ worked better than $\nu^* = 5$ with the Hastings PD in the last section: the variance of a $SI-\chi^2(\nu^*, \sigma^2)$ distribution (Appendix 1) for $\nu^* > 4$ is $\frac{2\nu^* \sigma^4}{(\nu^* - 2)(\nu^* - 4)}$, which goes down as $\nu^*$ increases, so when $\nu^* = 5$ produced an acceptance rate that was too low (in other words, the proposal distribution SD was too big) the right thing to do was to increase $\nu^*$.

*Metropolis results for model (2.7).* Section 3 of Appendix 2 contains a set of S+ functions to do Metropolis sampling in model (2.7) using the PD developed above. You can see how little needs to be changed from the Hastings code earlier in that appendix. To test
the code I used all of the same settings as with the Hastings results (among other things, this yielded a prior on $\lambda$ that was equivalent to the previous highly diffuse prior on $\sigma^2$) and took $\kappa = 6$ (I'll explain this value in Section 2.4). The ensuing run took 8.8 minutes at 333Mhz (the extra time was almost entirely due to writing out three monitored quantities—$\lambda$, $\sigma^2$, and $y^*$—instead of two) and produced results for $\sigma^2$ and $y^*$ that were identical to those from the Hastings approach, apart from MCMC sampling noise. Figure 2.5 is a plot of the results for $\lambda$, with the normal approximation (2.27) superimposed on top of the kernel density trace from the Metropolis output—you can see that $n = 100$ is sufficient to have reached asymptotic nirvana (to decent accuracy, at least) with this data set.

![Figure 2.5. Output of a Metropolis sampler in the Gaussian model (2.7): Time series trace (left panel) and density trace (right panel) for $\lambda = \log(\sigma^2)$. The solid curve on the right is based on a kernel density estimate from the 5,000 iterations; the dotted curve is the normal approximation (2.27), based on the MLE, to the posterior density.](image)

2.4 MCMC monitoring and convergence diagnostics

I have been promising for some time now to discuss methods for figuring out whether the chain is in equilibrium yet, and how long to run it after it has reached equilibrium. This has been an active
research area in the last 10 years (e.g., Brooks and Roberts, 1995; Cowles and Carlin, 1996) and will certainly continue to develop, but a number of useful methods have already been documented, as follows.

The first thing I often do is make a graph like Figure 2.6, which for want of a better name I will call an MCMC 4-plot. To create this picture, I reran my Hastings sampler from Section 2.3 on the parameter $\sigma^2$ with a far-from-optimal value of $\nu^*=-2.5$—and used a burn-in of 1,000 and a monitoring run of 5,000. This took about 1 second at 333Mhz and produced (on purpose) an abysmally low acceptance rate of only about 7%.

![Figure 2.6. MCMC 4-plot of results from the Hastings sampler in Section 2.3 on $\sigma^2$, with $n_B = 1000, n_M = 5000$, and $\nu^* = 2.5$.](image)

The upper left panel in Figure 2.6 is a time series trace of the 5,000 monitoring iterations. Here it shows pretty bad mixing: note that (a) the chain did not move at all for significant periods, and (b) if you ran a kind of mental “local smoother” through the plot, trying to estimate the mean of the time series near any given point, it would show a lot of wavy behavior, whereas white noise would just look like random fluctuations around a horizontal line. The upper right panel is a (considerably smoothed) kernel density trace of the monitored iterations\textsuperscript{18}, and (in spite of the poor mixing) already looks a lot like the correct answer (compare with the right panel in Figure 2.3).
The lower left and right panels in Figure 2.6 are plots of the autocorrelation and partial autocorrelation functions (ACF and PACF; e.g., Box and Jenkins, 1976) for the 5,000 correlated draws from the posterior for $\sigma^2$. If you have studied the time-domain approach to time series, you will recognize that these plots are exhibiting the textbook behavior of an $AR_1$ series with a first-order autocorrelation of about 0.9: the PACF has one big spike at lag 1 of size 0.9, and the rest of the spikes are negligible (the dotted lines are two standard error traces around 0 for judging which spikes are worth taking notice of), and the ACF shows a slow geometric-style decay from an autocorrelation of 0.9 at lag 1 to values near 0 out around lag 35 (or even later).

Taken together the panels of the 4-plot show a chain that could well be in equilibrium (I don’t see any vertical drift in the time series trace) but that likely needs to be run for considerably longer than 5,000 iterations to get accurate posterior summaries (because of the high serial correlation). With S+ handy, the easiest way to figure out how much longer is to invoke the MCMC diagnostic routines in a package called CODA.

**The CODA diagnostics.** CODA (Best et al., 1995) is a set of S+ functions available free on the web or by anonymous ftp from the Medical Research Council Biostatistics Unit in Cambridge, UK (see Appendix 2 for details on how to get the code). These functions offer six different kinds of MCMC convergence diagnostics, some of which I will now describe.

The simplest things you can get out of CODA are numerical estimates of the autocorrelation functions for each monitored quantity and the degree of cross-correlation exhibited by all the different time series you have generated, taken pairwise. I ran CODA on the Hastings output illustrated in Figure 2.6, obtaining the results in Tables 2.3–2.6 (I also monitored the predictive distribution for a future observation $y^*$). Section 1 of Appendix 2 gives an S+ function called preCODA to prepare the MCMC data set for reading by CODA.

**Autocorrelations.** Table 2.4, for example, gives the autocorrelations for $\sigma^2$ and $y^*$ and the degree of cross-correlation between them. The ACF for $\sigma^2$ is a numerical match to the upper left panel in Figure 2.6, and shows the slow decline in the autocorrelations (you have to go all the way out to nearly lag 50 with this choice of $\nu^*$ to get close to IID sampling). You can also see that there
is little correlation between $\sigma^2$ and $y^*$, and that—even with this $\nu^*—y^*$ looks like white noise.

Table 2.4. Autocorrelations and cross-correlations for the Hastings output illustrated in Figure 2.6.

**LAGS AND AUTOCORRELATIONS WITHIN EACH CHAIN:**

<table>
<thead>
<tr>
<th>Chain</th>
<th>Variable</th>
<th>Lag 1</th>
<th>Lag 5</th>
<th>Lag 10</th>
<th>Lag 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1</td>
<td>sigma2</td>
<td>0.5940</td>
<td>0.5560</td>
<td>0.3300</td>
<td>0.02100</td>
</tr>
<tr>
<td></td>
<td>y.star</td>
<td>0.00429</td>
<td>0.00159</td>
<td>0.01030</td>
<td>0.00600</td>
</tr>
</tbody>
</table>

**CROSS-CORRELATION MATRIX:**

Chain: hastings1

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>sigma2</th>
<th>y.star</th>
</tr>
</thead>
<tbody>
<tr>
<td>sigma2</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>y.star</td>
<td>-0.00716</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

_Geweke and Heidelberger-Welch._ Two other useful MCMC diagnostics produced by C\textsc{oda} are due to Geweke (1992) and Heidelberger and Welch (1983). Geweke proposed a simple method based on time series ideas. He reasoned that, if the chain were in equilibrium, the means of the first (say) 10% and the last (say) 50% of the iterates should be nearly equal. So to calculate his diagnostic he just does a $Z$-test of the hypothesis of equality of these two means, and reports the resulting $Z$ scores (on the usual standard normal scale), one for each monitored quantity. Thus Geweke $Z$-scores a lot bigger than (say) 2 in absolute value indicate that the mean level of the time series is still drifting, even after whatever burn-in you have already done, and you should rerun your chain with a longer burn-in before starting your monitoring. Here (Ta-
there is perhaps a hint that a longer burn-in would have been useful for \( \sigma^2 \).

Table 2.5. Geweke diagnostics
for the Hastings output illustrated in Figure 2.6.

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>hasting1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>---------</td>
<td>---</td>
</tr>
<tr>
<td>sigma2</td>
<td>-1.760</td>
<td></td>
</tr>
<tr>
<td>y.star</td>
<td>0.626</td>
<td></td>
</tr>
</tbody>
</table>

Heidelberger and Welch (1983) proposed a diagnostic approach that uses the Cramer-von Mises statistic\(^{20}\) to test for stationarity. CODA’s implementation of the Heidelberger-Welch approach goes like this:

- If overall stationarity fails for a given quantity being monitored, CODA discards the first 10% of the series for that quantity and recomputes the Cramer-von Mises statistic statistic, continuing in this manner until only the final 50% of the data remain.

- If stationarity still fails with the last half of the data, then CODA reports overall failure of the stationarity test.

- CODA also computes a half-width test, which tries to judge whether the portion of the series that passed the stationarity test is sufficient to estimate the posterior mean with a particular default accuracy. The idea is to use time-series methods to estimate the standard error of the mean of the MCMC draws and then compute half of the width of the resulting frequentist 95% interval estimate for this mean (namely, 1.96 times the standard error). If this is less than the default tolerance (in CODA, set to \( \epsilon \) times the sample mean, for \( \epsilon = 0.1 \)), the retained portion of the chain passes the half-width test. **NB** (1) This is not very stringent—if you use this test in CODA, you may well wish to make \( \epsilon \) smaller. (2) The half-width test is directly related to equations (2.11–2.13).

Here, as Table 2.6 indicates, even this rather poorly-mixing chain
gets over the Heidelberger-Welch hurdles with no problem, reinforcing the visual impression of no troubles with stationarity in Figure 2.6.

Table 2.6. Heidelberger-Welch diagnostics
for the Hastings output illustrated in Figure 2.6.

HEIDELBERGER AND WELCH STATIONARITY AND INTERVAL HALFWIDTH TESTS:

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>Stationarity test</th>
<th># of iters. to keep</th>
<th># of iters. to discard</th>
<th>C-vonM stat.</th>
</tr>
</thead>
<tbody>
<tr>
<td>sigma2</td>
<td>passed</td>
<td>5000</td>
<td>0</td>
<td>0.321</td>
</tr>
<tr>
<td>y.star</td>
<td>passed</td>
<td>5000</td>
<td>0</td>
<td>0.195</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>Halfwidth test Mean Halfwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>sigma2</td>
<td>passed</td>
</tr>
<tr>
<td></td>
<td>42.9</td>
</tr>
<tr>
<td></td>
<td>0.499</td>
</tr>
<tr>
<td>y.star</td>
<td>passed</td>
</tr>
<tr>
<td></td>
<td>404.0</td>
</tr>
<tr>
<td></td>
<td>0.167</td>
</tr>
</tbody>
</table>

Raftery-Lewis. The CODA diagnostic I find the most useful is due to Raftery and Lewis (1992). Given the output of an MCMC sampler, Raftery and Lewis address the question of how long to monitor the chain, and in doing so they recognize that this in turn should be based on the answer to another question: how accurate do you want the posterior summaries to be? So they ask you, the user, to specify three things:

- Which quantiles of the marginal posteriors are you most interested in? Usually the answer is the 2.5% and 97.5% points, since they are the basis of a 95% interval estimate.
- With what minimum probability do you want to achieve your accuracy goals? The default is 95%.
- How accurate would you like the estimated quantiles of interest to be? This, in turn, can be measured in two different ways: taking the 0.025 percentile as an example, you could either specify
that the quantile \( q \) corresponding to the 0.025 point in the CDF be accurate to a given tolerance, or that the area to the left of the reported quantile be within a given margin of 0.025. Raftery and Lewis have opted for the latter (which does not seem to me to be the more natural choice), and the CODA default on this scale is 0.005—in other words, the default tries to set it up so that if you report a nominal 95% interval by quoting the 0.025 and 0.975 points in the MCMC output, the actual posterior probability of your interval will be between 0.94 and 0.96.

Here is how their methods work. Given a particular quantity \( \theta \) that you have monitored and a particular quantile \( q \) of interest in \( \theta \)'s distribution, Raftery and Lewis dichotomize the output of the chain, replacing that output by a binary time series that is 1 if \( \theta_t \leq q \) and 0 otherwise. They then assert that this binary chain should be approximately Markovian, and use standard results for two-state Markov chains to estimate how long the chain should be run to achieve the desired accuracy for the chosen quantile.

Table 2.7. Raftery-Lewis diagnostics
for the Hastings output illustrated in Figure 2.6.

RAFTERY AND LEWIS CONVERGENCE DIAGNOSTIC:

Quantile = 0.025
Accuracy = +/- 0.005
Probability = 0.95

Chain: hastings1

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>Thin</th>
<th>Burn-in</th>
<th>Total</th>
<th>Lower bound</th>
<th>Dependence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(k)</td>
<td>(M)</td>
<td>(N)</td>
<td>(Min)</td>
<td>factor (I)</td>
</tr>
<tr>
<td>sigma2</td>
<td>1</td>
<td>58</td>
<td>68727</td>
<td>3746</td>
<td>18.3</td>
</tr>
<tr>
<td>y.star</td>
<td>1</td>
<td>2</td>
<td>3866</td>
<td>3746</td>
<td>1.03</td>
</tr>
</tbody>
</table>

As you can see from Table 2.7, Raftery and Lewis actually provide three kinds of estimates, in columns 2–4: what thinning ratio to use, how much additional burn-in would be useful the next time you run the chain, and the required length of (burn-in + monitoring) period—let’s call it \( \hat{\eta}_RL \)—to achieve your accuracy goals. Column 5 shows the length of the chain required to meet those
goals if it had been white noise, and column 6 reports the ratio of columns 4 and 5, which Raftery and Lewis call the dependence factor $I$.

I don't usually find the Raftery-Lewis thinning-ratio and extra-burn-in recommendations very useful (I tend to decide on thinning requirements based on storage considerations, and the recommended extra burn-in is usually trivially small). The column called Total (N) — $\hat{n}_{RL}$ — is interesting, though: its punchline in this case, having used a proposal distribution with a deeply suboptimal value of $\nu^*$, is that I need to rerun the chain for almost 70K iterations, to achieve the Raftery-Lewis default accuracy goals for the endpoints of my 95% interval for $\sigma^2$.

Of course, with the C program in Appendix 2 this is not hard: increasing $n_B$ to 5,000 and $n_M$ to 70,000 and storing every $n_T = 14$ iterate only takes about 11 seconds at 333Mhz. This passes all tests and reduces the first-order serial correlation of the stored iterates to 0.288, yielding a new $\hat{n}_{RL}$ of 6,756, which is more than the effective sample size of the new run (5,000). This sort of thing often happens—the first estimate of $\hat{n}_{RL}$ is a bit conservative because it isn't based on enough data yet. Now 6756 - 14 = 95K, so I reran the chain for 100K iterates, storing every 20th, producing a new $\hat{n}_{RL}$ of 5391 (!), at which point I decided that the resulting answers would be close enough for government work (no, actually I kept on, out of curiosity, and I had to go all the way out to 140K, storing every 28th, before I had 5,000 $\sigma^2$ draws that passed the default accuracy goals. The moral seems to be that $\hat{n}_{RL}$ may well be biased on the low side when based on a modest number of draws).

**Optimizing the proposal distribution.** Now that $\nu^* = 2.5$ has proven itself to be a rotten tuning constant for the Hastings proposal distribution, the availability of CODA's $\hat{n}_{RL}$ facility makes me wonder what the optimal $\nu^*$ is? Table 2.8, which is based on a series of runs with $n_B = 5,000$, $n_M = 400,000$, and $n_T = 1$, investigates this question. You can see that as $\nu^*$ increases from its smallest value in the table, so does the (estimated) acceptance probability $\hat{\alpha}$, but both the autocorrelation $\hat{\rho}$ and the default $\hat{n}_{RL}$ reach a minimum in the middle, around $\nu^* = 20–30$. Thus, as we saw by reasoning qualitatively in Section 2.3, the best acceptance probability will be somewhere in the middle (in this case around 0.44–0.51).

As an alternative to Raftery-Lewis, a different but related way
to figure out what \( n_M \) should be is based on equation (2.13), which addresses an accuracy goal for the posterior mean rather than for percentiles: how many draws should you take so that the estimate of the posterior mean of \( \theta \) you quote is correct to within a tolerance \( d \) with probability \((1 - \epsilon)\? For instance, the posterior mean of \( \sigma^2 \) in model (2.7) with the NB10 data and a diffuse prior, based on the 5,000 draws shown in Figure 2.3, is 42.2, with a posterior SD of 6.1. How much longer than 5,000 should I have run the chain to confidently quote all three of the significant figures in the estimate 42.2?

Table 2.8. Optimal choice of \( \nu^* \) in the Hastings proposal distribution for the normal variance model (2.7). \( \text{SSIF = sample size inflation factor (see text).} \)

<table>
<thead>
<tr>
<th>( \nu^* )</th>
<th>( \hat{\rho} )</th>
<th>( \hat{\alpha} )</th>
<th>( \hat{n}_{RL} ) (thousands)</th>
<th>SSIF ( \left( \frac{1 + \hat{\rho}}{1 - \hat{\rho}} \right) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>0.903</td>
<td>0.068</td>
<td>94.6</td>
<td>19.6</td>
</tr>
<tr>
<td>5.0</td>
<td>0.743</td>
<td>0.202</td>
<td>31.9</td>
<td>6.78</td>
</tr>
<tr>
<td>10.0</td>
<td>0.652</td>
<td>0.320</td>
<td>20.7</td>
<td>4.75</td>
</tr>
<tr>
<td>20.0</td>
<td>0.625</td>
<td>0.443</td>
<td>18.1</td>
<td>4.33</td>
</tr>
<tr>
<td>25.0</td>
<td>0.632</td>
<td>0.482</td>
<td>17.8</td>
<td>4.43</td>
</tr>
<tr>
<td>30.0</td>
<td>0.643</td>
<td>0.513</td>
<td>17.0</td>
<td>4.60</td>
</tr>
<tr>
<td>40.0</td>
<td>0.667</td>
<td>0.563</td>
<td>19.0</td>
<td>5.01</td>
</tr>
<tr>
<td>50.0</td>
<td>0.688</td>
<td>0.598</td>
<td>19.5</td>
<td>5.41</td>
</tr>
<tr>
<td>100.0</td>
<td>0.779</td>
<td>0.698</td>
<td>23.8</td>
<td>8.05</td>
</tr>
<tr>
<td>500.0</td>
<td>0.928</td>
<td>0.858</td>
<td>50.3</td>
<td>26.8</td>
</tr>
</tbody>
</table>

The answer to this question in turn depends on \( \nu^* \). The part of equation (2.13) that is sensitive to the proposal distribution is the ratio \( \left( \frac{1 + \hat{\rho}}{1 - \hat{\rho}} \right) \), which I have termed the \textit{sample size inflation factor (SSIF)} and listed in the last column of Table 2.8. This is the amount that \( n_M \) needs to be multiplied by to satisfy accuracy goal (2.12), compared with its required value under IID sampling. With the best \( \nu^* \)—the value that minimizes \( \hat{\rho} \), namely \( \nu^* \approx 20 \)—the SSIF for Hastings sampling in this problem with a proposal distribution of the form (2.15) is 4.33. Now for the final 2 in 42.2 to be right, I need \( d = 0.05 \), and if I pick 95% as the desired level of confidence, equation (2.13) says that \( n_M \) would have to be \( 4.33 \cdot 1.96^2 \cdot 0.05 \approx 248K \) when you contrast this with what people often do (burn-ins of 1–5K followed by monitoring runs of 5–10K are common), it would seem that most of us (myself included,
until I made this calculation) do not run our samplers for as long as perhaps we should.

Figure 2.7. A graphical version of Table 2.8: \( \hat{\rho} \) (solid line), \( \hat{\alpha} \) (small-dashed line), \( \hat{n}_{RL} \) (dotted line), and the SSIF (large-dashed line) are plotted against \( \nu^* \).

Figure 2.7 endeavors to wrap all of this up in one plot. The horizontal scale, expressed logarithmically, is \( \nu^* \); the left-hand vertical scale is \( \rho \); the right-hand vertical scale is \( \alpha \); and columns 2–5 of Table 2.8 are plotted against column 1 in that table. You can see that, while the \( \hat{n}_{RL} \) and SSIF criteria do not quite agree on the optimal \( \nu^* \), the region of near-optimality is broad, extending from about 10 to about 50, and this corresponds in turn to a broad range of target values for the acceptance probability, in this case from about 0.3 to about 0.6. Gelman et al. (1996) report something similar in the case of a Gaussian model with unknown mean rather than variance: with a Gaussian PD and a criterion that is different yet again from Raftery-Lewis and SSIF, they find the best \( \alpha \) to be about 0.4, with values from 0.3 to 0.6 not far from optimal.

**Optimal Hastings versus optimal Metropolis.** The calculations in the previous subsection can be repeated with the Metropolis sampler on \( \log(\sigma^2) \) introduced earlier. Table 2.9, which summarizes the results, demonstrates behavior similar to that seen for Hastings, with the following exceptions:
• As noted above, \( \kappa \) is a scale factor, whereas \( \nu^* \) is inversely related to scale, so as \( \kappa \) increases the acceptance probability goes down.

• Metropolis appears a bit better-behaved than Hastings in this example, at least as far as \( \hat{n}_{RL} \) is concerned (in contrast, they are about equally good—or bad—when measured by serial correlation and the SSIF).

The optimal \( \kappa \) is around 6, corresponding to an acceptance rate of about 0.44, although the region of near-optimality is again quite flat. This also agrees pretty well with what Gelman et al. (1996) found—in their problem the optimal multiplier, on the variance scale, was about 5.8.

Table 2.9. Optimal choice of \( \kappa \) in the Metropolis proposal distribution for the normal variance model (2.7).

<table>
<thead>
<tr>
<th>( \kappa )</th>
<th>( \hat{\rho} )</th>
<th>( \hat{\alpha} )</th>
<th>( \hat{n}_M ) (thousands)</th>
<th>SSIF ( \left( \frac{1+\hat{\epsilon}_n}{1-\hat{\rho}} \right) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.902</td>
<td>0.830</td>
<td>45.1</td>
<td>19.4</td>
</tr>
<tr>
<td>1.0</td>
<td>0.781</td>
<td>0.705</td>
<td>24.4</td>
<td>8.13</td>
</tr>
<tr>
<td>2.0</td>
<td>0.700</td>
<td>0.609</td>
<td>17.7</td>
<td>5.67</td>
</tr>
<tr>
<td>3.0</td>
<td>0.660</td>
<td>0.547</td>
<td>15.1</td>
<td>4.88</td>
</tr>
<tr>
<td>6.0</td>
<td>0.630</td>
<td>0.438</td>
<td>13.2</td>
<td>4.41</td>
</tr>
<tr>
<td>8.0</td>
<td>0.634</td>
<td>0.393</td>
<td>13.8</td>
<td>4.46</td>
</tr>
<tr>
<td>10.0</td>
<td>0.644</td>
<td>0.360</td>
<td>14.7</td>
<td>4.62</td>
</tr>
<tr>
<td>30.0</td>
<td>0.732</td>
<td>0.224</td>
<td>20.4</td>
<td>6.46</td>
</tr>
<tr>
<td>100.0</td>
<td>0.837</td>
<td>0.126</td>
<td>36.4</td>
<td>11.3</td>
</tr>
</tbody>
</table>

2.5 Gibbs sampling

Case study 2.1 (continued). I am finally ready to return to the geological example with which the chapter began. Recall that the statistical setup was

\[
(A, B) \sim p(A, B)
\]

\[
(y_i | A, B) \overset{\text{iid}}{\sim} U(A, B), \quad i = 1, \ldots, n.
\]  \hspace{1cm} (2.30)

for observed locations \( y_i \), in meters below ground, of finds of one taxon of ammonites—a particular kind of fossil mollusk (Tables 2.1 and 2.2). Here \( A \) and \( B \) are the true lifespan limits of this taxon; let's pick \( M. \text{dens.} \) as an example.

Bayesian inference in this problem involves the usual two steps,
beyond figuring out the likelihood: I need to specify a scientifically-reasonable prior, and I need to figure out how to compute the marginal posteriors, for instance \( p(A|y) \). Actually, there is another task, as well: I can consider reparameterizing, for instance (as I did earlier) by defining \( \mu = \frac{B + A}{2} \), the center of the true range, and \( \sigma = \frac{B - A}{2} \), a measure of the scale of this range, and re-expressing the model as

\[
(\mu, \sigma) \sim p(\mu, \sigma) \\
(y_i|\mu, \sigma) \overset{\text{ID}}{\sim} U(\mu - \sigma, \mu + \sigma).
\] (2.31)

I find it easier to think about things in this location-scale parameterization, so that's how I'll approach the problem here, regarding \( A = \mu - \sigma \) and \( B = \mu + \sigma \) as derived quantities to be monitored rather than elicited.

As far as the prior goes, the main thing that was known about these ammonites prior to Macellari (1986), from which the data in Tables 2.1 and 2.2 were taken, is that they were from the late Cretaceous period. On the meters-below-ground scale on Seymour Island in the Antarctic Peninsula, the source of the data, this period corresponded roughly to the range from \( L = 400 \)m to \( H = 1700 \)m. For any given taxon, this implies a prior in which \( \mu \) can be pretty much anywhere between \( L \) and \( H \), with no particular values favored, and \( \sigma \) is quite free too, subject to the restriction that \( L < \mu - \sigma \) and \( \mu + \sigma < H \). Rearranging these two inequalities and insisting that \( \sigma > 0 \), in keeping with a scale parameter, gives the prior

\[
\mu \sim U(L, H) \\
(\sigma|\mu) \sim U[0, \min(\mu - L, H - \mu)].
\] (2.32)

Here

\[
\min(\mu - L, H - \mu) = \left\{ \begin{array}{ll}
\mu - L & \text{for } L < \mu < \mu_* \\
H - \mu & \text{for } \mu_* < \mu < H
\end{array} \right.,
\] (2.33)

where \( \mu_* \) is such that \( \mu_* - L = H - \mu_* \); in other words, \( \mu_* = \frac{L + H}{2} = 1050 \), and another restriction that emerges is thus that \( \sigma < \mu_* - L = H - \mu_* = 650 \).

From (2.31), the likelihood for a single observation in this model is

\[
l(\mu, \sigma|y_i) = \frac{1}{2\sigma} I(\mu - \sigma < y_i < \mu + \sigma),
\] (2.34)

from which after a bit of thought you can see that the complete-
sample likelihood is

\[ l(\mu, \sigma | y) = \frac{1}{(2\sigma)^n} I[\mu - \sigma < \min(y)] I[\max(y) < \mu + \sigma]. \tag{2.35} \]

Now it is not a lot of fun to multiply (2.32) and (2.35), work out the normalizing constant, and integrate out one of the parameters to get the marginal posterior for the other one, so I am going to use MCMC here. Hastings and Metropolis are certainly possibilities (Problem 2.4), but I thought this would be a good chance to see the third main MCMC technique, Gibbs sampling, in action, so let's see how that goes.

**Gibbs sampling.** The idea behind Gibbs sampling, which (as I mentioned earlier) dates to work by Geman and Geman (1984) in image analysis, is a kind of what-if that is related to the EM algorithm (Baum et al., 1970; Dempster, Laird, and Rubin, 1978), a method developed to do maximum likelihood and Bayesian inference in models with missing information. Given a parameter vector \( \theta = (\theta_1, \ldots, \theta_k) \) with prior \( p(\theta) \), and a sample \( y \) with likelihood \( l(\theta | y) \), you may well notice that the full posterior \( p(\theta | y) = c p(\theta) l(\theta | y) \) is not so easy to work with, but it would become a lot easier if you only knew the value of some other (missing) information \( z \)—in other words, suppose that \( p(\theta | y, z) \) is more tractable than \( p(\theta | y) \), and could be used to estimate \( \theta \) (for instance, by taking the posterior mode \( \hat{\theta} \) of \( p(\theta | y, z) \)). Then given an initial estimate \( \hat{z} = z_0 \), you could construct \( p(\theta | y, \hat{z}) \), which would give rise to an estimate \( \hat{\theta} \), which should lead via \( p(z | y, \hat{\theta}) \) to a better estimate of \( z \), which would lead via \( p(\theta | y, \hat{z}) \) to an even better estimate of \( \theta \), and so on, around the mulberry bush.

Since marginal posteriors \( p(\theta_j | y) \) are of such central interest, a natural way to apply this sort of idea is to let \( \theta_j \) play the role of \( \theta \) above and let \( \theta_{(j)} \)—the \( \theta \) vector with component \( j \) omitted—play the role of \( z \). In the context of the ammonite data, for example, this suggests (1) sampling from \( p(\mu | y, \sigma) \), obtaining \( \mu \) (say), (2) then sampling from \( p(\sigma | y, \mu) \), obtaining \( \sigma \), (3) then sampling another \( \mu \) from \( p(\mu | y, \sigma) \), and so on. That, in a nutshell, is Gibbs sampling.

More precisely, for general \( k \) the algorithm is summarized by (2.36) below. Demonstrating that the resulting stochastic process \( \theta^{(t)} \) is indeed a Markov chain with the right equilibrium distribution and showing how Gibbs fits in with Hastings and Metropolis in the overall MCMC picture (Problem 2.5) are more complicated matters, but if you spot me that it works you can see that the algo-
The algorithm itself is pretty straightforward: the distributions $p(\theta_j | y, \theta_{(j)})$ are called the full conditionals for the model you're sampling from, and the rule is simply that you always use the most recent sampled values of the components of $\theta_{(j)}$ in defining and generating from the next full conditional. One iteration of the repeat loop in (2.36) is called a scan of the Gibbs sampler, and fills in one row of the MCMC data set (Table 2.3).

**Algorithm (Gibbs sampling)** (Geman and Geman, 1984). To construct a Markov chain whose equilibrium distribution is $p(\theta | y)$, Initialize $\theta^{(0)}$; $t \leftarrow 0$

Repeat \{ 
Sample $\theta_1^{(t+1)} \sim p(\theta_1 | y, (\theta_2^{(t)}, \ldots, \theta_k^{(t)}))$
Sample $\theta_2^{(t+1)} \sim p(\theta_2 | y, (\theta_1^{(t+1)}, \theta_3^{(t)}, \ldots, \theta_k^{(t)}))$
\hspace{1cm} \vdots \hspace{1cm} \vdots \hspace{1cm} \vdots \hspace{1cm} \vdots \hspace{1cm} \vdots
Sample $\theta_{k-1}^{(t+1)} \sim p(\theta_{k-1} | y, (\theta_1^{(t+1)}, \ldots, \theta_{k-2}^{(t+1)}, \theta_k^{(t)}))$
Sample $\theta_k^{(t+1)} \sim p(\theta_k | y, (\theta_1^{(t+1)}, \ldots, \theta_{k-1}^{(t+1)}))$
t $\rightarrow$ (t + 1) \}

**Working out the full conditionals.** To apply this idea to the ammonite data, I need to figure out the full conditionals $p(\mu | y, \sigma)$ and $p(\sigma | y, \mu)$. Considering $p(\mu | y, \sigma)$ first, notice that

$$p(\mu | y, \sigma) = \frac{p(\mu, y, \sigma)}{p(y, \sigma)} = cp(\mu)p(\sigma | \mu)I(\mu, \sigma | y). \tag{2.37}$$

The $c$ in (2.37) arises because I don't have to evaluate things like $p(y, \sigma)$ that don't involve $\mu$, since I am thinking of the left side of the equation as a function of $\mu$ for fixed $y$ and $\sigma$—indeed, anything that appears on the right side that is a function only of $y$ or $\sigma$ will just get absorbed into the proportionality constant. From (2.32) and (2.35), (2.37) becomes

$$p(\mu | y, \sigma) = \frac{c}{\min(\mu - L, H - \mu)} \cdot I[\sigma < \min(\mu - L, H - \mu)] \cdot I[\max(y) - \sigma < \mu < \min(y) + \sigma]. \tag{2.38}$$

At this point in building your own Gibbs sampler from scratch, you hope that the right side of an expression like (2.38) is the kernel of a density you recognize, so that it will be easy to sample from,
and in fact in many standard situations—for instance, Problems (2.8) and (2.9)—that is how things go. Here I am not so lucky: (2.38) is just a messy bunch of special cases, depending on whether \( \mu \) is bigger than one thing or smaller than another. I will spare you the details; suffice it to say that, specifying the full conditional for \( \mu \) in terms of its CDF \( F(\mu|y, \sigma) \) instead of its density, you get that

- For \( \max(y) - \min(y) < \sigma < \mu_* - \min(y) \),

\[
F(\mu|y, \sigma) = \begin{cases} 
0 & \mu < c_1 \\
\log\left(\frac{\mu - c_1}{\mu - c_2}\right) & c_1 < \mu < c_2 \\
1 & c_2 < \mu 
\end{cases},
\]  
(2.39)

where \( c_1 = \max[L + \sigma, \max(y) - \sigma] \) and \( c_2 = \min(y) + \sigma \); and

- For \( \mu_* - \min(y) < \sigma < \frac{H - L}{2} \equiv \sigma_* \),

\[
F(\mu|y, \sigma) = \begin{cases} 
0 & \mu < c_1 \\
c_4 \log\left(\frac{\mu - L}{c_1 - L}\right) & c_1 < \mu < \mu_* \\
c_4 \log\left[\left(\frac{\sigma^2}{(c_1 - L)(H - \mu)}\right)\right] & \mu_* < \mu < c_3 \\
1 & c_3 < \mu 
\end{cases},
\]  
(2.40)

where \( c_3 = \min(H - \sigma, c_2) \) and \( c_4 = \left\{ \log\left[\frac{\sigma^2}{(c_1 - L)(H - c_3)}\right] \right\}^{-1} \).

The reason I have focused on the CDF rather than the full conditional density for \( \mu \) is that the next thing I have to do is figure out how to sample from \( p(\mu|y, \sigma) \), and one of the easiest ways to do so is to recall (e.g., Ripley, 1987) that \( \tilde{\mu} = F^{-1}(U|y, \sigma) \) is a draw from \( p(\mu|y, \sigma) \) when \( U \sim U(0, 1) \). So the last step in sampling from \( \mu \)'s full conditional is inverting \( F(\mu|y, \sigma) \), which is straightforward: after some algebra you see that to make a draw \( \tilde{\mu} \) from \( p(\mu|y, \sigma) \), you can generate \( U \sim U(0, 1) \) and

\[
\text{If } \sigma < \mu_* - \min(y) \text{ set } \tilde{\mu} = L + (c_1 - L) \left(\frac{\sigma + c_1}{c_1 - L}\right)^U; \\
\text{else if } U < c_4 \log\left(\frac{\sigma}{c_1 - L}\right) \text{ set } \tilde{\mu} = L + (c_1 - L) \exp\left(\frac{U}{c_4}\right); \\
\text{else set } \tilde{\mu} = H - \exp\left[\frac{U}{c_4} - \log\left(\frac{\sigma}{c_1 - L}\right)\right].
\]  
(2.41)

The story for \( \sigma \)'s full conditional is considerably simpler:

\[
p(\sigma|y, \mu) = c p(\mu) p(\sigma|\mu) l(\mu, \sigma|y) \\
= \frac{c}{\sigma^n} \cdot I[c_8 < \sigma < c_6].
\]  
(2.42)
where $c_5 = \max[\mu - \min(y), \max(y) - \mu]$ and $c_6 = \min(\mu - L, H - \mu)$. (2.42) can be sampled from in the same way as $\mu$ was:

$$F(\sigma | y, \mu) = \begin{cases} 1 & \sigma < c_5 \vspace{.5em} \\ \frac{c_5^{1-n} - \sigma^{1-n}}{c_5^{1-n} - c_6^{1-n}} & c_5 < \sigma < c_6 \\ \frac{1}{c_5^{1-n}} & c_6 < \sigma \end{cases},$$

(2.43)

and to draw a $\hat{\sigma}$ from $p(\sigma | y, \mu)$ you just generate $U \sim U(0,1)$ and

Set $\hat{\sigma} = [(1-U)c_5^{1-n} + Uc_6^{1-n}]^{\frac{1}{1-n}}$. (2.44)

![Graphs showing time series and density traces for $\mu$ and $\sigma$.](image)

**Figure 2.8.** Time series and density traces for $\mu$ and $\sigma$ in the uniform model (2.31, 2.32) applied to the M. dens. $\alpha$ data.

**Results for the ammonite data.** Section 4 of Appendix 2 contains S+ code implementing this Gibbs sampler in the model (2.31, 2.32), and Figures 2.8 and 2.9 summarize the results when applied to the $M. \text{dens. \alpha}$ data from Table 2.1. I used a burn-in of 1,000 and a monitoring run (without thinning) of 10,000 from an initial value of $\sigma_0 = 190$ (a bit bigger than the smallest possible value $\frac{1}{2}[\max(y) - \min(y)] = 180$). This took about 2.5 minutes at 333MHz to produce output that passed all tests in Section 2.4, and resulted (for example) in a Monte Carlo SE for the posterior mean of $\mu$ of 0.22. Figure 2.8 shows the time series and density traces for $\mu$ and $\sigma$ (the ACF and PACF plots are not very interesting);
Figure 2.9 repeats for $A$ and $B$. Table 2.10 contains some numerical summaries for the four parameters.

![Graphs of A and B iterations and density plots](image)

Figure 2.9. *Like Figure 2.8 but summarizing the parameters $A$ and $B$.

A number of intriguing things emerge from even a cursory examination of the figures and table.

- The density trace for $\sigma$ looks more or less like that of a typical scale parameter, and the marginal posteriors for $A$ and $B$ look just about like they would have to, given their status as range-restriction parameters, but the density trace for $\mu$ is extremely peaked at its center of symmetry—not at all the Gaussian sort of shape you might expect for a location parameter.

- All four parameters are mixing well, and the first-order autocorrelations $\hat{\rho}_1$ and default $\hat{\text{r}}_{RL}$ values are much smaller than those we had come to expect with Hastings or Metropolis (actually, then, what I said a minute ago about the ACF and PACF plots isn’t true—they are interesting when compared with those from Hastings or Metropolis precisely because they *don’t* show much autocorrelation). This is a general feature of Gibbs—it usually produces parameter chains with less serial correlation (readers: what’s a good reference for this?).

- The posterior SDs for $\mu$ and $\sigma$ are remarkably small, given the variability of the data values and the small sample size ($n = 16$...
for \( M. \text{dens. } \alpha \): for instance, the posterior SD of \( \mu \) is only about 0.2% of the posterior mean. By comparison, the sample mean \( \hat{\mu} \) has standard error \( \frac{SD}{\sqrt{n}} = 25.3 \), so on the variance scale—where frequentists typically compare the performance of estimators—the posterior mean of \( \mu \), an alternative estimate of the center of symmetry, is 100 \[ \left( \frac{25.3}{20.7} \right)^3 - 1 \] = 50% more efficient than the usual (Gaussian-model-based) sample mean. This is connected to the "witch's hat" shape of the marginal posterior for \( \mu \) noted above.

Table 2.10. Numerical summaries of the four parameters in the uniform model (2.31, 2.32) applied to the M. dens. \( \alpha \) data.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mean</th>
<th>SD</th>
<th>95% Central Interval</th>
<th>( \hat{\rho}_1 )</th>
<th>Default ( \hat{n}_{RL} )</th>
<th>MLE ( \hat{n} ) (SE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mu )</td>
<td>994.4</td>
<td>20.7</td>
<td>(948, 1040)</td>
<td>0.044</td>
<td>8800</td>
<td>995 (14.6)</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>207.9</td>
<td>21.5</td>
<td>(183, 265)</td>
<td>0.54</td>
<td>5000</td>
<td>180 (13.7)</td>
</tr>
<tr>
<td>( A )</td>
<td>786.5</td>
<td>31.0</td>
<td>(700, 814)</td>
<td>0.32</td>
<td>9400</td>
<td>815 (20.0)</td>
</tr>
<tr>
<td>( B )</td>
<td>1202</td>
<td>28.6</td>
<td>(1180, 1280)</td>
<td>0.28</td>
<td>3800</td>
<td>1175 (20.0)</td>
</tr>
</tbody>
</table>

Some insight into what's going on here can be obtained by working out the MLEs of the four parameters in Table 2.10 and their standard errors. It is not hard to show (Problem 2.10) that in model (2.30, 2.31),

\[
\hat{A} = \min(y), \quad E(\hat{A}) = A + \frac{2\sigma}{n + 1} = A + \text{bias},
\]

\[
V(\hat{A}) = \frac{4n\sigma^2}{(n + 1)(n + 2)} = V(\hat{B}),
\]

\[
\hat{B} = \max(y), \quad E(\hat{B}) = B - \text{bias}, \quad (2.45)
\]

\[
\hat{\mu} = \frac{\hat{\beta} + \hat{\alpha}}{2}, \quad E(\hat{\mu}) = \mu, \quad V(\hat{\mu}) = \frac{2\sigma^2}{(n + 1)(n + 2)},
\]

\[
\hat{\sigma} = \frac{\hat{B} - \hat{\alpha}}{2}, \quad E(\hat{\sigma}) = \frac{n - 1}{n + 1} \sigma, \quad V(\hat{\sigma}) = \frac{n - 1}{n + 1} V(\hat{\mu}),
\]

where the hats denote the MLEs. (I computed the standard errors in Table 2.10 in the usual way, by plugging the MLE for \( \sigma \) into the variance expressions in (2.45).) Since the prior (2.32) I have used for \( (\mu, \sigma) \) is quite diffuse, the MLEs should be approximating the marginal posterior modes fairly closely (indeed you can verify
this from Figures 2.8 and 2.9), and the modes and means of these marginals are not all that different, so the variance formulas in (2.45) should provide some guidance as to the uncertainty we have about $\theta$ and $\sigma$ in light of the data.

There is something funny about these variances: they are of order \( \frac{1}{n^2} \), instead of the usual \( O\left(\frac{1}{n}\right) \) in location and scale problems. This is due to the extremely light tails of the uniform distribution—in effect, they fall off so rapidly (like a step function, in fact) that you can learn about the range-restriction parameters $A$ and $B$, and simple functions of them like $\mu$ and $\sigma$, at a much faster rate than with (say) Gaussian or lognormal data.

**Comparing the ammonite taxa.** I will finish this case study by applying the methodology developed above to all 13 taxa of ammonites in Tables 2.1 and 2.2. I used the same model and sampling strategy as those that produced Table 2.10 to generate 5,000 draws from each of the 13 posterior distributions for $\mu$, one for each taxon, and the results are summarized in Figures 2.10 and 2.11.

![Figure 2.10. Parallel boxplots of the 5,000 draws from the posteriors for $\mu$ in the ammonite uniform example, contrasting the 13 taxa in Tables 2.1 and 2.2.](image)

A natural way to examine these 13 posteriors is with parallel boxplots, as in Figure 2.10—I have used the same numbering scheme as Strauss and Sadler, who (for some reason) ranked the taxa in
increasing order of their smallest observations. Most of the distributions in this set of boxplots are close to symmetric, as you would expect from the model (in fact, sharp lack of symmetry in the posterior for $\mu$ could be a model diagnostic here). It is also interesting to note that some of the most unusual posteriors arise from the tiniest samples (taxa 4 and 11 each had a sample size of only 3, for instance).

![Boxplot of Ammonite Taxa](image)

Figure 2.11. *Plots of the posteriors for $\mu$ and the ranks of the $\mu$'s in the full ammonite data set. The vertical lines plot 95% central intervals and the superimposed points are medians in all cases.*

*The left-hand (solid) line for each taxon is the posterior for $\mu$; the right-hand (dotted) line summarizes the posterior for the ranks.*

One scientific question arising from the collection of the data in Tables 2.1 and 2.2 was the order in which the ammonite taxa secured their evolutionary niches in the Cretaceous, for instance as measured by the center $\mu$ of their true ranges. Figure 2.11 was created to help answer this question, and requires a bit of explaining. There are two vertical lines plotted for each taxon: on the left in each case (and referring to the left-hand vertical scale) is the 95% central interval for $\mu$ (with the median of the posterior superimposed as a dot); the right (dotted) line (with reference to the right-hand vertical scale) gives the 95% central interval for the ranks of the $\mu$'s.
CASE STUDY: MEASUREMENT OF PHYSICAL CONSTANTS

How did I get the intervals for the ranks? As noted by Spiegelhalter et al. (1995), whose rank analysis of an entirely different kind of data motivated Figure 2.11, an excellent feature of MCMC is the ease with which unusual and complicated functions of the underlying parameters may be monitored and summarized alongside the quantities appearing in your models. In this case, since I am treating the \( \mu \)'s for the various taxa as entirely independent in this modeling, all I had to do to monitor the ranks was (a) to make an MCMC data set with 5,000 rows and 13 columns, one for each of the sets of posterior draws for the different taxon \( \mu \)'s, and (b) create a new derived data set in which each row is replaced by the ranks of the observations in that row.

Two interesting things are immediately apparent from this plot:

- Uncertainty about the ranks of the \( \mu \)'s is considerably larger than that about the \( \mu \)'s themselves—notice in particular the disparity between the vertical line lengths for taxa 6–10. With some assurance we can say that the range-centers for taxa 0–4 came earlier in the Cretaceous than those for taxa 5–12, but that's about all we're pretty sure of. This contrasts with the rather sharper ranking conclusions that appear possible from looking directly at the posteriors for the \( \mu \)'s; and

- If you examine the dots in Figure 2.11 you will notice a pattern: for both the lowest taxa, 0–5, and the highest ones, 8–12, the rank median is farther from the center than the \( \mu \) median. Starting in Chapter 3 we will see examples of what are called shrinkage estimates, in which extreme values, in comparisons like those in this figure, are pulled back in toward the middle by switching over to a different model for the data. But because something has to get rank 1 even if it is only a little bit smaller than the second-smallest thing (and analogously for the upper end of the scale), switching attention from the underlying \( \mu \)'s to their underlying ranks evidently produces a set of anti-shrinkage or expansion estimates.

2.6 Case study: Measurement of physical constants

Back at the end of Chapter 1, when we were looking at the NB10 data (Case Study 1.2), you may remember that the Gaussian model of Section 1.8 didn't fit very well, because of a number of outliers in both tails (see Figures 1.2 and 1.4). At the time I said that it would
perhaps be good to expand the Gaussian model, by embedding it in the \( t \) family and adding a parameter \( \nu \) for tail-weight. We couldn’t fit that model in Chapter 1 because conjugacy is not available when \( \nu \) is treated as an unknown, but MCMC makes fitting models like this pretty close to routine, as I will now try to show.

In the notation of Chapter 1 the expanded model is

\[
(\mu, \sigma, \nu) \sim p(\mu, \sigma, \nu) \\
(y_i|\mu, \sigma, \nu) \overset{\text{IID}}{\sim} t_\nu(\mu, \sigma^2), \quad i = 1, \ldots, n. \tag{2.46}
\]

This model is actually not very hard to fit with Gibbs sampling—
I will conclude the chapter with an illustration of this using an MCMC package called BUGS—but I thought I would take the opportunity first to use (2.46) to lay out a fairly generic strategy for Bayesian model-fitting based on Metropolis sampling.

This is our first MCMC example in which the parameter vector \( \theta \) has dimension \( k \) bigger than 1, and the first thing you might think of in trying to apply Metropolis is to propose \( N(0, \kappa \sigma_j^2) \) moves, like I did in Section 2.3, separately and independently for each parameter \( \theta_j \). In other words—in model (2.46), for example—in filling in each row of the MCMC data set you might first sample a \( \mu \), then a \( \sigma \), and then a \( \nu \) (rather like Gibbs sampling, except that the draws in this case would be independent of all other values in the MCMC data set). This would be fairly easy to code, but it has a big potential flaw: if the parameters are highly correlated, then a lot of the moves you propose by treating them as independent will be implausible, and your acceptance rate will be far from optimal.

**A generic Metropolis sampling strategy.** To improve on this, the idea behind the generic strategy I want to look at here is that, possibly after appropriate reparameterization, the posterior distribution for \( \theta \) should be close to multivariate normal for moderate to large \( n \), say \( p(\theta|y) \sim N(\theta^*, \Sigma) \). This suggests a kind of generalization of the idea in Section 2.3: a random-walk Metropolis with multivariate normal proposal distribution, centered at where you are now, and with covariance matrix a multiple \( \kappa \) of \( \Sigma \), for suitably chosen \( \kappa \). This will accurately reflect any posterior correlations, thereby improving the efficiency of the sampling.

The steps of the strategy are thus as follows.

(1) Transform any components of \( \theta \) that live only on a subset of the real line to all of \( \mathbb{R} \). Rewrite the log likelihood in this new
parameterization, and recompute the log prior by including the appropriate Jacobian.

(2) Use pencil and paper or (more reliably, in complicated problems) a symbolic computing package to find the posterior mode \( \theta_m \). Symbolically obtain the Hessian \( H \) (the second partial derivative matrix) of the log posterior, evaluate it numerically at the posterior mode, and compute \( \hat{\Sigma} = -H^{-1}\theta_m \). If the prior is diffuse you can replace "posterior mode" by "MLE" and "log posterior" by "log likelihood."

(3) Code up and run a Metropolis sampler that makes \( N(0, \kappa \hat{\Sigma}) \) moves, varying \( \kappa \) to minimize the maximum of the SSIF or \( \hat{n}_{RL} \) values across the components of \( \theta \). Gelman et al. (1996) have shown that, in a particular class of problems that should give some guidance here, the optimal \( \kappa \) behaves roughly like \( \frac{5.8}{p} \), and the optimal acceptance rate decreases from about 0.44 for \( p = 1 \) to about 0.27 for \( p = 10 \), roughly along the curve \( 0.23 + \frac{0.28}{p} - \frac{0.046}{p^2} \).

If step (2) is too difficult, you will need another way to get an approximate \( \Sigma \). The simplest idea is probably to use the independent-component sampler I criticized a few paragraphs ago to get yourself started, and then switch over to step (3). One nice thing about MCMC is that, even with an inefficient proposal distribution, the output of the chain—once equilibrium has been reached—is a valid sample from the posterior. So you can try an iterative strategy like the following: start with a poorly-tuned proposal; run it a very long time; use the sample covariance matrix based on the columns of the resulting MCMC data set as an initial estimate \( \hat{\Sigma}_0 \) of \( \Sigma \); run for a long time with a multivariate normal proposal based on \( \hat{\Sigma}_0 \); use the sample covariance matrix from the columns of this MCMC data set to produce a better estimate \( \hat{\Sigma}_1 \); and so on.

This is called adaptive Metropolis-Hastings sampling (e.g., Gilks et al., 1997), and there is only one thing to watch out for: if you keep indefinitely refining the proposal distribution adaptively, based on the previous output of the chain, it has been shown that the sampler will not (necessarily) converge to the right equilibrium distribution. So you need to stop the adaptive process at some point before monitoring to produce the results you will announce, because if not you may well be monitoring the wrong distribution.

In a bit more detail, the alternative strategy is as follows.

(2') (a) Code up a Metropolis sampler that makes a series of \( N(0, \kappa_j \sigma_j^2) \)
moves, one for each parameter \( \theta_j \), obtaining estimates for the
\( \sigma_j^2 \) by whatever means you can think of (likelihood theory, it-
iterative guesswork, ...) and varying the \( \kappa_j \) so that the product
of the acceptance probabilities is as large as you can make it,
up to a maximum of about 0.5. Set \( s = 0 \).

(b) Run this sampler for a long time from a good starting value
and with a bigger-than-usual burn-in (use SSIF and/or \( \hat{n}_{RL} \)
values to define "large"), and use the sample covariances of
the columns of the resulting MCMC data set to construct an
estimate \( \Sigma_s \) of \( \Sigma \). If \( s > 0 \) and \( \Sigma_s \) and \( \Sigma_{s-1} \) don’t differ too
much, go to \( 4' \).

(3') Code up and run a Metropolis sampler that makes \( N(0, \kappa \hat{\Sigma}_s) \)
moves, varying \( \kappa \) to optimize the acceptance probability as usual.
When you have a \( \kappa \) you like, increment \( s \) and go back to \( 2'b \).

(4') Now, finally, make your monitoring run for the money using the
most recent \( \hat{\Sigma}_s \).

I have implemented strategy (1–3), using \textit{S+} and the symbolic
computing package \textit{Maple}, with model (2.46) using the NB10 data
(Problem 2.11 invites you to try strategy (1, 2'–4') on this same
example). This requires creating the new parameters \( \eta = \log(\sigma) \)
and \( \lambda = \log(\nu) \), and rewriting the model in terms of \( \theta = (\mu, \eta, \lambda) \).
The log likelihood function, in this parameterization, is (from Ap-
pendix 1)

\[
\log [l(\mu, \eta, \lambda) | y] = c + n \log \left[ \frac{e^\lambda + 1}{2} \right] - n\eta - n \log \left[ \frac{e^\lambda}{2} \right] - \frac{n\lambda}{2} - \frac{e^\lambda + 1}{2} \sum_{i=1}^{n} \log \left[ 1 + e^{-(\lambda+2\eta)}(y_i - \mu)^2 \right],
\]

where \( \Gamma(\cdot) \) is Euler’s gamma (generalized factorial) function (e.g.,
Abramowitz and Stegun, 1972).

**Prior elicitation in the NB10 t model.** By way of a prior
I have tried to bring in a modest amount of information that ac-
cords with the science of the problem. With \( n = 100 \) observations
it should be OK to use a prior with independent components, be-
cause any strong posterior correlations that should be present will
be accurately learned from the data, and having transformed to
the log scale for $\sigma$ and $\nu$ it should be reasonable to work with Gaussians, so I took

$$p(\mu, \eta, \lambda) = N(\mu | \mu_0, \sigma^2_\mu) \cdot N(\eta | \eta_0, \sigma^2_\eta) \cdot N(\lambda | \lambda_0, \sigma^2_\lambda).$$  \hspace{1cm} (2.48)

This reduces the elicitation problem to that of specifying the prior means and SDs for each of $\mu$, $\eta$, and $\lambda$.

- $\mu$ represents the true weight of the block of metal NB10, which is supposed to weight around 10g, and the observations are in micrograms below this nominal weight. So to give the National Bureau of Standards (NBS) the benefit of the doubt I should probably take $\mu_0 = 0$, but $\sigma_\mu$ should be big to reflect the possibility of substantial bias on the microgram scale. Based on previous results with similar weighing equipment at the NBS (Ku, 1969), I have chosen $\sigma_\mu = 500$ in what follows (see Problem 2.12 for a sensitivity analysis of the prior specification in this case study).

- $\nu$ indexes the tail-weight of the true distribution of measurement errors. Churchill Eisenhart (1979, personal communication), a leading statistician at NBS for decades, is on record as saying that “Measurement error processes in the physical sciences, when investigators report all their apparently valid data, tend to behave roughly like $t$ on about 7 degrees of freedom.” I have interpreted this expert judgment, a bit liberally, as an approximate statement that $P(2 < \nu < 20) = 0.95$. On the log scale this creates a 95% prior interval for $\lambda$ of $(0.69, 3.0)$, which in the Gaussian world implies $\lambda_0 = 1.84$ and $\sigma_\lambda = 0.59$.

- $\sigma$ is related to the true SD of the measurement errors made by the NBS weighing process, through $SD(y) = \sqrt{\frac{\nu}{\nu-2}} \sigma$ (as long as $\nu > 2$). A conservative reading of Ku (1969) suggests that errors on the order of 1–200 on the microgram scale are possible, which I will translate into the statement $P(1 < \sqrt{\frac{\nu}{\nu-2}} \sigma < 200) = 0.95$ for elicitation purposes. Taking $\nu \doteq 7$ for simplicity, for $\eta = \log(\sigma)$ this statement corresponds to the 95% prior interval $(-0.17, 5.13)$, leading to a Gaussian prior mean of $\eta_0 = 2.48$ and SD of $\sigma_\eta = 1.35$.

Before doing any sampling it is worth looking at the log posterior a bit to see if any pathologies should be anticipated. I can’t plot $p(\mu, \eta, \lambda | y)$ in all its glory, because we are condemned to three visual dimensions, but—like the blind men and the elephant in the
old story—I can try to create a mental image of the whole thing by looking at various views of it one by one. Figure 2.12 presents four such views of the log posterior in this problem. I drew the upper left panel by holding constant $\mu$ and $\eta$ at plausible values, not (perhaps) too far from their posterior modes (I took the sample mean for $\mu$ and the log of the sample SD for $\eta$), and tracing out the log posterior as a function of $\lambda$. If the posterior is multivariate normal this plot should look locally quadratic around its maximum, and—while it lacks a bit in the symmetry department—it is at least bowl-shaped down with only a single maximum. The other two similar plots (not shown), obtained by fixing $(\mu, \lambda)$ and $(\eta, \lambda)$, are also reasonably well-behaved—in particular, there are no signs of multimodality.

The other three panels in Figure 2.12 are contour plots of the log posterior, obtained by fixing one component of $\theta$ at a time and letting the other two vary. Each of these graphs should look like a set of concentric ellipses if the posterior is close to multivariate normal, and as long as you cast a slightly generous eye on the $(\eta, \lambda)$
plot you can see that things are not terribly far from MVN. It's also interesting to note that $\eta$ and $\lambda$ are fairly strongly positively correlated in the posterior, which on reflection makes sense: if I gave you a moderate-$n$ sample of data with a few points that may or may not be outliers, it would be hard for you to tell if the underlying story was (small $\nu$, small $\sigma$)—in other words, the data really are $t_\nu$—or (large $\nu$, large $\sigma$), which is like saying that the data are really (close to) Gaussian but just with a big SD. Thus scale and shape are confounded in the $t$ family.

OK, now I'm ready to try to build my Metropolis sampler. I've done step (1) already (I didn't have to compute any Jacobians because it was easy to elicit on the transformed scale). Appendix 2 Section 5 gives some code in the symbolic computing language Maple to get the approximate posterior covariance matrix $\hat{\Sigma}$. It's easy to specify the log posterior in Maple since it has a built-in $\log[\Gamma(\cdot)]$ function, and then you just have to ask Maple to differentiate the log posterior symbolically and solve the resulting MAP (maximum a posteriori) equations numerically to find the mode.

It turns out that Maple could not find the mode $\theta_m$ without some help, in the form of range restrictions on where $\theta_m$ might be, but after I gave it a rather broad hint of this kind it was able to solve the likelihood equations in this problem with no trouble, in about 2.5 seconds at 333MHz. To finish the calculation off I just have to ask Maple to calculate the Hessian $H$ symbolically and evaluate it numerically at the mode, and then take $\hat{\Sigma} = -\left[H^{-1}\right]_{\theta_m}$, as noted in the generic strategy above. The results are $\theta_m = (404.3, 1.346, 1.260)$ and

$$
\hat{\Sigma} = \begin{pmatrix}
0.215 & 0.00299 & 0.00808 \\
0.00299 & 0.0119 & 0.0149 \\
0.00808 & 0.0149 & 0.0749
\end{pmatrix},
$$

leading to approximate standard errors ($\sqrt{0.216} = 0.46, 0.11, 0.27$) for $(\hat{\mu}, \hat{\eta}, \hat{\lambda})$ and approximate correlations ($\frac{0.00299}{\sqrt{0.216 \cdot 0.0119}} = 0.059$, 0.064, 0.50) for $(\hat{\mu}, \hat{\eta}), (\hat{\mu}, \hat{\lambda}), (\hat{\eta}, \hat{\lambda})$. Thus $\mu$ is around $404.3 \pm 0.46$; $\eta$ is about $1.346 \pm 0.11$, meaning that $\sigma$ is likely to be in the range $(\exp(1.346 - 2 \cdot 0.11) = 3.1, \exp(1.346 + 2 \cdot 0.11) = 4.8)$; and $\lambda$ is around $1.260 \pm 0.27$, so that $\nu$ is probably in the interval (2.1, 6.0). All of this is useful information in extracting the full posterior.

Appendix 2 Section 6 contains some $S+$ functions to implement the generic Metropolis strategy above with the NB10 data (note
how little would have to be changed to use this sampler on a completely different problem). With \( \kappa = 2 \simeq \frac{5.8}{3} \), a single long run of 45,000, storing every 9th iterate, after a burn-in of 2000 from a starting value of \( \theta_0 = (404.6, 1.699, 1.946) = [\hat{y}, \log(\sqrt{\frac{5}{3}} s), \log(7)] \) took about 10 minutes at 333Mhz and produced the output summarized in Table 2.11 and Figure 2.13. All CODA diagnostics were well-behaved, and the thinning by a factor of 6 resulted in fairly low serial correlations and default \( \hat{n}_{RL} \) values. The Metropolis acceptance rate was 0.31, which is near-optimal for \( p = 3 \) based on Gelman et al. (1996)'s results, so I didn't try to look for a better \( \kappa \).

Table 2.11. Numerical summaries of the three original-scale parameters in the \( t \) model (2.46, 2.48) applied to the NB10 data, using the monitoring strategy described in the text.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mean</th>
<th>SD</th>
<th>95% Central Interval</th>
<th>( \hat{p}_1 )</th>
<th>Default ( \hat{n}_{RL} )</th>
<th>MLE (SE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mu )</td>
<td>404.3</td>
<td>0.48</td>
<td>(403.4, 405.3)</td>
<td>0.20</td>
<td>5000</td>
<td>404.3 (0.46)</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>3.92</td>
<td>0.44</td>
<td>(3.14, 4.87)</td>
<td>0.17</td>
<td>4100</td>
<td>3.70 (0.42)</td>
</tr>
<tr>
<td>( \nu )</td>
<td>3.75</td>
<td>1.1</td>
<td>(2.15, 6.44)</td>
<td>0.16</td>
<td>3900</td>
<td>3.01 (0.86)</td>
</tr>
</tbody>
</table>

The table and figure bring up several interesting points.

- The posterior mean of the scale parameter \( \sigma \) is substantially lower than the sample SD \( s = 6.5 \), but this is to be expected since \( V(y) = \frac{\nu}{\nu-2} \sigma^2 \) in this model (as long as \( \nu > 2 \)). Indeed, the sample average of the quantity \( \frac{s^2}{\nu-2} \) across the 4,971 rows of the MCMC data set with \( \nu > 2 \) is 44.4, not far from the sample variance 41.8.

- Bayesian and ML inferences with these data (with my prior, at least) are similar for \( \mu \), but the posterior means are about 6% and 25% larger than the MLEs for \( \sigma \) and \( \nu \), respectively, and the MLE standard errors are smaller than the posterior SDs. Part of this difference comes from the prior, part from the difference between means and modes for skewed distributions, and part from the way ML inference (sometimes inaccurately) deals with uncertainty about \( \theta_{(j)} \) when summarizing uncertainty about \( \theta_j \). Note in particular how much smaller the SE for \( \hat{\eta}_{MLE} \) is than the posterior SE for \( \nu \).
Figure 2.13. MCMC output summaries in the NB10 \(t\) model: density traces of \(\mu\), \(\sigma\), and \(\nu\) (reading clockwise from the upper left), and scatterplot of \(\lambda\) versus \(\eta\) (compare with the lower right contour plot in Figure 2.12).

- Most importantly, recall from Chapter 1 that when we (incorrectly) assumed a Gaussian model for these data, the posterior mean and SD of \(\mu\) (which has the same meaning in both models, and is thus comparable) were 404.3 and 0.65. With the \(t\) model the posterior mean is the same but the posterior SD, 0.47, is substantially (28\%) smaller.

  - This makes sense from a frequentist robustness point of view: if—in view of the outliers—you were to use a trimmed mean instead of the sample mean, you would lop off the smallest and largest (say) 2 observations, calculate the mean \(\bar{y}_T\) (also about 404.3) and SD \(s_T\) (considerably smaller: 4.25) of the rest, and (in effect) use \(\frac{s_T}{\sqrt{n}}\) = 0.43 as your standard error.

  - However, the conclusion is interesting from a Bayesian robustness point of view: when I expand the Gaussian model by embedding it in the \(t_\nu\) family for unknown \(\nu\), my model uncertainty has increased (because the former model is a special case of the latter, obtained by pretending you know that \(\nu = \infty\)), but evidently in this case my inferential uncertainty
about the quantity of principal interest—\( \mu \)—has decreased. This is partly because the \( t_\nu \) model with small \( \nu \) fits better and partly because it turns out that the Gaussian is a very conservative choice for inference about location parameters (in fact, it minimizes Fisher information for such parameters in [essentially] the [whole] class of symmetric distributions; see Draper, 1997).

One last point to consider before leaving this case study concerns model diagnostics, a topic I will take up in more detail in Chapter 4. I have been reasonably careful about MCMC diagnostics in this chapter, but it is all too easy in the midst of looking at CODA output to forget that (a) model diagnostics are equally important and (b) MCMC diagnostics have little or nothing to say directly about the fit of the model to the data.

Figure 2.14. Simple model diagnostic plots in the NB10 example.

The left panel relates the correlation of the \( t_\nu \) quantiles and the sorted NB10 data values to \( \nu \), and the right panel is a \( t_\nu \) qqplot of the NB10 data with \( \nu = 2 \).

Figures 1.2 and 1.5 demonstrated that the Gaussian model fit the NB10 data poorly, but did not directly show that the \( t \) model fits well. Figure 2.14, on the other hand, provides some evidence that the \( t \) family is appropriate for the NB10 data. With access to the CDF of the \( t_\nu \) distribution it is easy enough to make a \( t \)
qqplot of a data set, but what should you choose for \( \nu \)? The left panel of Figure 2.14 gives one simple answer: I have plotted the correlation between the \( t_\nu \) quantiles and the sorted NB10 data values\(^\ddagger\) as a function of \( \nu \), and you can see that this is maximized for \( \nu \approx 2 \). So the right panel of the figure gives a \( t_2 \) qqplot, which does indeed fit the data pretty much like a glove. It is interesting to consider (Problem 2.13) why the apparently optimal value of \( \nu \) in this sense is not particularly well supported by the posterior for \( \nu \) summarized in Table 2.11.

\[ \textbf{A Gibbs shortcut: BUGS.} \] The generic Metropolis \( \star \) analysis above (or—better—its analogue in \( \mathcal{C} \)) is a reasonably satisfying way to implement MCMC in many problems, but it would be nice if there were a rather more user-friendly environment in which to get MCMC results. There have been several attempts to date to supply fairly narrowly-targeted MCMC packages, including \texttt{bpois} (for Poisson regression; Doss and Narasimhan, 1994) and \texttt{MCSim} (by Frédéric Bois; see Carlin and Louis, 1996); the most successful general-purpose attempt so far—by quite a margin—has been the Gibbs sampling package \texttt{BUGS}, developed by David Spiegelhalter, Wally Gilks, and colleagues at the MRC Biostatistics Unit in Cambridge (UK). The program is available for free, in a variety of hardware and operating system configurations, at http://www.mrc-bsu.cam.ac.uk or by anonymous ftp at ftp.mrc-bsu.cam.ac.uk, and may be run either in interactive or batch mode. The authors have provided excellent documentation for their code, including an extensive set of worked examples.

At first thought, writing a generic Gibbs sampling package sounds like a daunting task—for instance, how would you automatically figure out the full conditional distributions for an arbitrarily specified model? The authors of \texttt{BUGS} (Gilks et al., 1994) have succeeded in achieving considerable generality by means of two fairly mild forms of limitation, as follows.

- With a few exceptions, \texttt{BUGS} is restricted to fitting models expressible as \textbf{directed acyclic graphs} (DAGs) (e.g., Whittaker, 1990; Lauritzen et al., 1990). Figure 2.15 presents a visual representation of the DAG implied by the NB10 \( t \) model (2.46, 2.48). In pictures such as these, \textit{constants} (not present here) are denoted by rectangles; \textit{stochastic nodes} are variables given a distribution by the model, and are denoted by circles; and \textit{directed links} between nodes are indicated by arrows: solid arrows denote
stochastic dependence, and dashed arrows (not present here) indicate deterministic relationships. The directed links basically specify what depends on what in the model: the node into which an arrow points is dependent on the node from which the arrow came.

Figure 2.15. DAG representation of the variables in the NB10 t model (2.46, 2.48).

The term directed in the acronym DAG means that all the lines connecting nodes in the graph have to have arrows (so that you can figure out the dependence structure); and acyclic means that there are no subsets of the graph in which you can start somewhere, follow the arrows along, and end up back where you started. The DAG in Figure 2.15 is incredibly simple (see the examples manuals that come with BUGS for illustrations of how complicated things can get)—about all you can learn from this DAG is that the $y_i$ depend on $\mu, \sigma,$ and $\nu$ (in graphical models jargon, the three latter nodes are parents of the $y_i$, or equivalently $y_i$ is a descendant (or child) of each of the three parameters), and that $\mu, \sigma,$ and $\nu$ are independent of each other (in the prior specification).

The reason BUGS sticks to DAGs (and this is essentially a null restriction in this book—I cannot think of any model we will look at that is not a DAG) is that it is (relatively) easy to specify the full conditionals with DAGs, at least up to multiplicative constants (which are irrelevant in Gibbs sampling): the full conditional for a given stochastic node is just (proportional to) the product of the distribution specified at that node and the distributions specified at all of its children. Thus the tree structure implied by the DAG (and the acyclic assumption ensures that it is indeed a tree, and not something more complicated) allows the full conditionals to be read off directly.
In addition to a general way to figure out the full conditionals, BUGS needs a strategy for sampling from them. The developers have adopted a three-part approach to dealing with this, as follows. First, the code contains a simple expert system that tries to recognize conjugacy and make use of standard methods of sampling from conjugate distributions, if this avenue is successful. Next, if this fails, BUGS tries to convince itself that the full conditional distributions of the model are log-concave, which just means that on the log scale these distributions should be bowl-shaped down\textsuperscript{25}. The reason for this restriction is that Gilks and Wild (1992) and Gilks (1992) have developed a clever adaptive-rejection sampling method that relies on the log-concavity to create a progressively more efficient rejection sampler over time as the sampling proceeds, and BUGS uses this approach when conjugacy fails. Finally, if neither conjugacy nor log-concavity is available, BUGS can sample from an arbitrary full conditional if it is discretized. More support points in the discrete approximation to the real full conditional will obviously increase the accuracy of the approximation, but too many such points will produce a very slow sampler indeed.

I will conclude this chapter with an alternative analysis of the NB10 data using Gibbs sampling in BUGS. Table 2.12 gives the principal input file for the BUGS reanalysis, which is given a name with suffix .bug in the directory where you want to run BUGS. You can see that the syntax is clear and resembles that of S+ in some respects (to aid users of the former who are already familiar with the latter). You name your model; specify the values of any constants; tell BUGS which of the names you will be using in the program should be thought of as variables, and with what (vector or matrix) dimensions; tell it the names of the files from which it can read in the data (suffix .dat) and the initial values (suffix .in) for the MCMC sampling; specify the priors; specify the likelihood; and define any derived quantities for monitoring.

Appendix 2 Section 7 contains the other files used to make BUGS runs in this problem. After writing the .bug file and making sure that the .dat and .in files are correctly specified, you run BUGS either interactively or in batch mode. In the former case you type bugs and enter commands from the keyboard; in the latter you collect all of these commands into a file with suffix .cmd and type (for instance) backbugs nb10.3.cmd. The main advantages of batch
Table 2.12. BUGS file nb10.3.bug for a Gibbs sampling reanalysis of the NB10 data.

```plaintext
model nb10.3;  # Naming the model.
const
  n = 100, g = 101;  # Defining the constants.
var
  mu, tau, u, grid[g],
  nu, y[n], sigma;  # Specifying the variables.
data y in "nb10-y.dat",
  grid in "nb10-grid.dat";
init in "nb10.3.in";  # Reading in the data
  # and initial values.
{
  mu ~ dnorm( 0.0, 4.0E-6 );  # Specifying the priors
  tau ~ dgamma( 0.25, 0.12 );  # for mu, sigma, and nu
  u ~ dcat( grid[1] );  # (see text).
  nu <- 1.0 + u / 7.0;  #
  for ( i in 1:n ) {  # Specifying the likelihood.
    y[i] ~ dtr( mu, tau, nu );  #
  }
  sigma <- 1.0 / sqrt( tau );  # Defining a derived quantity.
}
```

Either way—interactive or batch—your first command will be, for example, compile "nb10.3.bug", after which BUGS will tell you about all your syntax errors and quit if it finds any. It often takes several iterations of editing the .bug file before you have a clean compile, which is why most people run BUGS interactively until they have (ahem) gotten all the bugs out before going to batch mode. After the compilation, your next command is usually something like update(1000), which will (in this case) perform a burn-in of 1,000 iterations; after that you prepare for the monitoring phase by issuing a series of commands that tell BUGS what, and how, to monitor. I have used monitor(mu, 14), monitor(sigma, 14), and monitor(nu, 14) followed by update(70000) in Appendix 2 Section 7—this has the effect of requiring BUGS to store monitored values for \( \mu, \sigma, \) and \( \nu \) (without doing so for any other variables.
in Table 2.12) but with a thinning ratio of 14 across the 70,000 iterations, so that only \( \frac{70000}{14} = 5000 \) values are actually written to disk for each variable. The command \( q() \) makes BUGS actually do the writing out to disk, after which it tidies up a bit and quits.

**Model specification in BUGS.** Specifying the likelihood in Table 2.12 is pretty straightforward: BUGS is able to work with something like 21 different built-in distributions that routinely arise in Bayesian analysis, of which \( t_\nu(\mu, \sigma^2) \) is one. Variances (and related scale parameters) in BUGS are always specified by working directly instead with precisions; the quantity \( \tau \) in the BUGS statement \( y[i] \sim \text{dt}(\mu, \tau, \nu) \); is just given by \( \tau = \frac{1}{\sigma} \). This explains why I have to define the derived quantity \( \text{sigma} \leftarrow \frac{1.0}{\text{sqrt}(\tau)} \);—I would rather monitor \( \sigma \) than \( \tau \). The \( \text{for}( \text{i in 1:n} ) \) loop is just telling BUGS in its language that, conditional on \((\mu, \tau, \nu)\), the \( n \) observed NB10 measurements \( y_i \) are IID \( t_\nu(\mu, \sigma^2) \).

In making this BUGS run I wanted to more or less duplicate the earlier Metropolis analysis of this problem, and it turned out that specifying the prior equivalently in BUGS required a bit more work. First I tried a literal translation of the independent normal and lognormal priors for \( \mu, \sigma, \) and \( \nu \) used previously:

\[
\begin{align*}
\mu & \sim \text{dnorm}(0.0, 4.0E-6) ; \\
\sigma & \sim \text{dnorm}(2.48, 0.549) ; \\
\nu & \sim \text{dnorm}(1.84, 2.87) ; \\
\tau & \leftarrow \frac{1.0}{\text{pow}(\sigma, 2)} ;
\end{align*}
\]

Here \( \text{dnorm}(0.0, 4.0E-6) \) means a normal distribution with mean 0 and precision 0.000004, which corresponds directly to the earlier specification \( \sigma_\mu = 500 \), and the parameters of the lognormals are the same as those I used earlier (with the variability again on the precision scale—for example, \( \sigma_\lambda = 0.59 \) earlier, which translates into a prior precision for \( \nu = e^\lambda \) of \( \frac{1}{0.59^2} = 2.87 \)). However, at the end of the compile phase with this prior BUGS announced

```
Error in file: nb10.1.bug
   for node: sigma
   -- error --
Unable to choose update method for node
```

which is its way of saying it cannot verify that the full conditional for \( \sigma \) is log concave with this model, and that therefore it doesn’t know how to sample.

So I said to myself, OK, given that BUGS likes to work with variance parameters on the precision scale and the conjugate prior
for precisions in Gaussian models is the gamma distribution (this follows from the conjugate prior for the variance being the scaled inverse $\chi^2$), it probably doesn’t like the lognormal prior for $\sigma$, because the induced prior on $\tau$ is not gamma. So next I tried

\[
\begin{align*}
mu & \sim \text{dnorm}(0.0, 4.0E-6); \\
\tau & \sim \text{dgamma}(0.001, 0.001); \\
u & \sim \text{dnorm}(1.84, 2.87);
\end{align*}
\]

starting (initially) with an extremely diffuse prior for $\tau$ (the $\Gamma(\epsilon, \epsilon)$ prior for precisions, for small $\epsilon$ like 0.001, has a big spike near 0 but is close to flat over the entire rest of the real line; see Spiegelhalter et al., 1995). However this time BUGS said

```
Error in file: nb10.2.bug
for node: nu
   -- error --
```

meaning that the prior for $\sigma$ (through $\tau$) was OK, but now it was having the same log-concavity trouble with $\nu$.

So conjugacy and log-concavity (appear to) fail for $\nu$, leaving the discretization approach as the only way to work with parameters like this in BUGS. The code will allow you to work with arbitrary discrete distributions with support points $\{1, \ldots, K\}$, for $K \leq 500$ (which is plenty to get good accuracy), and you can then transform the $1$–$K$ scale linearly to any other finite range. To approximate the prior I used in the generic Metropolis approach, I want to create a discretized version of the lognormal distribution with mean and SD (on the log scale) $\lambda_0 = 1.84$ and $\sigma_{\lambda} = 0.59$, except truncated to an interval $(l, h)$ wide enough to include the entire likely posterior for $\nu$. To achieve this I (i) chose $l = 1.1, h = 15.4$, and $K = 101$; (ii) got S+ to work out the mass at each point based on its lognormal CDF; and (iii) stored these 101 numbers in the file nb10-grid.dat. Then the two statements (a) $u \sim \text{dcat(grid[])}$ and (b) $nu <- 1.0 + u / 7.0$ in the .bug file act (a) to create a random draw $u$ from the discretized distribution spread out from 1 to 101 and (b) to transform this distribution to live on $(1.1, 15.4)$, as desired.

The last thing to specify is the hyperparameters of the gamma prior on $\tau$. I did this by reasoning that if $\sigma \sim LN(2.48, 1.35^2)$ then $\sigma^2 \sim LN(4.96, 2.70^2)$ and then finding the hyperparameters of an inverse gamma distribution for $\sigma^2$ (and thus a gamma distribution for $\tau$) that was a good visual match to $LN(4.96, 2.70^2)$, obtaining ($\alpha = 0.25, \beta = 0.12$).
BUGS results. At this point I was ready for a first try at results. I chose a burn-in of 1,000 and a short monitoring run of 4,000, obtaining MCMC output that passed the CODA Heidelberger-Welch tests but which had first-order serial correlations of (0.31, 0.58, 0.95) for $(\mu, \sigma, \nu)$, leading to default $\hat{n}_{RL}$ values of (4.2K, 6.0K, 67K), respectively. Evidently (a discretized version of) Gibbs sampling is not mixing very well on the degrees of freedom parameter with this data set. So I reran BUGS with the .cmd file in Section 7 of Appendix 2, using a burn-in of 1,000 and a monitoring run of 70K (storing every 14th iterate), and obtained results that both yielded good $\hat{n}_{RL}$ values and agreed up to Monte Carlo noise with those in Table 2.11 and Figure 2.13. The only problem is that this second BUGS run took 95 minutes at 333Mhz, versus 7 minutes for the generic Metropolis approach to achieve the same MCMC accuracy: discretization really slows BUGS down. You can show (Problem 2.14) that a from-scratch Gibbs sampler in this problem is considerably more competitive with Metropolis; on the other hand, writing a .bug file is considerably easier than programming up your own sampler from the beginning.

If you are fairly new to MCMC, I encourage you both (a) to give BUGS a chance in a number of other problems—as their examples documentation shows, when you stick with conjugate-style priors the BUGS success stories include problems in random-effects logistic and Weibull regression, extra-Poisson variation, latent class models, predictor-variable measurement error, order constraints, changepoints, spatial smoothing, and genetic pedigree analysis—and (b) to write a number of your own samplers from scratch, to develop your intuition about which MCMC strategy is most likely to get you to the finish line most quickly in the applications of principal interest to you.

2.7 Additional reading

[xx this section is incomplete] Gamerman (1997) and lots of references therein; various chapters in Gilks et al. (1996); Gelman et al. chapter 11; Carlin and Louis chapter 8; the MCMC preprint library; manuscript readers: please let me know of any important MCMC references I have omitted (bearing in mind the nature of the material presented here).
2.8 Problems

[xx this section is still quite rough]

2.1 [xx sensitivity analysis on effects of outliers in ammonite data]

2.2 (A.1) [xx conjugate analysis of the uniform model is possible if one of the two parameters is known but not if both are unknown]

2.3 [xx if \((\theta_i, t = n_B + 1, \ldots, n_B + n_M)\) is a valid sample from the posterior for \(\theta\), then \([f(\theta_i), t = n_B + 1, \ldots, n_B + n_M]\) is a valid sample from the posterior for \(f(\theta)\) for all reasonable \(f\)]

2.4 [xx try Hastings and/or Metropolis out on the ammonite problem]

2.5 [xx Explain how Gibbs fits in with Hastings and Metropolis in the overall MCMC picture]

2.6 [xx figure out induced prior on \((A, B)\) in the ammonite problem—reasonable?] 

2.7 [xx sensitivity analysis on specification of hyperparameters \(L\) and \(H\) in the ammonite problem]

2.8 [xx Standard situation in which the full conditionals are recognizable and easy to sample from]

2.9 [xx Another standard situation in which the full conditionals are recognizable and easy to sample from]

2.10 [xx show that in model (2.30, 2.31) the MLEs are as advertised]

2.11 [xx try strategy (1–2′–4′) on the NB10 data]

2.12 [xx sensitivity analysis of the prior specification in NB10 case study]

2.13 [xx Explain why the apparently optimal value of \(\nu\) in the sense of Figure 2.14 is not particularly well supported by the posterior for \(\nu\) summarized in Table 2.11]

2.14 [xx Show that a from-scratch Gibbs sampler in the NB10 problem is considerably more competitive with Metropolis]

2.15 [xx Express the \(t\) model hierarchically as a scale mixture of normals, draw the DAG, and explain the conditional independence relationships]

2.16 [xx Standard situation in which both {Metropolis or Hastings} from scratch and BUGS are reasonably straightforward. Monitor yourself in human and computer time to see how long it takes you to get (what should be) similar answers. Also contrast the amount of incremental learning arising from both strategies.]
2.17 (N2) [xx Try simulated annealing (Note 9) on the NB10 problem as an alternative way to find the posterior mode.]

2.9 Notes

2.1 I am grateful to Rob Weiss for drawing my attention to this article, and to Dimitris Fouskakis for digitizing Figure 1 from it.

2.2 A Poisson process with intensity \( \lambda \) (e.g., Feller, 1968 and/or Ross, 1970) is a stationary, continuous-time, positive-integral-valued stochastic process \( N(t) \) which (conceptually) counts the number of occurrences of something of interest to you in the time interval \([0,t]\) (so that \( N(0) = 0 \)), and which satisfies the following:

- \( \{N(t), t \geq 0\} \) has independent increments, meaning that for all \( t_0 < t_1 < \cdots < t_n \) the quantities \( \{N(t_i) - N(t_{i-1})\}, i = 1, \ldots, n \) are independent for all \( n \geq 1 \); and
- For all \( s \) and \( t \), the number \( N(t+s) - N(s) \) of occurrences in any interval of length \( t \) has a Poisson distribution with mean \( \lambda t \).

Several strong conclusions about \( N(t) \) immediately arise from these strong assumptions—for instance, the interarrival times are exponential, and given that \( N(t) = n \), the \( n \) arrival times have the same distribution as the order statistics of a sample of size \( n \) from the \( U(0,t) \) distribution.

2.3 Two other approaches worth mentioning are

- **Reference analysis** (Bernardo, 1979), which tries to develop highly diffuse priors and straightforward updating strategies for as wide a variety of standard likelihoods as possible. However (Bernardo and Smith, 1994), this approach has trouble with multiparameter problems, hierarchical models, and prediction, rendering it less general than the other methods on which I focus; and
- **Numerical quadrature** (Smith et al., 1985), which uses ideas from the standard numerical analysis literature on quadrature (Bayesians are not the only people who have to evaluate high-dimensional integrals, after all) modified to the Bayesian context. People working in this area report considerable success in models with small \( k \) (less than about 7),
but the approach seems problematic with a large number of parameters.

2.4 The basic idea (e.g., Bernardo and Smith, 1994) relies on simple Taylor series calculations. In the continuous case, for instance, with $\theta$ a $k$-vector of parameters and $y$ an $n$-vector of univariate outcomes, write $p(\theta|y)$ as proportional to $\exp\{\log[p(\theta)] + \log[l(\theta|y)]\}$; expand each of the log terms inside the exp{} about their respective maxima, keeping only the constant, linear (which vanish), and quadratic bits; and collect like terms together. Then under relatively mild regularity conditions guaranteeing that the remainder terms go to 0 with increasing $n$ (e.g., LeCam and Yang, 1990), $p(\theta|y)$ should be close for large $n$ to a multivariate normal distribution with mean vector

$$H_n^{-1} \left[ H_0 \theta_0 + H(\hat{\theta}) \hat{\theta} \right]$$

(2.50)

and $k$ by $k$ covariance matrix

$$H_n^{-1} = \left[ H_0 + H(\hat{\theta}) \right]^{-1},$$

(2.51)

where $H_0$ is $-1$ times the Hessian (matrix of second derivatives) of the log prior evaluated at the prior mode $\theta_0$ and $H(\hat{\theta})$ is $-1$ times the Hessian of the log likelihood evaluated at the MLE $\hat{\theta}$:

$$H_0 = \left( -\frac{\partial^2 \log[p(\theta)]}{\partial \theta_l \partial \theta_j} \right)_{\theta=\theta_0}, \quad H(\hat{\theta}) = \left( -\frac{\partial^2 \log[l(\theta|y)]}{\partial \theta_l \partial \theta_j} \right)_{\theta=\hat{\theta}}.$$

(2.52)

Notice how similar these expressions are to the corresponding formulae (1.26) and (1.28), in the simple model from Chapter 1 with Gaussian prior and likelihood for an unknown $\mu$: the posterior mean is a weighted average of a prior measure of center $\theta_0$ and a data measure of center $\hat{\theta}$, weighted by the multivariate analogue of their respective precisions, and the posterior precision (matrix) $H_n$ is the sum of the prior ($H_0$) and data ($H(\hat{\theta})$) precisions.

2.5 The main exception I am thinking of is the beautiful Laplace-style investigation conducted by Mosteller and Wallace (1964, 1984) into the authorship of the Federalist papers.

2.6 Intuitively speaking, a (discrete-time) Markov chain (e.g., Feller, 1968; Roberts, 1996; Gamerman, 1997) is a stochastic process unfolding in time in such a way that the past and future
states of the process are independent given the present state—in other words, to figure out where the chain is likely to go next you don’t need to pay attention to where it’s been, you just need to consider where it is now. More formally, a stochastic process \( \{\theta_t, t \in T\}, T = \{0, 1, \ldots\} \), with state space \( S \) is Markov if, for any set \( A \in S \),
\[
P(\theta_{t+1} \in A|\theta_0, \ldots, \theta_t) = P(\theta_{t+1} \in A|\theta_t). \tag{2.53}
\]

The theory of Markov chains is harder mathematically if \( S \) is continuous (e.g., Tierney, 1996), which is what we need for MCMC with real-valued parameters, but most of the main ideas emerge with discrete state spaces, and I will assume discrete \( S \) in the intuitive discussion below. Generalizations to continuous time are also possible (e.g., Feller, 1971) but are not relevant here.

The idea in MCMC is (a) to set things up so that the Markov chain converges to an equilibrium or stationary distribution, and (b) to further contrive that this distribution is \( p(\theta|y) \). To achieve the first goal, the chain needs to satisfy three properties:

- It must be irreducible, which basically means that no matter where it starts the chain has to be able to reach any other state in a finite number of iterations with positive probability;
- It must be aperiodic, meaning that for all states \( i \) the set of possible sojourn times, to get back to \( i \) having just left it, can have no divisor bigger than 1. This forces the chain to mix freely among its possible states rather than oscillating back and forth within a subset of \( S \); and
- It must be positive recurrent, meaning that (a) for all states \( i \), if the process starts at \( i \) it will return to \( i \) with probability 1, and (b) the expected length of waiting time till the first return to \( i \) is finite. Notice that this is a bit delicate: wherever the chain is now, we insist that it must certainly come back here, but we don’t expect to have to wait forever for this to happen.

A positive recurrent and aperiodic chain is called ergodic, and it turns out that such chains possess a unique stationary (or equilibrium, or invariant) distribution \( \pi \), characterized by the relation
\[
\pi(j) = \sum_i \pi(i)P_{ij}(t) \tag{2.54}
\]
for all states \( j \) and times \( t \geq 0 \), where \( P_{ij}(t) = P(\theta_t = j | \theta_0 = i) \) is the transition matrix of the chain. Informally, the stationary distribution characterizes the behavior that the chain will settle into after it has been run for a long time, regardless of its initial state.

The MCMC point of having set up all this machinery is the ergodic theorem: if \( \{\theta_t\} \) is ergodic and \( f \) is any real-valued function for which \( E_\pi[f(\theta)] \) is finite, then with probability 1

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

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 \( p(\theta|y) \) (see the next endnote)—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 (Sections 2.3 and 2.4).

2.7 We may as well look at the stationary distribution in the case of a continuous state space \( S \), for instance \( \mathbb{R}^k \). As noted (for example) by Tierney (1996), to pin down the distribution of a Markov chain \( \theta^{(t)} \) when \( S \) is continuous, you need to know two things: its initial distribution across the states in \( S \), and its transition kernel, the continuous analogue of the transition matrix in Note 6: for any \( \theta \in S, A \subset S \), and \( t \geq 0 \), this is the function \( P(\theta, A) = P(\theta_{t+1} \in A | \theta_t = \theta) \). The transition kernel just specifies the distribution of the chain’s location at time \( (t+1) \) given that it was at \( \theta \) at time \( t \).

The argument (one version of it, at least; e.g., Gilks et al., 1996b) for deriving the stationary distribution proceeds in four steps. (1) By looking at the Hastings algorithm in (2.6) and thinking about the possible moves at any given time \( t \), you can see that the transition kernel of the Hastings sampler satisfies the equation

\[
P(\theta_{t+1} | \theta_t) = f(\theta_{t+1} | \theta_t) \alpha_H(\theta_t, \theta_{t+1}) + I(\theta_{t+1} = \theta_t).
\]
\[ 1 - \int f(\theta|\theta_t) \alpha_H(\theta_t, \theta) \, d\theta, \]  
(2.56)

where \( I(\cdot) \) is the indicator function (the first right-side term in (2.56) picks up the possibility that the chain moves, and the second that it stays put). (2) If you expand out the definitions of both \( \alpha_H(\theta_t, \theta_{t+1}) \) and \( \alpha_H(\theta_{t+1}, \theta_t) \) (by which I mean, for instance, \( \alpha_H(\theta_t, \theta_{t+1}) = \text{something if such-and-such is true and something else if not} \) and form the ratio \( \frac{\alpha_H(\theta_t, \theta_{t+1})}{\alpha_H(\theta_{t+1}, \theta_t)} \), you will see that (Metropolis et al. and) Hastings picked the acceptance probabilities so that

\[ \frac{\alpha_H(\theta_t, \theta_{t+1})}{\alpha_H(\theta_{t+1}, \theta_t)} = \frac{p(\theta_{t+1}|y) f(\theta_t|\theta_{t+1})}{p(\theta_t|y) f(\theta_{t+1}|\theta_t)}. \]  
(2.57)

(3) This, together with (2.56) and some algebra, shows that the chain satisfies the detailed balance equation,

\[ p(\theta_t|y) P(\theta_{t+1}|\theta_t) = p(\theta_{t+1}|y) P(\theta_t|\theta_{t+1}), \]  
(2.58)

which is the crucial thing that gives what we want: (4) Integrating (2.56) over the possible values of \( \theta_t \) and plugging in detailed balance yields

\[ p(\theta_{t+1}|y) = \int p(\theta_t|y) P(\theta_{t+1}|\theta_t) \, d\theta_t, \]  
(2.59)

which is the continuous-state-space version of (2.54). This has demonstrated that, if the Markov chain created by the Hastings algorithm (2.6) has a stationary distribution, then that distribution must be \( p(\theta|y) \); see Tierney (1996) for details and precise conditions that ensure convergence. **NB** Detailed balance is closely related to reversibility of the chain: in the language used here, a Markov chain is reversible if it is positive recurrent with stationary distribution \( p(\cdot|y) \) satisfying the detailed balance condition (2.58).

2.8 This terminology is slightly nonstandard. Most people talk about Metropolis-Hastings sampling without specifying in the name whether the proposal distribution is symmetric (Metropolis; see (2.21)) or not (Hastings), but I will often retain the distinction in what follows.

2.9 **Simulated annealing** (SA; e.g., Geman and Geman 1984) is a stochastic optimization method for maximizing a (nearly) arbitrary real-valued function \( f(\theta) \) (\( \theta \in \mathbb{R}^k \)), based on a nice idea
that fits in well with the other MCMC approaches in this chapter. If $f$ is unimodal then any standard method should find the mode without much trouble, for instance Newton-Raphson from even a not-very-good starting point, so to make things tougher suppose $f$ has one or more local maxima in addition to the global one.

**Algorithm** (*simulated annealing*). To maximize a posterior distribution $p(\theta | y)$, choose a proposal distribution (PD) $f(\theta | \theta_t)$ and a cooling schedule $T_t$, define

$$
\alpha_{SA}(\theta_t, \theta^*) = \exp\left\{ -\frac{\log[p(\theta_t | y)] - \log[p(\theta^* | y)]}{T_t} \right\},
$$

and

Initialize $\theta_0$; $t \leftarrow 0$

Repeat {
    Sample $\theta^* \sim f(\theta | \theta_t)$
    If $p(\theta^* | y) > p(\theta_t | y)$ then $\theta_{t+1} \leftarrow \theta^*$
    else {
        Sample $u \sim U(0,1)$
        If $u \leq \alpha_{SA}(\theta_t, \theta^*)$ then $\theta_{t+1} \leftarrow \theta^*$
        else $\theta_{t+1} \leftarrow \theta_t$
    }
    $t \rightarrow (t+1)$
}

(2.60)

A greedy stochastic hill-climbing strategy in this situation might proceed like this: at time $t$ in the search, (a) generate a new candidate place to consider moving to, say $\theta^*$, and (b) compare $f(\theta_t)$ and $f(\theta^*)$. If the new place is better (higher) then move there (set $\theta_{t+1} = \theta^*$); otherwise discard this $\theta^*$, go back to (a), generate another candidate, and so on. With high likelihood this will eventually get you to the top of the nearest hill, but once you are there it won't allow you to jump away from this hill and find a higher peak (if any) somewhere else. SA improves on this by allowing you to sometimes go downhill (early on in the search process, at least), in the hope that by temporarily making things worse you will eventually wander to the highest place of all. SA implements this by using a rule of the form {if $\theta^*$ is better, then by all means move there, but if it's worse, move there anyway
with probability \( \alpha_{SA} \). Formally, the algorithm, in the context of maximization of a posterior \( p(\theta|y) \), is summarized in (2.60).

The idea behind the cooling schedule in SA is the following. Imagine you were wandering around in the plane (this is like generating proposed places to move when \( \theta \) has \( k = 2 \) components) looking for the maximum of \( p(\theta|y) \), which is (let's say) highly concentrated in a small region, and you were far from that region—the higher the peak was, the easier it would be for you to spot it. This suggests trying to maximize an exaggerated or heightened version of the posterior, for instance \( [p(\theta|y)]^{1/2} \) for \( T < 1 \), instead of \( p \) itself, and the closer \( T \) is to 0 the more exaggerated \( [p(\theta|y)]^{1/2} \) will be. So while you are letting the iteration counter \( t \) run from 1 to \( n_M \) (say), making new proposed moves \( \theta^* \) all the while, why not let \( T \) get smaller and smaller as a function of \( t \) as well? \( T_t \) is called the temperature parameter in the SA algorithm, and any method for monotonically decreasing it from a starting value \( T_0 \) (1.0, say) to a final value \( T_f \) (0.001, say) is called a cooling schedule. There are a number of possibilities; one that often seems to work well (Stander and Silverman, 1994) is a geometric decline, \( T_t = T_0 \gamma^t \) for \( \gamma = \left( \frac{T_f}{T_0} \right)^{1/n_M} \). To find the global mode you run the algorithm repeatedly, each time with a large \( n_M \) (like 10,000), and using a (widely dispersed) variety of starting values \( \theta_0 \), possibly also varying \( T_0 \). Early on in these runs \( \alpha_{SA} \) will be fairly large and you will often jump to locally inferior places, but as \( T_t \) approaches 0 so does \( \alpha_{SA} \) and the process eventually “freezes” at one particular mode. There is no guarantee that this is the global max (which is why you should run it with a number of different \( \theta_0 \)), but SA's willingness to go downhill as well as up often allows it to out-perform greedier search methods at “bump-hunting.”

As I mentioned in the main text, I like SA better than the Gelman-Rubin strategy for finding multiple modes, for the following reason. A bit of algebra should convince you that if, instead of varying the temperature, you hold \( T \) constant at 1, the acceptance probability \( \alpha_{SA} \) coincides with \( \alpha_M \), the acceptance rate (2.21) from Metropolis sampling: indeed when \( T = 1 \), SA and Metropolis are identical. Thus you can solve two important problems with one piece of software by writing an SA routine—cool the process by sending \( T \downarrow 0 \) to find the mode(s), or hold
the temperature constant at 1 to extract the usual posterior marginal and predictive summaries with Metropolis.

2.10 The (sample) autocorrelation function (ACF) for a time series $\theta_t$ (e.g., Box and Jenkins, 1976; Chatfield, 1996) simply measures the degree to which knowledge of the past of the series is linearly predictive of its future. Specifically, this function is given by

$$\hat{\rho}_k = \frac{\sum_{t=1}^{n-k} (\theta_t - \bar{\theta})(\theta_{t-k} - \bar{\theta})}{\sum_{t=1}^{n} (\theta_t - \bar{\theta})^2}$$  \hspace{1cm} (2.61)

as $k$, measuring the number of lags backwards in the series, varies from $0, \ldots, n-1$, where $n$ is the number of observed time points and $\bar{\theta} = \frac{1}{n} \sum_{t=1}^{n} \theta_t$. The first-order autocorrelation (or serial correlation) $\hat{\rho}_1$ is often the star of the show in MCMC work, because the columns in the MCMC data set often behave a lot like autoregressive processes of order 1 (see the next endnote).

2.11 The autoregressive process of order 1 with lag-1 serial correlation $\rho$ (e.g., Box and Jenkins, 1976, Chatfield, 1996), abbreviated $AR_1(\rho)$, is modeled as follows:

$$\theta_t - \mu = \alpha_1 (\theta_{t-1} - \mu) + z_t,$$  \hspace{1cm} (2.62)

where the $z_t$ are white noise, assumed IID $N(0, \sigma^2_z)$. Decent approximate estimates of the mean and regression parameters are given by the intuitively obvious $\hat{\mu} = \bar{\theta}$ and $\hat{\alpha}_1 = \hat{\rho}_1$. The autocorrelation function of an $AR_1$ process is $\rho_k = \rho_1^k$—in other words, for positive $\rho_1$, a plot of the sample autocorrelations should show a steady geometric decay—and the partial autocorrelation function (PACF; endnote 20) has a spike of height $\rho_1$ at lag 1 and is zero thereafter—so the sample ACF and PACF should (in theory) make diagnosing an $AR_1$ process pretty straightforward.

2.12 This makes the sub-chain, observed only at the times at which a move actually takes place, a martingale (e.g., Breiman, 1968), a fact that helps to establish some useful properties of MCMC samplers.

2.13 For many purposes in working with the $SI-\chi^2$ distribution, the factor $(\sigma^2)^{\frac{1}{2}}$ in (2.14) can be treated as a throw-away constant, but not when (2.15) or something like it is used as a PD: to compute the acceptance probability (2.5) the log PD has to be evaluated with arguments $(\sigma^2, \sigma^2)$ half the time and $(\sigma^2, \sigma^2)$ the
other half, so that the term \((\sigma^2)^{\frac{\text{first}}{2}}\) does not cancel in evaluating (2.5). The first time I programmed up the Hastings sampler for this model I made the mistake of ignoring this factor and got results in which the posterior mean depended on \(\nu^\prime\), which of course cannot happen if the implementation is correct. I am indebted to Bill Browne for helping me spot this error.

2.14 All timings in this book were made on a dedicated DECalpha Unix workstation running at 333MHz, or on one or another of a variety of Unix SPARCstation and UltraSPARC CPU servers (with the appropriate conversion in timings made to 333MHz).

2.15 You can show (e.g., Bernardo and Smith, 1994) that the predictive distribution for \(y^\ast\) given \(y\) in this model is a scaled \(t\), with degrees of freedom \((\nu_p + n)\), mean \(\mu\), and scale parameter \(\frac{\nu_p + n}{\nu_p + n}\).

2.16 Another option is to write the slow bits of your program in Fortran or C and call them from within S+; see Venables and Ripley (1997).

2.17 There is a direct analogy between the form and scale of a proposal distribution in MCMC and the choice of kernel and window width in density estimation, and the same results (e.g., Silverman, 1986) apply: it doesn’t matter too much whether you use (say) a Gaussian or uniform PD (kernel); what matters a lot is to get the PD scale (window width) right.

2.18 I actually computed the density estimate on the \(\log(\sigma^2)\) scale and back-transformed it, and I used a big window width, namely 0.25[\(\text{max}\{\log(\sigma^2)\} - \text{min}\{\log(\sigma^2)\}\)]. The S+ code is

```r
  d <- density( log( sigma2 ), width = ( max( log( sigma2 ) ) - min( log( sigma2 ) ) ) / 4 )
  plot( exp( d$x ), d$y, type = 'l', xlab = 'sigma2', ylab = 'Density', xlim = c( 25, 75 ) )
```

2.19 The partial autocorrelation function (PACF) \(\phi_{kk}\) at lag \(k\) of a time series measures “the excess correlation not accounted for by an AR_{k-1} model” (Chatfield, 1996), and may be estimated by “successively fitting AR_{p} processes for \(p = 1, 2, \ldots\) and picking out the estimates of the last coefficients fitted at each stage” (Box and Jenkins, 1976). Thus the PACF is a kind of direct diagnostic for AR_{p} processes: if something is (say) AR_{1} then there will be no “excess correlation not accounted for by an AR_{1}
model," so that the PACF at lag 2 and thereafter will be zero. Specifically, for AR1 models, \( \phi_{11} = \rho_1 \) and \( \phi_{22} = \frac{\rho_2 - \rho_1^2}{1 - \rho_1^2} \), so that \( \phi_{22} = 0 \) for an AR1 process (because for such a process \( \rho_2 = \rho_1^2 \); see endnote 11). When least squares or something equivalent is used to estimate the \( \phi_{kk} \), you can show that \( SE(\hat{\phi}_{kk}) = \frac{1}{\sqrt{n}} \) for \( k \geq p + 1 \) when the series is \( AR_p \) (and a similar result applies for the ACF), which explains the little horizontal dotted lines in Figure 2.6: it is natural, in diagnosing the order of an \( AR_p \) process from the PACF, to look for the first lag \( k = p + 1 \) for which \( \phi_{kk} \) and all higher partial autocorrelations do not differ significantly from 0, and conclude tentatively that the series is \( AR_p \). On this basis both the ACF and PACF in Figure 2.6 are nearly perfect \( AR_1 \).

2.20 [xx details on Heidelberger-Welch]

2.21 Some of the results in this subsection are joint with Bill Browne, who has more extensive findings of this type in his PhD dissertation, Browne (1998).

2.22 Of course, as usual in the design of sampling experiments, if I decide that I can't afford 248K iterations I can take solace in less stringent requirements. For instance, if all I want is to make sure that the Monte Carlo standard error for the posterior mean of \( \hat{\tau}^2 \) is 0.05, then I get to divide 248K by 1.96^2, producing the more pleasant-sounding target of 65K iterations, and if I relax \( d \) further from 0.05 to 0.1 I can divide again by \( \left( \frac{0.1}{0.05} \right)^2 \) to obtain a requirement of 16K iterations, which brings me into the Raftery-Lewis default neighborhood. Also, to be fair, this is Hastings, not Gibbs (see Section 2.5), where the serial correlations are usually lower.

2.23 This can be shown directly, using reasoning like that laid out in Note 7, but the similarity between Gibbs and Metropolis-Hastings (MH) should suggest an easier route—if you could show that Gibbs is a special case of MH, then you would be done. But this is not hard, as follows. As I will examine in Section 2.6, given a parameter vector \( \theta \) of dimension \( k \), you could either make an MH update on all of \( \theta \) at once, or you could divide \( \theta \) into subcomponents or blocks (which may have only one element of \( \theta \) in them) and update the blocks sequentially. The latter approach, which people call single-component Metropolis-Hastings, was in fact the one originally proposed by Metropolis et al. Once a
series of proposal distributions for each block is specified, possibly conditional on the current results in some or all of the other blocks, it is straightforward (see, e.g., Gilks et al., 1996b) to show that Gibbs is just a special case of single-component MH in which the acceptance probability is always 1.

2.24 This is the same idea on which the Shapiro-Wilk (1965) test for normality is based: essentially they compute the squared correlation between the ordered data values and the expected quantiles of a standard normal, which is just a numerical summary of the fit to the Gaussian as diagnosed by the usual normal qqplot.

2.25 If the log density of the full conditional is sufficiently smooth to have a second derivative, log concavity means that this derivative should be everywhere non-negative. Gilks (1992), who developed adaptive-rejection sampling with Wild (Gilks and Wild, 1992), actually requires less than this: only that if you pick three values \( \theta_1 \) in the support of the full conditional \( p(\theta) \) (say), with \( \theta_1 < \theta_2 < \theta_3 \), and define the points \( P_i = [\theta_i, p(\theta_i)] \), the gradient of the chord joining \( P_2 \) and \( P_3 \) can be no larger than that of the chord joining \( P_1 \) and \( P_2 \).

2.26 In fact log-concavity holds in this model for both the full conditionals for \( \sigma \) and \( \nu \), but BUGS seems unable to verify this fact.
References


REFERENCES


REFERENCES


REFERENCES


Stander J, Silverman BW (1994). [xx supply this]


