Chapter 6

Bayesian Statistics

6.1 Multidimensional Bayesian Analysis

This chapter takes up Bayesian statistics. Modern Bayesian statistics relies heavily on computers, computation, programming, and algorithms, so that will be the major focus of this chapter. We cannot give a complete treatment here, but there are several good books that cover these topics in more depth. See, for example, Gelman et al. [2004], Liu [2004], Marin and Robert [2007], or Robert and Casella [1997].

Recall the framework of Bayesian inference from Section 2.5.

- We posit a parametric family of distributions $\{p(y \mid \theta)\}$.
- We express our old knowledge of θ through a prior probability density $p(\theta)$.
- The previous two items combine to yield $p(y, \theta)$ and, ultimately, $p(\theta \mid y)$.
- The posterior density $p(\theta \mid y)$ represents our new state of knowledge about θ.

The posterior density is

$$p(\theta \mid y) = \frac{p(\theta)p(y \mid \theta)}{\int p(\theta)p(y \mid \theta)\, d\theta} \propto p(\theta)p(y \mid \theta). \tag{6.1}$$

So far, so good. But in many interesting applications, θ is multi-dimensional and problems arise when we want to examine the posterior. Equation 6.1 tells us how to evaluate the posterior at any value of θ, but that's not always sufficient for getting a sense of which values of θ are most likely, somewhat likely, unlikely, etc. One way to develop a feeling for a multidimensional posterior is to examine a marginal posterior density, say

$$p(\theta_1 \mid y) = \int \cdots \int p(\theta_1, \ldots, \theta_k \mid y)\, d\theta_2 \ldots d\theta_k. \tag{6.2}$$

Unfortunately, the integral in Equation 6.2 is often not analytically tractable and must be integrated numerically. Standard numerical integration techniques such as quadrature may work well in low dimensions, but in Bayesian statistics Equation 6.2 is often sufficiently high dimensional that standard techniques are unreliable. Therefore, new numerical integration techniques are needed. The most important of these is called Markov chain Monte Carlo integration, or MCMC. Other techniques can be found in the references at the beginning of the chapter. For the purposes of this book, we investigate MCMC. But first, to get a feel for Bayesian analysis, we explore posteriors in low dimensional, numerically tractable examples.

The general situation is that there are multiple parameters $\theta_1, \ldots, \theta_k$, and data $y_1, \ldots, y_n$. We may be interested in marginal, conditional, or joint distributions of the parameters either *a priori* or *a posteriori*. Some examples:

- $p(\theta_1, \ldots, \theta_k)$, the joint prior
- $p(\theta_1, \ldots, \theta_k \,|\, y_1, \ldots, y_n)$, the joint posterior
- $p(\theta_1 \,|\, y_1, \ldots, y_n) = \int \cdots \int p(\theta_1, \ldots, \theta_n \,|\, y_1, \ldots, y_n)\, d\theta_2 \cdots d\theta_k$, the marginal posterior of θ_1
- $$\begin{aligned} p(\theta_2, \ldots, \theta_k \,|\, \theta_1, y_1, \ldots, y_n) &= p(\theta_1, \ldots, \theta_k \,|\, y_1, \ldots, y_n)/p(\theta_1 \,|\, y_1, \ldots, y_n) \\ &\propto p(\theta_1, \ldots, \theta_k \,|\, y_1, \ldots, y_n), \end{aligned}$$

  the conditional joint posterior density of $(\theta_2, \ldots, \theta_k)$ given θ_1, where the "$\propto$" means that we substitute θ_1 into the numerator and treat the denominator as a constant.

The examples in this section illustrate the ideas.

Example 6.1 (Ice Cream Consumption, cont.)
This example continues Example 3.5 in which weekly ice cream consumption is modelled as a function of temperature and possibly other variables. We begin with the model in Equation 3.10:

$$\text{consumption} = \beta_0 + \beta_1 \text{temperature} + \text{error}$$

or

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

where $\epsilon_1, \ldots, \epsilon_{30} \sim$ i.i.d.N$(0, \sigma)$. For now, let us suppose that σ is known. Later we'll drop that assumption. Because σ is known, there are only two parameters: β_0 and β_1.

For a Bayesian analysis we need a prior distribution for them; then we can compute the posterior distribution. For now we adopt the following prior without comment. Later we will see why we chose this prior and examine its consequences.

$$\begin{aligned}\beta_0 &\sim \mathsf{N}(\mu_0, \sigma_0)\\ \beta_1 &\sim \mathsf{N}(\mu_1, \sigma_1)\\ \beta_0 &\perp \beta_1\end{aligned}$$

i. e.

$$\begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix} \sim \mathsf{N}\left(\begin{pmatrix}\mu_0\\ \mu_1\end{pmatrix}, \begin{pmatrix}\sigma_0^2 & 0\\ 0 & \sigma_1^2\end{pmatrix}\right)$$

for some choice of $(\mu_0, \mu_1, \sigma_0, \sigma_1)$. The likelihood function is

$$\begin{aligned}\ell(\beta_0, \beta_1) &= p(y_1, \ldots, y_{30} \,|\, \beta_0, \beta_1)\\ &= \prod_{i=1}^{30} p(y_i \,|\, \beta_0, \beta_1)\\ &= \prod_{i=1}^{30} \left(\frac{1}{\sqrt{2\pi}\sigma}\right) e^{-\frac{1}{2}\left(\frac{y_i - (\beta_0 + \beta_1 x_i)}{\sigma}\right)^2}\end{aligned}$$

To find the posterior density we will use matrix notation. Let $\boldsymbol{\beta} = \begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix}$, $\boldsymbol{\mu} = \begin{pmatrix}\mu_0\\ \mu_1\end{pmatrix}$, $\Sigma = \begin{pmatrix}\sigma_0^2 & 0\\ 0 & \sigma_1^2\end{pmatrix}$, $\vec{Y} = (Y_1, \ldots, Y_n)^t$ and

$$X = \begin{pmatrix}1 & \text{temperature}_1\\ 1 & \text{temperature}_2\\ \vdots & \vdots\\ 1 & \text{temperature}_{30}\end{pmatrix}.$$

Conditional on $\boldsymbol{\beta}$, $\vec{Y}$ has a 30-dimensional Normal distribution with mean $X\boldsymbol{\beta}$ and covariance matrix $\sigma^2 I_{30}$. The posterior density is proportional to the prior times the likelihood.

$$p(\boldsymbol{\beta} \,|\, \vec{Y}) \propto e^{-\frac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\mu})^t \Sigma^{-1} (\boldsymbol{\beta}-\boldsymbol{\mu}) - \frac{1}{2}(\vec{Y} - X\boldsymbol{\beta})^t (\vec{Y} - X\boldsymbol{\beta})/\sigma^2} \tag{6.3}$$

At this point we observe that the exponent is a quadratic form in $\boldsymbol{\beta}$. Therefore the posterior density will be a two-dimensional Normal distribution for $\boldsymbol{\beta}$ and we just have

to complete the square to find the mean vector and covariance matrix. The exponent is, apart from a factor of $-\frac{1}{2}$ and some irrelevant constants involving the y_i's,

$$\boldsymbol{\beta}^t[\Sigma^{-1} + X^tX/\sigma^2]\boldsymbol{\beta} - 2\boldsymbol{\beta}^t[\Sigma^{-1}\boldsymbol{\mu} + X^t\vec{Y}/\sigma^2] + \cdots = (\boldsymbol{\beta} - \boldsymbol{\mu}^*)^t(\Sigma^*)^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}^*) + \cdots$$

where $\Sigma^* = (\Sigma^{-1} + X^tX/\sigma^2)^{-1}$ and $\boldsymbol{\mu}^* = \Sigma^*(\Sigma^{-1}\boldsymbol{\mu} + X^t\vec{Y}/\sigma^2)$. Therefore, the posterior distribution of $\boldsymbol{\beta}$ given $\vec{Y}$ is Normal with mean $\boldsymbol{\mu}^*$ and covariance matrix Σ^*. It is worth noting (1) that the posterior precision matrix $(\Sigma^*)^{-1}$ is the sum of the prior precision matrix Σ^{-1} and a part that comes from the data, X^tX/σ^2 and (2) that the posterior mean is $\Sigma^*(\Sigma^{-1}\boldsymbol{\mu} + X^t\vec{Y}/\sigma^2) = \Sigma^*(\Sigma^{-1}\boldsymbol{\mu} + (X^tX/\sigma^2)(X^tX)^{-1}X^t\vec{Y})$, a matrix weighted average of the prior mean $\boldsymbol{\mu}$ and the least-squares estimate $(X^tX)^{-1}X^t\vec{Y}$ where the weights are the two precisions Σ^{-1} and X^tX/σ^2.

The derivation of the posterior distribution does not depend on any particular choice of $(\boldsymbol{\mu}, \Sigma)$, but it does depend on the fact that the prior distribution was Normal because that's what gives us the quadratic form in the exponent. That's one reason we took the prior distribution for $\boldsymbol{\beta}$ to be Normal: it made the calculations easy.

Now let's look at the posterior more closely, see what it implies for (β_0, β_1), and see how sensitive the conclusions are to the choice of $(\boldsymbol{\mu}, \Sigma)$. We're also treating σ as known, so we'll need a value. Let's start with the choice

$$\boldsymbol{\mu} = \begin{pmatrix}\mu_0\\ \mu_1\end{pmatrix} = \begin{pmatrix}0\\ 0\end{pmatrix}; \qquad \Sigma = \begin{pmatrix}\sigma_0^2 & 0\\ 0 & \sigma_1^2\end{pmatrix} = \begin{pmatrix}10^6 & 0\\ 0 & 10^6\end{pmatrix}; \qquad \text{and} \qquad \sigma = 0.05. \tag{6.4}$$

The large diagonal entries in Σ say that we have very little *a priori* knowledge of $\boldsymbol{\beta}$. We can use `R` to calculate the posterior mean and covariance.

```
ic <- read.table ( "data/ice cream.txt", header=T )

mu <- c ( 0, 0 )
Sig <- diag ( rep ( 10^6, 2 ) )
sig <- 0.05
x <- cbind ( 1, ic$temp )
y <- ic$IC

Sigstar <- solve ( solve(Sig) + t(x) %*% x / (sig^2) )
mustar <- Sigstar %*% ( solve(Sig) %*% mu + t(x) %*% y / (sig^2) )
```

The result is

$$\boldsymbol{\mu}^* = \begin{pmatrix}.207\\ .003\end{pmatrix} \qquad \text{and} \qquad \Sigma^* = \begin{pmatrix}8.54\times10^{-4} & -1.570\times10^{-5}\\ -1.570\times10^{-5} & 3.20\times10^{-7}\end{pmatrix}. \tag{6.5}$$

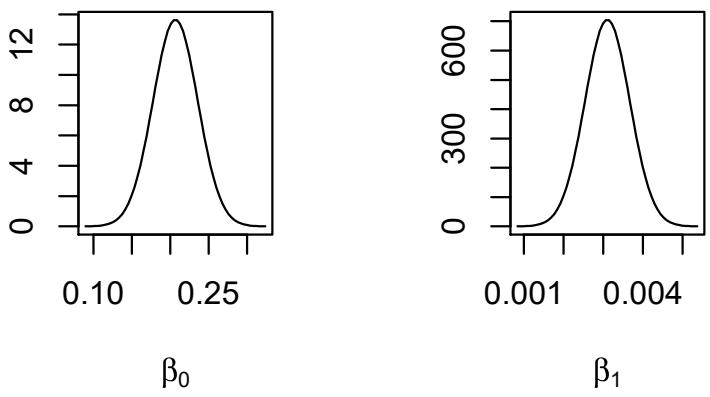

Figure 6.1: Posterior densities of β_0 and β_1 in the ice cream example using the prior from Equation 6.4.

Note:

- `solve` performs matrix inversions and solves systems of linear equations.
- `t(x)` is the transpose of `x`.
- `%*%` performs matrix multiplication.
- The matrix product `t(x) %*% y` is so common it has a special shortcut, `crossprod ( x, y )`. We could have used `crossprod` to find Σ^* and $\boldsymbol{\mu}^*$.

Figure 6.1 shows the posterior densities. The posterior density of β_0 is not very meaningful because it pertains to ice cream consumption when the temperature is 0. Since our data was collected at temperatures between about 25 and 75, extrapolating to temperatures around 0 would be dangerous. And because β_0 is not meaningful, neither is the joint density of (β_0, β_1). On the other hand, our inference for β_1 *is* meaningful. It says that ice cream consumption goes up about .003 ($\pm .001$ or so) pints per person for every degree increase in temperature. You can verify whether that's sensible by looking at Figure 3.8.

Figure 6.1 was produced by the following snippet.

```
opar <- par ( mfrow=c(1,2) )
m  <- mustar[1]
sd <- sqrt ( Sigstar[1,1] )
x <- seq ( m-4*sd, m+4*sd, length=60 )
plot ( x, dnorm ( x, m, sd ), type="l",
       xlab=expression(beta[0]), ylab="" )
m  <- mustar[2]
sd <- sqrt ( Sigstar[2,2] )
x <- seq ( m-4*sd, m+4*sd, length=60 )
plot ( x, dnorm ( x, m, sd ), type="l",
       xlab=expression(beta[1]), ylab="" )
```

Now we'd like to investigate the role of the prior density. We notice that the prior SD of β_1 was 10^3 while the posterior SD is $\sqrt{3.2 \times 10^{-7}} \approx 5.7 \times 10^{-4}$. In other words, the data has reduced the uncertainty by a huge amount. Because there's so much information in the data, we expect the prior to have little influence. We can verify that by considering priors with different SD's and comparing their posteriors. To that end, consider the prior

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_0 \\ \mu_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}; \qquad \Sigma = \begin{pmatrix} \sigma_0^2 & 0 \\ 0 & \sigma_1^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}; \qquad \text{and} \qquad \sigma = 0.05. \tag{6.6}$$

With prior 6.6, the posterior would be

$$\boldsymbol{\mu}^* = \begin{pmatrix} .207 \\ .003 \end{pmatrix} \qquad \text{and} \qquad \Sigma^* = \begin{pmatrix} 8.53 \times 10^{-4} & -1.568 \times 10^{-5} \\ -1.568 \times 10^{-5} & 3.19 \times 10^{-7} \end{pmatrix},$$

nearly identical to posterior 6.5. That's because the data contain much more information than the prior, so the prior plays a negligible role in determining the posterior distribution. The posterior precision matrix (inverse covariance matrix) is $\Sigma^{-1} + X^t X/\sigma^2$. In our case, $X^t X/\sigma^2 = \begin{pmatrix} 30 & 1473 \\ 1473 & 80145 \end{pmatrix}$, which has entries much bigger than those in Σ^{-1}, whichever prior we use. Even if we did a careful job of assessing our prior, it would be influential only if our prior precision matrix had entries of the same order of magnitude as $X^t X/\sigma^2$. Since that's unlikely — our true *a priori* variances are probably not as small as 1/30 — there's little to be gained by choosing the prior carefully and much effort to be saved by using an arbitrary prior, as long as it has reasonably large variances.

ring	ID	xcoor	ycoor	spec	dbh	1998	1999	2000
1	11003	0.71	0.53	pita	19.4	0	0	0
1	11004	1.26	2.36	pita	14.1	0	0	4
1	11011	1.44	6.16	pita	19.4	0	6	0
1	11013	3.56	5.84	pita	21.6	0	0	0
1	11017	3.75	8.08	pita	10.8	0	0	0
				⋮				
6	68053	0.82	10.73	pita	14.4	0	0	0
6	68055	-2.24	13.34	pita	11	0	0	0
6	68057	-0.78	14.21	pita	8	0	0	0
6	68058	0.76	14.55	pita	10.6	0	0	0
6	68059	1.48	13	pita	21.2	0	5	10

Table 6.1: The numbers of pine cones on trees in the FACE experiment, 1998–2000.

Example 6.2 (Pine Cones)
One possible result of increased CO_2 in the atmosphere is that plants will use some of the excess carbon for reproduction, instead of growth. They may, for example produce more seeds, produce bigger seeds, produce seeds earlier in life, or produce seeds when they, the plants, are smaller. To investigate this possibility in the Duke FACE experiment (See Example 1.12 and its sequels.) a graduate student went to the FACE site each year and counted the number of pine cones on pine trees in the control and treatment plots (LaDeau and Clark [2001] and Example 3.8).

The data are in Table 6.1. The first column is *ring*. Rings 1, 5, and 6 were control; 2, 3, and 4 were treatment. The next column, *ID*, identifies each tree uniquely; *xcoor* and *ycoor* give the location of the tree. The next column, *spec*, gives the species; *pita* stands for *pinus taeda*, or loblolly pine, the dominant canopy tree in the FACE experiment. The column *dbh* gives diameter at breast height, a common way for foresters and ecologists to measure the size of a tree. The final three columns show the number of pine cones in 1998, 1999, and 2000. We investigate the relationship between dbh and the number of pine cones, and whether that relationship is the same in the control and treatment plots.

Figures 6.2, 6.3, and 6.4 plot the numbers of pine cones as a function of dbh in the years 1998–2000. In 1998, very few trees had pine cones and those that did had very few. But by 1999, many more trees had cones and had them in greater number. There does not appear to be a substantial difference between 1999 and 2000. As a quick check of our visual impression we can count the fraction of pine trees having pine cones each year, by ring. The following R code does the job.

```
for ( i in 1:6 ) {
  good <- cones$ring == i
  print ( c ( sum ( cones$X1998[good] > 0 ) / sum(good),
              sum ( cones$X1999[good] > 0 ) / sum(good),
              sum ( cones$X2000[good] > 0 ) / sum(good) ) )
}
[1] 0.0000000 0.1562500 0.2083333
[1] 0.05633803 0.36619718 0.32394366
[1] 0.01834862 0.21100917 0.27522936
[1] 0.05982906 0.39316239 0.37606838
[1] 0.01923077 0.10576923 0.22115385
[1] 0.04081633 0.19727891 0.18367347
```

Since there's not much action in 1998 we will ignore the data from that year. The data show a greater contrast between treatment (rings 2, 3, 4) and control (rings 1, 5, 6) in 1999 than in 2000. So for the purpose of this example we'll use the data from 1999. A good scientific investigation, though, would use data from all years.

We're looking for a model with two features: (1) the probability of cones is an increasing function of dbh and of the treatment and (2) given that a tree has cones, the number of cones is an increasing function of dbh and treatment. Here we describe a simple model with these features. The idea is (1) a logistic regression with covariates dbh and treatment for the probability that a tree is sexually mature and (2) a Poisson regression with covariates dbh and treatment for the number of cones given that a tree is sexually mature. Let Y_i be the number of cones on the i'th tree. Our model is

$$
\begin{aligned}
x_i &= \begin{cases} 1 & \text{if the } i\text{'th tree had extra } CO_2 \\ 0 & \text{otherwise} \end{cases} \\
\theta_i &= \begin{cases} 1 & \text{if the } i\text{'th tree is sexually mature} \\ 0 & \text{otherwise} \end{cases} \\
\pi_i &= \mathrm{P}[\theta_i = 1] = \frac{\exp(\beta_0 + \beta_1 \mathsf{dbh}_i + \beta_2 x_i)}{1 + \exp(\beta_0 + \beta_1 \mathsf{dbh}_i + \beta_2 x_i)} \\
\phi_i &= \exp(\gamma_0 + \gamma_1 \mathsf{dbh}_i + \gamma_2 x_i) \\
Y_i &\sim \mathsf{Poi}(\theta_i \phi_i)
\end{aligned}
\tag{6.7}
$$

This model is called a *zero-inflated Poisson model.* There are six unknown parameters: β_0, β_1, β_2, γ_0, γ_1, γ_2. We must assign prior distributions and compute posterior

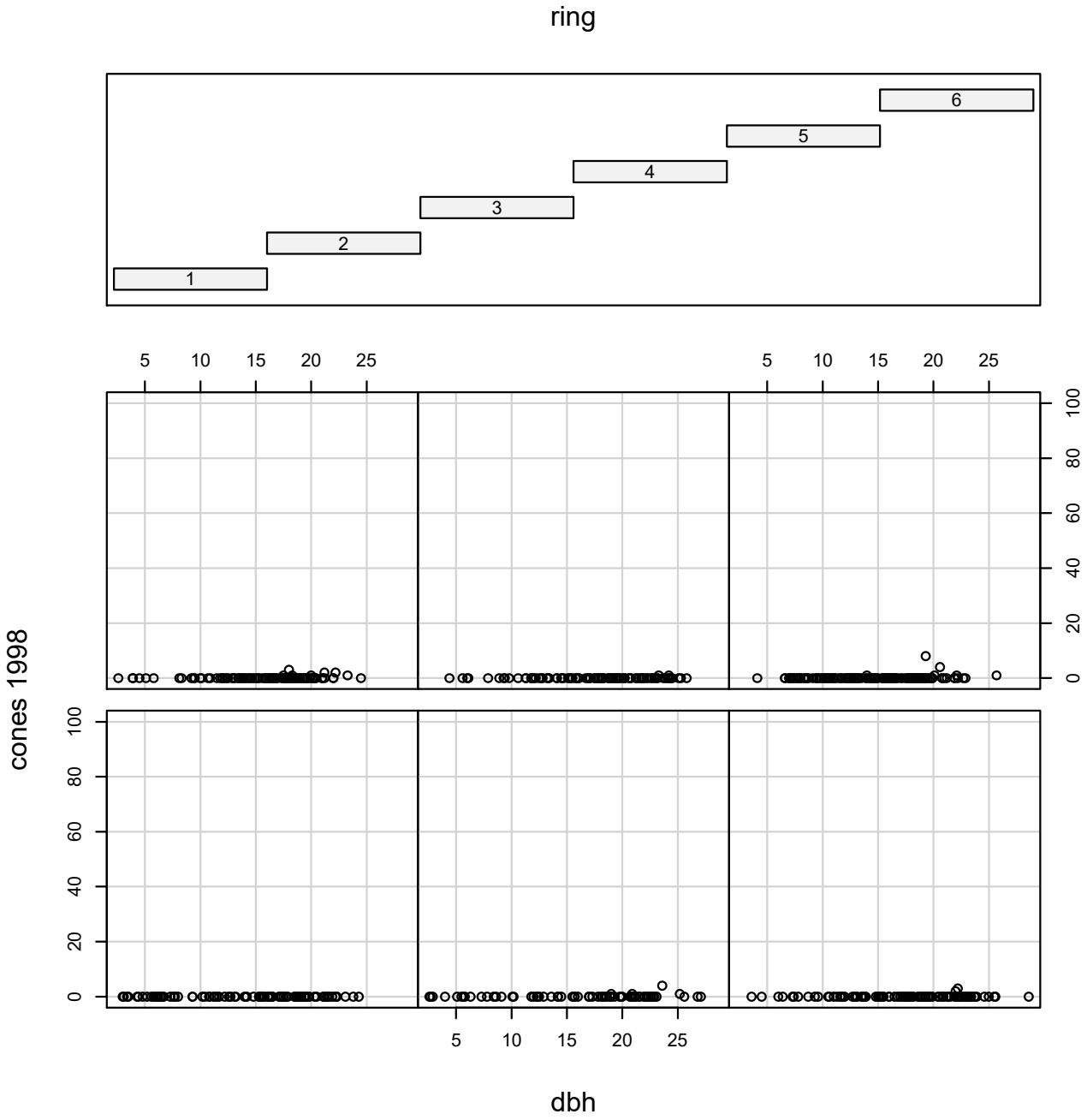

Figure 6.2: Numbers of pine cones in 1998 as a function of dbh

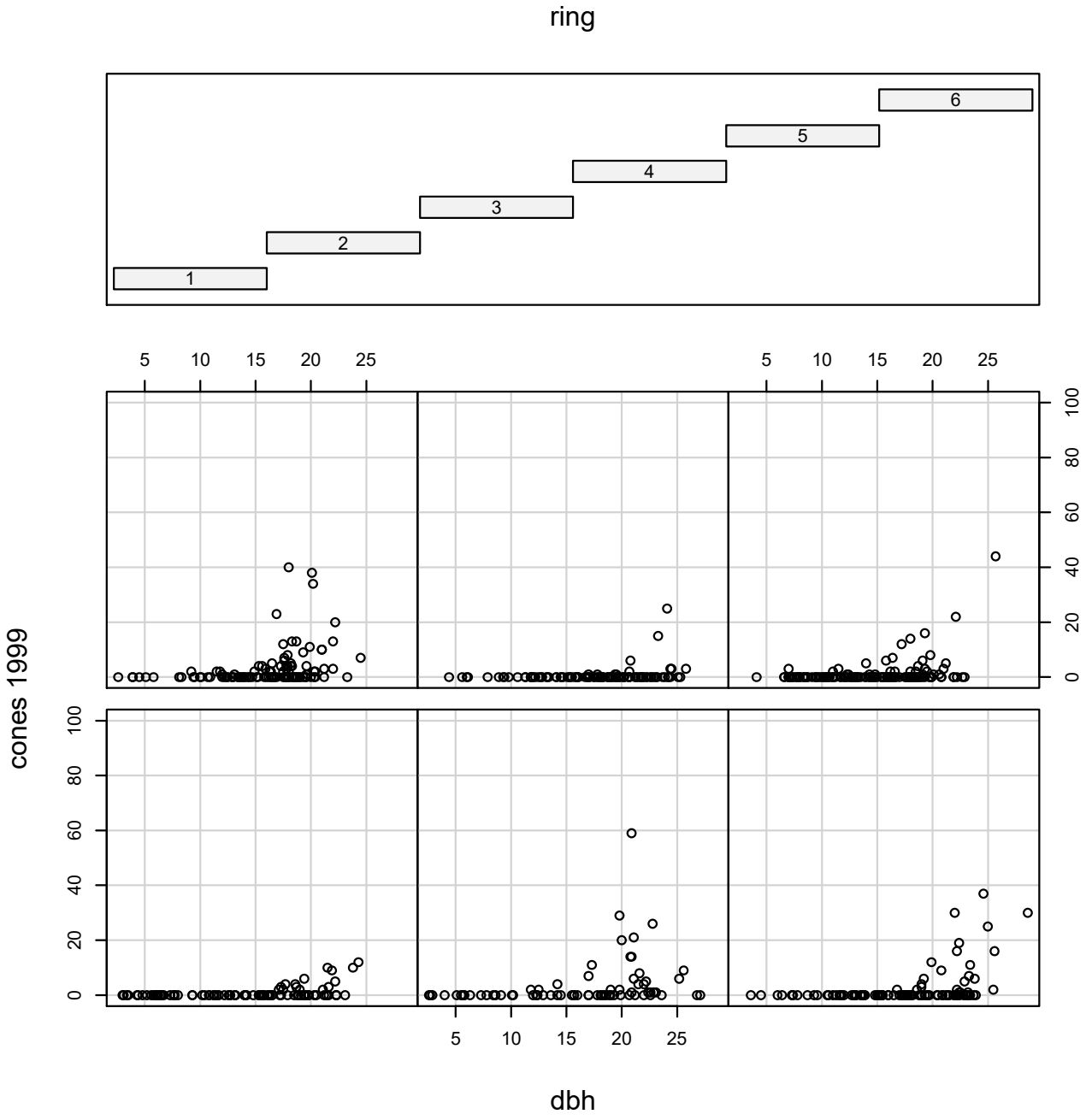

Figure 6.3: Numbers of pine cones in 1999 as a function of dbh

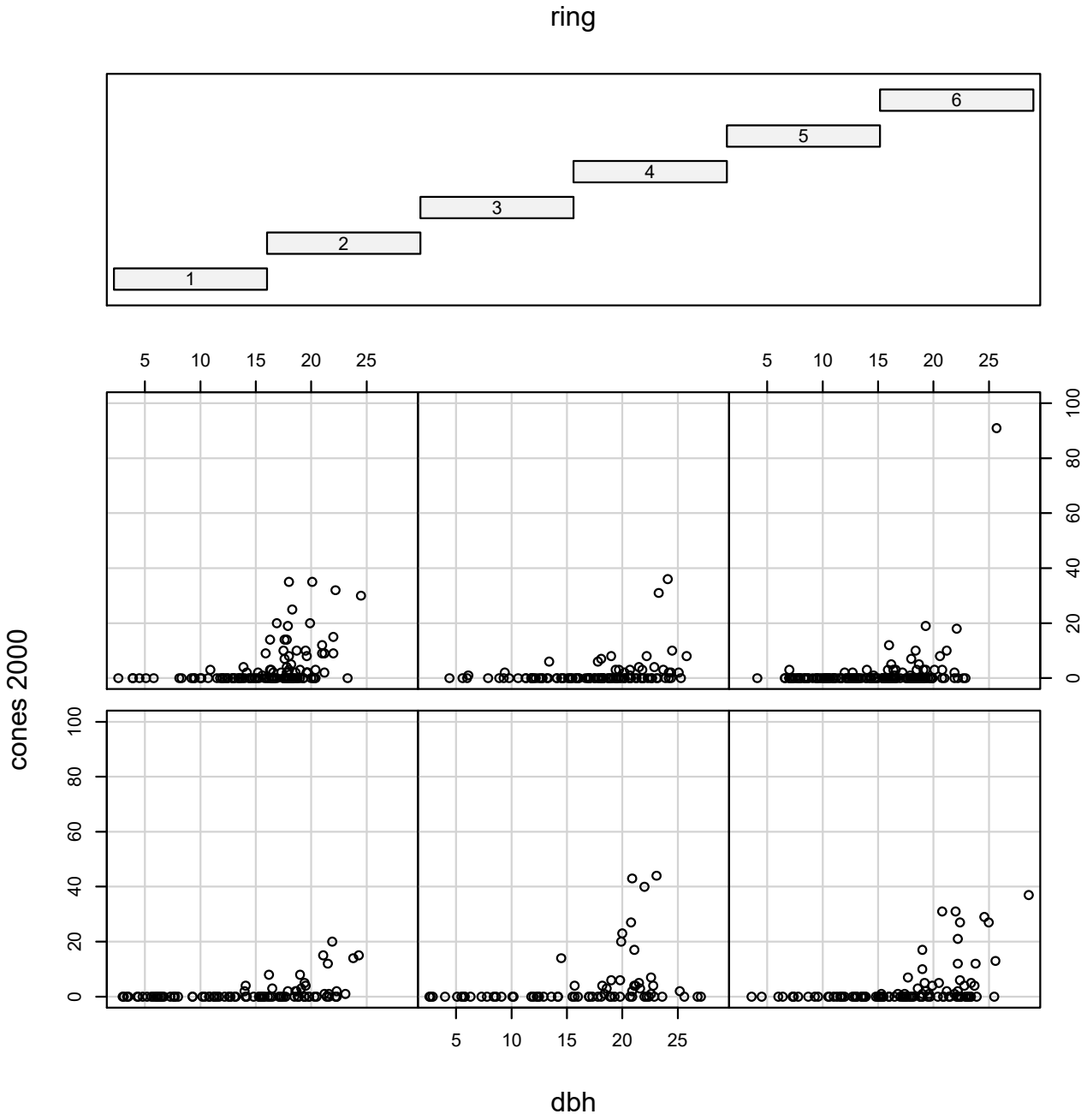

Figure 6.4: Numbers of pine cones in 2000 as a function of dbh

distributions of these parameters. In addition, each tree has an indicator θ_i and we will be able to calculate the posterior probabilities $\mathrm{P}[\theta_i = 1 \,|\, y_1, \ldots, y_n]$ for $i = 1, \ldots, n$.

We start with the priors $\beta_0, \beta_1, \beta_2, \gamma_0, \gamma_1, \gamma_2 \sim$ i.i.d.U$(-100, 100)$. This prior distribution is not, obviously, based on any substantive prior knowledge. Instead of arguing that this is a sensible prior, we will later check the robustness of conclusions to specification of the prior. If the conclusions are robust, then we will argue that almost any sensible prior would lead to roughly the same conclusions.

To begin the analysis we write down the joint distribution of parameters and data.

$$
\begin{aligned}
&p(y_1, \ldots, y_n, \beta_0, \beta_1, \beta_2, \gamma_0, \gamma_1, \gamma_2) \\
&\qquad = p(\beta_0, \beta_1, \beta_2, \gamma_0, \gamma_1, \gamma_2) \times p(y_1, \ldots, y_n \,|\, \beta_0, \beta_1, \beta_2, \gamma_0, \gamma_1, \gamma_2) \\
&\qquad = \left(\frac{1}{200}\right)^6 \mathbf{1}_{(-100,100)}(\beta_0) \times \mathbf{1}_{(-100,100)}(\beta_1) \times \mathbf{1}_{(-100,100)}(\beta_2) \times \mathbf{1}_{(-100,100)}(\gamma_0) \\
&\qquad\qquad \times \mathbf{1}_{(-100,100)}(\gamma_1) \times \mathbf{1}_{(-100,100)}(\gamma_2) \\
&\times \prod_{i:y_i>0} \left(\frac{\exp(\beta_0 + \beta_1 \mathsf{dbh}_i + \beta_2 x_i)}{1 + \exp(\beta_0 + \beta_1 \mathsf{dbh}_i + \beta_2 x_i)} \right. \\
&\qquad\qquad \left. \times \frac{\exp(-\exp(\gamma_0 + \gamma_1 \mathsf{dbh}_i + \gamma_2 x_i)) \exp(\gamma_0 + \gamma_1 \mathsf{dbh}_i + \gamma_2 x_i)^{y_i}}{y_i!} \right) \\
&\times \prod_{i:y_i=0} \left(\frac{1}{1 + \exp(\beta_0 + \beta_1 \mathsf{dbh}_i + \beta_2 x_i)} + \right. \\
&\qquad\qquad \left. \frac{\exp(\beta_0 + \beta_1 \mathsf{dbh}_i + \beta_2 x_i)}{1 + \exp(\beta_0 + \beta_1 \mathsf{dbh}_i + \beta_2 x_i)} \exp(-\exp(\gamma_0 + \gamma_1 \mathsf{dbh}_i + \gamma_2 x_i)) \right) \qquad (6.8)
\end{aligned}
$$

In Equation 6.8 each term in the product $\prod_{i:y_i>0}$ is

$$\mathrm{P}[i\text{'th tree is sexually mature}] \times p(y_i \,|\, i\text{'th tree is sexually mature})$$

while each term in $\prod_{i:y_i=0}$ is

$$\mathrm{P}[i\text{'th tree is immature}] + \mathrm{P}[i\text{'th tree is mature but produces no cones}].$$

The posterior $p(\beta_0, \beta_1, \beta_2, \gamma_0, \gamma_1, \gamma_2 \,|\, y_1, \ldots, y_n)$ is proportional, as a function of $(\beta_0, \beta_1, \beta_2, \gamma_0, \gamma_1, \gamma_2)$, to Equation 6.8. Similarly, conditional posteriors such as $p(\beta_0 \,|\, \beta_1, \beta_2, \gamma_0, \gamma_1, \gamma_2, y_1, \ldots, y_n)$ are proportional, as a function of β_0, to Equation 6.8. But that doesn't allow for much simplification; it allows us to ignore only the factorials in the denominator.

To learn about the posterior in, say, Equation 6.8 it is easy to write an R function that accepts $(\beta_0, \beta_1, \beta_2, \gamma_0, \gamma_1, \gamma_2)$ as input and returns 6.8 as output. But that's quite a complicated function of $(\beta_0, \beta_1, \beta_2, \gamma_0, \gamma_1, \gamma_2)$ and it's not obvious how to use the function or what it says about any of the six parameters or the θ_i's. Therefore, in Section 6.2 we present an algorithm that is very powerful for evaluating the integrals that often arise in multivariate Bayesian analyses.

6.2 The Metropolis, Metropolis-Hastings, and Gibbs Sampling Algorithms

In "Markov chain Monte Carlo", the term "Monte Carlo" refers to evaluating an integral by using many random draws from a distribution. To fix ideas, suppose we want to evaluate Equation 6.2. Let $\vec{\theta} = (\theta_1, \ldots, \theta_k)$. If we could generate many samples $\vec{\theta}_1, \ldots, \vec{\theta}_M$ of $\vec{\theta}$ (where $\vec{\theta}_i = (\theta_{i,1}, \ldots, \theta_{i,k})$) from its posterior distribution then we could approximate Equation 6.2 by

1. discarding $\theta_{i,2}, \ldots, \theta_{i,k}$ from each iteration,
2. retaining $\theta_{1,1}, \ldots, \theta_{M,1}$,
3. using $\theta_{1,1}, \ldots, \theta_{M,1}$ and standard density estimation techniques (page 105) to estimate $p(\theta_1 \,|\, y)$, or
4. for any set A, using
$$\frac{\text{number of } \theta_{i,1}\text{'s in } A}{M}$$
as an estimate of $\mathrm{P}[\theta_1 \in A \,|\, y]$.

That's the idea behind Monte Carlo integration.

The term "Markov chain" refers to how the samples $\vec{\theta}_1, \ldots, \vec{\theta}_M$ are produced. In a Markov chain there is a *transition density* or *transition kernel* $k(\vec{\theta}_i \,|\, \vec{\theta}_{i-1})$ which is a density for generating $\vec{\theta}_i$ given $\vec{\theta}_{i-1}$. We first choose $\vec{\theta}_1$ almost arbitrarily, then generate $(\vec{\theta}_2 \,|\, \vec{\theta}_1)$, $(\vec{\theta}_3 \,|\, \vec{\theta}_2)$, and so on, in succession, for as many steps as we like. Each $\vec{\theta}_i$ has a density $p_i \equiv p(\vec{\theta}_i)$ which depends on $\vec{\theta}_1$ and the transition kernel. But,

1. under some fairly benign conditions (See the references at the beginning of the chapter for details.) the sequence $p_1, p_2, \ldots$ converges to a limit p, the *stationary distribution*, that does not depend on $\vec{\theta}_1$;

2. the transition density $k(\vec{\theta}_i \,|\, \vec{\theta}_{i-1})$ can be chosen so that the stationary distribution p is equal to $p(\vec{\theta} \,|\, y)$;

3. we can find an m such that $i > m \Rightarrow p_i \approx p = p(\vec{\theta} \,|\, y)$;

4. then $\vec{\theta}_{m+1}, \ldots, \vec{\theta}_M$ are, approximately, a sample from $p(\vec{\theta} \,|\, y)$.

The Metropolis-Hastings algorithm [Metropolis et al., 1953, Hastings, 1970] is one way to construct an MCMC algorithm whose stationary distribution is $p(\vec{\theta} \,|\, y)$. It works according to the following steps.

1. Choose a proposal density $g(\vec{\theta}^* \,|\, \vec{\theta})$.

2. Choose $\vec{\theta}_1$.

3. For $i = 2, 3, \ldots$

   - Generate a proposal $\vec{\theta}^*$ from $g(\vec{\theta}^* \,|\, \vec{\theta}_{i-1})$.
   - Set
   $$r \equiv \min \left\{ 1, \frac{p(\vec{\theta}^* \,|\, y) g(\vec{\theta}_{i-1} \,|\, \vec{\theta}^*)}{p(\vec{\theta}_{i-1} \,|\, y) g(\vec{\theta}^* \,|\, \vec{\theta}_{i-1})} \right\}. \tag{6.9}$$
   - Set
   $$\vec{\theta}_i = \begin{cases} \vec{\theta}^* & \text{with probability } r, \\ \vec{\theta}_{i-1} & \text{with probability } 1 - r. \end{cases}$$

Step 3 define the transition kernel k. In many MCMC chains, the acceptance probability r may be strictly less than one, so the kernel k is a mixture of two parts: one that generates a new value of $\vec{\theta}_{i+1} \neq \vec{\theta}_i$ and one that sets $\vec{\theta}_{i+1} = \vec{\theta}_i$.

To illustrate MCMC, suppose we want to generate a sample $\theta_1, \ldots, \theta_{10{,}000}$ from the Be(5, 2) distribution. We arbitrarily choose a proposal density $g(\theta^* \,|\, \theta) = \mathrm{U}(\theta - .1, \theta + .1)$ and arbitrarily choose $\theta_1 = 0.5$. The following R code draws the sample.

```
samp <- rep ( NA, 10000 )
samp[1] <- 0.5
for ( i in 2:10000 ) {
  prev <- samp[i-1]
  thetastar <- runif ( 1, prev - .1, prev + .1 )
  r <- min ( 1, dbeta(thetastar,5,2) / dbeta(prev,5,2) )
  if ( rbinom ( 1, 1, r ) == 1 )
    new <- thetastar
```

```
    else
      new <- prev
    samp[i] <- new
  }
```

The top panel of Figure 6.5 shows the result. The solid curve is the Be(5, 2) density and the histogram is made from the Metropolis-Hastings samples. They match closely, showing that the algorithm performed well.

Figure 6.5 was produced by

```
par ( mfrow=c(3,1) )
hist ( samp[-(1:1000)], prob=TRUE, xlab=expression(theta),
         ylab="", main="" )
x <- seq(0,1,length=100)
lines ( x, dbeta(x,5,2) )
plot ( samp, pch=".", ylab=expression(theta) )
plot ( dbeta(samp,5,2), pch=".", ylab=expression(p(theta)) )
```

The code `samp[-(1:1000)]` discards the first 1000 draws in the hope that the sampler will have converged to its stationary distribution after 1000 iterations.

Assuming that convergence conditions have been met and that the algorithm is well-constructed, MCMC chains are guaranteed eventually to converge and deliver samples from the desired distribution. But the guarantee is asymptotic and in practice the output from the chain should be checked to diagnose potential problems that might arise in finite samples.

The main thing to check is *mixing*. An MCMC algorithm operates in the space of $\vec{\theta}$. At each iteration of the chain, i.e., for each value of i, there is a current location $\vec{\theta}_i$. At the next iteration the chain moves to a new location $\vec{\theta}_i$. In this way the chain explores the $\vec{\theta}$ space. While it is exploring it also evaluates $p(\vec{\theta}_i)$. In theory, the chain should spend many iterations at values of $\vec{\theta}$ where $p(\vec{\theta})$ is large — and hence deliver many samples of $\vec{\theta}$'s with large posterior density — and few iterations at values where $p(\vec{\theta})$ is small. For the chain to do its job it must find the mode or modes of $p(\vec{\theta})$, it must move around in their vicinity, and it must move between them. The process of moving from one part of the space to another is called *mixing*.

The middle and bottom panels of Figure 6.5 illustrate mixing. The middle panel plots θ_i vs. i. It shows that the chain spends most of its iterations in values of θ between about 0.6

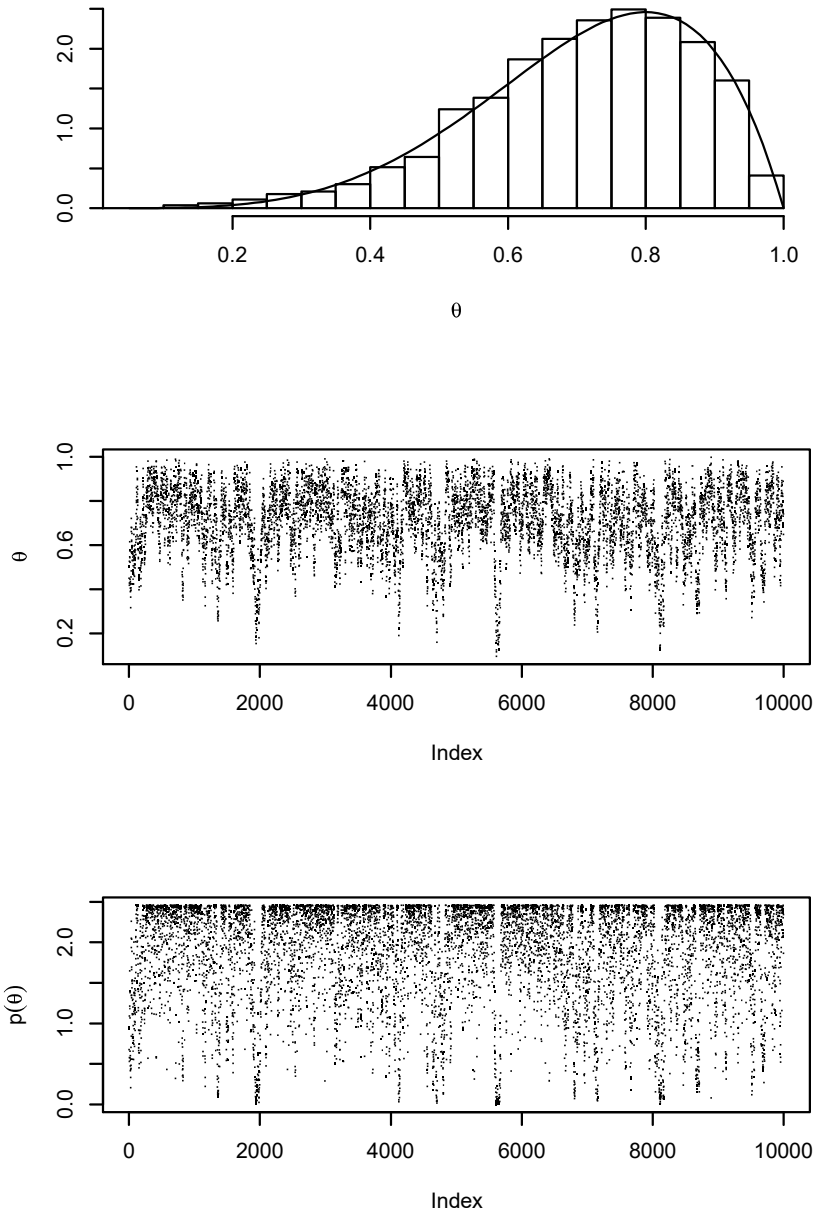

Figure 6.5: 10,000 MCMC samples of the Be(5, 2) density. **Top panel**: histogram of samples from the Metropolis-Hastings algorithm and the Be(5, 2) density. **Middle panel**: θ_i plotted against i. **Bottom panel**: $p(\theta_i)$ plotted against i.

and 0.9 but makes occasional excursions down to 0.4 or 0.2 or so. After each excursion it comes back to the mode around 0.8. The chain has taken many excursions, so it has explored the space well. The bottom panel plots $p(\theta_i)$ vs. i. It shows that the chain spent most of its time near the mode where $p(\theta) \approx 2.4$ but made multiple excursions down to places where $p(\theta)$ is around 0.5, or even less. This chain mixed well.

To illustrate poor mixing we'll use the same MCMC algorithm but with different proposal kernels. First we'll use $(\theta^* \mid \theta) = \mathrm{U}(\theta - 100, \theta + 100)$ and change the corresponding line of code to
`thetastar <- runif ( 1, prev - 100, prev + 100 )`. Then we'll use $(\theta^* \mid \theta) = \mathrm{U}(\theta - .00001, \theta + .00001)$ and change the corresponding line of code to
`thetastar <- runif ( 1, prev - .00001, prev + .00001 )`. Figure 6.6 shows the result. The left-hand side of the figure is for $(\theta^* \mid \theta) = \mathrm{U}(\theta - 100, \theta + 100)$. The top panel shows a very much rougher histogram than Figure 6.5; the middle and bottom panels show why. The proposal radius is so large that most proposals are rejected; therefore, $\theta_{i+1} = \theta_i$ for many iterations; therefore we get the flat spots in the middle and bottom panels. The plots reveal that the sampler explored fewer than 30 separate values of θ. That's too few; the sampler has not mixed well. In contrast, the right-hand side of the figure — for $(\theta^* \mid \theta) = \mathrm{U}(\theta - .00001, \theta + .00001)$ — shows that θ has drifted steadily downward, but over a very small range. There are no flat spots, so the sampler is accepting most proposals, but the proposal radius is so small that the sampler hasn't yet explored most of the space. It too has not mixed well.

Plots such as the middle and bottom plots of Figure 6.6 are called *trace* plots because they trace the path of the sampler.

In this problem, good mixing depends on getting the proposal radius not too large and not too small, but just right (Hassall [1909]). To be sure, if we run the MCMC chain long enough, all three samplers would yield good samples from $\mathrm{Be}(5, 2)$. But the first sampler mixed well with only 10,000 iterations while the others would require many more iterations to yield a good sample. In practice, one must examine the output of one's MCMC chain to diagnose mixing problems. No diagnostics are fool proof, but not diagnosing is foolhardy.

Several special cases of the Metropolis-Hastings algorithm deserve separate mention.

Metropolis algorithm It is often convenient to choose the proposal density $g(\vec{\theta}^* \mid \vec{\theta})$ to be symmetric; i.e., so that $g(\vec{\theta}^* \mid \vec{\theta}) = g(\vec{\theta} \mid \vec{\theta}^*)$. In this case the Metropolis ratio $p(\vec{\theta}^* \mid y)g(\vec{\theta}_{i-1} \mid \vec{\theta}^*)/p(\vec{\theta}_{i-1} \mid y)g(\vec{\theta}^* \mid \vec{\theta}_{i-1})$ simplifies to $p(\vec{\theta}^* \mid y)/p(\vec{\theta}_{i-1} \mid y)$. That's what happened in the $\mathrm{Be}(5, 2)$ illustration and why the line
`r <- min ( 1, dbeta(thetastar,5,2) / dbeta(prev,5,2) )` doesn't involve g.

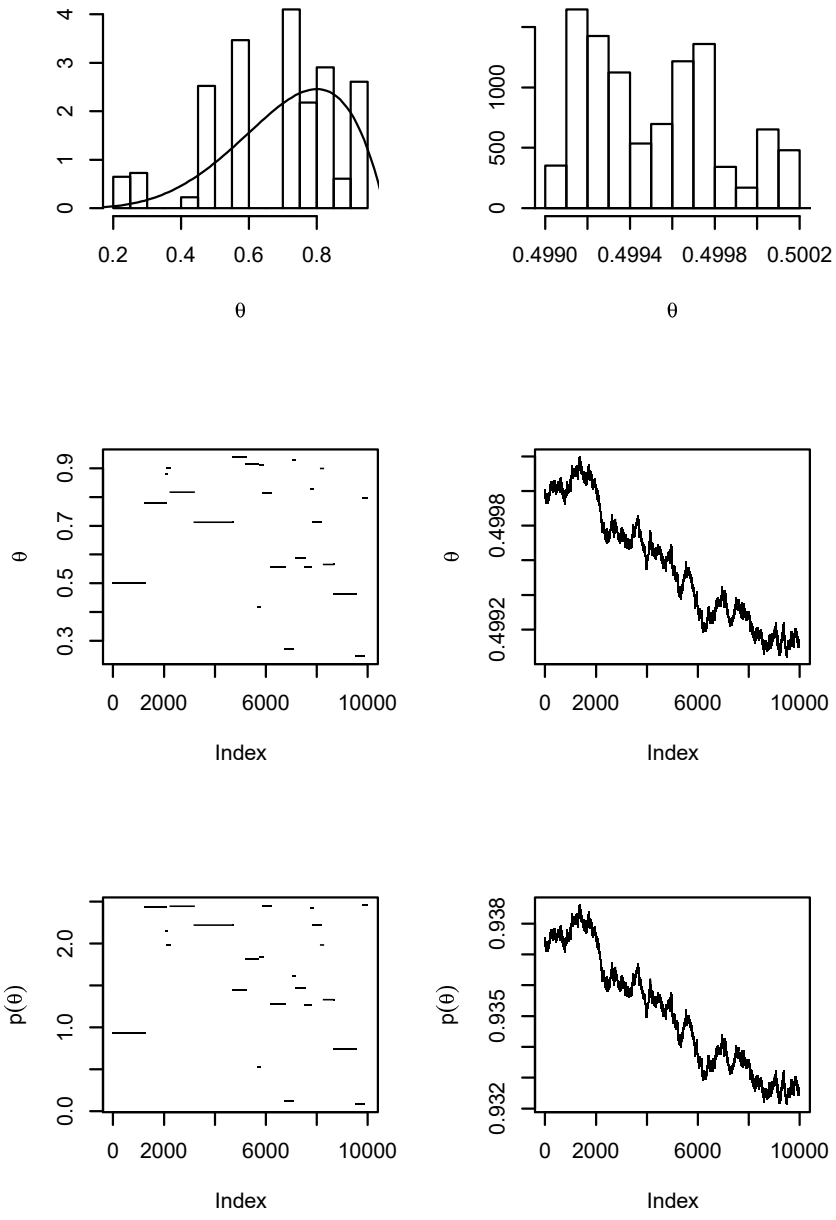

Figure 6.6: 10,000 MCMC samples of the Be(5, 2) density. **Left column**: $(\theta^* \mid \theta) = \mathrm{U}(\theta - 100, \theta + 100)$; **Right column**: $(\theta^* \mid \theta) = \mathrm{U}(\theta - .00001, \theta + .00001)$. **Top**: histogram of samples from the Metropolis-Hastings algorithm and the Be(5, 2) density. **Middle**: θ_i plotted against i. **Bottom**: $p(\theta_i)$ plotted against i.

Independence sampler It may be convenient to choose $g(\vec{\theta}^* \mid \vec{\theta}) = g(\vec{\theta}^*)$ not dependent on $\vec{\theta}$. For example, we could have used `thetastar <- runif(1)` in the Be(5, 2) illustration.

Multiple transition kernels We may construct multiple transition kernels, say $g_1, \ldots, g_m$. Then for each iteration of the MCMC chain we can randomly choose $j \in \{1, \ldots, m\}$ and make a proposal according to g_j. We would do this either for convenience or to improve the convergence rate and mixing properties of the chain.

Gibbs sampler [Geman and Geman, 1984] In many practical examples, the so-called *full conditionals* or *complete conditionals* $p(\theta_j \mid \vec{\theta}_{(-j)}, y)$ are known and easy to sample for all j, where $\theta_{(-j)} = (\theta_1, \ldots, \theta_{j-1}, \theta_{j+1}, \ldots, \theta_k)$. In this case we may sample $\theta_{i,j}$ from
$p(\theta_j \mid \theta_{i,1}, \ldots, \theta_{i,j-1}, \theta_{i-1,j+1}, \ldots, \theta_{i-1,k})$ for $j = 1, \ldots, k$ and set $\vec{\theta}_i = (\theta_{i,1}, \ldots, \theta_{i,k})$. We would do this for convenience.

The next example illustrates several MCMC algorithms on the pine cone data of Example 6.2.

Example 6.3 (Pine Cones, cont)
In this example we try several MCMC algorithms to evaluate and display the posterior distribution in Equation 6.8. Throughout this example, we shall, for compactness, refer to the posterior density as $p(\vec{\theta})$ instead of $p(\vec{\theta} \mid y_1, \ldots, y_n)$.

First we need functions to return the prior density and the likelihood function.

```
dprior <- function ( params, log=FALSE ) {
  logprior <- (   dunif ( params["b0"], -100, 100, log=TRUE )
                + dunif ( params["b1"], -100, 100, log=TRUE )
                + dunif ( params["b2"], -100, 100, log=TRUE )
                + dunif ( params["g0"], -100, 100, log=TRUE )
                + dunif ( params["g1"], -100, 100, log=TRUE )
                + dunif ( params["g2"], -100, 100, log=TRUE )
              )
  if (log) return (logprior)
  else return (exp(logprior))
}

lik <- function ( params, n.cones=cones$X2000, dbh=cones$dbh,
                  trt=cones$trt, log=FALSE ) {
  zero <- n.cones == 0
```

```
  tmp1 <- params["b0"] + params["b1"] * dbh + params["b2"] * trt
  tmp2 <- params["g0"] + params["g1"] * dbh + params["g2"] * trt
  etmp1 <- exp(tmp1)
  etmp2 <- exp(tmp2)

  loglik <- (    sum ( tmp1[!zero] )
              - sum ( etmp2[!zero] )
              + sum ( n.cones[!zero] * tmp2[!zero] )
              + sum ( log ( 1 + etmp1[zero] * exp ( -etmp2[zero] ) ) )
              - sum ( log ( 1 + etmp1 ) )
            )
  if (log) return (loglik)
  else return (exp(loglik))
}
```

Now we write a proposal function. This ones makes $(\vec{\theta}^* \mid \vec{\theta}) \sim \mathrm{N}(\vec{\theta}, .1\mathbf{I}_6)$, where $\mathbf{I}_6$ is the 6×6 identity matrix.

```
g.all <- function ( params ) {
  sig <- c(.1,.1,.1,.1,.1,.1)
  proposed <- mvrnorm ( 1, mu=params, Sigma=diag(sig) )
  return ( list ( proposed=proposed, ratio=1 ) )
}
```

Finally we write the main part of the code. Try to understand it; you may have to write something similar. Notice an interesting feature of `R`: assigning names to the components of `params` allows us to refer to the components by name in the `lik` function.

```
# initial values
params <- c ( "b0"=0, "b1"=0, "b2"=0, "g0"=0, "g1"=0, "g2"=0 )

# number of iterations
mc <- 10000

# storage for output
mcmc.out <- matrix ( NA, mc, length(params)+1 )
```

```
# the main loop
for ( i in 1:mc ) {
  prop <- g.all ( params )
  new <- prop$proposed
  log.accept.ratio <- (   dprior ( new, log=TRUE )
                        - dprior ( params, log=TRUE )
                        + lik ( new, log=TRUE )
                        - lik ( params, log=TRUE )
                        - log ( prop$ratio )
                      )
  accept.ratio <- min ( 1, exp(log.accept.ratio) )

  if ( as.logical ( rbinom(1,1,accept.ratio) ) )
    params <- new

  mcmc.out[i,] <- c ( params, lik ( params, log=TRUE ) )
}
```

Figure 6.7 shows trace plots of the output. The plots show that the sampler did not move very often; it did not mix well and did not explore the space effectively.

Figure 6.7 was produced by the following snippet.

```
par ( mfrow=c(4,2), mar=c(4,4,1,1)+.1 )
for ( i in 1:6 )
  plot ( mcmc.out[,i], ylab=names(params)[i], pch="." )
plot ( mcmc.out[,7], ylab=expression(p(theta)), pch="." )
```

When samplers get stuck, sometimes it's because the proposal radius is too large. So next we try a smaller radius: `sig <- rep(.01,6)`. Figure 6.8 shows the result. The sampler is still not mixing well. The parameter β_0 travelled from its starting point of $\beta_0 = 0$ to about $\beta_0 \approx -1.4$ or so, then seemed to get stuck; other parameters behaved similarly. Let's try running the chain for more iterations: `mc <- 100000`. Figure 6.9 shows the result. Again, the sampler does not appear to have mixed well. Parameters β_0 and β_1, for example, have not yet settled into any sort of steady-state behavior and

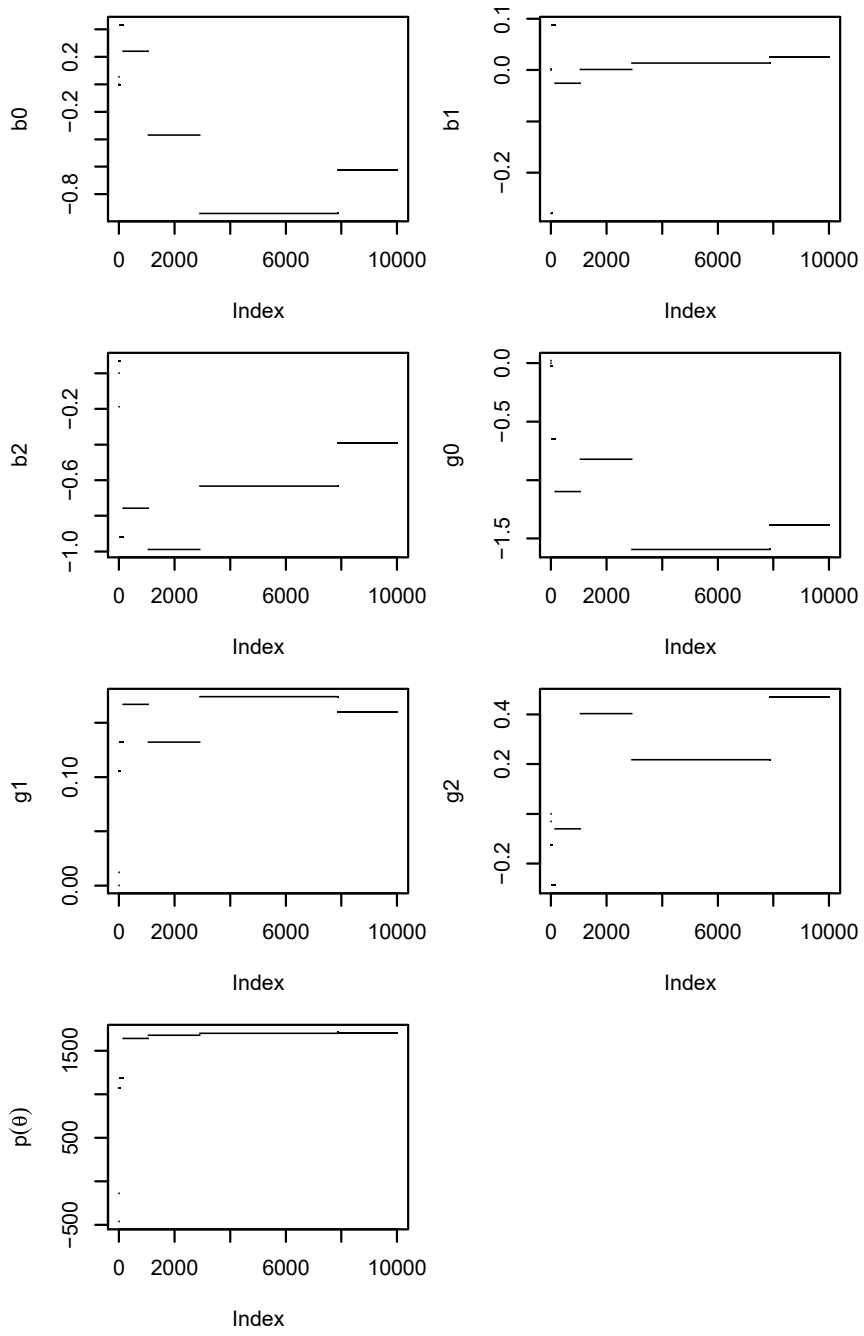

Figure 6.7: Trace plots of MCMC output from the pine cone code on page 365.

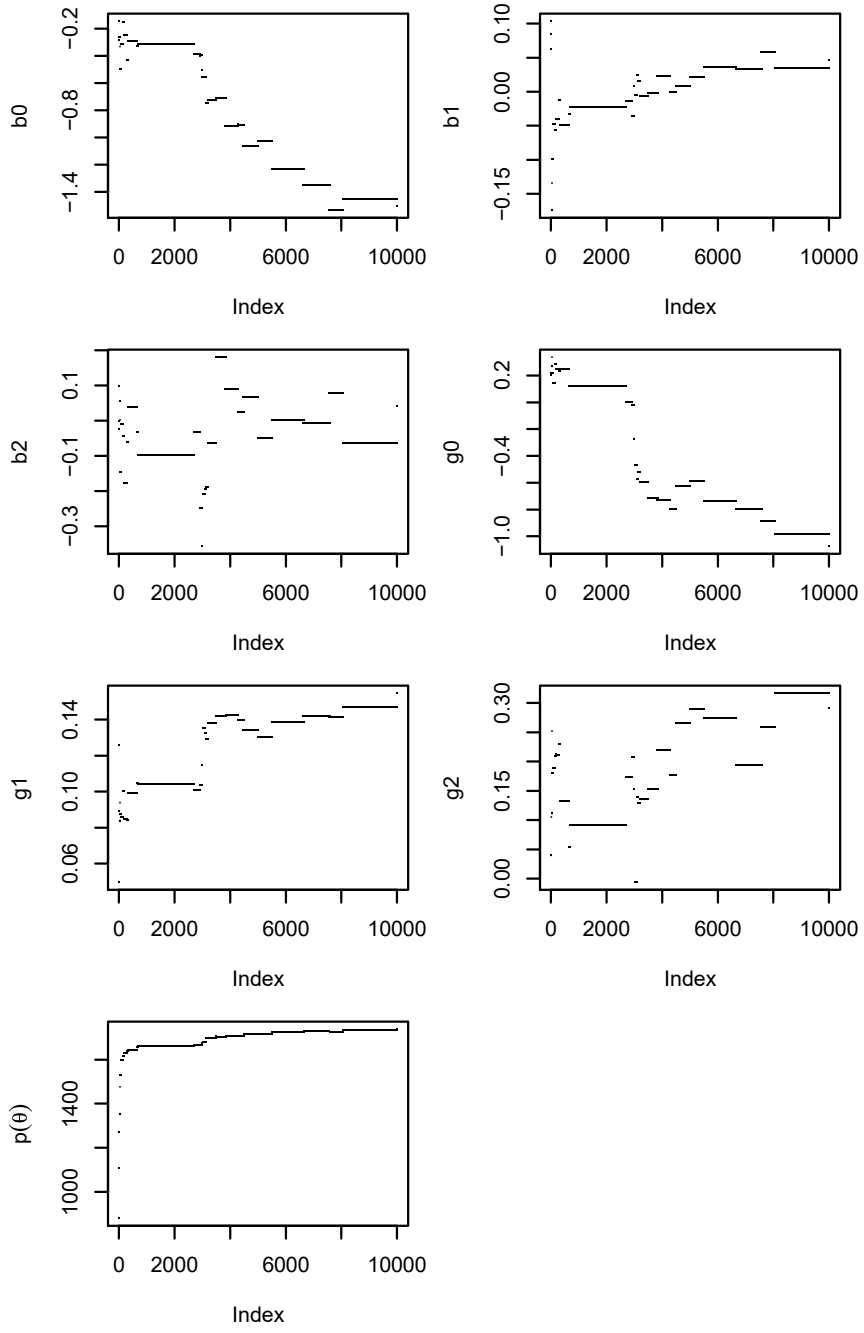

Figure 6.8: Trace plots of MCMC output from the pine cone code with a smaller proposal radius.

$p(\vec{\theta})$ seems to be steadily increasing, indicating that the sampler may not yet have found the posterior mode.

It is not always necessary to plot every iteration of an MCMC sampler. Figure 6.9 plots every 10'th iteration; plots of every iteration look similar. The figure was produced by the following snippet.

```
par ( mfrow=c(4,2), mar=c(4,4,1,1)+.1 )
plotem <- seq ( 1, 100000, by=10 )
for ( i in 1:6 )
  plot ( mcmc.out[plotem,i], ylab=names(params)[i], pch="." )
plot ( mcmc.out[plotem,7], ylab=expression(p(theta)), pch="." )
```

The sampler isn't mixing well. To write a better one we should try to understand why this one is failing. It could be that proposing a change in all parameters simultaneously is too dramatic, that once the sampler reaches a location where $p(\vec{\theta})$ is large, changing all the parameters at once is likely to result in a location where $p(\vec{\theta})$ is small, therefore the acceptance ratio will be small, and the proposal will likely be rejected. To ameliorate the problem we'll try proposing a change to only one parameter at a time. The new proposal function is

```
g.one <- function ( params ) {
  sig <- c ( "b0"=.1, "b1"=.1, "b2"=.1, "g0"=.1, "g1"=.1, "g2"=.1 )
  which <- sample ( names(params), 1 )
  proposed <- params
  proposed[which] <- rnorm ( 1, mean=params[which], sd=sig[which] )
  return ( list ( proposed=proposed, ratio=1 ) )
}
```

which randomly chooses one of the six parameters and proposes to update that parameter only. Naturally, we edit the main loop to use `g.one` instead of `g.all`. Figure 6.10 shows the result. This is starting to look better. Parameters β_2 and γ_2 are exhibiting steady-state behavior; so are β_0 and β_1, after iteration 10,000 or so ($x = 1000$ in the plots). Still, γ_0 and γ_1 do not look like they have converged.

Figure 6.11 illuminates some of the problems. In particular, β_0 and β_1 seem to be linearly related, as do γ_0 and γ_1. This is often the case in regression problems; and we

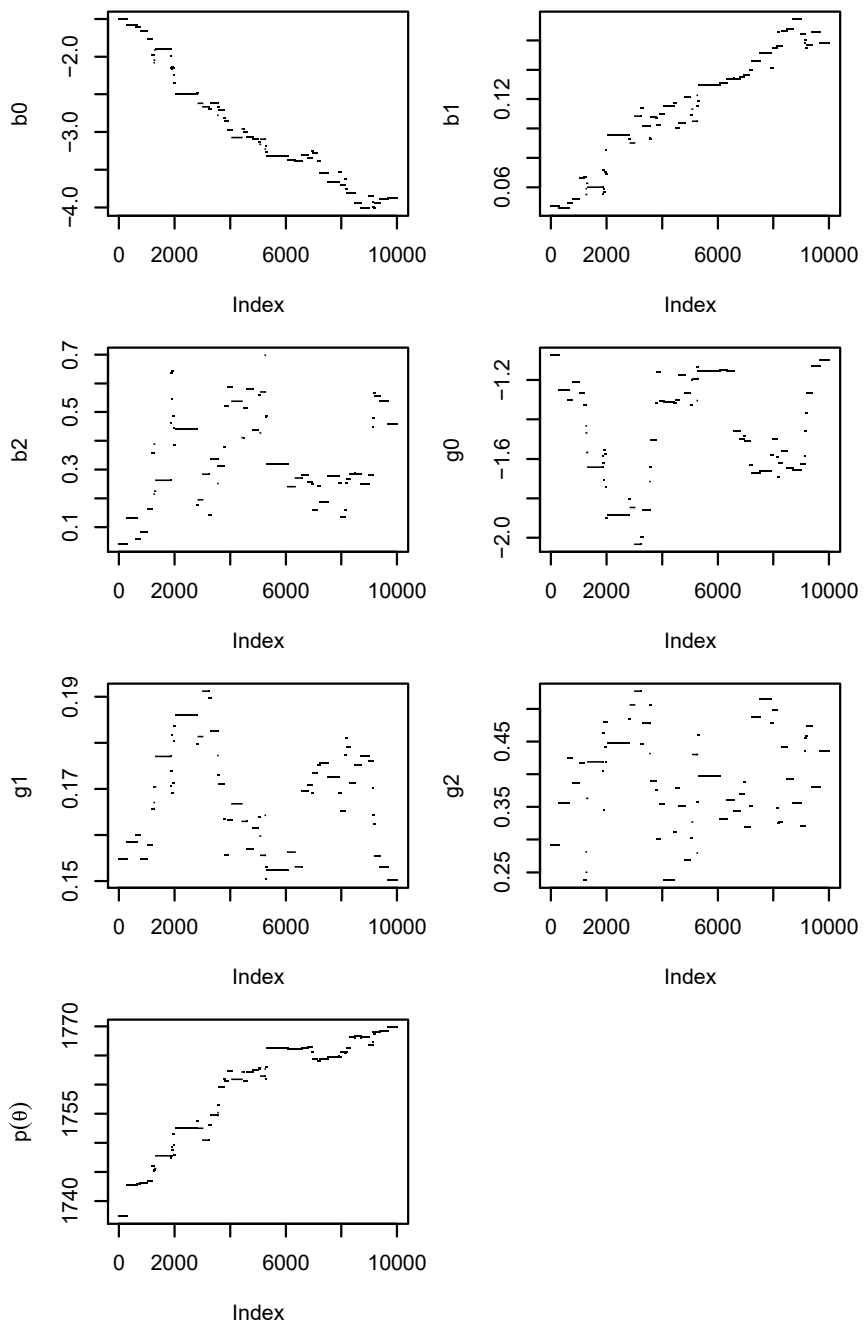

Figure 6.9: Trace plots of MCMC output from the pine cone code with a smaller proposal radius and 100,000 iterations. The plots show every 10'th iteration.

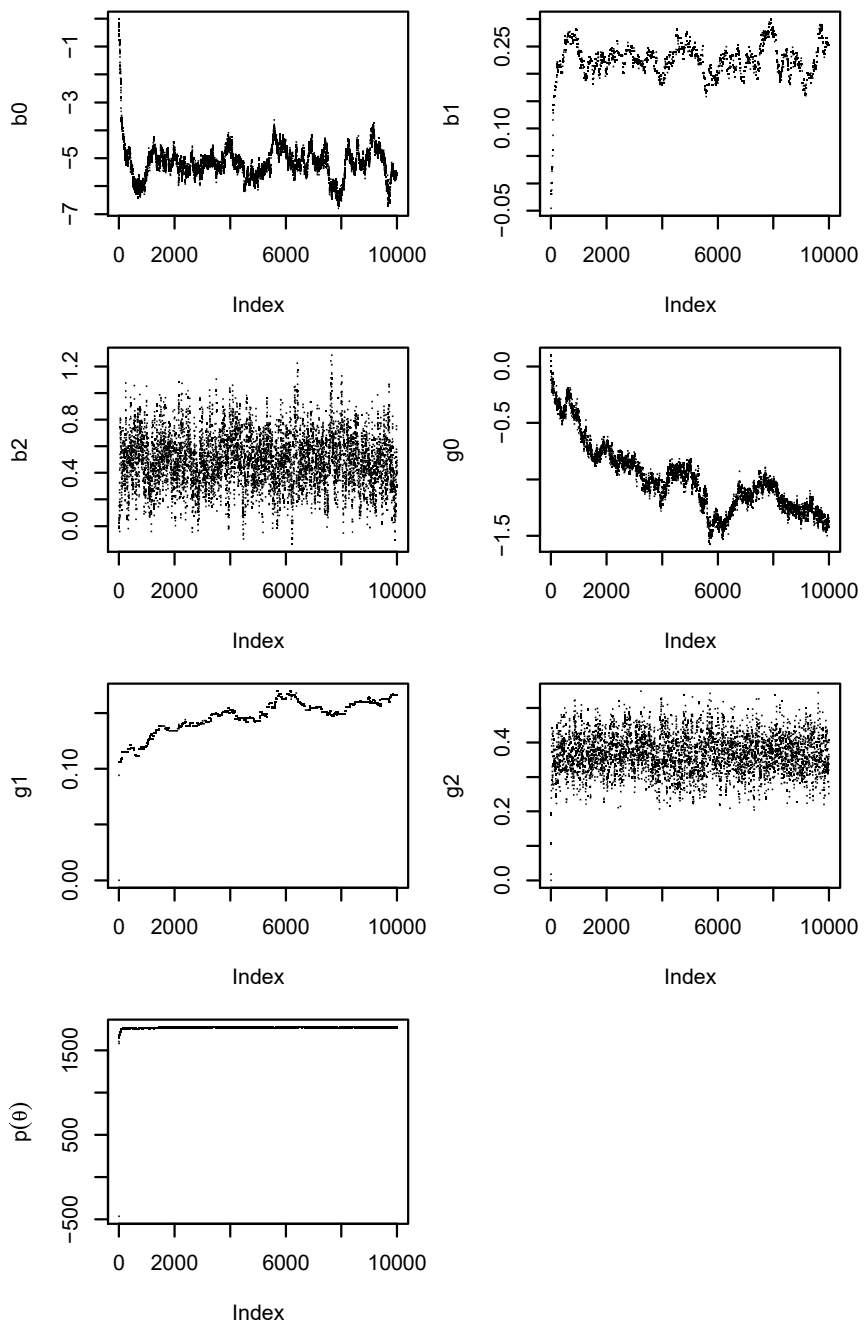

Figure 6.10: Trace plots of MCMC output from the pine cone code with proposal function `g.one` and 100,000 iterations. The plots show every 10'th iteration.

have seen it before for the pine cones in Figure 3.15. In the current setting it means that $p(\vec{\theta}|\, y_1, \ldots, y_n)$ has ridges: one along a line in the (β_0, β_1) plane and another along a line in the (γ_0, γ_1) plane.

Figure 6.11 was produced by the following snippet.

```
plotem <- seq ( 10000, 100000, by=10 )
pairs ( mcmc.out[plotem,], pch=".",
        labels=c(names(params),"density") )
```

As Figure 6.10 shows, it took the first 10,000 iterations or so for β_0 and β_1 to reach a roughly steady state and for $p(\vec{\theta})$ to climb to a reasonably large value. If those iterations were included in Figure 6.11, the points after iteration 10,000 would be squashed together in a small region. Therefore we made `plotem <- seq ( 10,000, 100000, by=10 )` to drop the first 9999 iterations from the plots.

If our MCMC algorithm proposes a move along the ridge, the proposal is likely to be accepted. But if the algorithm proposes a move that takes us off the ridge, the proposal is likely to be rejected because p would be small and therefore the acceptance ratio would be small. But that's not happening here: our MCMC algorithm seems not to be stuck, so we surmise that it is proposing moves that are small compared to the widths of the ridges. However, because the proposals are small, the chain does not explore the space quickly. That's why γ_0 and γ_1 appear not to have reached a steady state. We could improve the algorithm by proposing moves that are roughly parallel to the ridges. And we can do that by making multivariate Normal proposals with a covariance matrix that approximates the posterior covariance of the parameters. We'll do that by finding the covariance of the samples we've generated and using it as the covariance matrix of our proposal distribution. The R code is

```
Sig <- cov ( mcmc.out[10000:100000,-7])
g.group <- function ( params ) {
  proposed <- mvrnorm ( 1, mu=params, Sigma=Sig )
  return ( list ( proposed=proposed, ratio=1 ) )
}
```

We drop the first 9999 iterations because they seem not to reflect $p(\vec{\theta})$ accurately. Then we calculate the covariance matrix of the samples from the previous MCMC sampler.

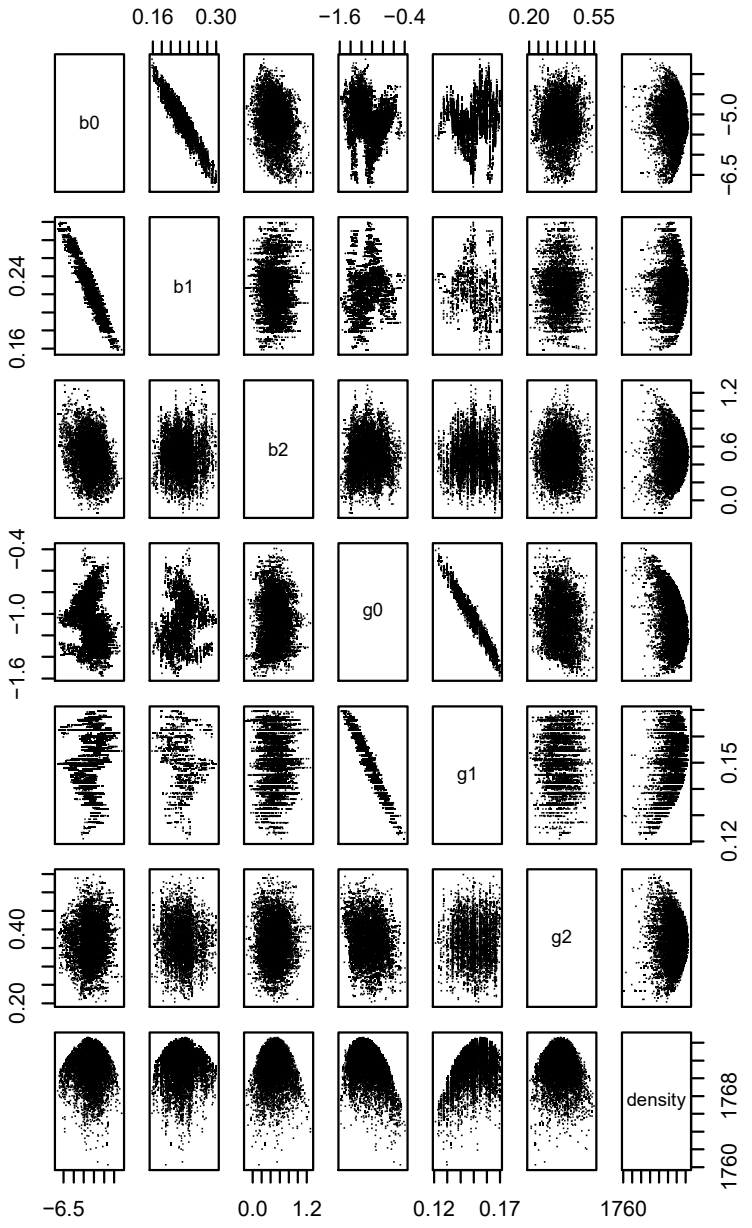

Figure 6.11: Pairs plots of MCMC output from the pine cones example.

That covariance matrix is used in the proposal function. The results are shown in Figures 6.12 and 6.13. Figure 6.12 shows that the sampler seems to have converged after the first several thousand iterations. The posterior density has risen to a high level and is hovering there; all six variables appear to be mixing well. Figure 6.13 confirms our earlier impression that the posterior density seems to be approxiately Normal — at least, it has Normal-looking two dimensional marginals — with β_0 and β_1 highly correlated with each other, γ_1 and γ_2 highly correlated with each other, and no other large correlations. The sampler seems to have found one mode and to be exploring it well.

Figures 6.12 and 6.13 were produced with the following snippet.

```
plotem <- seq ( 1, 100000, by=10 )
par ( mfrow=c(4,2), mar=c(4,4,1,1)+.1 )
for ( i in 1:6 )
  plot ( mcmc.out[plotem,i], ylab=names(params)[i], pch="." )
plot ( mcmc.out[plotem,7], ylab=expression(p(theta)), pch="." )

plotem <- seq ( 1000, 100000, by=10 )
pairs ( mcmc.out[plotem,], pch=".",
        labels=c(names(params),"density") )
```

Now that we have a good set of samples from the posterior, we can use it to answer substantive questions. For instance, we might want to know whether the extra atmospheric CO_2 has allowed pine trees to reach sexual maturity at an earlier age or to produce more pine cones. This is a question of whether β_2 and γ_2 are positive, negative, or approximately zero. Figure 6.14 shows the answer by plotting the posterior densities of β_2 and γ_2. Both densities put almost all their mass on positive values, indicating that $\mathrm{P}[\beta_2 > 0]$ and $\mathrm{P}[\gamma_2 > 0]$ are both very large, and therefore that pines trees with excess CO_2 mature earlier and produce more cones than pine trees grown under normal conditions.

Figure 6.14 was produced by the following snippet.

```
par ( mfrow=c(1,2) )
plot ( density ( mcmc.out[10000:100000,"b2"] ),
```

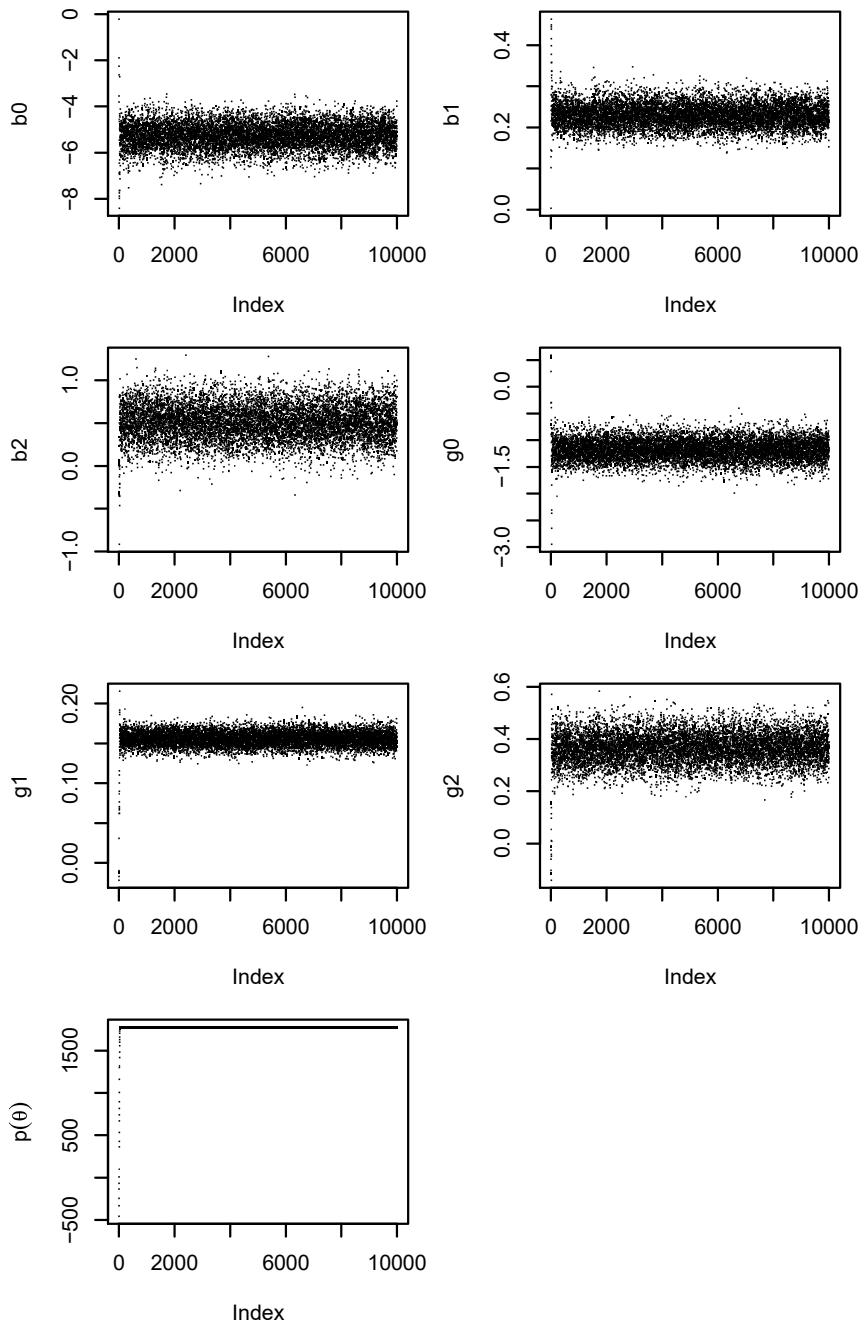

Figure 6.12: Trace plots of MCMC output from the pine cone code with proposal function `g.group` and 100,000 iterations. The plots show every 10'th iteration.

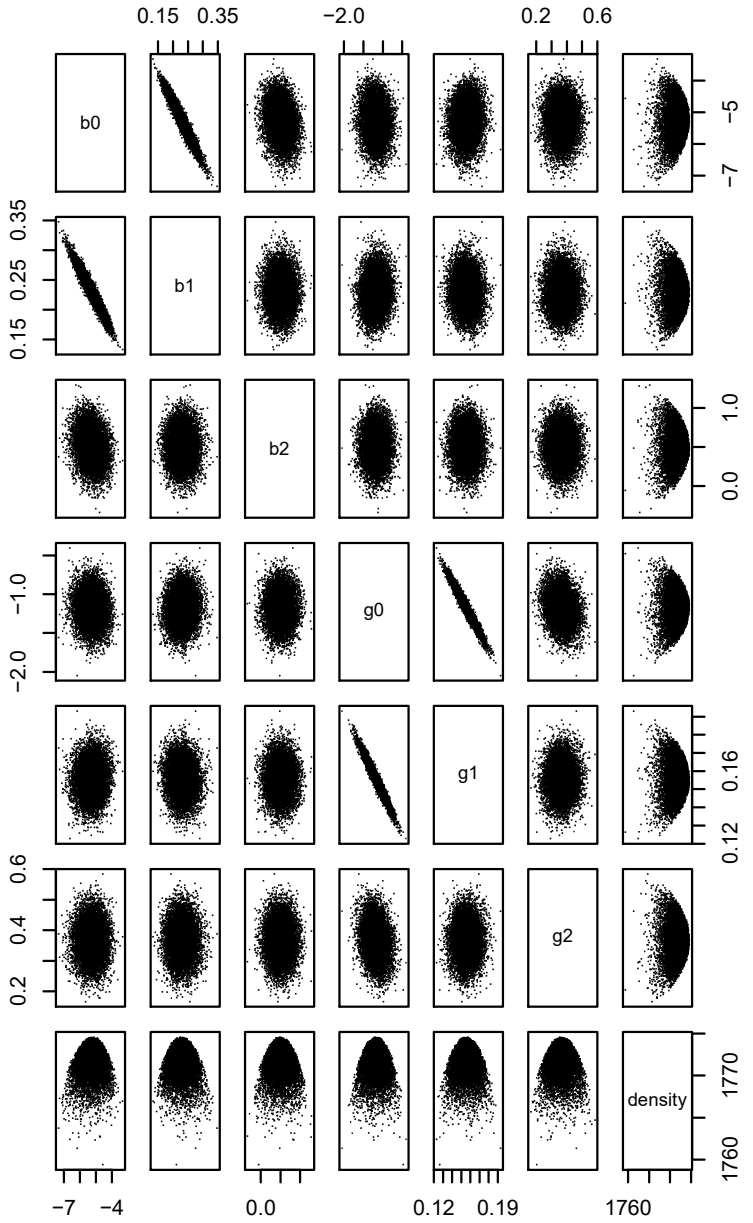

Figure 6.13: Pairs plots of MCMC output from the pine cones example with proposal `g.group`.

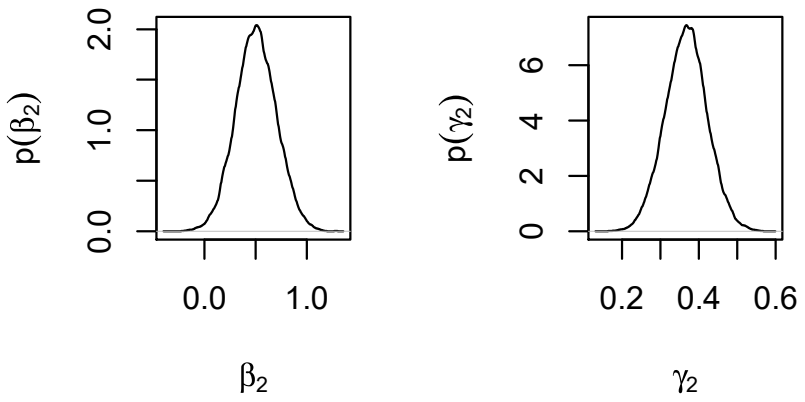

Figure 6.14: Posterior density of β_2 and γ_2 from Example 6.3.

```
       xlab=expression(beta[2]),
       ylab=expression(p(beta[2])), main="" )
plot ( density ( mcmc.out[10000:100000,"g2"] ),
       xlab=expression(gamma[2]),
       ylab=expression(p(gamma[2])), main="" )
```

6.3 Exercises

1. This exercise follows from Example 6.1.

   (a) Find the posterior density for C_{50}, the expected amount of ice cream consumed when the temperature is 50 degrees, by writing C_{50} as a linear function of (β_0, β_1) and using the posterior from Example 6.1.

   (b) Find the posterior density for C_{50} by reparameterizing. Instead of working with parameters (β_0, β_1), work with parameters (C_{50}, β_1). Write the equation for Y_i as a linear function of (C_{50}, β_1) and find the new X matrix. Use that new matrix and a convenient prior to calculate the posterior density of (C_{50}, β_1).