20

Bayesian Data Analysis and MCMC

20.1 Introduction

Bayesian statistics is based up a philosophy different from that of other methods of statistical inference. In Bayesian statistics all unknowns, and in particular unknown parameters, are considered to be random variables and their probability distributions specify our beliefs about their likely values. Estimation, model selection, and uncertainty analysis are implemented by using Bayes's theorem to update our beliefs as new data are observed.

Non-Bayesians distinguish between two types of unknowns, parameters and latent variables. To a non-Bayesian, parameters are fixed quantities without probability distributions while latent variables are random unknowns with probability distributions. For example, to a non-Bayesian, the mean μ, the moving average coefficients $\theta_1, \ldots, \theta_q$, and the white noise variance σ_ϵ^2 of an MA(q) process are fixed parameters while the unobserved white noise process itself consists of latent variables. In contrast, to a Bayesian, the parameters and the white noise process are both unknown random quantities. Since this chapter takes a Bayesian perspective, there is no need to distinguish between the parameters and latent variables, since they can now be treated in the same way. Instead, we will let $\boldsymbol{\theta}$ denote the vector of all unknowns and call it the "parameter vector." In the context of time series forecasting, for example, $\boldsymbol{\theta}$ could include both the unobserved white noise and the future values of the series being forecast.

A hallmark of Bayesian statistics is that one *must* start by specifying prior beliefs about the values of the parameters. Many statisticians have been reluctant to use Bayesian analysis since the need to start with prior beliefs seems too subjective. Consequently, there have been heated debates between Bayesian and non-Bayesian statisticians over the philosophical basis of statistics. However, much of mainstream statistical thought now supports the more pragmatic notion that we should use whatever works satisfactorily.

D. Ruppert, D.S. Matteson, *Statistics and Data Analysis for Financial Engineering*, Springer Texts in Statistics,
DOI 10.1007/978-1-4939-2614-5_20

If one has little prior knowledge about a parameter, this lack of knowledge can be accommodated by using a so-called noninformative prior that provides very little information about the parameter relative to the information supplied by the data. In practice, Bayesian and non-Bayesian analyses of data usually arrive at similar conclusions when the Bayesian analysis uses only weak prior information so that knowledge of the parameters comes predominately from the data.

Moreover, in finance and many other areas of application, analysts often have substantial prior information and are willing to use it. In business and finance, there is no imperative to strive for objectivity as there is in scientific study. The need to specify a prior can be viewed as a strength, not a weakness, of the Bayesian view of statistics, since it forces the analyst to think carefully about how much and what kind of prior knowledge is available.

There has been a tremendous increase in the use of Bayesian statistics over the past few decades, because the Bayesian philosophy is becoming more widely accepted and because Bayesian estimators have become much easier to compute. In fact, Bayesian techniques often are the most satisfactory way to compute estimates for complex models. We have heard one researcher say "I am not a Bayesian but I use Bayesian methods" and undoubtedly others would agree.

For an overview of this chapter, assume we are interested in a parameter vector $\boldsymbol{\theta}$. A Bayesian analysis starts with a *prior* probability distribution for $\boldsymbol{\theta}$ that summarizes all prior knowledge about $\boldsymbol{\theta}$; "prior" means before the data are observed. The likelihood is defined in the same way in a non-Bayesian analysis, but in Bayesian statistics the likelihood has a different interpretation—the likelihood is the conditional distribution of the data given $\boldsymbol{\theta}$. The key step in Bayesian inference is the use of Bayes's theorem to combine the prior knowledge about $\boldsymbol{\theta}$ with the information in the data. This is done by computing the conditional distribution of $\boldsymbol{\theta}$ given the data. This distribution is called the *posterior distribution*. In many, if not most, practical problems, it is impossible to compute the posterior analytically and numerical methods are used instead. A very successful class of numerical Bayesian methods is Markov chain Monte Carlo (MCMC), which simulates a Markov chain in such a way that the stationary distribution of the chain is the posterior distribution of the parameters. The simulated data from the chain are used to compute Bayes estimates and perform uncertainty analysis.

20.2 Bayes's Theorem

Bayes's theorem applies to both discrete events and to continuously distributed random variables. We will start with the case of discrete events. The continuous case is covered in Sect. 20.3.

Suppose that $B_1, \ldots, B_K$ is a partition of the sample space $\mathcal{S}$ (the set of all possible outcomes). By "partition" is meant that $B_i \cap B_j = \emptyset$ if $i \neq j$ and $B_1 \cup B_2 \cup \cdots \cup B_K = \mathcal{S}$. For any set A, we have that

$$A = (A \cap B_1) \cup \cdots \cup (A \cap B_K),$$

and therefore, since $B_1, \ldots, B_K$ are disjoint,

$$P(A) = P(A \cap B_1) + \cdots + P(A \cap B_K). \tag{20.1}$$

It follows from (20.1) and the definition of conditional probability that

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{P(A)} = \frac{P(A|B_j)P(B_j)}{P(A|B_1)P(B_1) + \cdots + P(A|B_K)P(B_K)}. \tag{20.2}$$

Equation (20.2) is called *Bayes's theorem*, and is also known as Bayes's rule or Bayes's law. Bayes's theorem is a simple, almost trivial, mathematical result, but its implications are profound. The importance of Bayes's theorem comes from its use for updating probabilities. Here is an example, one that is far too simple to be realistic but that illustrates how Bayes's theorem can be applied.

Example 20.1. Bayes's theorem in a discrete case

Suppose that our prior knowledge about a stock indicates that the probability θ that the price will rise on any given day is either 0.4 or 0.6. Based upon past data, say from similar stocks, we believe that θ is equally likely to be 0.4 or 0.6. Thus, we have the *prior* probabilities

$$P(\theta = 0.4) = 0.5 \quad \text{and} \quad P(\theta = 0.6) = 0.5.$$

We observe the stock for five consecutive days and its price rises on all five days. Assume that the price changes are independent across days, so that the probability that the price rises on each of five consecutive days is θ^5. Given this information, we may suspect that θ is 0.6, not 0.4. Therefore, the probability that θ is 0.6, given five consecutive price increases, should be greater than the prior probability of 0.5, but how much greater? As notation, let A be the event that the prices rises on five consecutive days. Then, using Bayes's theorem, we have

$$\begin{aligned} P(\theta = 0.6|A) &= \frac{P(A|\theta = 0.6)P(\theta = 0.6)}{P(A|\theta = 0.6)P(\theta = 0.6) + P(A|\theta = 0.4)P(\theta = 0.4)} \\ &= \frac{(0.6)^5(0.5)}{(0.6)^5(0.5) + (0.4)^5(0.5)} \\ &= \frac{(0.6)^5}{(0.6)^5 + (0.4)^5} = \frac{0.07776}{0.07776 + 0.01024} = 0.8836. \end{aligned}$$

Thus, our probability that θ is 0.6 was 0.5 before we observed five consecutive price increases but is 0.8836 after observing this event. Probabilities before observing data are called the *prior probabilities* and the probabilities conditional on observed data are called the *posterior probabilities*, so the prior probability that θ equals 0.6 is 0.5 and the posterior probability is 0.8836. □

Bayes's theorem is extremely important because it tells us exactly how to update our beliefs in light of new information. Revising beliefs after receiving additional information is something that humans do poorly without the help of mathematics.[1] There is a human tendency to put either too little or too much emphasis on new information, but this problem can be mitigated by using Bayes's theorem for guidance.

20.3 Prior and Posterior Distributions

We now assume that $\boldsymbol{\theta}$ is a continuously distributed parameter vector. The *prior distribution* with density $\pi(\boldsymbol{\theta})$ expresses our beliefs about $\boldsymbol{\theta}$ prior to observing data. The likelihood function is interpreted as the conditional density of the data $\boldsymbol{Y}$ given $\boldsymbol{\theta}$ and written as $f(\boldsymbol{y}|\boldsymbol{\theta})$. Using Eq. (A.19), the joint density of $\boldsymbol{\theta}$ and $\boldsymbol{Y}$ is the product of the prior and the likelihood; that is,

$$f(\boldsymbol{y}, \boldsymbol{\theta}) = \pi(\boldsymbol{\theta}) f(\boldsymbol{y}|\boldsymbol{\theta}). \tag{20.3}$$

The marginal density of $\boldsymbol{Y}$ is found by integrating $\boldsymbol{\theta}$ out of the joint density so that

$$f(\boldsymbol{y}) = \int \pi(\boldsymbol{\theta}) f(\boldsymbol{y}|\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{20.4}$$

and the conditional density of $\boldsymbol{\theta}$ given $\boldsymbol{Y}$ is

$$\pi(\boldsymbol{\theta}|\boldsymbol{Y}) = \frac{\pi(\boldsymbol{\theta}) f(\boldsymbol{Y}|\boldsymbol{\theta})}{f(\boldsymbol{Y})} = \frac{\pi(\boldsymbol{\theta}) f(\boldsymbol{Y}|\boldsymbol{\theta})}{\int \pi(\boldsymbol{\theta}) f(\boldsymbol{Y}|\boldsymbol{\theta}) d\boldsymbol{\theta}}. \tag{20.5}$$

Equation (20.5) is another form of Bayes's theorem. The density on the left-hand side of (20.5) is called the *posterior density* and gives the probability distribution of $\boldsymbol{\theta}$ after observing the data $\boldsymbol{Y}$.

Notice our use of π to denote densities of $\boldsymbol{\theta}$, so that $\pi(\boldsymbol{\theta})$ is the prior density and $\pi(\boldsymbol{\theta}|\boldsymbol{Y})$ is the posterior density. In contrast, f is used to denote densities of the data, so that $f(\boldsymbol{y})$ is the marginal density of the data and $f(\boldsymbol{y}|\boldsymbol{\theta})$ is the conditional density given $\boldsymbol{\theta}$.

Bayesian estimation and uncertainty analysis are based upon the posterior. The most common Bayes estimators are the mode and the mean of the

[1] See Edwards (1982) .

posterior density. The mode is called the *maximum a posteriori estimator*, or *MAP estimator*. The mean of the posterior is

$$E(\boldsymbol{\theta}|\boldsymbol{Y}) = \int \boldsymbol{\theta}\, \pi(\boldsymbol{\theta}|\boldsymbol{Y}) d\boldsymbol{\theta} = \frac{\int \boldsymbol{\theta}\, \pi(\boldsymbol{\theta}) f(\boldsymbol{Y}|\boldsymbol{\theta}) d\boldsymbol{\theta}}{\int \pi(\boldsymbol{\theta}) f(\boldsymbol{Y}|\boldsymbol{\theta}) d\boldsymbol{\theta}} \tag{20.6}$$

and is also called the posterior expectation.

Example 20.2. Updating the prior beliefs about the probability that a stock price will increase

We continue Example 20.1 but change the simple, but unrealistic, prior that said that θ was either 0.4 or 0.6 to a more plausible prior where θ could be any value in the interval [0, 1], but with values near 1/2 more likely. Specifically, we use a Beta(2,2) prior so that

$$\pi(\theta) = 6\theta(1-\theta), \quad 0 < \theta < 1.$$

Let Y be the number of times the stock price increases on five consecutive days. Then Y is Binomial(n, θ) and the density of Y is

$$f(y|\theta) = \binom{5}{y} \theta^y (1-\theta)^{5-y}, \quad y = 0, 1, \ldots, 5.$$

Since we observed that $Y = 5$, $f(Y|\theta) = f(5|\theta) = \theta^5$ and the posterior density is

$$\pi(\theta|5) = \frac{6\,\theta(1-\theta)\theta^5}{\int 6\,\theta(1-\theta)\theta^5 d\theta} = 56\,\theta^6(1-\theta),$$

which is a Beta(7,2) density.

The prior and posterior densities are shown in Fig. 20.1. The posterior density is shifted towards the right compared to the prior because five consecutive days saw increased prices. The 0.05 lower and upper quantiles of the posterior distribution are 0.529 and 0.953, respectively, and are shown on the plot. Thus, there is 90 % posterior probability that θ is between 0.529 and 0.953. For this reason, the interval [0.529, 0.953] is called a 90 % *posterior interval* and provides us with the set of likely values of θ. Posterior intervals are Bayesian analogs of confidence intervals and are discussed further in Sect. 20.6. Posterior intervals are also called *credible intervals*.

The posterior expectation is

$$\int_0^1 \theta\, \pi(\theta|5) d\theta = \int_0^1 56\,\theta^7(1-\theta) d\theta = \frac{56}{72} = 0.778. \tag{20.7}$$

The MAP estimate is $6/7 = 0.857$ and its location is shown by a dotted vertical line in Fig. 20.1.

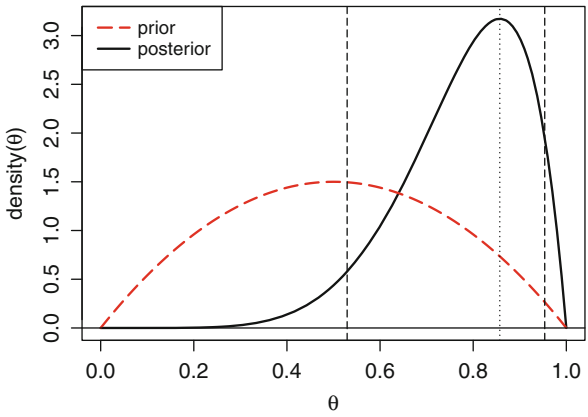

Fig. 20.1. *Prior and posterior densities in Example 20.2. The dashed vertical lines are at the lower and upper* 0.05*-quantiles of the posterior, so they mark off a* 90 *% equal-tailed posterior interval. The dotted vertical line shows the location of the posterior mode at* $\theta = 6/7 = 0.857$.

The posterior CDF is

$$\int_0^\theta \pi(x|5)dx = \int_0^\theta 56x^6(1-x)dx = 56\left(\frac{\theta^7}{7} - \frac{\theta^8}{8}\right), \quad 0 \le \theta \le 1.$$

□

20.4 Conjugate Priors

In Example 20.2, the prior and the posterior were both beta distributions. This is an example of a family of conjugate priors. A family of distributions is called a *conjugate prior family* for a statistical model (or, equivalently, for the likelihood) if the posterior is in this family whenever the prior is in the family. Conjugate families are convenient because they make calculation of the posterior straightforward. All one needs to do is to update the parameters in the prior. To see how this is done, we will generalize Example 20.2.

Example 20.3. Computing the posterior density of the probability that a stock price will increase—general case of a conjugate prior

Suppose now that the prior for θ is Beta(α, β) so that the prior density is

$$\pi(\theta) = K_1\theta^{\alpha-1}(1-\theta)^{\beta-1}, \tag{20.8}$$

where K_1 is a constant. As we will see, knowing the exact value of K_1 is not important, but from (A.14) we know that $K_1 = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$. The parameters in

a prior density must be known, so here α and β are chosen by the data analyst in accordance with the prior knowledge about the value of θ. The choice of these parameters will be discussed later.

Suppose that the stock price is observed on n days and increases on Y days (and does not increase on $n - Y$ days). Then the likelihood is

$$f(Y|\theta) = K_2\theta^Y(1-\theta)^{n-Y}, \tag{20.9}$$

where $K_2 = \binom{n}{Y}$ is another constant. The joint density of θ and Y is

$$\pi(\theta)f(Y|\theta) = K_3\theta^{\alpha+Y-1}(1-\theta)^{\beta+n-Y-1}, \tag{20.10}$$

where $K_3 = K_1K_2$. Then, the posterior density is

$$\pi(\theta|Y) = \frac{\pi(\theta)f(Y|\theta)}{\int_0^1 \pi(\theta)f(Y|\theta)d\theta} = K_4\theta^{\alpha+Y-1}(1-\theta)^{\beta+n-Y-1}. \tag{20.11}$$

where

$$K_4 = \frac{1}{\int_0^1 \theta^{\alpha+Y-1}(1-\theta)^{\beta+n-Y-1}d\theta}. \tag{20.12}$$

The posterior distribution is Beta$(\alpha + Y, \beta + n - Y)$.

We did not need to keep track of the values of $K_1, \ldots, K_4$. Since (20.11) is proportional to a Beta$(\alpha + Y, \beta + n - Y)$ density and since all densities integrate to 1, we can deduce that the constant of proportionality is 1 and the posterior is Beta$(\alpha + Y, \beta + n - Y)$. It follows from (A.14) that

$$K_4 = \frac{\Gamma(\alpha+\beta+n)}{\Gamma(\alpha+Y)\Gamma(\beta+n-Y)}.$$

It is worth noticing how easily the posterior can be found. One simply updates the prior parameters α and β to $\alpha + Y$ and $\beta + n - Y$, respectively.

Using the results in Appendix A.9.7 about the mean and variance of beta distributions, the mean of the posterior is

$$E(\theta|Y) = \frac{\alpha+Y}{\alpha+\beta+n} \tag{20.13}$$

and the posterior variance is

$$\begin{aligned}\text{var}(\theta|\boldsymbol{Y}) &= \frac{(\alpha+Y)(\beta+n-Y)}{(\alpha+\beta+n)^2(\alpha+\beta+n+1)} \\ &= \frac{E(\theta|Y)\{1-E(\theta|Y)\}}{(\alpha+\beta+n+1)}.\end{aligned} \tag{20.14}$$

For values of α and β that are small relative to Y and n, $E(\theta|Y)$ is approximately equal to the MLE, which is Y/n. If we had little prior knowledge

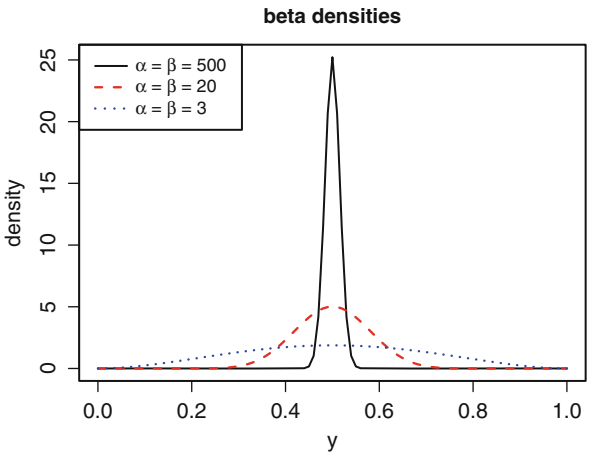

Fig. 20.2. *Examples of beta probability densities with* $\alpha = \beta$.

of θ, we might take both α and β close to 0. However, since θ is the probability of a positive daily return on a stock, we might be reasonably certain that θ is close to 1/2. In that case, choosing $\alpha = \beta$ and both fairly large (so that the prior precision is large) makes sense. One could plot several beta densities with $\alpha = \beta$ and decide which seem reasonable choices of the prior. For example, Fig. 20.2 contains plots of beta densities with $\alpha = \beta = 3$, 20, and 500. When 500 is the common value of α and β, then the prior is quite concentrated about 1/2. This prior could be used by someone who is rather sure that θ is close to 1/2. Someone with less certainty might instead prefer to use $\alpha = \beta = 20$, which has almost all of the prior probability between 0.3 and 0.6. The choice $\alpha = \beta = 3$ leads to a very diffuse prior and would be chosen if one had very little prior knowledge of θ and wanted to "let the data speak for themselves."

The posterior mean in (20.13) has an interesting interpretation. Suppose that we had prior information from a previous sample of size $\alpha+\beta$ and in that sample the stock price increased α times. If we combined the two samples, then the total sample size would be $\alpha+\beta+n$, the number of days with a price increase would be $\alpha+Y$, and the MLE of θ would be $(\alpha+Y)/(\alpha+\beta+n)$, the posterior mean given by (20.13). We can think of the prior as having as much information as would be given by a prior sample of size $\alpha+\beta$ and $\alpha/(\alpha+\beta)$ can be interpreted as the MLE of θ from that sample. Therefore, the three priors in Fig. 20.2 can be viewed as having as much information as samples of sizes 6, 40, and 1000. For a fixed value of $E(\theta|Y)$, we see from (20.14) that the posterior variance of θ becomes smaller as α, β, or n increases; this makes sense since n is the sample size and $\alpha+\beta$ quantifies the amount of information in the prior.

Since it is not necessary to keep track of constants, we could have omitted them from the previous calculations and, for example, written (20.8) as

$$\pi(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}. \tag{20.15}$$

In the following examples, we will omit constants in this manner. □

Example 20.4. Posterior distribution when estimating the mean of a normal population with known variance

Suppose $Y_1, \ldots, Y_n$ are i.i.d. $N(\mu, \sigma^2)$ and σ^2 is known. The unrealistic assumption that σ^2 is known is made so that we can start simple and will be removed later.

The conjugate prior for μ is the family of normal distributions. To show this, assume that the prior on μ is $N(\mu_0, \sigma_0^2)$ for known values of μ_0 and σ_0^2. We learned in Example 20.3 that it is not necessary to keep track of quantities that do not depend on the unknown parameters (but could depend on the data or known parameters), so we will keep track only of terms that depend on μ.

Simple algebra shows that the likelihood is

$$\begin{aligned} f(Y_1, \ldots, Y_n|\mu) &= \prod_{i=1}^{n} \left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{1}{2\sigma^2}(Y_i - \mu)^2 \right\} \right] \\ &\propto \exp\left\{ -\frac{1}{2\sigma^2}\left(-2n\overline{Y}\mu + n\mu^2\right) \right\}. \end{aligned} \tag{20.16}$$

The prior density is

$$\pi(\mu) = \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left\{ -\frac{1}{2\sigma_0^2}(\mu - \mu_0)^2 \right\} \propto \exp\left\{ -\frac{1}{2\sigma_0^2}(-2\mu\mu_0 + \mu^2) \right\}. \tag{20.17}$$

A *precision* is the reciprocal of a variance, and we let $\tau = 1/\sigma^2$ denote the population precision. Multiplying (20.16) and (20.17), we can see that the posterior density is

$$\begin{aligned} \pi(\mu|Y_1, \ldots, Y_n) &\propto \exp\left\{ \left(\frac{n\overline{Y}}{\sigma^2} + \frac{\mu_0}{\sigma_0^2} \right)\mu - \left(\frac{n}{2\sigma^2} + \frac{1}{2\sigma_0^2} \right)\mu^2 \right\} \\ &= \exp\left\{ (\tau_{\overline{Y}}\overline{Y} + \tau_0\mu_0)\mu - \frac{1}{2}(\tau_{\overline{Y}} + \tau_0)\mu^2 \right\}, \end{aligned} \tag{20.18}$$

where $\tau_{\overline{Y}} = n\tau = n/\sigma^2$ and $\tau_0 = 1/\sigma_0^2$, so that $\tau_{\overline{Y}}$ is the precision of $\overline{Y}$ and τ_0 is the precision of the prior distribution.

One can see that $\log\{\pi(\mu|Y_1, \ldots, Y_n)\}$ is a quadratic function of μ, so $\pi(\mu|Y_1, \ldots, Y_n)$ is a normal density. Therefore, to find the posterior distribution we need only compute the posterior mean and variance. The posterior mean is the value of μ that maximizes the posterior density, that is, the posterior mode, so to calculate the posterior mean, we solve

$$0 = \frac{\partial}{\partial \mu} \log\{\pi(\mu|Y_1, \ldots, Y_n)\} \tag{20.19}$$

and find that the mean is

$$E(\mu|Y_1, \ldots, Y_n) = \frac{\tau_{\overline{Y}}\overline{Y} + \tau_0\mu_0}{\tau_{\overline{Y}} + \tau_0} = \frac{\frac{n\overline{Y}}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}}. \tag{20.20}$$

We can see from (A.10) that the precision of a normal density $f(y)$ is -2 times the coefficient of y^2 in $\log\{f(y)\}$. Therefore, the posterior precision is -2 times the coefficient of μ^2 in (20.18). Consequently, the posterior precision is $\tau_{\overline{Y}} + \tau_0 = n/\sigma^2 + 1/\sigma_0^2$, and the posterior variance is

$$\text{Var}(\mu|Y_1 \ldots, Y_n) = \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}}. \tag{20.21}$$

In summary, the posterior distribution is

$$N\left(\frac{\frac{n\overline{Y}}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}}, \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}}\right) = N\left(\frac{\tau_{\overline{Y}}\overline{Y} + \tau_0\mu_0}{\tau_{\overline{Y}} + \tau_0}, \frac{1}{\tau_{\overline{Y}} + \tau_0}\right). \tag{20.22}$$

We can see that the posterior precision $(\tau_{\overline{Y}} + \tau_0)$ is the sum of the precision of $\overline{Y}$ and the precision of the prior; this makes sense since the posterior combines the information in the data with the information in the prior.

Notice that as $n \to \infty$, the posterior precision $\tau_{\overline{Y}}$ converges to ∞ and the posterior distribution is approximately

$$N(\overline{Y}, \sigma^2/n). \tag{20.23}$$

What this result tells us is that as the amount of data increases, the effect of the prior becomes negligible. The posterior density also converges to (20.23) as $\sigma_0 \to \infty$ with n fixed, that is, as the prior becomes negligible because the prior precision decreases to zero.

A common Bayes estimator is the posterior mean given by the right-hand side of (20.20). Many statisticians are neither committed Bayesians nor committed non-Bayesians and like to look at estimators from both perspectives. A non-Bayesian would analyze the posterior mean by examining its bias, variance, and mean-squared error. We will see that, in general, the Bayes estimator is biased but is less variable than $\overline{Y}$, and the tradeoff between bias and variance is controlled by the choice of the prior.

To simplify notation, let $\widehat{\mu}$ denote the posterior mean. Then

$$\widehat{\mu} = \delta\overline{Y} + (1 - \delta)\mu_0, \tag{20.24}$$

where $\delta = \tau_{\overline{Y}}/(\tau_{\overline{Y}} + \tau_0)$, and $E(\widehat{\mu}|\mu) = \delta\mu + (1 - \delta)\mu_0$, so the bias of $\widehat{\mu}$ is $\{E(\widehat{\mu}|\mu) - \mu\} = (\delta - 1)(\mu - \mu_0)$ and $\widehat{\mu}$ is biased unless $\delta = 1$ or $\mu_0 = \mu$.

We will have $\delta = 1$ only in the limit as the prior precision τ_0 converges to 0 and $\mu_0 = \mu$ means that the prior mean is exactly equal to the true parameter, but of course this beneficial situation cannot be arranged since μ is not known.

The variance of $\widehat{\mu}$ is

$$\text{Var}(\widehat{\mu}|\mu) = \frac{\delta^2\sigma^2}{n},$$

which is less than $\text{Var}(\overline{Y}) = \sigma^2/n$, except in the extreme case where $\delta = 1$. We see that smaller values of δ lead to more bias but smaller variance. The best bias–variance tradeoff minimizes the mean square error of $\widehat{\mu}$, which is

$$\text{MSE}(\widehat{\mu}) = \text{BIAS}^2(\widehat{\mu}) + \text{Var}(\widehat{\mu}) = (\delta - 1)^2(\mu - \mu_0)^2 + \frac{\delta^2\sigma^2}{n}. \tag{20.25}$$

It is best, of course, to have $\mu_0 = \mu$, but this is not possible since μ is unknown. What is known is $\delta = \tau_{\overline{Y}}/(\tau_{\overline{Y}} + \tau_0)$ and δ can be controlled by the choice of τ_0.

Figure 20.3 shows the MSE as a function of $\delta \in (0, 1)$ for three values of $\mu - \mu_0$, which is called the "prior bias" since it is the difference between the true value of the parameter and the prior mean. In this figure $\sigma^2/n = 1/2$. For each of the two larger values of the prior bias, there is a range of values of δ where the Bayes estimator has a smaller MSE than $\overline{Y}$, but if δ is below this range, then the Bayes estimator has a larger MSE than $\overline{Y}$ and the range of "good" δ-values decreases as the prior bias increases. If the prior bias is large and δ is too small, then the MSE of the Bayes estimator can be quite large since it converges to the squared prior bias as $\delta \to 0$; see (20.25) or Fig. 20.3. This result shows the need either to have a good prior guess of μ or to keep the prior precision small so that δ is large. However, when δ is large, then the Bayes estimator cannot improve much over $\overline{Y}$ and, in fact, converges to $\overline{Y}$ as $\delta \to 1$.

In summary, it can be challenging to choose a prior that offers a substantial improvement over $\overline{Y}$. One way to do this is to combine several related

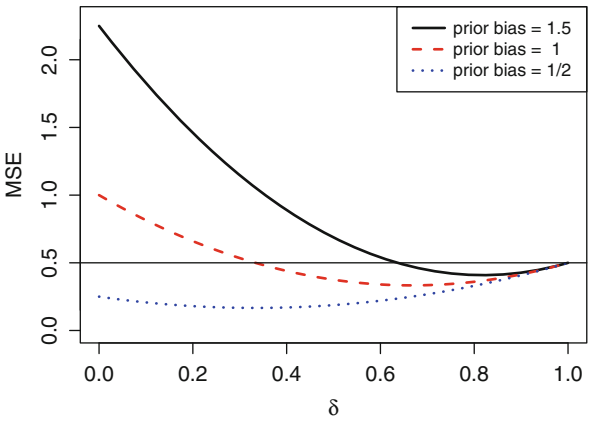

Fig. 20.3. *MSE versus δ for three values of "prior bias" $= \mu - \mu_0$ when $\sigma^2/n = 1/2$. The horizontal line represents the MSE of the maximum likelihood estimator ($\overline{Y}$).*

estimation problems using a hierarchical prior; see Sect. 20.8. When it is not possible to combine related problems and there is no other way to get information about μ, then the prudent data analyst will forgo the attempt to improve upon the MLE and instead will choose a small value for the prior precision τ_0. □

Example 20.5. Posterior distribution when estimating a normal precision

Now suppose that $Y_1, \ldots, Y_n$ are i.i.d. with a known mean μ and an unknown variance σ^2 and precision $\tau = 1/\sigma^2$. We will show that the conjugate priors for τ are the gamma distributions and we will find the posterior distribution of τ. Define $s^2 = n^{-1}\sum_{i=1}^{n}(Y_i - \mu)^2$, which is the MLE of σ^2.

Simple algebra shows that the likelihood is

$$f(Y_1, \ldots, Y_n|\tau) \propto \exp\left(-\frac{1}{2}n\tau s^2\right)\tau^{n/2}. \tag{20.26}$$

Let the prior distribution be the gamma distribution with shape parameter α and scale parameter b which has density

$$\pi(\tau) = \frac{\tau^{\alpha-1}}{\Gamma(\alpha)b^{\alpha}}\exp(-\tau/b) \propto \tau^{\alpha-1}\exp(-\tau/b). \tag{20.27}$$

Multiplying (20.26) and (20.27), we see that the posterior density for τ is

$$\pi(\tau|Y_1, \ldots, Y_n) \propto \tau^{n/2+\alpha-1}\exp\{-(ns^2/2 + b^{-1})\tau\}, \tag{20.28}$$

which shows that the posterior distribution is gamma with shape parameter $n/2 + \alpha$ and scale parameter $\left(ns^2/2 + b^{-1}\right)^{-1}$; that is,

$$\pi(\tau|Y_1, \ldots, Y_n) = \text{Gamma}\left\{n/2 + \alpha, \left(ns^2/2 + b^{-1}\right)^{-1}\right\}. \tag{20.29}$$

We have shown that gamma distributions are conjugate for a normal precision parameter.

The expected value of a gamma distribution is the product of the shape and scale parameters, so the posterior mean of τ is

$$E(\tau|Y_1, \ldots, Y_n) = \frac{\frac{n}{2} + \alpha}{\frac{ns^2}{2} + b^{-1}}.$$

Notice that $E(\tau|Y_1, \ldots, Y_n)$ converges to s^{-2} as $n \to \infty$, which is not surprising since the MLE of σ^2 is s^2, so that the MLE of τ is s^{-2}. □

20.5 Central Limit Theorem for the Posterior

For large sample sizes, the posterior distribution obeys a central limit theorem that can be roughly stated as follows:

Result 20.6. *Under suitable assumptions and for large enough sample sizes, the posterior distribution of* $\boldsymbol{\theta}$ *is approximately normal with mean equal to the true value of* $\boldsymbol{\theta}$ *and with variance equal to the inverse of the Fisher information matrix.*

This result is also known as the *Bernstein–von Mises Theorem*. See Sect. 20.14 for references to a precise statement of the theorem.

This theorem is an important result for several reasons. First, a comparison with Result 5.1 shows that the Bayes estimator and the MLE have the same large-sample distributions. In particular, we see that for large sample sizes, the effect of the prior becomes negligible, because the asymptotic distribution does not depend on the prior. Moreover, the theorem shows a connection between confidence and posterior intervals that is discussed in the next section.

One of the assumptions of this theorem is that the prior remains fixed as the sample size increases, so that eventually nearly all of the information comes from the data. The more informative the prior, the larger the sample size needed for the posterior distribution to approach its asymptotic limit.

20.6 Posterior Intervals

Bayesian posterior intervals were mentioned in Example 20.2 and will now be discussed in more depth.

Posterior intervals have a different probabilistic interpretation than confidence intervals. The theory of confidence intervals views the parameter as fixed and the interval as random because it is based on a random sample. Thus, when we say "the probability that the confidence interval will include the true parameter is ...," it is the probability distribution of the interval, not the parameter, that is being considered. Moreover, the probability expresses the likelihood *before* the data are collected about what will happen after the data are collected. For example, if we use 95 % confidence, then the probability is 0.95 that we will obtain a sample whose interval covers the parameter. After the data have been collected and the interval is known, a non-Bayesian will say that either the interval covers the parameter or it does not, so the probability that the interval covers the parameter is either 1 or 0, though, of course, we do not know which value is the actual probability.

In the Bayesian theory of posterior intervals, the opposite is true. The sample is considered fixed since we use posterior probabilities, that is, probabilities conditional on the data. Therefore, the posterior interval is considered a fixed quantity. But in Bayesian statistics, parameters are treated as random. Therefore, when a Bayesian says "the probability that the posterior interval will include the true parameter is ...," the probability distribution being considered is the posterior distribution of the parameter. The random quantity is the parameter, the interval is fixed, and the probability is after the data have been collected.

Despite these substantial philosophical differences between confidence and posterior intervals, in many examples where both a confidence interval and a posterior interval have been constructed, one finds that they are nearly equal. This is especially common when the prior is relatively noninformative compared to the data, for example, in Example 20.3 if $\alpha + \beta$ is much smaller than n.

There are solid theoretical reasons based on central limit theorems why confidence and posterior intervals are nearly equal for large sample sizes. By Result 20.6 (the central limit theorem for the posterior), a large-sample posterior interval for the ith component of $\boldsymbol{\theta}$ is

$$E(\theta_i|\boldsymbol{Y}) \pm z_{\alpha/2}\sqrt{\text{var}(\theta_i|\boldsymbol{Y})}. \tag{20.30}$$

By Results 5.1 and 7.6 (the univariate and multivariate central limit theorems for the MLE), the large-sample confidence interval (5.20) based on the MLE and the large-sample posterior interval (20.30) will approach each other as the sample size increases. Therefore, practically-minded non-Bayesian data analysts are often happy to use a posterior interval and interpret it as a large-sample approximation to a confidence interval. Except in simple problems, all confidence intervals are based on large-sample approximations. This is true for confidence intervals that use profile likelihood, the central limit theorem for the MLE and Fisher information, or the bootstrap, in other words, for all of the major methods for constructing confidence intervals.

There are two major types of posterior intervals, highest probability and equal-tails. Let $\psi = \psi(\boldsymbol{\theta})$ be a scalar function of the parameter vector $\boldsymbol{\theta}$ and let $\pi(\psi|\boldsymbol{Y})$ be the posterior density of ψ. A highest-probability interval is of the form $\{\psi : \pi(\psi|\boldsymbol{Y}) > k\}$ for some constant k. As k increases from 0 to ∞, the posterior probability of this interval decreases from 1 to 0, and k is chosen so that the probability is $1 - \alpha$. If $\pi(\psi|\boldsymbol{Y})$ has multiple modes, then the set $\{\psi : \pi(\psi|\boldsymbol{Y}) > k\}$ might not be an interval and in that case it should be called a posterior set or posterior region rather than a posterior interval. In any case, this region has the interpretation of being the smallest set with $1-\alpha$ posterior probability. When the highest-posterior region is an interval, it can be found by computing all intervals that range from the α_1-lower quantile of $\pi(\psi|\boldsymbol{Y})$ to the α_2-upper quantile of $\pi(\psi|\boldsymbol{Y})$, where $\alpha_1 + \alpha_2 = \alpha$, and the using the shortest of these intervals.

The equal-tails posterior interval has lower and upper limits equal to the lower and upper $\alpha/2$-quantiles of $\pi(\psi|\boldsymbol{Y})$. The two types of intervals coincide when $\pi(\psi|\boldsymbol{Y})$ is symmetric and unimodal, which will be at least approximately true for large samples by the central limit theorem for the posterior.

Posterior intervals of either type are easy to compute when using Monte Carlo methods; see Sect. 20.7.3.

Example 20.7. Posterior interval for a normal mean when the variance is known

This example continues Example 20.4. By (20.20) and (20.21), a $(1-\alpha)$ 100 % posterior interval for μ is

$$\frac{\tau_{\overline{Y}}\overline{Y} + \tau_0\mu_0}{\tau_{\overline{Y}} + \tau_0} \pm z_{\alpha/2}\sqrt{\frac{1}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}}}, \tag{20.31}$$

where $z_{\alpha/2}$ is the $\alpha/2$-upper quantile of the standard normal distribution.

If either $n \to \infty$ or $\sigma_0 \to \infty$, then the information in the prior becomes negligible relative to the information in the data because $\tau_{\overline{Y}}/\tau_0 \to \infty$, and the posterior interval converges to

$$\overline{Y} \pm z_{\alpha/2}\frac{\sigma}{\sqrt{n}},$$

which is the usual non-Bayesian confidence interval. □

20.7 Markov Chain Monte Carlo

Although the Bayesian calculations in the simple examples of the last few sections were straightforward, this is generally not true for problems of practical interest. Frequently, the integral in the denominator of posterior density (20.5) is impossible to calculate analytically. The same is true of the integral in the numerator of the posterior mean given by (20.6). Because of computational difficulties, until approximately 1990 Bayesian data analysis was much less widely used than now. Fortunately, Monte Carlo simulation methods for approximating posterior densities and expectations have been developed. They have been a tremendous advance and not only have they made Bayesian methods practical, but also they have led to the solution of applied problems that heretofore could not be tackled.

The most widely applicable Monte Carlo method for Bayesian analysis simulates a Markov chain whose stationary distribution is the posterior. The sample from this chain is used for Bayesian inference. This technique is called *Markov chain Monte Carlo*, or *MCMC*. The `BUGS` language implements MCMC. There are three widely used versions of BUGS, `OpenBUGS`, `WinBUGS`, and `JAGS`. Most `BUGS` programs will run on all three versions, but there are exceptions. `JAGS` is in one way the most versatile of the three versions since it is the only one that will run under MacOS.[2]

This section is an introduction to MCMC and `BUGS`. First, we discuss Gibbs sampling, the simplest type of MCMC. Gibbs sampling works well when it is applicable, but it is applicable only to limited set of problems. Next, the Metropolis–Hastings algorithm is discussed. Metropolis–Hastings is applicable to nearly every type of Bayesian analysis. `BUGS` is a sophisticated program that is able to select an MCMC algorithm that is suitable for a particular model.

[2] `OpenBUGS` will run on a Mac under WINE.

20.7.1 Gibbs Sampling

Gibbs sampling is the simplest MCMC method. Suppose that the parameter vector $\boldsymbol{\theta}$ can be partitioned into M subvectors so that

$$\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\theta}_M \end{pmatrix}.$$

Let $[\boldsymbol{\theta}_j|\boldsymbol{Y}, \boldsymbol{\theta}_k, k \neq j]$ be the conditional distribution of $\boldsymbol{\theta}_j$ given the data $\boldsymbol{Y}$ and the values of the other subvectors; $[\boldsymbol{\theta}_j|\boldsymbol{Y}, \boldsymbol{\theta}_k, k \neq j]$ is called the *full conditional distribution* of $\boldsymbol{\theta}_j$. Gibbs sampling is feasible if one can sample from each of the full conditionals.

Gibbs sampling creates a Markov chain that repeatedly samples the subvectors $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M$ in the following manner. The chain starts with an arbitrary starting value $\boldsymbol{\theta}^{(0)}$ for the parameter vector $\boldsymbol{\theta}$. Then the subvector $\boldsymbol{\theta}_1^{(1)}$ is sampled from the full conditional $[\boldsymbol{\theta}_1|\boldsymbol{Y}, \boldsymbol{\theta}_k, k \neq 1]$ with each of the remaining subvectors $\boldsymbol{\theta}_k, k \neq 1$, set at its current value which is $\boldsymbol{\theta}_k^{(0)}$. Next $\boldsymbol{\theta}_2^{(1)}$ is sampled from $[\boldsymbol{\theta}_2|\boldsymbol{Y}, \boldsymbol{\theta}_k, k \neq 2]$ with $\boldsymbol{\theta}_k, k \neq 2$, set at its current value, which is $\boldsymbol{\theta}_k^{(1)}$ for $k = 1$ and $\boldsymbol{\theta}_k^{(0)}$ for $k \geq 2$. One continues it this way until each of $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M$ has been updated and one has $\boldsymbol{\theta}^{(1)} = (\boldsymbol{\theta}_1^{(1)}, \ldots, \boldsymbol{\theta}_M^{(1)})^\mathsf{T}$.

Then $\boldsymbol{\theta}^{(2)}$ is found starting at $\boldsymbol{\theta}^{(1)}$ in the same way that $\boldsymbol{\theta}^{(1)}$ was obtained starting at $\boldsymbol{\theta}^{(0)}$. Continuing in this way, we obtain the sequence $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(N)}$ that is a Markov chain with the remarkable property that its stationary distribution is the posterior distribution of $\boldsymbol{\theta}$. Moreover, regardless of the starting value $\boldsymbol{\theta}^{(0)}$, the chain will converge to the stationary distribution. After convergence to the stationary distribution, the Markov chain samples the posterior distribution and the MCMC sample is used to compute posterior expectations, quantiles, and other characteristics of the posterior distribution.

Since the Gibbs sample does not start in the stationary distribution, the first N_0 iterations are discarded as a burn-in period for an appropriately chosen value of N_0. We will assume that this has been done and $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(N)}$ is the sample from the chain after the burn-in period. In Sect. 20.7.5, methods for choosing N_0 are discussed.

Example 20.8. Gibbs sampling for a normal mean and precision

In Example 20.7, we found the posterior for a normal mean when the precision is known, and in Example 20.5, we found the posterior for a normal precision when the mean is known. These two results specify the two full conditionals and allow one to apply Gibbs sampling to the problem of estimating a normal mean and precision when both are unknown. The idea is simple. A starting value $\tau^{(0)}$ for τ is selected. The starting value might be the MLE, for example. However, there are advantages to using multiple chains with random starting values that are *overdispersed*, meaning that their probability distribution is more scattered than that posterior distribution; see Sect. 20.7.5.

Then, treating τ as known and equal to $\tau^{(0)}$, $\mu^{(1)}$ is drawn randomly from its Gaussian full conditional posterior distribution given in (20.22). Note: the starting value $\tau^{(0)}$ for the population precision τ should not be confused with the precision τ_0 in the prior for μ; $\tau^{(0)}$ is used only once, to start the Gibbs sampling algorithm; after burn-in, the Gibbs sample will not depend on the actual value of $\tau^{(0)}$. In contrast, τ_0 is fixed and is part of the posterior so the Gibbs sample should and will depend on τ_0.

After $\mu^{(1)}$ has been sampled, μ is treated as known and equal to $\mu^{(1)}$ and $\tau^{(1)}$ is drawn from the full conditional (20.29). Gibbs sampling continues in this way, alternatively between sampling μ and τ from their full conditionals. □

20.7.2 Other Markov Chain Monte Carlo Samplers

It is often difficult or impossible to sample directly from the full conditionals of the posterior and then Gibbs sampling is infeasible. Fortunately, there is a large variety of other sampling algorithms that can be used when Gibbs sampling cannot be used. Programming Monte Carlo algorithms "from scratch" is beyond the scope of this book but is explained in the references in Sect. 20.14. The `BUGS` language discussed in Sect. 20.7.4 allows analysts to use MCMC without the time-consuming and error-prone process of programming the details.

20.7.3 Analysis of MCMC Output

The analysis of MCMC output typically examines scalar-valued functions of the parameter vector $\boldsymbol{\theta}$. The analysis should be performed on each scalar quantity of interest. Let $\psi = \psi(\boldsymbol{\theta})$ be one such function. Suppose $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$ is an MCMC sample from the posterior distribution of $\boldsymbol{\theta}$, either from a single Markov chain or from combining multiple chains, and define $\psi_i = \psi(\boldsymbol{\theta}_i)$. We will assume that the burn-in period and the chain lengths are sufficient so that $\psi_1, \ldots, \psi_N$ is a representative sample from the posterior distribution of ψ. Methods for diagnosing convergence and adequacy of the Monte Carlo sample size are explained in Sect. 20.7.5.

The MCMC sample mean $\overline{\psi} = N^{-1} \sum_{i=1}^{N} \psi_i$ estimates the posterior expectation $E(\psi|\boldsymbol{Y})$, which is the most common Bayes estimator. The MCMC sample standard deviation $s_\psi = \left\{(N-1)^{-1} \sum_{i=1}^{N} (\psi_i - \overline{\psi})^2\right\}^{1/2}$ estimates the posterior standard deviation of ψ and will be called the *Bayesian standard error*. If the sample size of the data is sufficiently large, then the posterior distribution will be approximately normal by Result 20.6 and an approximate $(1-\alpha)$ posterior interval for ψ is

$$\overline{\psi} \pm z_{\alpha/2}\, s_\psi. \tag{20.32}$$

Interval (20.32) is an MCMC approximation to (20.30).

However, one need not use this normal approximation to find posterior intervals. If $L(\alpha_1)$ is the α_1-lower sample quantile and $U(\alpha_2)$ is the α_2-upper sample quantile of $\psi_1, \ldots, \psi_N$, then $[L(\alpha_1), U(\alpha_2)]$ is a $1-(\alpha_1+\alpha_2)$ posterior interval. For an equal-tailed posterior interval, one uses $\alpha_1 = \alpha_2 = \alpha/2$. For a highest-posterior density interval, one chooses α_1 and α_2 on a fine grid such that $\alpha_1 + \alpha_2 = \alpha$ and $U(\alpha_2) - L(\alpha_1)$ is minimized. One should check that the posterior density of ψ is unimodal using a kernel density estimate. If there are several modes and sufficiently deep troughs between them, then highest-posterior density posterior region could be a union of intervals, not a single interval. However, even in this somewhat unusual case, $[L(\alpha_1), U(\alpha_2)]$ might still be used as the shortest $1-\alpha$ posterior *interval*.

Kernel density estimates can be used to visualize the shapes of the posterior densities. As an example, see Fig. 20.4 discussed in Example 20.9 ahead. Most automatic bandwidth selectors for kernel density estimation are based on the assumption of an independent sample. When applied to MCMC output, they might undersmooth. If the `density()` function in `R` is used, one might correct this undersmoothing by using a value of the `adjust` parameter greater than the default value of 1. However, Fig. 20.4 uses the default value and the amount of smoothing seems adequate; this could be due to the large Monte Carlo sample size, $N = 10{,}000$.

20.7.4 JAGS

`JAGS` is a implementation of the `BUGS` (Bayesian analysis Using Gibbs Sampling) program that can be run from Windows, Mac OS, or Linux. `JAGS` can be used as a standalone program or it can be called from within `R` using the `rjags` package.

Example 20.9. Using `JAGS` *to fit a t-distribution to returns*

In this example, a t-distribution will be fit to S&P 500 returns using `JAGS` called from `R`. The `BUGS` program below is in the file `univt.bug`. The program will run under any of `OpenBUGS`, `WinBUGS`, or `JAGS`. In this example, `JAGS` will be used.

```
model{
for(i in 1:N)
{
   r[i] ~ dt(mu, tau, k)
}
mu ~ dnorm(0.0, 1.0E-6)
tau ~ dgamma(0.1, 0.01)
k ~ dunif(2, 50)
sigma2 <- (k / (k - 2)) / tau
sigma <- sqrt(sigma2)
}
```

In BUGS, dnorm(mu,tau) is the normal distribution with mean equal to mu and precision equal to tau. Also, dt(mu, tau, k) is the t-distribution with mean equal to mu, degrees of freedom equal to k, and inverse scale parameter equal to the square root of tau (so tau is proportional to, rather than equal to, the precision and the constant of proportionality is $\sqrt{(k-2)/k}$). In the BUGS program, the "for loop" (lines 3–5) specifies the likelihood and lines 6–8 specify the priors for mu, tau, and k. Line 9 computes the variance from tau and line 10 computes the standard deviation.

The R program is:

```
1  library(rjags)
2  library("Ecdat")
3  data(SP500)
4  r = SP500$r500
5  N = length(r)
6  data = list(r = r, N = N)
7  inits = function(){list(mu = rnorm(1, mean = mean(r),
8     sd = 2 * sd(r)), tau = runif(1, 0.2/var(r), 2/var(r)),
9     k = runif(1, 2.5, 10))}
10 t1 = proc.time()
11 univt.mcmc <- jags.model("univt.bug", data = data, inits = inits,
12    n.chains = 3, n.adapt = 1000, quiet = FALSE)
13 nthin = 20
14 univt.coda = coda.samples(univt.mcmc, c("mu", "k", "sigma"),
15    100*nthin, thin = nthin)
16 summary(univt.coda, digits = 2)
17 t2 = proc.time()
18 (t2 - t1) / 60
19 pdf("basic_plot.pdf", width = 4, height = 7)  ##  Figure 20.4
20 par(mfrow = c(4, 2))
21 plot(univt.coda, auto.layout = F)   ##  Figure 20.4
22 graphics.off()
23 gelman.diag(univt.coda)
24 effectiveSize(univt.coda)
25 pdf("gelman_plot.pdf", width = 6, height = 6)  ##  Figure 20.6
26 gelman.plot(univt.coda)
27 graphics.off()
28 dic.samples(univt.mcmc, 100*nthin, thin = nthin, type = "pD:)
```

Line 6 creates a data list that is given to JAGS and lines 7–9 creates a function inits() that generates starting values for each chain. The function jags.model() at lines 11–12 creates an object univt.mcmc of class jags containing a graphical model description of the model specified in the file univt.bug. This object is one of the arguments of coda.samples() at lines 14–15. The function coda.samples() produces an object univt.coda of class mcmc.list containing MCMC output. Objects of this class can be used as input to functions in the coda package such as gelman.diag(), effectiveSize(), gelman.plot(), and summary(). Line 18 prints the

computation time in minutes. The computation time for this example was about 6 minutes, but this number is, of course, hardware dependent.

The output from line 16 is:

```
> summary(univt.coda, digits = 2)

Iterations = 3020:5000
Thinning interval = 20
Number of chains = 3
Sample size per chain = 100

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

           Mean        SD  Naive SE Time-series SE
k     6.0451630 0.5443919 3.143e-02      3.137e-02
mu    0.0005129 0.0001850 1.068e-05      1.071e-05
sigma 0.0103017 0.0002078 1.200e-05      1.146e-05

2. Quantiles for each variable:

           2.5%       25%       50%       75%     97.5%
k     5.1407126 5.6780998 6.0380664 6.4039149 7.1849194
mu    0.0001289 0.0003821 0.0005046 0.0006191 0.0008763
sigma 0.0099304 0.0101560 0.0102910 0.0104427 0.0107435
```

Figure 20.4 produced at line 21 contains trace plots and kernel density estimates for `mu`, `k`, and `sigma` arranged alphabetically. The trace plots are simply time series plots of the three chains. The interpretation of trace plots is discussed in Sect. 20.7.5. The Gelman plot produced by line 26 is shown later as Fig. 20.6.

The diagnostics from lines 23–28 will discuss briefly here and described in more detail later. The Gelman diagnostics produced by line 23 are:

```
> gelman.diag(univt.coda)
Potential scale reduction factors:

      Point est. Upper C.I.
k           1.00       1.01
mu          1.00       1.01
sigma       1.02       1.06

Multivariate psrf

1.02
```

The effective sample sizes calculated at line 24 are:

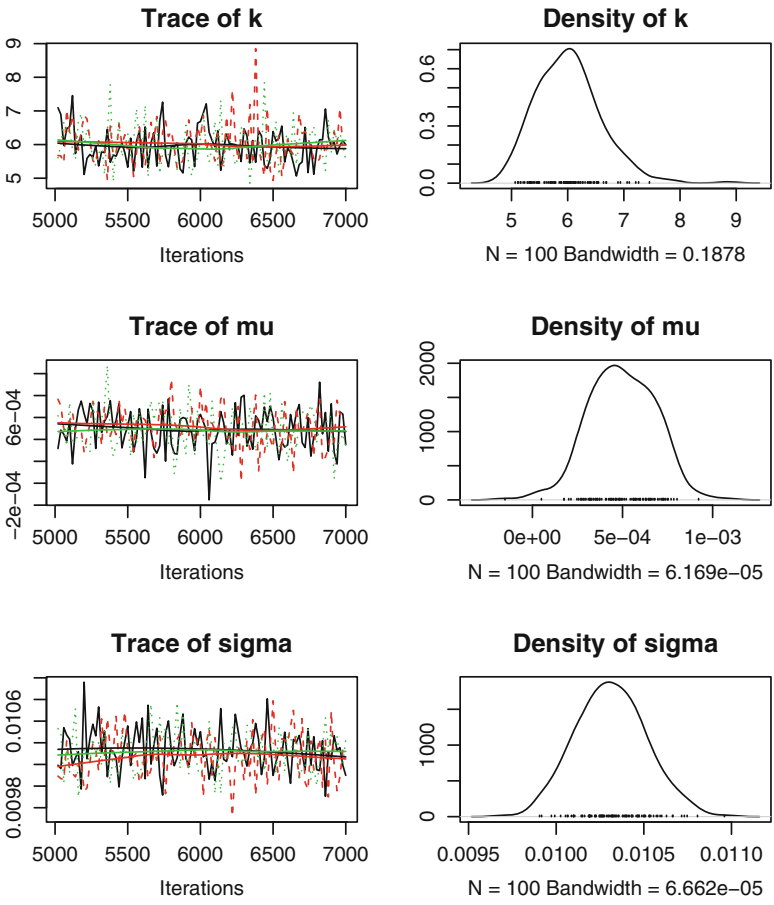

Fig. 20.4. *Trace plots and kernel density estimates in Example 20.9.*

```
> effectiveSize(univt.coda)
       k       mu    sigma
366.8874 300.0000 300.0000
```

Gelman diagnostics and effective sample sizes are discussed soon in Sect. 20.7.5. DIC and p_D, which are discussed in Sect. 20.7.6 and produced at line 28, are DIC $= -18,062$ and $p_D = 2.664$:

```
> dic.samples(univt.mcmc, 100*nthin, thin = nthin, type = "pD")
  |**************************************************| 100%
Mean deviance:  -18065
penalty 2.664
Penalized deviance: -18062
```

□

20.7.5 Monitoring MCMC Convergence and Mixing

The length N_0 of the burn-in period must be sufficiently large that the Markov chain has converged to the stationary distribution by the end of burn-in. The length N of the chain after burn-in must be large enough that moments, quantiles, and other quantities computed from the MCMC sample are accurate estimates of the corresponding characteristics of posterior. Markov chains are dependent sequences and the chains used in MCMC typically have positive autocorrelation. Because of the autocorrelation, to achieve accurate estimates Markov chain samples must be larger, often far larger, than would be necessary with independent sampling. A chain that moves about the posterior slowly is said to mix poorly. The slower the mixing of the chain, the larger the sample size needed for accurate estimation.

In principle, one long Markov chain is all that is needed to sample the posterior. However, if several chains are generated, then one can compare them to decide if the burn-in period N_0 and chain length N are sufficiently large. If the amount of between-chain variation in the chain means is large relative to the within-chain variation, then the chains are mixing poorly. Consequently, diagnostics for convergence and mixing can be based on between- and within-chain variation.

Between-chain variability will be artificially low if the chains have similar starting values. For this reason, it is recommended that the starting values be randomly sampled from a distribution with greater dispersion than the posterior. For example, one might use a Gaussian or t-distribution with mean equal to the MLE and covariance matrix equal to k times the inverse Fisher information for some $k > 1$, e.g., $k = 1.5$ or 2.

Example 20.10. Good mixing and poor mixing

Excellent and poor mixing are contrasted in Fig. 20.5. The model is linear regression with two predictor variables and i.i.d. Gaussian noise. There are two simulated data sets. In the first data set the predictors are highly correlated (sample correlation = 0.996). Trace plots for this data set are in the top row. In the second data set the predictors are independent and the trace plots are in the middle row. Except for this difference in the amount of collinearity, the two data sets have the same distributions. In both of these cases, there are three chains and for each chain there is a burn-in period of $N_0 = 100$ iterations and then 1000 iterations that are retained. In each row, trace plots, are shown for the three regression coefficients (intercept and two slopes) and for the residual precision (inverse variance).

In each case the three chains were started at randomly chosen initial values. The probability distribution was centered at the least-squares and "overdispersed" relative to the posterior distribution. Specifically, the regression coefficients have a Gaussian starting value distribution centered at the least-squares estimate and with covariance matrix 1.5 times the covariance matrix of the least-squares estimator. The noise variance had a starting distribution that

was uniformly distributed between 0.25 and 4 times the least-squares estimate [e.g., $\widehat{\sigma}^2_\epsilon$ in (9.16)] of the noise variance. By using overdispersed starting values, one can discover how quickly the chains move from their starting values to the stationary distribution. The chains for the regression coefficients move very quickly in the middle row but slowly in the top row. The residual precision is unaffected by collinearity and moves quickly even in the high collinearity case.

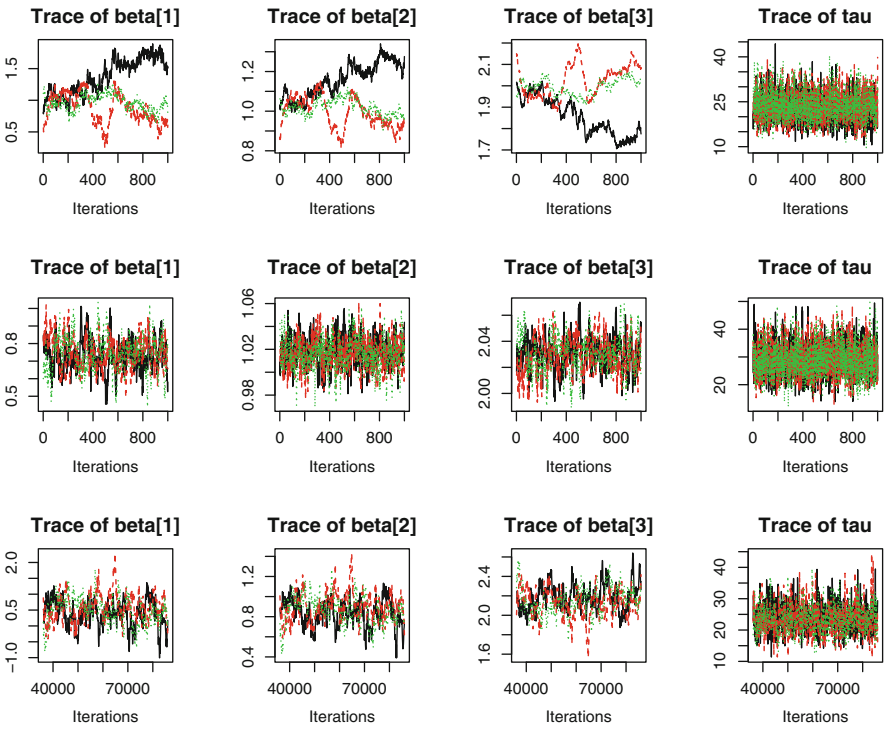

Fig. 20.5. *MCMC analysis of a linear regression model with two predictor variables. Simulated data. Trace plots of the regression coefficients and residual precision for three chains. The burn-in period was* 100 *and the chain lengths are* 1000*. The trace plots contain the MCMC output after the burn-in period. The intercept is beta[1] and the slopes are beta[2] and beta[3].* **Top row:** *The two predictors are highly correlated and the strong collinearity is causing poor mixing of the regression coefficients. Notice that the chains have not converged to the stationary distribution by the start of the sampling period and that the between-chain variation is large.* **Middle row:** *The burn-in period was* 100 *and the chain lengths are* 1000 *as in the top row. The two predictors are independent and there is very good mixing because there is no collinearity. Notice that the chains have converged to the stationary distribution by the start of the sampling period and there is little between-chain variation.* **Bottom row:** *Same data set as the top row but with a burn-in period of* 5000 *and chain lengths of* 30,000*. The chains have been thinned so that only every* 10*th iteration is retained.*

One solution to poor mixing is to increase the burn-in period and the chain lengths. The bottom row uses the same data set as in the top row but with a longer burn-in (5000 iterations) and longer chains (30,000 iterations). The chains have been thinned so that only every 10th iteration is retained. Thinning can speed the analysis of the MCMC output by reducing the Monte Carlo sample size and can improve the appearance of trace plots—a trace plot of 3 chains of 30,000 iterations each would be almost completely filled in. The chains appear to have converged to the stationary distribution by the end of the burn-in and to mix reasonably well over 30,000 iterations (3000 after thinning).

The BUGS code for this model is:

```
model{
for(i in 1:N){
y[i] ~ dnorm(mu[i],tau)
mu[i] <- x[i,1]*beta[1] + x[i,2]*beta[2] +  x[i,3]*beta[3]
}
for(i in 1:3)beta[i] ~ dnorm(0,.00001)
tau ~ dgamma(0.01,0.01)
}
```

The R code is:

```
library(rjags)
library(coda)
library(mvtnorm)
set.seed(90201)
N = 50
beta1 = 1
beta2 = 2
alpha = 1
x1 = rnorm(N, mean = 3, sd = 2)
x2 = x1 + rnorm(N, mean = 3, sd = 0.2)
x = cbind(rep(1, N), x1, x2)
y = alpha + beta1 * x1 + beta2 * x2 + rnorm(N, mean = 0, sd = 0.2)
data = list(y = y, x = x, N = N)
summ = summary(lm(y ~ x1 + x2))
betahat = as.numeric(summ$coeff)[1:3]
covbetahat = summ$sigma^2 * solve(t(x) %*% x)
inits = function(){list(beta = as.numeric(rmvnorm(n = 1,
   mean = betahat, sigma=1.5 * covbetahat)),
   tau=runif(1, 1/(4 * summ$sigma^2), 4 / summ$sigma^2))}
regr <- jags.model("lin_reg_vect.bug", data = data, inits = inits,
   n.chains = 3, n.adapt = 1000, quiet = FALSE)
regr.coda = coda.samples(regr, c("beta", "tau"), 1000, thin = 1)
regr.coda.largeN = coda.samples(regr, c("beta", "tau"),
   50000, thin = 100)
#####   no collinearity  #####
set.seed(90201)
```

```
x1 = rnorm(N, mean = 3, sd = 2)
x2 = rnorm(N,mean = 3, sd = 2) + rnorm(N, mean = 3, sd = 0.2)
x = cbind(rep(1, N), x1, x2)
y = alpha + beta1 * x1 + beta2 * x2 + rnorm(N, mean = 0, sd = 0.2)
data = list(y = y, x = x, N = N)
summ = summary(lm(y ~ x1 + x2))
betahat = as.numeric( summ$coeff )[1:3]
covbetahat = summ$sigma^2 * solve(t(x) %*% x)
inits=function(){list(beta = as.numeric(rmvnorm(n = 1,
   mean = betahat, sigma = 1.5 * covbetahat)) ,
   tau=runif(1, 1 / (4 * summ$sigma^2), 4 / summ$sigma^2))}
regr.noco <- jags.model("lin_reg_vect.bug", data = data,
   inits = inits, n.chains = 3, n.adapt = 1000, quiet = FALSE)
regr.coda.noco = coda.samples(regr.noco, c("beta", "tau"),
   1000, thin = 1)
pdf("linRegMCMC.pdf", width = 7, height = 6)
par(mfrow=c(3,4))
traceplot(regr.coda)
traceplot(regr.coda.noco)
traceplot(regr.coda.largeN)
graphics.off()
```

□

We now introduce two widely used diagnostics, `Rhat` and `n.eff`. Suppose one samples M chains, each of length N after burn-in. Let $\boldsymbol{\theta}_{i,j}$ be the ith iterate from the jth chain and let $\psi_{i,j} = \psi(\boldsymbol{\theta}_{i,j})$ for some scalar-valued function ψ. For example, to extract the kth parameter, one would use $\psi(\boldsymbol{x}) = x_k$, or ψ might compute the standard deviation or the variance from the precision. We also use ψ to denote the estimand $\psi(\boldsymbol{\theta})$.

Let

$$\overline{\psi}_{\cdot,j} = N^{-1} \sum_{i=1}^{N} \psi_{i,j} \tag{20.33}$$

be the mean of the jth chain and let

$$\overline{\psi}_{\cdot,\cdot} = M^{-1} \sum_{j=1}^{M} \psi_{\cdot,j}. \tag{20.34}$$

$\overline{\psi}_{\cdot,\cdot}$ is the average of the chain means and is the Monte Carlo approximation to $E(\psi|\boldsymbol{Y})$. Then define

$$B = \frac{N}{M-1} \sum_{j=1}^{M} \left(\overline{\psi}_{\cdot,j} - \overline{\psi}_{\cdot,\cdot}\right)^2. \tag{20.35}$$

B/N is the sample variance of the chain means. Define

$$s_j^2 = (N-1)^{-1} \sum_{i=1}^{N} \left(\psi_{i,j} - \overline{\psi}_{\cdot,j}\right)^2, \tag{20.36}$$

the variance of the jth chain, and define

$$W = M^{-1} \sum_{j=1}^{M} s_j^2. \tag{20.37}$$

W is the pooled within-chain variance. The two variances, B and W, are combined into

$$\widehat{\text{var}}^+(\psi|\boldsymbol{Y}) = \frac{N-1}{N} W + \frac{1}{N} B, \tag{20.38}$$

where, as before, $\boldsymbol{Y}$ is the data.

To assess convergence, one can use

$$\widehat{R} = \sqrt{\frac{\widehat{\text{var}}^+(\psi|\boldsymbol{Y})}{W}}. \tag{20.39}$$

$\widehat{R}$ is called the "potential scale reduction factor" in output produced by the function `gelman.diag()`; see the output in Example 20.9. $\widehat{R}$ is also called the "shrink factor" or sometime the "Gelman shrink factor."

When the chains have not yet reached the stationary distribution, the numerator $\widehat{\text{var}}^+(\psi|\boldsymbol{Y})$ inside the radical is an upward-biased estimate of $\text{var}(\psi|\boldsymbol{Y})$ and the denominator W is a downward-biased estimator of this quantity. Both biases converge to 0 as the burn-in period and Monte Carlo sample size increase. Therefore, larger values of $\widehat{R}$ indicate nonconvergence. If $\widehat{R}$ is approximately equal to 1, say at most 1.1, then the chains are considered to have converged to the stationary distribution and $\widehat{\text{var}}^+(\psi|\boldsymbol{Y})$ can be used as an estimate of $\text{var}(\psi|\boldsymbol{Y})$. A larger value of $\widehat{R}$ is an indication that a longer burn-in period is needed. A small value of $\widehat{R}$ is evidence that the burn-in period is adequate, but we need another diagnostic, the effective sample size, to know if the sampling period was long enough.

The *effective sample size* of the chain is

$$N_{\text{eff}} = MN \frac{\widehat{\text{var}}^+(\psi|\boldsymbol{Y})}{B}. \tag{20.40}$$

The interpretation of N_{eff} is that the Markov chain can estimate the posterior expectation of ψ with approximately the same precision as would be obtained from an independent sample from the posterior of size N_{eff}. (Of course, it is usually impossible to actually obtain an independent sample, which is why MCMC is used.)

N_{eff} is derived by comparing the variance of $\overline{\psi}_{\cdot,\cdot}$ from Markov chain sampling with the variance of $\overline{\psi}_{\cdot,\cdot}$ under hypothetical independent sampling. Since

$\overline{\psi}_{\cdot,\cdot}$ is the average of the means of M independent chains and since B/N is the sample variance of these M chain means,

$$M^{-1}\frac{B}{N} \tag{20.41}$$

estimates the Monte Carlo variance of $\overline{\psi}_{\cdot,\cdot}$. Suppose instead of sampling M chains, each of length N, one could take an independent sample of size N^* from the posterior. The Monte Carlo variance of the mean of this sample would be

$$\frac{\text{var}(\psi|\boldsymbol{Y})}{N^*},$$

which can be estimated by

$$\frac{\widehat{\text{var}}^{+}(\psi|\boldsymbol{Y})}{N^*}. \tag{20.42}$$

By definition N_{eff} is the value of N^* that makes (20.41) equal to (20.42) and therefore N^* is given by (20.40). Because B/N is the sample variance of M chains and because M is typically quite small, often between 2 and 5, B has considerable Monte Carlo variability. Therefore, N_{eff} is at best a crude estimate of the effective sample size.

$\widehat{R}$ and N_{eff} are computed by the functions `gelman.diag()` and `effectiveSize()` in the `coda` package. The function `gelman.plot()` in the `coda` package plots the $\widehat{R}$ evaluated at various times along the simulation. The documentation for this function notes that "A potential problem with gelman.diag is that it may mis-diagnose convergence if the shrink factor happens to be close to 1 by chance. By calculating the shrink factor at several points in time, gelman.plot shows if the shrink factor has really converged, or whether it is still fluctuating." Figure 20.6 shows the Gelman plot from Example 20.9. The dashed red line is an upper 97.5 % confidence limit. Although $\widehat{R}$ varies during the simulations, it appears to have converged to values close to 1 by the end of the simulations.

How large should N_{eff} be? Of course, larger means better Monte Carlo accuracy, but larger values of N_{eff} require more or longer chains, so we do not want N_{eff} to be unnecessarily large. The effect of N_{eff} on estimation error can be seen by decomposing the estimation error $\psi - \overline{\psi}_{\cdot,\cdot}$ into two parts, which will be called E_1 and E_2:

$$\psi - \overline{\psi}_{\cdot,\cdot} = \{\psi - E(\psi|\boldsymbol{Y})\} + \left\{E(\psi|\boldsymbol{Y}) - \overline{\psi}_{\cdot,\cdot}\right\} = E_1 + E_2. \tag{20.43}$$

If $E\{\psi|\boldsymbol{Y}\}$ could be computed exactly so that it, not $\overline{\psi}_{\cdot,\cdot}$, would be the estimator of ψ, then E_1 would be the only error. E_2 is the error due to the Monte Carlo approximation of $E\{\psi|\boldsymbol{Y}\}$ by $\overline{\psi}_{\cdot,\cdot}$. The two errors E_1 and E_2 are uncorrelated, so

$$\text{var}\{(\psi - \overline{\psi}_{\cdot,\cdot})|\boldsymbol{Y}\} = \text{var}(E_1|\boldsymbol{Y}) + \text{var}(E_2|\boldsymbol{Y})$$

$$= \text{var}(\psi|\boldsymbol{Y}) + \frac{\text{var}(\psi|\boldsymbol{Y})}{N_{\text{eff}}}$$
$$= \text{var}(\psi|\boldsymbol{Y}) \left(1 + \frac{1}{N_{\text{eff}}}\right)$$

by the definitions of $\text{var}(\psi|\boldsymbol{Y})$ and N_{eff} and using the approximation $\widehat{\text{var}}^{+}(\psi|\boldsymbol{Y}) \approx \text{var}(\psi|\boldsymbol{Y})$. Using the Taylor series approximation $\sqrt{1+\delta} \approx 1 + \delta/2$ for small values of δ, we see that

$$\sqrt{\text{var}\{(\psi - \overline{\psi}_{\cdot,\cdot})|\boldsymbol{Y}\}} \approx \sqrt{\text{var}(\psi|\boldsymbol{Y})} \left(1 + \frac{1}{2N_{\text{eff}}}\right). \tag{20.44}$$

Recall that $\sqrt{\text{var}\{(\psi - \overline{\psi}_{\cdot,\cdot})|\boldsymbol{Y}\}}$ is the "Bayesian standard error." If $N_{\text{eff}} \geq 50$, then we see from (20.44) that the standard error is inflated by Monte Carlo error by at most 1 %. Thus, one might use the rule-of-thumb that N_{eff} should be at least 50. Remember, however, that N_{eff} is estimated only crudely because the number of chains is small. Thus, we might want to have N_{eff} at least 100 to provide some leeway for error in the estimation of N_{eff}.

The value of N_{eff} can vary between different choices of ψ. Recall that the values of N_{eff} from Example 20.9 were:

```
> effectiveSize(univt.coda)
       k       mu    sigma
366.8874 300.0000 300.0000
```

In this example, N_{eff} is 300 for `mu` and `sigma` and only slightly larger for `k`. (In more complex models, much greater variation in N_{eff} is common.) In this example, 300 is the Monte Carlo sample size since there are three chains, each of length 100 after burn-in and thinning.[3] Therefore, in this simple example, MCMC sampling is as effective as independent sampling; this is not a typical case.

For convenience, $\widehat{R}$ values in Example 20.9 are listed again:

```
> gelman.diag(univt.coda)
Potential scale reduction factors:

      Point est. Upper C.I.
k           1.00       1.01
mu          1.00       1.01
sigma       1.02       1.06

Multivariate psrf

1.02
```

[3] The effective sample size can be larger than the actual sample size if there is negative correlation, but negative correlation is unlikely. It is more likely that some of the effective sample sizes exceed the actual sample sizes due to random variation, i.e., estimation error.

One can see that $\widehat{R}$ is at most 1.02 for all of the parameters that were monitored, which is another indication that the amount of MCMC sampling was sufficient. Even the 95 % upper confidence limits are satisfactory, at most 1.06.

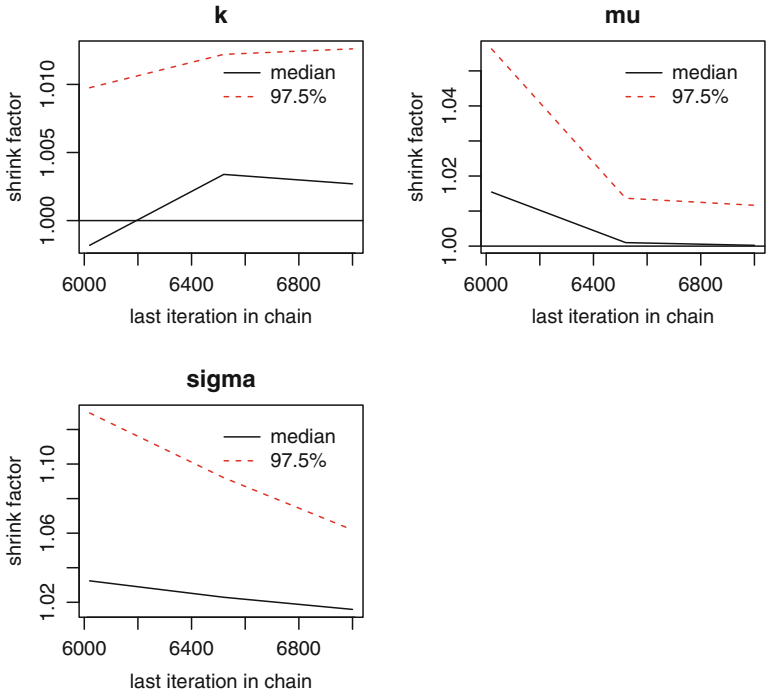

Fig. 20.6. *Gelman plot in Example 20.9.*

20.7.6 DIC and p_D for Model Comparisons

In this section, we introduce two widely used statistics, DIC and p_D. DIC is used to compare several models for the same data set and is a Bayesian analog of AIC. p_D is a Bayesian analog to the number of parameters in the model.

Recall from Sect. 5.12 that the deviance, denoted now by $D(\boldsymbol{Y}, \boldsymbol{\theta})$, is minus twice the log-likelihood, and AIC defined by (5.29) is

$$\text{AIC} = D(\boldsymbol{Y}, \boldsymbol{\theta}_{\text{ML}}) + 2p, \tag{20.45}$$

where $\widehat{\boldsymbol{\theta}}_{\text{ML}}$ is the MLE and p is the dimension of $\boldsymbol{\theta}$. A Bayesian analog of the MLE is the posterior mean, the usual Bayes estimator, which can be estimated by MCMC.

We need a Bayesian analog of p, the number of parameters. It may seem strange at first that we do not simply use p itself as in a non-Bayesian analysis. After all, the number of parameters has not changed just because we now have a prior and are using Bayesian estimation. However, the prior information used in a Bayesian analysis somewhat constrains the estimated parameters, which makes the *effective* number of parameters less than p. To appreciate why this is true, consider an example where there are d returns on equities that are believed to be similar. Assume the returns have a multivariate normal distribution. Let's focus on the d expected returns, call them $\mu_1, \ldots, \mu_d$. To a non-Bayesian, there are two ways to model $\mu_1, \ldots, \mu_d$. The first is to assume that they are all equal, say to μ, and then there is only one parameter to model the means. The other possibility is to assume that the expected returns are not equal so that there are d parameters.

A Bayesian can achieve a compromise between these two extreme by specifying a prior such that $\mu_1, \ldots, \mu_d$ are similar but not identical. For example, we could assume that they are i.i.d. $N(\mu, \sigma_\mu^2)$, and σ_μ^2 would specify the degree of similarity. It is important to appreciate that σ_μ^2 can be estimated from the data, that is, there is no need to specify in advance the degree of similarity between the means. The result of using such prior information is that the *effective* number of parameters to specify $\mu_1, \ldots, \mu_d$ is greater than 1 but less than d.

The effective number of parameters is defined as

$$p_D = \widehat{D}_{\text{avg}} - D(\boldsymbol{Y}, \overline{\boldsymbol{\theta}}), \tag{20.46}$$

where

$$\overline{\boldsymbol{\theta}} = (NM)^{-1} \sum_{j=1}^{M} \sum_{i=1}^{N} \boldsymbol{\theta}_{i,j}$$

is the average of the MCMC sample of $\boldsymbol{\theta}_{i,j}$ and estimates the posterior expectation of θ, and

$$\widehat{D}_{\text{avg}} = (NM)^{-1} \sum_{j=1}^{M} \sum_{i=1}^{N} D(\boldsymbol{Y}, \boldsymbol{\theta}_{i,j})$$

is an MCMC estimate of

$$D_{\text{avg}} = E\{D(\boldsymbol{Y}, \boldsymbol{\theta}) | \boldsymbol{Y}\}. \tag{20.47}$$

In analogy with (20.45), DIC is defined as

$$\text{DIC} = D(\boldsymbol{Y}, \overline{\boldsymbol{\theta}}) + 2p_D.$$

By (20.46), we have as well that

$$\text{DIC} = \widehat{D}_{\text{avg}} + p_D.$$

The function `dic.samples()` in the `rjags` package reports $\widehat{D}_{\text{avg}}$ as the "mean deviance," p_D as the "penalty," and DIC as the "penalized deviance." See the output in Example 20.9.

As the following example illustrates, p_D is primarily a measure of the posterior variability of $\boldsymbol{\theta}$, which increases as p increases or the amount of prior information about $\boldsymbol{\theta}$ decreases relative to the information in the sample.

Example 20.11. p_D when estimating a normal mean with known precision

Suppose that $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ are i.i.d. $N(\mu, 1)$, so $\boldsymbol{\theta} = \mu$ in this example. Then the log-likelihood is

$$\begin{aligned}\log\{L(\mu)\} &= -\frac{1}{2}\sum_{i=1}^{n}(Y_i-\mu)^2 - \frac{n}{2}\log(2\pi)\\ &= -\frac{1}{2}\left\{\sum_{i=1}^{n}(Y_i-\overline{Y})^2 + n(\overline{Y}-\mu)^2\right\} - \frac{n}{2}\log(2\pi),\end{aligned}$$

and so

$$D(\boldsymbol{Y},\mu) = \sum_{i=1}^{n}(Y_i-\overline{Y})^2 + n(\overline{Y}-\mu)^2 + n\log(2\pi). \tag{20.48}$$

When p_D is computed, quantities not depending on μ cancel with the subtraction in (20.46). Therefore, for the purpose of computing p_D, we can use

$$D(\boldsymbol{Y},\mu) = n(\overline{Y}-\mu)^2. \tag{20.49}$$

Then

$$D\{\boldsymbol{Y}, E(\mu|\boldsymbol{Y})\} = \{\overline{Y} - E(\mu|\boldsymbol{Y})\}^2, \tag{20.50}$$

and

$$\begin{aligned}D_{\text{avg}} &= n\,E\{(\overline{Y}-\mu)^2|\boldsymbol{Y}\}\\ &= n\Big(\{\overline{Y}-E(\mu|\boldsymbol{Y})\}^2 + E\big[\{E(\mu|\boldsymbol{Y})-\mu\}^2|\boldsymbol{Y}\big]\Big)\\ &= n\Big[\{\overline{Y}-E(\mu|\boldsymbol{Y})\}^2 + \text{Var}(\mu|\boldsymbol{Y})\Big]\\ &= D\{\boldsymbol{Y}, E(\mu|\boldsymbol{Y})\} + n\text{Var}(\mu|\boldsymbol{Y}),\end{aligned} \tag{20.51}$$

because $\{\overline{Y} - E(\mu|\boldsymbol{Y})\}$ and $\{E(\mu|\boldsymbol{Y}) - \mu\}$ are conditionally uncorrelated given $\boldsymbol{Y}$. Therefore,

$$\begin{aligned}p_D &= \widehat{D}_{\text{avg}} - D\{\boldsymbol{Y}, E(\mu|\boldsymbol{Y})\}\\ &\approx D_{\text{avg}} - D\{\boldsymbol{Y}, E(\mu|\boldsymbol{Y})\} = n\text{Var}(\mu|\boldsymbol{Y}) = \frac{n}{n+\tau_0},\end{aligned} \tag{20.52}$$

where the last equality uses (20.21) and τ_0 is the prior precision for μ. The approximation ("$\approx$") in (20.52) becomes equality as the Monte Carlo sample size N increases to ∞.

As $\tau_0 \to 0$, the amount of prior information becomes negligible and the right-hand side of (20.52) converges to $p = 1$. Conversely, as $\tau_0 \to \infty$, the amount of prior information increases without bound and the right-hand side of (20.52) converges to 0. This is an example of a general phenomenon—more prior information means less effective parameters. □

Generally, $p_D \approx p$ when p is small and there is little prior information. In other cases, such as when d means are modeled as coming from a common normal distribution, p_D could be considerably less than p—see Example 20.12.

When comparing models using DIC, smaller is better, though, like AIC and BIC, DIC should never be used blindly. Often subject-matter considerations or model simplicity will lead an analyst to select a model other than the one minimizing DIC.

The function `dic.sample()` returns both DIC and p_D, as can be seen in the output from Example 20.9 which was:

```
> dic.samples(univt.mcmc, 100*nthin, thin = nthin, type = "pD")
  |**************************************************| 100%
Mean deviance:  -18065
penalty 2.664
Penalized deviance: -18062
```

20.8 Hierarchical Priors

A common situation is having a number of parameters that are believed to have similar, but not identical, values. For example, the expected returns on several equities might be thought similar. In such cases, it can be useful to pool information about the parameters to improve the specification of the prior, because the use of good prior information will improve the accuracy of the estimation. A effective method for pooling information is a Bayesian analysis with a so-called "hierarchical prior" that allows one to shrink the estimates toward each other or toward some other target. An example of the latter would be shrinking the sample covariance matrix of returns toward an estimate from the CAPM or another factor model. This type of shrinkage would achieve a tradeoff between the high variability of the sample covariance matrix and the potential bias of the covariance matrix estimator from a factor model when the factor model does not fit perfectly.

As before, let the likelihood be $f(\boldsymbol{y}|\boldsymbol{\theta})$. The likelihood is the first layer (or stage) in the hierarchy. So far in this chapter, the prior density of $\boldsymbol{\theta}$, which is the second layer, has been $\pi(\boldsymbol{\theta}|\boldsymbol{\gamma})$, where the parameter vector $\boldsymbol{\gamma}$ in the prior has a known value, say $\boldsymbol{\gamma}_0$. For example, in Example 20.3 the prior had a beta distribution with both parameters fixed.

In a *hierarchical* or multistage prior, $\boldsymbol{\gamma}$ is unknown and has its own prior $\pi(\boldsymbol{\gamma}|\boldsymbol{\delta})$ (the third layer). Typically, $\boldsymbol{\delta}$ has a known value, though one can add further layers to the hierarchy by making $\boldsymbol{\delta}$ unknown with its own prior, and so forth.

It is probably easiest to understand hierarchical priors using examples.

Example 20.12. Estimating expected returns on midcap stocks

This example uses the `midcapD.ts` dataset. This data set contains 500 daily returns on 20 midcap stocks and on the market and was used in Example 5.2.

The data set will be divided into the "training data," which contains the first 100 days of returns and the "test" data containing the last 400 days of returns. Only the training data will be used for estimation. The test data will be used to compare the estimates from the training data. The test data sample size was chosen intentionally to be relatively large so that we can consider the mean returns from the test data to be the "true" expected returns on the 20 stocks, though, of course, this is only an approximation. The "true" expected returns will be estimated using the training data.

We will compare three possible estimators of the true expected returns.

(a) sample means (the 20 mean returns on the midcap stocks for the first 100 days);
(b) pooled estimation (total shrinkage where every expected return has the same estimate);
(c) Bayes estimation with a hierarchical prior (shrinkage).

Method (a) is the "usual" non-Bayesian estimator where each expected return is estimated by the sample mean of that stock. In method (b), every expected return has the same estimate, which is the "mean of means," that is, the average of the 20 means from (a). Bayes shrinkage, which will be explained in this example, shrinks the 20 individual means toward the mean of means using a hierarchical prior. Bayesian shrinkage is a compromise between (a) and (b). Shrinkage was also used in Example 16.10 though in that example the amount of shrinkage was chosen arbitrarily because Bayesian methods had not yet been introduced.

Let $R_{i,t}$ be the tth daily return on i stock expressed as a percentage. For Bayesian shrinkage, the first layer will be the simple model

$$R_{i,t} = \mu_i + \epsilon_{i,t},$$

where $\epsilon_{i,t}$ are i.i.d. $N(0, \sigma_\epsilon^2)$. This model has several unrealistic aspects: (a) the assumption that the standard deviation of $\epsilon_{i,t}$ does not depend on i; (b) the assumption that $\epsilon_{i,t}$ and $\epsilon_{i',t}$ are independent (we know that there will be cross-sectional correlations); (c) the assumption that there are no GARCH effects; (d) the assumption that the $\epsilon_{i,t}$ are normally distributed rather than

having heavy tails. Nonetheless, for the purpose of estimating expected returns, this model should be adequate. Remember, "all models are wrong but some models are useful," and, of course, what is "useful" depends on the objectives of the analysis.

The hierarchy prior has second layer

$$\mu_i \sim \text{ i.i.d. } N(\alpha, \sigma_\mu^2).$$

The assumption here is that the expected returns for the 20 midcap stocks have been sampled from a large population of expected returns, perhaps of all midcap stocks or even a larger population. The mean of that population is α and the standard deviation is σ_μ.

If we used a non-hierarchical prior, then we would need to specify values of α and σ_μ. This is exactly what was done in Example 20.4, except in that example σ_ϵ^2 also was known. We probably have a rough idea of the values of α and σ_μ, but it is unlikely that we have precise information about them, and we saw in Example 20.4 that a rather accurate specification of the prior is needed for the Bayes estimator to improve upon the sample means. In fact, the Bayes estimator can easily be inferior to the sample means if the prior is poorly chosen.

The third layer will be a prior on α and σ_μ and will let us use the data to estimate these parameters. It is important to appreciate why we can estimate α and σ_μ in this example, but they could not be estimated in Example 20.4. The reason is that we now have 20 expected returns (the μ_i) that are distributed with the same mean α and standard deviation σ_μ. In contrast, in Example 20.4 there is only a single μ and so it not possible to estimate the mean and variance of the population from which this μ was sampled.

Because there is now a substantial amount of information in the data about α, σ_ϵ^2, and σ_μ^2, we could use fairly noninformative priors for them to "let the data speak for themselves."

The `BUGS` program for this example is:

```
model{
for (i in 1:n)
{
  for (j in 1:m)
  {
    x1[i,j] ~ dnorm(mu[j], tau_eps)
  }
}
for (j in 1:m)
{
  mu[j] ~ dnorm(alpha, tau_mu)
}
alpha ~ dnorm(0.0, 1.0E-3)
tau_eps ~ dgamma(0.1, 0.01)
tau_mu ~ dgamma(0.1, 0.01)
```

```
sigma_eps <- 1 / sqrt(tau_eps)
sigma_mu <- 1 / sqrt(tau_mu)
}
```

The R code for this example is below:

```
library(rjags)
dat = read.csv("midcapD.ts.csv")
market = 100 * as.matrix(dat[, 22])
x = 100 * as.matrix(dat[, -c(1, 22)])
m = 20
k = 100
x1 = x[1:k, ]
x2 = x[(k+1):500, ]
mu1 = apply(x1, 2, mean)
mu2 = apply(x2, 2, mean)
means = apply(x1, 2, mean)
sd2 = apply(x1, 2, sd)
tau_mu = 1 / mean(sd2^2)
tau_eps = 1 / sd(means)^2
n = k
data = list(x1 = x1, n = n, m = m)
inits.midCap = function(){list(alpha = 0.001, mu = means,
   tau_eps = tau_eps, tau_mu = tau_mu)}
midCap <- jags.model("midCap.bug", data = data, inits = inits.midCap,
   n.chains = 3, n.adapt = 1000, quiet = FALSE)
nthin = 20
midCap.coda = coda.samples(midCap, c("mu", "tau_mu", "tau_eps",
   "alpha", "sigma_mu", "sigma_eps"), 500 * nthin, thin = nthin)
summ.midCap = summary(midCap.coda)
summ.midCap
post.means = summ.midCap[[1]][2:21, 1]
pdf("midcap.pdf", width = 6, height = 3.75)
par(mfrow = c(1, 2))
plot(c(rep(1, m), rep(2, m)), c(mu1, mu2),
     xlab = "estimate                          target",ylab = "mean",
     main = "sample means",
     ylim = c(-0.3, 0.7), axes = FALSE)
axis(2)
axis(1, labels = FALSE, tick = TRUE, lwd.tick = 0)
for (i in 1:m){lines(1:2, c(mu1[i], mu2[i]), col = i)}
plot(c(rep(1, m), rep(2, m)), c(post.means, mu2),
     xlab = "estimate                          target", ylab = "mean",
     main = "Bayes",
     ylim=c(-0.3, 0.7), axes = FALSE)
axis(2)
axis(1, labels = FALSE, tick = TRUE, lwd.tick = 0)
for (i in 1:m){lines(1:2, c(post.means[i], mu2[i]) ,col=i)}
graphics.off()
options(digits = 2)
```

```
sum((mu1 - mu2)^2 )
sum((post.means - mu2)^2)
sum((mean(mu1) - mu2)^2)
```

The output is below.

```
> summ.midCap

Iterations = 20:10000
Thinning interval = 20
Number of chains = 3
Sample size per chain = 500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean        SD  Naive SE Time-series SE
alpha      0.08730  0.102433 2.645e-03      3.101e-03
mu[1]      0.11121  0.171456 4.427e-03      4.546e-03
mu[2]      0.12128  0.169230 4.369e-03      4.892e-03
mu[3]      0.07871  0.170849 4.411e-03      4.702e-03

 (edited to save space)

mu[19]     0.05082  0.175466 4.531e-03      4.422e-03
mu[20]     0.03997  0.184614 4.767e-03      4.873e-03
sigma_eps  4.30691  0.067810 1.751e-03      1.629e-03
sigma_mu   0.14970  0.067054 1.731e-03      1.671e-03
tau_eps    0.05395  0.001699 4.386e-05      4.074e-05
tau_mu    75.70128 68.715669 1.774e+00      1.738e+00

2. Quantiles for each variable:

              2.5%       25%      50%      75%     97.5%
alpha     -0.11465  0.017800  0.08600  0.15560   0.29111
mu[1]     -0.21768 -0.001040  0.10910  0.21864   0.43408
mu[2]     -0.21107  0.018025  0.11704  0.22382   0.47727
mu[3]     -0.27262 -0.032547  0.08392  0.19634   0.40900

 (edited to save space)

mu[19]    -0.30441 -0.059642  0.05612  0.16521   0.38525
mu[20]    -0.34436 -0.069959  0.04462  0.16479   0.37048
sigma_eps  4.17808  4.261352  4.30714  4.35226   4.43733
sigma_mu   0.06155  0.100600  0.13853  0.18225   0.31293
tau_eps    0.05079  0.052792  0.05390  0.05507   0.05729
tau_mu    10.21311 30.107752 52.10738 98.81013 263.94725
```

The posterior means of σ_μ and σ_ϵ are 0.150 % and 4.31 %, respectively (the returns are as percentages). If we look at precisions instead of standard

deviations, then we find that the posterior means of τ_μ and τ_ϵ are 75.7 and 0.0540. Using the notation of (20.24), in the present example $\tau_{\overline{Y}}$ is $100\tau_\epsilon = 5.4$ and $\tau_0 = \tau_\mu = 75.7$. Therefore, δ in (20.24) is $5.4/(5.4 + 75.7) = 0.067$. Recall that δ close to 0 (far from 1) results in substantial shrinkage, so δ equal to 0.064 causes a great amount of shrinkage of the sample means toward the mean of means, as can be seen in Fig. 20.7.

To compare the estimators, we use the sum of squared errors (SSE) defined as

$$\text{SSE} = \sum_{i=1}^{20} (\widehat{\mu}_i - \mu_i)^2,$$

where μ_i is the ith "true" mean from the test data and $\widehat{\mu}_i$ is an estimate from the training data. The values of the SSE are found in Table 20.1. The SSE for the sample means is about 11 (1.9/0.18) times larger than for the Bayes estimate. Clearly, shrinkage is very successful in this example.

Interestingly, complete shrinkage to the pooled mean is even better than Bayesian shrinkage. Bayesian shrinkage attempts to estimate the optimal amount of shrinkage, but, of course, it cannot do this perfectly. Although complete shrinkage is better than Bayesian shrinkage in this example, complete shrinkage is, in general, dangerous since it will have a large SSE in examples where the true means differ more than in this case. If one has a strong prior belief that the true means are very similar, one should use this

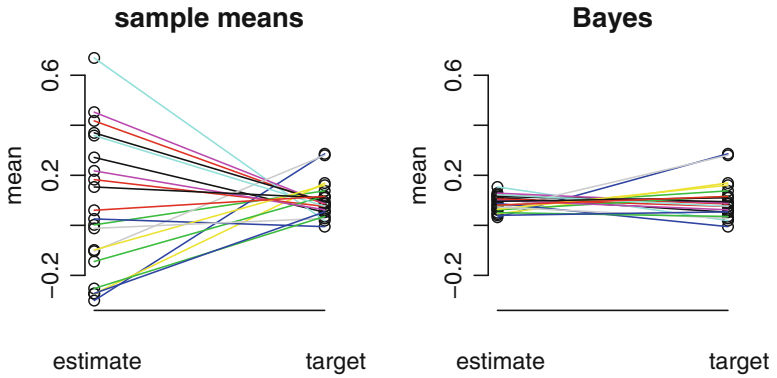

Fig. 20.7. *Estimation of the average returns for* 20 *midcap stocks. "Target" is the quantity being estimated, specifically the average return over* 400 *days of test data. "Estimate" is an estimate based on the* 100 *previous days of training data. On the left, the estimates are the* 20 *individual sample means. On the right, the estimates are the sample means shrunk toward their mean. In each panel, the estimate and target for each stock are connected by a line. On the left, the sample means of the training data are so variable that the stocks with smaller (larger) means in the training data often have larger (smaller) means in the test data. The Bayes estimates on the right are much closer to the targets.*

belief when specifying a prior for σ_μ. Instead of using a noninformative prior as in this example, one would use a prior more concentrated near 0. □

Table 20.1. Sum of squared errors (SSE) for three estimators of the expected returns of 20 midcap stocks.

Estimate	SSE
(a) sample means	1.9
(b) pooled mean	0.12
(c) Bayes	0.18

20.9 Bayesian Estimation of a Covariance Matrix

In this section, we assume that $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$ is an i.i.d. sample from a d-dimensional $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ distribution or a d-dimensional $t_\nu(\boldsymbol{\mu}, \boldsymbol{\Lambda})$ distribution. We will focus on estimation of the covariance matrix $\boldsymbol{\Sigma}$ of a multivariate normal distribution or the scale matrix $\boldsymbol{\Lambda}$ of a multivariate t-distribution. The *precision matrix* is defined as $\boldsymbol{\Sigma}^{-1}$ or $\boldsymbol{\Lambda}^{-1}$ for the Gaussian and t-distributions, respectively. This definition is analogous to the univariate case where the precision is defined as the reciprocal of the variance or squared scale parameter.

We will start with Gaussian distributions.

20.9.1 Estimating a Multivariate Gaussian Covariance Matrix

In the multivariate Gaussian case, the conjugate prior for the precision matrix $\boldsymbol{\Sigma}^{-1}$ is the Wishart distribution. The Wishart distribution, denoted by Wishart$(\nu, \boldsymbol{A})$, has a univariate parameter ν called the degrees of freedom and a matrix parameter $\boldsymbol{A}$ that can be any nonsingular covariance matrix. There is a simple definition of the Wishart$(\nu, \boldsymbol{A})$ distribution when ν is an integer. Let $\boldsymbol{Z}_i, \ldots, \boldsymbol{Z}_n$ be i.i.d. $N(\boldsymbol{\mu}, \boldsymbol{A})$. In this case, the distribution of

$$\sum_{i=1}^{n} (\boldsymbol{Z}_i - \boldsymbol{\mu})(\boldsymbol{Z}_i - \boldsymbol{\mu})^\mathsf{T}$$

is Wishart$(n, \boldsymbol{A})$. Also, the distribution of

$$\sum_{i=1}^{n} (\boldsymbol{Z}_i - \overline{\boldsymbol{Z}})(\boldsymbol{Z}_i - \overline{\boldsymbol{Z}})^\mathsf{T} \tag{20.53}$$

is Wishart$(n-1, \boldsymbol{A})$. Because the sum in (20.53) is $n-1$ times the sample covariance matrix, the Wishart distribution is important for inference about the covariance matrix of a Gaussian distribution.

The density of a Wishart$(\nu, \boldsymbol{A})$ distribution for any positive value of ν is

$$f(\boldsymbol{W}) = C(\nu, d)\,|\boldsymbol{A}|^{-\nu/2}\,|\boldsymbol{W}|^{(\nu-d-1)/2} \exp\left\{-\frac{1}{2}\mathrm{tr}(\boldsymbol{A}^{-1}\boldsymbol{W})\right\} \tag{20.54}$$

with normalizing constant

$$C(\nu, d) = \left\{2^{\nu d/2}\,\pi^{d(d-1)/4} \prod_{i=1}^{d} \Gamma\left(\frac{\nu+1-i}{2}\right)\right\}^{-1}.$$

The argument $\boldsymbol{W}$ is a nonsingular covariance matrix. The expected value is $E(\boldsymbol{W}) = \nu\boldsymbol{A}$. In the univariate case ($d = 1$), the Wishart distribution is a gamma distribution.

If $\boldsymbol{W}$ is Wishart$(\nu, \boldsymbol{A})$ distributed, then the distribution of $\boldsymbol{W}^{-1}$ is called the inverse Wishart distribution with parameters ν and $\boldsymbol{A}^{-1}$ and denoted Inv-Wishart$(\nu, \boldsymbol{A}^{-1})$.

Let $\boldsymbol{Y} = (\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n)$ denote the data. To derive the full conditional for the precision matrix $\boldsymbol{\Sigma}^{-1}$, assume that $\boldsymbol{\mu}$ is known. We know from (7.15) that the likelihood is

$$f(\boldsymbol{Y}|\boldsymbol{\Sigma}^{-1}) = \prod_{i=1}^{n}\left[\frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{Y}_i - \boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y}_i - \boldsymbol{\mu})\right\}\right].$$

After some simplification,

$$f(\boldsymbol{Y}|\boldsymbol{\Sigma}^{-1}) \propto |\boldsymbol{\Sigma}^{-1}|^{n/2} \exp\left\{-\frac{1}{2}\sum_{i=1}^{n}(\boldsymbol{Y}_i - \boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y}_i - \boldsymbol{\mu})\right\}.$$

Define

$$\boldsymbol{S} = \sum_{i=1}^{n}(\boldsymbol{Y}_i - \boldsymbol{\mu})(\boldsymbol{Y}_i - \boldsymbol{\mu})^{\mathsf{T}}.$$

Next

$$\sum_{i=1}^{n}(\boldsymbol{Y}_i - \boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y}_i - \boldsymbol{\mu}) = \mathrm{tr}\left\{\sum_{i=1}^{n}(\boldsymbol{Y}_i - \boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y}_i - \boldsymbol{\mu})\right\} = \mathrm{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}). \tag{20.55}$$

The first equality in (20.55) is the trivial result that a scalar is also a 1×1 matrix and equal to its trace. The second equality uses the result that $\mathrm{tr}(\boldsymbol{AB}) = \mathrm{tr}(\boldsymbol{BA})$ for any matrices $\boldsymbol{B}$ and $\boldsymbol{A}$ such that the products $\boldsymbol{BA}$ and $\boldsymbol{AB}$ are defined. If follows that

$$f(\boldsymbol{Y}|\boldsymbol{\Sigma}^{-1}) \propto |\boldsymbol{\Sigma}^{-1}|^{n/2} \exp\left\{-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{S})\right\}. \tag{20.56}$$

Suppose that the prior on the precision matrix $\boldsymbol{\Sigma}^{-1}$ is Wishart$(\nu_0, \boldsymbol{\Sigma}_0^{-1})$. Then the prior density is

$$\pi(\boldsymbol{\Sigma}^{-1}) \propto |\boldsymbol{\Sigma}^{-1}|^{(\nu_0-d-1)/2} \exp\left\{-\frac{1}{2}\text{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_0)\right\}. \tag{20.57}$$

Since the posterior density is proportional to the product of the prior density and the likelihood, it follows from (20.56) and (20.57) that the posterior density is

$$\pi(\boldsymbol{\Sigma}^{-1}|\boldsymbol{Y}) \propto |\boldsymbol{\Sigma}^{-1}|^{(n+\nu_0-d-1)/2} \exp\left[-\frac{1}{2}\text{tr}\left\{\boldsymbol{\Sigma}^{-1}(\boldsymbol{S}+\boldsymbol{\Sigma}_0)\right\}\right]. \tag{20.58}$$

Therefore, the posterior distribution of $\boldsymbol{\Sigma}^{-1}$ is Wishart$\left\{n+\nu_0, (\boldsymbol{S}+\boldsymbol{\Sigma}_0)^{-1}\right\}$. The posterior expectation is

$$E(\boldsymbol{\Sigma}^{-1}|\boldsymbol{Y}) = (n+\nu_0)\left\{(\boldsymbol{S}+\boldsymbol{\Sigma}_0)^{-1}\right\}. \tag{20.59}$$

If ν_0 and $\boldsymbol{\Sigma}_0$ are both small, then

$$E(\boldsymbol{\Sigma}^{-1}|\boldsymbol{Y}) \approx n\boldsymbol{S}^{-1} \tag{20.60}$$

The MLE of $\boldsymbol{\Sigma}$ is $n^{-1}\boldsymbol{S}$, so the MLE of $\boldsymbol{\Sigma}^{-1}$ is $n\boldsymbol{S}^{-1}$. Therefore, for small values of ν_0 and $\boldsymbol{\Sigma}_0$, the Bayesian estimator of $\boldsymbol{\Sigma}^{-1}$ is close to the MLE.

The full conditional for $\boldsymbol{\Sigma}^{-1}$ can be combined with a model for $\boldsymbol{\mu}$ to estimate both parameters. For application to asset returns, a hierarchical prior for $\boldsymbol{\mu}$ such as in Example 20.12 might be used. In either case, an MCMC analysis would be straightforward.

20.9.2 Estimating a Multivariate-*t* Scale Matrix

The Wishart distribution is not a conjugate prior for the scale matrix of a multivariate t-distribution, but it can be used as the prior nonetheless, since MCMC does not require the use of conjugate priors.

Example 20.13. Estimating the correlation matrix of the `CRSPday` *data*

In Example 7.4, the correlation matrix of the `CRSPday` returns data was estimated by maximum likelihood. In this example, the MLE will be compared to a Bayes estimate and the two estimates will be found to be very similar. The `BUGS` program used in this example is

```
model{
for(i in 1:N)
{
  y[i,1:m] ~ dmt(mu[], tau[,], df_likelihood)
}
mu[1:m] ~ dmt(mu0[], Prec_mu[,], df_prior)
tau[1:m,1:m] ~  dwish(Prec_tau[,], df_wishart)
lambda[1:m,1:m] <- inverse(tau[,])
}
```

In the BUGS program, mu is the mean vector, tau is the precision matrix, lambda is the scale matrix of the returns. Also, dmt is the multivariate-t distribution, and dwish is the Wishart distribution.

At the time of this writing (Dec 2014), JAGS does not have a sampler that will sample the posterior from this model. Therefore, WinBUGS will be used and will be called using the bugs() function in the R2WinBUGS package. The R code is below.

```
library(R2WinBUGS)
library(MASS)  # need to mvrnorm
library(MCMCpack) # need for rwish
library(mnormt)
data(CRSPday, package = "Ecdat")
y = CRSPday[,4:7]
N = dim(y)[1]
m = dim(y)[2]
mu0 = rep(0,m)
Prec_mu = diag(rep(1, m)) / 10000
Prec_tau =  diag(rep(1, m)) / 10000
df_wishart = 6
df_likelihood = 6
df_prior = 6
data = list(y = y, N = N, Prec_mu = Prec_mu,
   Prec_tau = Prec_tau,
   mu0 = mu0, m = m, df_likelihood = df_likelihood,
   df_prior = df_prior, df_wishart = df_wishart)
inits_t_CRSP = function(){list(mu = mvrnorm(1, mu0,
   diag(rep(1, m) / 100)),
   tau = rwish(6, diag(rep(1, m)) / 100))}
library(R2WinBUGS)
multi_t.sim = bugs(data, inits_t_CRSP ,
   model.file = "mult_t_CRSP.bug",
   parameters = c("mu", "tau"), n.chains = 3,
   n.iter = 2200, n.burnin = 200, n.thin = 2,
   program = "WinBUGS", bugs.seed = 13, codaPkg = FALSE)
print(multi_t.sim, digits = 2)
tauhat = multi_t.sim$mean$tau
lambdahat = solve(tauhat)
sdinv = diag(1/sqrt(diag(lambdahat)))
cor = sdinv %*% lambdahat %*% sdinv
print(cor,digits=4)
```

The data list that is an input to the BUGS program contain y which is the matrix of returns, df_likelihood which is the degrees of freedom of the t-distribution in the likelihood, mu0 which is the prior mean for mu, df_prior which is the degrees of freedom in the t prior on mu, and df_wishart which is the degrees of freedom of the Wishart prior on tau.

Ideally, df_likelihood should be an unknown parameter, but WinBUGS does not allow this parameter to be estimated. Instead, we fix it at the MLE

(rounded to 6) computed in Example 7.4. The need to fix this parameter at the MLE is due to limitations of `WinBUGS` and could, with considerably more effort, be circumvented by programming the MCMC in `R` or another language rather than using `WinBUGS`.

Note that `codaPkg = FALSE` was specified in the call to `bugs()`; this was not necessary since it is the default. When `codaPkg = FALSE` then `bugs()` returns an object of class `bugs` which cannot be used directly by functions in the `coda` package since these functions take objects of class `mcmc.list`. However, the function `as.mcmc.list()` will convert a `bugs` object to an `mcmc.list` object.

There were three chains, each of length 2000 after a burn-in of 200 and thinned to every second iteration. Thus, the total sample size was 3000 after thinning. The convergence to the stationary distribution and mixing were both quite rapid. N_{eff} was at least 1500 and $\widehat{R}$ essentially 1 for all parameters, which indicate adequate burn-in and chain lengths.

```
> print(multi_t.sim, digits = 2)
Inference for Bugs model at "mult_t_CRSP.bug", fit using WinBUGS,
 3 chains, each with 2200 iterations (first 200 discarded), n.thin = 2
 n.sims = 3000 iterations saved
           mean   sd   2.5%    25%    50%    75%  97.5% Rhat n.eff
mu[1]         0    0      0      0      0      0      0    1  3000
mu[2]         0    0      0      0      0      0      0    1  3000
mu[3]         0    0      0      0      0      0      0    1  3000
mu[4]         0    0      0      0      0      0      0    1  1800
tau[1,1]  14706  473  13780  14390  14690  15020  15630    1  3000

 (edited to save space)

tau[4,4]  65197 2102  61180  63738  65190  66600  69360    1  3000
deviance -69858   61 -69980 -69900 -69860 -69820 -69730    1  3000

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = Dbar-Dhat)
pD = 13.8 and DIC = -69843.9
DIC is an estimate of expected predictive error (lower deviance is better).
```

Since μ is close to zero, `multi_t.sim` needed to be printed again, this time with more digits than 2:

```
             mean      sd     2.5%      25%      50%      75%    97.5% Rhat n.eff
mu[1]     9.4e-04 2.4e-04  4.6e-04  7.7e-04  9.4e-04  1.1e-03  1.4e-03    1  3000
mu[2]     4.4e-04 2.9e-04 -1.3e-04  2.4e-04  4.6e-04  6.4e-04  1.0e-03    1  3000
mu[3]     6.9e-04 2.3e-04  2.3e-04  5.3e-04  6.9e-04  8.4e-04  1.1e-03    1  3000
mu[4]     7.7e-04 1.3e-04  5.1e-04  6.8e-04  7.7e-04  8.6e-04  1.0e-03    1  1800
```

The Bayes estimate of the precision matrix was converted to a correlation matrix at lines 29–33 of the `R` program. The estimated correlation matrix is below.

```
       [,1]   [,2]   [,3]   [,4]
[1,] 1.0000 0.3191 0.2841 0.6756
[2,] 0.3191 1.0000 0.1586 0.4696
[3,] 0.2841 0.1586 1.0000 0.4300
[4,] 0.6756 0.4696 0.4300 1.0000
```

In Example 7.4, the MLE of the correlation matrix was found to be

```
$cor
       [,1]   [,2]   [,3]   [,4]
[1,] 1.0000 0.3192 0.2845 0.6765
[2,] 0.3192 1.0000 0.1584 0.4698
[3,] 0.2845 0.1584 1.0000 0.4301
[4,] 0.6765 0.4698 0.4301 1.0000
```

Notice the similarity between the Bayes estimate and the MLE. □

20.9.3 Non-Wishart Priors for the Covariate Matrix

We saw in Example 20.13 that a Wishart prior with noninformative choices of the prior parameters more or less replicates maximum likelihood estimation. Often, however, one wishes to shrink the covariance matrix toward some target, perhaps a estimate from a factor model. See Example 20.16.

20.10 Stochastic Volatility Models

Stochastic volatility models are an alternative to GARCH models for modeling conditional heteroscedasticity. In the ARIMA/GARCH models of Chap. 14, there was a single white noise process that drove both the conditional mean and the conditional variance. In contrast, stochastic volatility models use one white noise process to drive the conditional expectation and another to drive the conditional variance. Therefore, stochastic volatility models are more challenging to fit because the unobserved volatility process is driven by its own white noise process, which, of course, is also unobserved; see (20.63) below. Bayesian analysis is particularly good at dealing with unobserved variables and seems the best way to meet the challenge of fitting stochastic volatility models.

We will illustrate stochastic volatility models with the model

$$Y_t = \mu + \sum_{j=1}^{k} \beta_j X_{j,t} + a_t, \tag{20.61}$$

where Y_t is an observed process, e.g., the returns on an asset, and $X_{j,t}$, $j = 1, \ldots, k$, is the jth covariate at time t and could be a lagged value of Y_t. Also, a_t is a weak white noise process with conditional heteroscedasticity. Specifically,

$$a_t = \sqrt{h_t}\epsilon_t \tag{20.62}$$

where $\log(h_t)$ follows the ARMA(p,q) process

$$\log(h_t) = \beta_0 + \sum_{j=1}^{p} \phi_j \log(h_{t-j}) + \sum_{j=1}^{q} \theta_j v_{t-j} + v_t, \tag{20.63}$$

ϵ_t and v_t are mutually independent iid white noises, and $\text{Var}(\epsilon_t) = 1$. Notice from (20.62) that $\sqrt{h_t}$ is the conditional standard deviation of Y_t. As mentioned above, none of the variables in (20.63) are observable.

Example 20.14. Fitting an ARMA(1,1) stochastic volatility model to the S&P 500 stock returns

As an illustration, model (20.61)–(20.63) will be fit to daily S&P 500 log returns from January 2011 through October 2014. Model (20.61) will be used with no covariates so that $Y_t = \mu + a_t$. An ARMA(1,1) stochastic volatility model will be used so that $\log(h_t) = \beta_0 + \phi \log(h_{t-1}) + \theta v_{t-1} + v_t$. In (20.62) ϵ_t has a t-distribution.

A `BUGS` program to fit the stochastic volatility model is below.

```
1  model
2  {
3   for (i in 1:N)
4   {
5     y[i] ~ dt(mu, tau[i], nu)
6    }
7    logh[1] ~ dnorm(0, 1.0E-6)
8    for(i in 2:N)
9    {
10     logh[i] ~ dnorm(beta0 + phi * logh[i-1] + theta * v[i-1], tau_v)
11     v[i] <- logh[i] - beta0 + phi * logh[i-1] + theta * v[i-1]
12   }
13   for (i in 1:N)
14   {
15     tau[i] <- exp(-logh[i])
16     h[i] <- 1/tau[i]
17   }
18   mu ~ dnorm(0.0, 1)
19   beta0 ~ dnorm(0, 0.0001)
20   phi ~ dnorm(0.4, 0.0001)
21   theta ~ dnorm(0, 0.0001)
22   tau_v ~ dgamma(0.01, 0.01)
23   v[1] ~ dnorm(0, 0.001)
24   nu ~ dunif(1,30)
25   sigma_v <- 1 / sqrt(tau_v)
26 }
```

Line 5 specifies the likelihood conditional on h_t. Line 7 gives a prior for the h_1. Lines 8–12 specify model (20.63) starting at $t = 2$ with $p = q = 1$; line 23 gives v_1 a noninformative prior to start this recursion. Lines 18–25 specify diffuse priors on μ, β_0, ϕ, θ, and σ_v.

S&P 500 prices from Jan 3, 2011 to Oct 31, 2014 are in the file `S&P500_new.csv`. There are 964 log returns starting on Jan 4, 2011. The following `R` code computes the log returns and fit the stochastic volatility model to the log returns. The MCMC took about 10 minutes.

```
1  library(rjags)
2  dat = read.csv("S&P500_new.csv")
3  prices = dat$Adj.Close
4  y = diff(log(prices))
5  #####  get initial estimates  #####
6  N = length(y)
7  logy2 = log(y^2)
8  fitar = lm(logy2[2:N] ~ logy2[1:(N - 1)])
9  beta0Init = as.numeric(fitar$coef[1])
10 phiInit = as.numeric(fitar$coef[2])
11 sfitar = summary(fitar)
12 tauInit = 1/sfitar$sigma^2
13 #####  Set up for MCMC  #####
14 N = length(y)
15 data = list(y = y, N = N)
16 inits_stochVol_ARMA11 = function(){list(mu = rnorm(1, mean = mean(y),
17    sd = sd(y) / sqrt(N)), logh = log(y^2),
18    beta0 = runif(1, beta0Init * 1.5, beta0Init/1.5),
19    phi = runif(1,phiInit / 1.5, phiInit * 1.5),
20    tau_v = runif(1, tauInit / 1.5, tauInit * 1.5),
21    theta = runif(1, -0.5, 0.5))}
22 stochVol_ARMA11 <- jags.model("stochVol_ARMA11.bug", data = data,
23    inits = inits_stochVol_ARMA11,
24    n.chains = 3, n.adapt = 1000, quiet = FALSE)
25 nthin = 20
26 stochVol_ARMA.coda = coda.samples(stochVol_ARMA11, c("mu", "beta0",
27    "phi", "theta", "tau_v", "nu", "tau"), 100 * nthin, thin = nthin)
28 summ_stochVol_ARMA11 = summary(stochVol_ARMA.coda)
29 head(summ_stochVol_ARMA11[[1]], 8)
30 tail(summ_stochVol_ARMA11[[1]], 8)
31 dic.stochVol_ARMA11 = dic.samples(stochVol_ARMA11, 100 * nthin,
32    thin = nthin, type = "pD")
33 dic.stochVol_ARMA11
```

Lines 8–12 compute rough initial values for β_0, ϕ, and σ_v by using Y_t^2 a proxy for h_t and regressing $\log(Y_t^2)$ on $\log(Y_{t-1}^2)$.

Since nearly 1000 variables are monitored, we do not want to look at the output for all of them. Instead, the MCMC output is summarized for β_0, μ, ϕ, θ, τ_v, and the first and last few h_i.

```
> head(summ_stochVol_ARMA11[[1]], 8)
                Mean           SD     Naive SE Time-series SE
beta0  -1.327118e+01 6.986557e+00 4.033691e-01   2.956974e+00
mu      8.977085e-04 2.290611e-04 1.322485e-05   1.197846e-05
nu      2.116160e+01 5.785812e+00 3.340440e-01   3.884508e-01
phi     3.651321e-05 2.254858e-02 1.301843e-03   8.364831e-03
tau[1]  2.379568e+05 4.180067e+05 2.413363e+04   2.413332e+04
tau[2]  6.815322e+04 5.764209e+04 3.327968e+03   3.933972e+03
tau[3]  6.472217e+04 5.089951e+04 2.938685e+03   3.247533e+03
```

```
tau[4]  6.030660e+04 4.190914e+04 2.419625e+03   2.815956e+03
> tail(summ_stochVol_ARMA11[[1]], 8)
                Mean           SD     Naive SE Time-series SE
tau[959] 1.328635e+04 7.236487e+03 4.177988e+02   4.432135e+02
tau[960] 1.468882e+04 8.124815e+03 4.690864e+02   5.062925e+02
tau[961] 1.318162e+04 7.091475e+03 4.094265e+02   3.959164e+02
tau[962] 1.514622e+04 8.918848e+03 5.149299e+02   5.163099e+02
tau[963] 1.574039e+04 1.065051e+04 6.149072e+02   6.165450e+02
tau[964] 1.403511e+04 8.598674e+03 4.964447e+02   5.021540e+02
tau_v    5.161879e+00 1.409690e+00 8.138847e-02   1.765265e-01
theta    4.819691e-01 1.219201e-02 7.039058e-04   3.262414e-03

> dic.stochVol_ARMA11
Mean deviance:  -6653
penalty 119.6
Penalized deviance: -6533
```

DIC and pD are called the "penalized deviance" and the "penalty" in the output of `dic.samples()` and in this example are -6536 and 127.3, respectively. □

20.11 Fitting GARCH Models with MCMC

Like stochastic volatility models, GARCH models are easy to fit by MCMC. One reason for fitting a GARCH model with `BUGS` is that this model can then be compared using DIC with a stochastic volatility model that is also fit using `BUGS`.

The following `BUGS` program fits a GARCH(1,1) model with t-distributed noise. This program runs under `JAGS` but crashes with no useful error messages under `OpenBUGS` and `WinBUGS`. The model is

$$\begin{aligned} y_t &= \mu + a_t \\ a_t &= \sqrt{h_t}\epsilon_t \\ h_t &= \alpha_0 + \alpha_1 a_{t-1}^2 + \beta_1 h_{t-1} \\ \epsilon_t &\sim t_\nu(0,1). \end{aligned}$$

```
model{
for (t in 1:N)
{
  y[t] ~ dt(mu, tau[t], nu)
  a[t] <- y[t] - mu
  tau[t] <- 1/h[t]
}
for (t in 2:N)
{
  h[t] <- alpha0 + alpha1 * pow(a[t-1], 2) + beta1 * h[t-1]
```

```
}
mu ~ dnorm(0, 0.001)
h[1] ~ dunif(0, 0.0012)
alpha0 ~ dunif(0, 0.2)
alpha1 ~ dunif(0.00001, 0.8)
beta0 ~ dunif(0.00001, 0.8)
nu ~ dunif(1,30)
}
```

Example 20.15. Fitting a GARCH(1,1) model to the S&P 500 stock returns

The following R code fits a GARCH(1,1) model to the data in Example 20.14.

```
library(rjags)
dat = read.csv("S&P500_new.csv")
prices = dat$Adj.Close
y = diff(log(prices))
N = length(y)
data = list(y = y, N = N)
inits_garch11 = function(){list(alpha0 = runif(1, 0.001, 0.25),
    beta1 = runif(1, 0.001, 0.25), mu = runif(1, 0.001, 0.25),
    alpha1 = runif(1, 0.001, 0.25), nu = runif(1, 2, 10))}
garch11 <- jags.model("garch11.bug", data=data,
    inits = inits_garch11,
    n.chains = 3, n.adapt = 1000, quiet = FALSE)
nthin = 20
garch11.coda = coda.samples(garch11,c("mu", "beta1", "alpha0",
   "alpha1", "nu", "tau"), 100*nthin, thin = nthin)
dic.garch11 = dic.samples(garch11, 100*nthin, thin = nthin)
dic.garch11
diffdic(dic.garch11, dic.stochVol_ARMA11)
summ_garch11 = summary(garch11.coda)
head(summ_garch11[[1]])
tail(summ_garch11[[1]])
```

The output is below.

```
> dic.garch11
Mean deviance:  -6508
penalty 5.737
Penalized deviance: -6502
> diffdic(dic.garch11, dic.stochVol_ARMA11)
Difference: 30.90573
Sample standard error: 15.12843
> head(summ_garch11[[1]])
               Mean           SD     Naive SE Time-series SE
alpha0 3.875413e-06 9.405668e-07 5.430365e-08   6.060383e-08
alpha1 1.287614e-01 2.170856e-02 1.253344e-03   1.465383e-03
beta1  7.677071e-01 2.621658e-02 1.513615e-03   1.870831e-03
mu     8.663882e-04 2.290809e-04 1.322599e-05   1.247927e-05
```

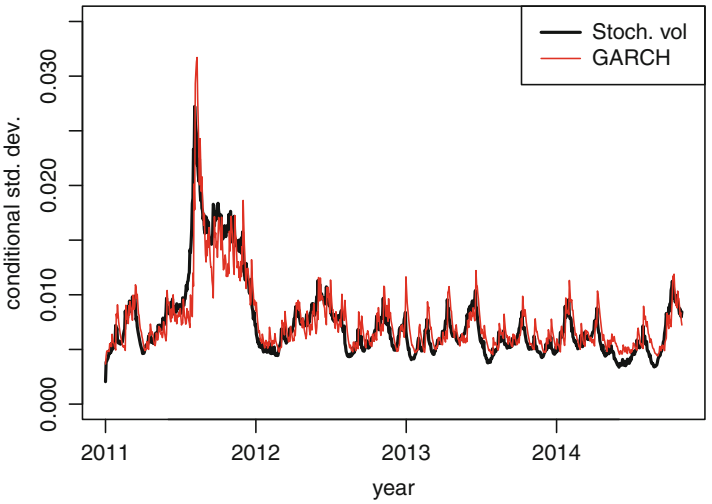

Fig. 20.8. *The conditional standard deviations of the log returns estimated by the stochastic volatility model in Example 20.14 and the GARCH(1,1) model.*

```
nu      7.570399e+00 1.990408e+00 1.149163e-01    1.077795e-01
tau[1] 7.432585e+04 1.242142e+05 7.171511e+03    7.376238e+03
> tail(summ_garch11[[1]])
              Mean       SD  Naive SE Time-series SE
tau[959] 10765.65 1050.613  60.65714       56.26545
tau[960] 12495.45 1207.416  69.71019       64.84224
tau[961] 15145.85 1500.491  86.63091       81.57595
tau[962] 14266.32 1339.490  77.33549       72.44932
tau[963] 17144.03 1653.541  95.46724       90.31518
tau[964] 19112.62 1869.545 107.93821      107.54300
```

Line 18 of the `R` code uses the function `diffdic()` in the `rjags` package to compare the stochastic volatility and GARCH models by DIC. In the output from `diffdic()`, we see that DIC for the GARCH model is larger than for the stochastic volatility model; the difference between the two DIC values is 30.9 but with a standard error of 15.1. Figure 20.8 compares the estimates of the conditional standard deviations of the log returns by the two models. Figure 20.9 compares the ACF's of the squared standardized residuals[4] from the two models and also assuming that the log returns are i.i.d. Despite the large difference in DIC, the two models are about equally successful in modeling the conditional heteroscedasticity.

□

[4] A residual is standardized by dividing it by its conditional standard deviation.

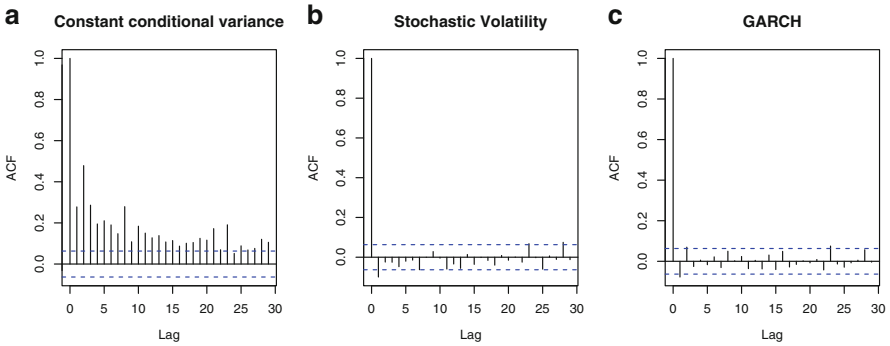

Fig. 20.9. *ACF's of the squared standardized residuals. (a) Assuming a constant conditional standard deviation. The model is* $R_t = \mu + \epsilon_t$ *where* ϵ_t *is independent white noise. (b) Assuming an ARMA(1,1) stochastic volatility model. (c) Assuming a GARCH(1,1) model.*

20.12 Fitting a Factor Model

Factor models can be fit easily using `JAGS`. An advantage of a Bayesian analysis of a factor model is that a hierarchial model can be used to shrink the betas towards each other.

Example 20.16. Fitting a one factor model to stock returns

In this example, the factor will be the returns on the S&P 500 so this is a Bayesian version of the CAPM. The model that will be used is

$$R_{j,t} = \beta_j R_{M,t} + \epsilon_{j,t}.$$

where, as in Chap. 17, $R_{j,t}$ is the return on the j stock at time t, $j = 1, \ldots, 10$, and $R_{M,t}$ is the return on the S&P 500 at time t. It is assumed that for $j = 1, \ldots, 10$, $\{\epsilon_{j,t},\ t = 1, \ldots\}$ are mutually independent i.i.d. white noise processes with $\text{var}(\epsilon_{j,t}) = \sigma^2_{\epsilon,j}$. For simplicity we have assumed that all the alphas are zero.

We will put a hierarchical on $\beta_1, \ldots, \beta_{10}$, specifically

$$\beta_j \sim N(\mu_\beta, \sigma^2_\beta),$$

with non-informative priors on μ_β and σ^2_β.

The `BUGS` program is below. At line 11, μ_β is called `meanbeta` and σ_β^{-2} is called `taubeta`. Also, `tauepsilon` on line 6 is σ_ϵ^{-2}.

```
1 model{
2 for (t in 1:N)
3   {
4   for (j in 1:m)
```

```
    {
        R[t,j] ~ dnorm(beta[j]*mkt[t], tauepsilon[j])
     }
    }
for (j in 1:m)
{
  beta[j] ~ dnorm(meanbeta, taubeta)
  tauy[j] ~ dgamma(0.1, 0.001)
}
meanbeta ~ dnorm(1, 0.000001)
taubeta ~ dunif(1, 100)
}
```

This example is a continuation of Example 16.11 in that it uses the stock price data in the file `Stock_Bond.csv`. Since there are nearly 5000 days of returns, one can create 20 blocks of returns, each block with 250 days except that the last block would be somewhat short of 250. Each block can be used for training (parameter estimation) and the next block for testing.

The `R` program below illustrates this strategy using only the first two blocks, the first as training data and the second as test data. During training, the optimal allocation vector $\boldsymbol{w}$ is estimated. Here "optimal" means maximizing the expected utility of the returns using the utility function

$$U(R;\lambda) = 1 - \exp\{-\lambda(1+R)\}. \tag{20.64}$$

The value of λ is set equal to 3 at line 38 of the `R` code. A value of λ in the range 2 to 8 is reasonable since these are daily returns and typically in the range ± 0.05. Also, in (20.64) we are implicitly assuming that the initial wealth is equal to 1, since the initial wealth is not explicitly included there.

The first part of the `R` program fits the factor model:

```
library(rjags)
dat = read.csv("Stock_Bond.csv")
y = dat[, c(3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 22)]
n = dim(y)[1]
m = dim(y)[2] - 1
r = y[-1, ] / y[-n, ] - 1
k1 = 250
k2 = k1 + 250
rtrain = r[1:k1, 1:m]
mkt_train = r[1:k1, 11]
rtest = r[(k1+1):k2, 1:m]
data = list(R = rtrain, N = k1, mkt = mkt_train, m = m)
inits.Capm = function(){list(beta = rep(1,m))}
Capm.jags <- jags.model("BayesCapm.bug.R", data = data,
   inits = inits.Capm, n.chains = 1, n.adapt = 1000, quiet = FALSE)
nthin = 10
N = 500
Capm.coda = coda.samples(Capm.jags,
```

```
    c("beta", "tauepsilon", "taubeta"),
    N * nthin, thin = nthin)
MCMC_out = Capm.coda[[1]]
summ = as.matrix(summary(Capm.coda)[[1]][,1])
beta = summ[1:10]
taubeta = summ[11]
tauy = summ[12:21]
sigmaepsilon = tauepsilon^(-.5)
```

The next section of R code defines the utility function and finds the optimal allocation vector $\boldsymbol{w}$ by using quadratic programming as in Example 16.11. The vector $\boldsymbol{w}$ is found twice, once using the sample mean vector and covariance matrix of the training sample returns to estimate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in Eq. (16.20) (model-free) and once using the factor model to estimate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ (CAPM).

```
ExUtil = function(w)
{
  -1 + exp(-lambda * (1 + t(w) %*% mu) +
  lambda^2 * t(w) %*% Omega %*% w / 2 )
}

mu_model_free = colMeans(rtrain)
Omega_model_free = cov(rtrain)
mu_Capm = beta * mean(mkt)
Omega_Capm = beta %o% beta * var(mkt_train) + diag(sigmaepsilon^2)

lambda = 3
library(quadprog)
mu = mu_model_free
Omega = Omega_model_free
opt1 = solve.QP(Dmat = as.matrix(lambda^2 * Omega),
   dvec = lambda * mu, Amat = as.matrix(rep(1,10)),
   bvec = 1, meq = 1)
w_model_free = opt1$solution

mu = mu_Capm
Omega = Omega_Capm
opt2 = solve.QP(Dmat = as.matrix(lambda^2 * Omega),
   dvec = lambda * mu, Amat = as.matrix(rep(1,10)),
   bvec = 1, meq = 1)
w_Capm = opt2$solution
```

Next, the utility of the portfolio's returns is averaged over the test data using the model-free and CAPM based estimates of the optimal portfolio. Also, the mean and standard deviations of the portfolio's returns on the test data are computed.

```
return_model_free = as.matrix(rtest) %*% w_model_free
ExUt_model_free = mean(1 - exp(-lambda * return_model_free))

```

```
return_Capm = as.matrix(rtest) %*% w_Capm
ExUt_Capm = mean(1 - exp(-lambda * return_Capm))

print(c(ExUt_model_free, ExUt_Capm), digits = 2)
print(c(mean(return_model_free), mean(return_Capm)), digits = 2)
print(c(sd(return_model_free), sd(return_Capm)), digits = 2)
```

The output is below. We see that the portfolio selected using the CAPM estimates outperformed the portfolio based on the sample mean vector and covariance matrix. Interestingly, the CAPM selected portfolio not only has a higher average utility, but it also has both a higher mean and a lower standard deviation of the returns compared to the model-free estimates. Usually a higher expected return comes with higher risk (larger standard deviation), but a better estimate can achieve a higher return without higher risk.

```
> print(c(ExUt_model_free, ExUt_Capm), digits = 2)
[1] -0.0179  0.0023
> print(c(mean(return_model_free), mean(return_Capm)), digits = 2)
[1] 0.00060 0.00099
> print(c(sd(return_model_free), sd(return_Capm)), digits = 2)
[1] 0.067 0.012
```

These results are based on only one block of training data and one block of test data, and by themselves they are not a convincing demonstration of the superiority of CAPM estimates. The analysis could be continued using all twenty blocks of data; see Exercise 4.

Figure 20.10 plots the allocation vectors, $\boldsymbol{w}$, using model-free and CAPM-based estimators. The model-free $\boldsymbol{w}$ oscillates widely and has substantial short-selling. In contrast, the CAPM-based $\boldsymbol{w}$ is much closer to assigning equal weights to the 10 stocks and has minimal short selling.

Another issue is how hierarchical Bayes estimation of the CAPM compares with ordinary least-squares estimation. Figure 20.11 plots the least-squares estimates of $\boldsymbol{\beta}$ versus the Bayes estimates. With the exception of the largest beta, the Bayes estimates are only slightly shrunk together and are similar to the least-squares estimates. This suggest that the Bayes estimates will, at best, be only a moderate improvement over least-squares in this example. Most of the improvement is due to using the CAPM to estimate the expected returns. □

20.13 Sampling a Stationary Process

This section provides the theory behind the statistics B, W, and $\widehat{\text{var}}^+(\psi|\boldsymbol{Y})$ used in Sect. 20.7.5 to monitor MCMC convergence and mixing.

Suppose that $Y_1, Y_2, \ldots, Y_n$ is a sample from a stationary process with mean μ and autocovariance function $\gamma(h)$. Let $\overline{Y} = n^{-1} \sum_{i=1}^{n} Y_i$ be the sample mean. Then

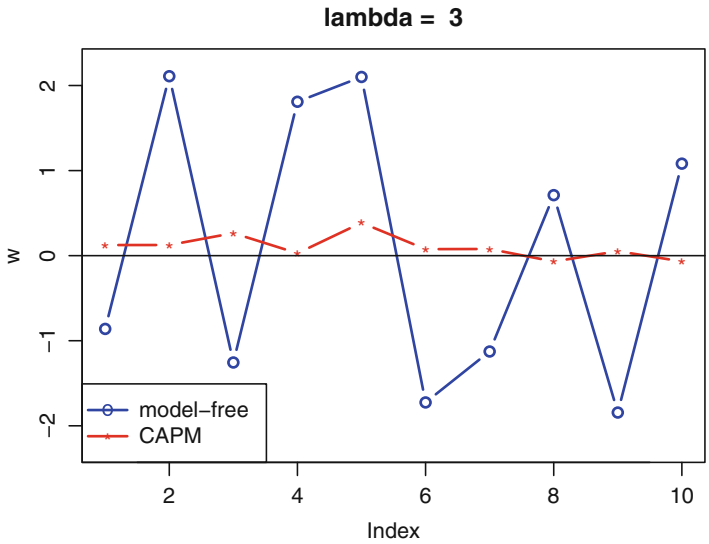

Fig. 20.10. *Allocation (or weight) vectors using the sample mean vector and covariance matrix (model-free) and CAPM-based estimates.*

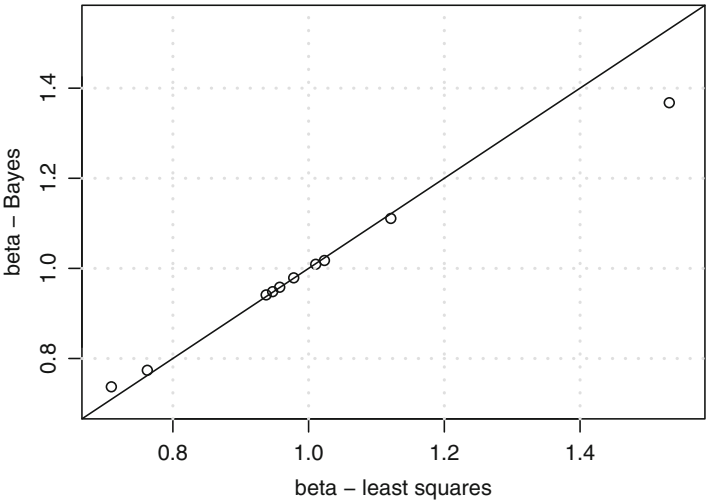

Fig. 20.11. *Plot of the Bayes estimator of β versus the least-squares estimator.*

$$\begin{aligned}\text{var}(\overline{Y}) &= n^{-2}\sum_{i=1}^{n}\sum_{j=1}^{n}\text{Cov}(Y_i, Y_j)\\ &= n^{-2}\sum_{i=1}^{n}\sum_{j=1}^{n}\gamma(i-j)\\ &= n^{-2}\left\{n\gamma(0) + 2\sum_{h=1}^{n-1}\gamma(h)(n-h)\right\}\\ &= \frac{\gamma(0)}{n}R_n, \end{aligned} \tag{20.65}$$

where $R_n = \left\{1 + 2\sum_{h=1}^{n-1}\rho(h)\left(1 - \frac{h}{n}\right)\right\}$. If $Y_1, Y_2, \ldots, Y_n$ is an uncorrelated process (white noise), then $R_n = 1$ and (20.65) agrees with (7.13).

Most stationary processes generated by MCMC have $\rho(h) \geq 0$ for all h so that R_n is inflated by the autocorrelation. The inflation can be severe. Consider the case of a stationary AR(1) process, $Y_n = \phi Y_{n_1} + \epsilon_i$. AR(1) processes often are reasonably good approximations to MCMC processes. For an AR(1) we can approximate R_n:

$$R_n \approx \left\{1 + 2\sum_{h=1}^{\infty}\rho(h)\right\} = \left\{2\sum_{h=0}^{\infty}\phi^h - 1\right\} = \left(\frac{2}{1-\phi} - 1\right) = \frac{1+\phi}{1-\phi}, \tag{20.66}$$

where we have used summation formula for geometric series (3.4) with $T = \infty$. Notice that the right-hand side of (20.66) increases without bound as $\phi \to 1$.

From the identity

$$\sum_{i=1}^{n}(Y_i - \mu)^2 = \sum_{i=1}^{n}(Y_i - \overline{Y})^2 + n(\overline{Y} - \mu)^2,$$

we obtain

$$E\left\{\sum_{i=1}^{n}(Y_i - \overline{Y})^2\right\} = \gamma(0)(n - R_n) \tag{20.67}$$

since $\gamma(0) = E\left\{(Y_i - \mu)^2\right\}$ and $\gamma(0)R_n = E\left\{n(\overline{Y} - \mu)^2\right\}$ by definitions. Therefore, an unbiased estimate of the process variance $\gamma(0)$ is

$$\widehat{\gamma}(0) = \frac{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}{n - R_n}. \tag{20.68}$$

When the process is uncorrelated so that $R_n = 1$, the right-hand side of (20.68) is the sample variance (A.7). For positively autocorrelated processes, $R_n > 1$ and the sample variance (which uses 1 in place of R_n) is biased downward.

To obtain an unbiased estimate of $\gamma(0)$, one can use

$$\frac{\sum_{i=1}^{n}(Y_i - \overline{Y})^2 + \widehat{\gamma(0)R_n}}{n}, \tag{20.69}$$

where $\widehat{\gamma(0)R_n}$ is an unbiased estimator of $\gamma(0)R_n$. There are several methods for estimating $\gamma(0)R_n$. The simplest uses several independent realizations of the process. Let $\overline{Y}_1, \ldots, \overline{Y}_M$ be the means of M independent realizations of the process and let $\overline{Y} = M^{-1} \sum_{j=1}^{m} \overline{Y}_j$. Then

$$\widehat{\gamma(0)R_n} = \frac{\sum_{j=1}^{M} (\overline{Y}_j - \overline{Y})^2}{M - 1} \tag{20.70}$$

is an unbiased estimator of $\gamma(0)R_n$. The statistic $\widehat{\text{var}}^{+}(\psi|\boldsymbol{Y})$ used in Sect. 20.7.5 for MCMC monitoring is a special case of (20.68) and (20.70).

20.14 Bibliographic Notes

There are many excellent books on Bayesian statistics. Gelman, Carlin, Stern, Dunson, Vehtari, and Rubin (2013) and Carlin and Louis (2008) are introductions to Bayesian statistics written at about the same mathematical level as this book. Box and Tiao (1973) is a classic work on Bayesian statistics with a wealth of examples and still worth reading despite its age. Berger (1985) is a standard reference on Bayesian analysis and decision theory. Bernardo and Smith (1994) and Robert (2007) are more recent books on Bayesian theory. Rachev, Hsu, Bagasheva, and Fabozzi (2008) covers many applications of Bayesian statistics to finance.

Albert (2007) is an excellent introduction to Bayesian computations in `R`. Chib and Greenberg (1995) explain how the Metropolis–Hastings algorithm works and why its stationary distribution is the posterior. Congdon (2001, 2003) covers the more recent developments in Bayesian computing with an emphasis on OpenBUGS software. There are other Bayesian Monte Carlo samplers besides MCMC, for example, importance sampling. Robert and Casella (2005) discuss these as well as MCMC. Gelman et al., (2013) have examples of Bayesian computations in `R` and `OpenBUGS` in an appendix. Lunn, Thomas, Best, and Spiegelhalter (2000) describe the design of OpenBUGS. Lunn, Jackson, Best, and Spiegelhalter (2013) is a comprehensive introduction to BUGS.

The diagnostics $\widehat{R}$ and N_{eff} are due to Gelman and Rubin (1992) though Sect. 20.7.5 uses the somewhat different notation of Gelman et al., (2013). Spiegelhalter, Best, Carlin, and van der Linde (2002) proposed DIC and p_D. The Gelman plot was introduced by Brooks and Gelman (1998). Kass et al. (1998) discuss their practical experiences with MCMC.

Bayesian modeling of yield curves models is discussed by Chib and Ergashev (2009). Bayesian time series are discussed by Albert and Chib (1993), Chib and Greenberg (1994), and Kim, Shephard, and Chib (1998); the first two papers cover ARMA process and the last covers ARCH and stochastic volatility models. There is a vast literature on the important and difficult problem of Bayesian estimation of covariance matrices with nonconjugate priors. Daniels and Kass (1999) review some of the literature in addition to providing their own suggestions.

We have not discussed empirical Bayes inference, but Carlin and Louis (2000) can be consulted for an introduction to that literature. Empirical Bayes inference uses a hierarchical prior but estimates the parameters in the lower level in a non-Bayesian manner and then, treating those parameter as known and fixed, performs a Bayesian analysis. The result is shrinkage estimation much like that achieved by a Bayesian analysis. The advantage of an empirical Bayes analysis is that it can be somewhat simpler than a fully Bayesian analysis. The disadvantage is that it underestimates uncertainty because estimated parameters in the prior are treated as if they were known. There are shrinkage estimators that are not exactly Bayesian or even empirical Bayes procedures. Ledoit and Wolf (2003) propose a shrinkage estimator for the covariance matrix of stock returns. Their shrinkage target is an estimate from a factor model, for example, the CAPM. Shrinkage estimation goes back at least to Stein (1956) and is often called Stein estimation.

The central limit theorem for the posterior is discussed by Gelman et al. (2013), Lehmann (1983), and van der Vaart (1998), in increasing order of technical level.

See Greyserman, Jones, and Strawderman (2006) for more information on portfolio selection by Bayesian methods.

20.15 R Lab

20.15.1 Fitting a *t*-Distribution by MCMC

In this section of the lab, you will fit the t-distribution to monthly returns on IBM using `JAGS` to estimate the posterior distribution by MCMC sampling.

Run the following `R` code to load the `rjags` package, input the data, and prepare the data for use by `JAGS`.

```
library(rjags)
data(CRSPmon, package = "Ecdat")
ibm = CRSPmon[ , 2]
r = ibm
N = length(r)
ibm_data = list(r = r, N = N)
```

Next, put the following `BUGS` code in a text file. I will assume that you name this file `univt.bug`, though you can use another name provided you make appropriate changes in the `R` code that follows. `BUGS` code is somewhat similar to, but not the same as, `R` code. For example, in `R` "`dt`" is the t-density, but in `BUGS` it is the t-distribution.

```
model{
for (t in 1:N)
{
```

```
   r[t] ~ dt(mu, tau, k)
}
mu ~ dnorm(0.0, 1.0E-6)
tau ~ dgamma(0.1, 0.01)
nu ~ dunif(2, 50)
sigma2 <- (k / (k - 2)) / tau
sigma <- sqrt(sigma2)
}
```

BUGS programs are difficult to debug, so be careful to enter the code exactly as it appears here. It has been tested and runs as written, but any error will cause problems. Our experience is that JAGS is better at providing error messages and easier to debug than WinBUGS and OpenBUGS.

The BUGS code above provides a description of the statistical model and specifies the prior distributions. The model states that the data are i.i.d. from a t-distribution. The ~ symbol assigns a distribution to a random variable so `y[i] ~ dt(mu, tau, k)` gives the likelihood of the data. Here `mu`, `tau`, and `k` are the mean, precision, and degrees of freedom, respectively, of the t-distribution. For a t-distribution, the precision is $\tau = 1/\lambda^2$ where λ is the scale parameter. Also, `mu ~ dnorm(0.0, 1.0E-6)` specifies the prior for the mean `mu` to be normal with mean 0 and precision 1.0E-6. The precision of a normal distribution is the reciprocal of its variance, so here the prior variance of μ is 1.0E6.

The symbol <- is used to assign a value (rather than a distribution) to a variable. Thus, `sigma <- 1/sqrt(tau)` makes `sigma` the scale parameter of the t-distribution of the data. In R, "=" can be used in place of "<-" for assigning a value to a variable, but this is not true in BUGS. The parameter `sigma` is not needed, but, by defining this variable in the BUGS program, we generate a sample from its posterior distribution.

Next, run the following R code that defines a function `inits()`. This function is used to generate random starting values for the chains.

```
inits = function(){list(mu = rnorm(1, 0, 0.3),
   tau = runif(1, 1, 10), k = runif(1, 1, 30))}
```

The next code uses the `jags.model()` and `coda.samples()` functions in the `rjags` package. Notice that the arguments specify the data, the function to create initial values of the chains, the file containing the BUGS program, the parameters to be monitored and returned, the number of chains, the number of iterations per chain, the number of iterations to discard as burn-in, the amount of thinning.

```
univ_t <- jags.model("univt.bug", data = ibm_data,
  inits = inits, n.chains = 3, n.adapt = 1000, quiet = FALSE)
nthin = 2
univ_t.coda = coda.samples(univ_t, c("mu", "tau", "k",
   "sigma"), n.iter = 500 * nthin, thin = nthin)
```

Next, print and plot the results.

```
summary(univ_t.coda)
effectiveSize(univ_t.coda)
gelman.diag(univ_t.coda)
```

Problem 1

(a) Which parameter mixes best according to N_{eff} in the output?
(b) Which parameter mixes worst according to N_{eff} in the output?
(c) Give a 95 % posterior interval for the degrees-of-freedom parameter.

Next, plot the results to check for convergence to the stationary distribution (posterior distribution) using Gelman plots and trace plots.

```
gelman.plot(univ_t.coda)
par(mfrow = c(2, 2))
traceplot(univ_t.coda)
```

Plotting the ACFs gives much insight into how well the chains are mixing. The less autocorrelation, the better. The function `autocorr.plot()` plots ACFs separately for each chain.

```
par(mfrow = c(2, 2))
autocorr.plot(univ_t.coda, auto.layout = FALSE)
```

Problem 2

(a) Which parameter mixes best and which mixes worse according to the ACF plots? Explain your answers.
(b) Find the posterior skewness and kurtosis of the degrees of freedom parameter.

The function `densityplot()` gives kernel density estimate from each chain.

```
library(lattice)
densityplot(univ_t.coda)
```

Problem 3 *Which posterior densities are most skewed?*

The kurtosis of a t-distribution is $3(\nu-2)/(\nu-4)$ if $\nu > 4$ and is $+\infty$ if $\nu \leq 4$. Variables in `R` can have infinite values: `Inf` is $+\infty$ and `-Inf` is $-\infty$, so `R` can handle infinite values of kurtosis if they occur.

Problem 4 *Write* R *code to compute 1500 MCMC values of the kurtosis. (1500 = 3*2000/2; there are 3 chains of length 2000 after burn-in and they are thinned to every 2nd iteration.)*

(a) Find the 0.01, 0.05, 0.25, 0.5, 0.75, 0.95, and 0.99 quantiles of the posterior distribution of the kurtosis of IBM returns.
(b) Estimate the posterior probability that the kurtosis of the distribution of IBM returns is finite.
(c) Compute the 0.01, 0.05, 0.25, 0.5, 0.75, 0.95, and 0.99 quantiles of the bootstrap distribution of the sample kurtosis of IBM. Take 1000 resamples using both a model-free and a model-based bootstrap. Compare the two sets of bootstrap quantiles with the posterior quantiles in (a).
(d) Compare 90 % bootstrap basic percentile confidence intervals for the kurtosis with the 90 % posterior interval. Which interval is shortest? Why might it be shortest?

20.15.2 AR Models

In this section of the lab, you will fit an AR(1) model to the changes in the log of the GDP. First, run the following code to process the data. Notice that the log-GDP time series is differenced before fitting.

```
library(rjags)
data(Tbrate, package = "Ecdat")
#  r = the 91-day treasury bill rate
#  y = the log of real GDP
#  pi = the inflation rate
del_dat = diff(Tbrate)
y = del_dat[,2]
N = length(y)
GDP_data=list(y = y, N = N)
```

Next create a file called `ar1.bug` containing the following WinBUGS code.

```
model{
for(i in 2:N){
   y[i] ~ dnorm(mu + phi * (y[i-1] - mu), tau)
}
mu ~ dnorm(0, 0.00001)
phi ~ dnorm(0, 0.00001)
tau ~ dgamma(0.1, 0.0001)
sigma <- 1/sqrt(tau)
}
```

Finally, run the following code to fit an AR(1) model using `JAGS` and also using R's `arima()` function to compute the MLE, which will be compared with the Bayes estimator.

```
inits = function(){list(mu = rnorm(1, 0, 2 * sd(y) / sqrt(N)),
   phi = rnorm(1, 0, 0.3), tau = runif(1, 1, 10))}
ar1 <- jags.model("ar1.bug", data = GDP_data, inits = inits,
   n.chains = 3, n.adapt = 1000, quiet = FALSE)
nthin = 20
ar1.coda = coda.samples(ar1, c("mu", "phi", "sigma"),
   n.iter = 500 * nthin, thin = nthin)
summary(ar1.coda, digits = 3)
arima(y, order = c(1, 0, 0))
```

Problem 5 *Construct time series and ACF plots of the parameters* `phi` *and* `sigma`.

(a) Do you believe that the MCMC sample size is adequate? Why or why not? Is the burn-in iterations adequate? Why or why not? If you feel that either the number of iterations or the length of the burn-in period is inadequate, then rerun with a larger burn-in period and/or MCMC sample size.

(b) Compute the MLEs for this model using `arima()`*. How closely do the Bayes estimates and MLEs agree? Could you explain any possible disagreement?*

(c) The model in the `BUGS` *program does not assume that the time series is in its stationary distribution. In fact, the model does not even assume that there is a stationary distribution. Explain why.*

(d) Modify the `BUGS` *program to utilize the marginal distribution of* y_1*, assuming that the process starts in its stationary distribution.*

20.15.3 MA Models

Next you will fit an MA(1) to simulated data. The function `arima.sim()` is used to create the data.

```
library(rjags)
set.seed(5640)
N = 600
y = arima.sim(n = N, list(ma = -0.5), sd = 0.4)
y = as.numeric(y) + 3
q = 5
ma.sim_data = list(y = y, N = N, q = q)
inits.ma = function(){list(mu = rnorm(1, mean(y), 2*sd(y)/sqrt(N)),
   theta = rnorm(1, -0.05, 0.1), tau = runif(1, 5, 8))}
```

Put the following `BUGS` program in the file `ma1.bug`. This program not only fits the MA(1) model but also predicts `q` steps ahead; `q` is an input parameter chosen by the user and, from the viewpoint of `BUGS`, `q` is part of the data and is set equal to 5 in the code above. The predicted values will be included in the output and called `ypred`.

```
model{
for (i in 2:N)
{
   w[i] <- y[i] - mu - theta * w[i-1]
}
w[1] ~ dnorm(0, 0.01)
for (i in 2:N)
{
   y[i] ~ dnorm(mu + theta * w[i-1], tau)
}
mu ~ dnorm(0, 0.0001)
theta ~ dnorm(0, 0.0001)
tau ~ dgamma(0.01, 0.0001)
sigma <- 1/sqrt(tau)
for (i in 1:q)
{
   ypred[i] ~ dnorm(theta * w[N + i - 1], tau)
   w[i + N] <- ypred[i] - theta * w[N + i - 1]
}
}
```

Now run this R code.

```
ma1 <- jags.model("ma1.bug", data = ma.sim_data, inits = inits.ma,
   n.chains = 3, n.adapt = 1000, quiet = FALSE)
nthin = 5
ma1.coda = coda.samples(ma1, c("mu", "theta", "sigma", "ypred"),
   n.iter = 500 * nthin, thin = nthin)
summary(ma1.coda)
```

Problem 6

(a) Do you believe that the MCMC sample size is adequate? Why or why not? If you feel it is inadequate, than rerun JAGS *with a larger MCMC sample size. Is the length of the burn-in periods adequate?*

(b) Construct time series and ACF plots of the parameters theta, sigma, ypred[1], *and* ypred[2]. *What do the plots tell us about MCMC mixing and convergence?*

(c) Find a 90 % posterior interval for the next observation after the observed data.

20.15.4 ARMA Models

Create a simulated sample from an ARMA(1,1) process with the following R code.

```
set.seed(5640)
N = 600
```

```
y = arima.sim(n = N, list(ar = 0.9, ma = -0.5), sd = 0.4)
y = as.numeric(y)
```

Problem 7 *Create* `BUGS` *and* `R` *code to fit the ARMA(1,1) model to the simulated data. Monitor the result to make certain that the MCMC sample size is large enough.*

(a) Discuss how well the chains mix and whether the Monte Carlo sample size is adequate.
(b) Find 99 % posterior intervals for the AR and MA parameters.

20.16 Exercises

1. Show in Example 20.2 that the MAP estimator is 6/7.
2. Verify (20.26).
3. In the derivation of (20.51), it was stated that "$\{\overline{Y} - E(\mu|\boldsymbol{Y})\}$ and $\{E(\mu|\boldsymbol{Y}) - \mu\}$ are conditionally uncorrelated given $\boldsymbol{Y}$." Verify this statement.
4. Continue the analysis in Example 20.16. Divide the data into 20 blocks of 250 days each, except that the last block will have only 212 days. Use each of the first 19 blocks as training data with the subsequent block as test data. How does portfolio selection based on the CAPM compare with model-free estimation when averaged over the 19 pairs of training and test data sets?
5. One of the strength of fitting models by MCMC using `BUGS` is that a very wide range of models can be fit. As an example, in this exercise a regression model with MA(1) errors will be used. For data, use the first 1500 returns[5] on GM and on the S&P 500 index in the data set `Stock_Bond.csv`. Fit the model
$$R_t = \beta R_{M,t} + \epsilon_t + \theta\epsilon_{t-1}.$$
where R_t is the tth return on GM, $R_{M,t}$ is the tth return on the S&P 500, and $\epsilon_1, \ldots, \epsilon_{1500}$ are i.i.d. $N(0, \sigma_\epsilon^2)$. Use non-informative priors on β, θ, and σ_ϵ^2.
6. Expand the model in Exercise 5 so that $\epsilon_1, \ldots$ is a GARCH(1,1) process. Revise the `BUGS` and `R` code of Exercise 5 to fit this expanded model.
7. So far we have treated the sample mean vector and covariance matrix as fixed when considering the risk of a portfolio. Stated differently, estimation risk has been ignored. A methodology for taking risk due to estimation error into account was proposed by Greyserman, Jones, and Strawderman (2006). Assume that the vector of returns $\boldsymbol{R}$ is $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ distributed. Let $(\boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)})$, $k = 1, \ldots, K$, be an MCMC sample from the posterior distribution of $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. For each k, let $R^{(k)}$ be $N(\boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)})$ distributed. Then

[5] `JAGS` had trouble when the full data set was used, probably because there are nearly 5000 latent variables. This problem is likely hardware dependent.

$\boldsymbol{R}^{(1)}, \ldots, \boldsymbol{R}^{(K)}$ is a sample from the posterior predictive distribution of $\boldsymbol{R}$ and take uncertainty about $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ into account and

$$K^{-1} \sum_{k=1}^{K} U\{X_0(1 + \boldsymbol{w}^\mathsf{T} \boldsymbol{R}^{(k)})\} \tag{20.71}$$

estimates the expected utility if the allocation vector is $\boldsymbol{w}$. Here, as in Sect. 16.8, X_0 is the initial wealth and U is the utility function. Continue the analysis in Example 20.16 using the CAPM model and maximize (20.71). Maximizing (20.71) is a nonlinear optimization problem, so a good starting value is essential. As a starting value, use the $\boldsymbol{w}$ found in Example 20.16 that ignores estimation error.

References

Albert, J. (2007) *Bayesian Computation with R*, Springer, New York.

Albert, J. H. and Chib, S. (1993) Bayes inference via Gibbs sampling of autoregressive time series subject to Markov mean and variance shifts, *Journal of Business & Economic Statistics*, 11, 1–15.

Berger, J. O. (1985) *Statistical Decision Theory and Bayesian Analysis* 2nd ed., Springer-Verlag, Berlin.

Bernardo, J. M., and Smith, A. F. M. (1994) *Bayesian Theory*, Wiley, Chichester.

Box, G. E. P., and Tiao, G. C. (1973) *Bayesian Inference in Statistical Analysis*, Addison-Wesley, Reading, MA.

Brooks, S. P. and Gelman, A. (1998) General Methods for Monitoring Convergence of Iterative Simulations. *Journal of Computational and Graphical Statistics*, 7, 434–455.

Carlin, B. P., and Louis, T. A. (2000) Empirical Bayes: Past, present and future. *Journal of the American Statistical Association*, **95**, 1286–1289.

Carlin, B. , and Louis, T. A. (2008) *Bayesian Methods for Data Analysis*, 3rd ed., Chapman & Hall, New York.

Chib, S., and Ergashev, B. (2009) Analysis of multifactor affine yield curve models. *Journal of the American Statistical Association*, **104**, 1324–1337.

Chib, S., and Greenberg, E. (1994) Bayes inference in regression models with ARMA(p, q) errors. *Journal of Econometrics*, **64**, 183–206.

Chib, S., and Greenberg, E. (1995) Understanding the Metropolis–Hastings algorithm. *American Statistician*, **49**, 327–335.

Congdon, P. (2001) *Bayesian Statistical Modelling*, Wiley, Chichester.

Congdon, P. (2003) *Applied Bayesian Modelling*, Wiley, Chichester.

Daniels, M. J., and Kass, R. E. (1999) Nonconjugate Bayesian estimation of covariance matrices and its use in hierarchical models. *Journal of the American Statistical Association*, **94**, 1254–1263.

Edwards, W. (1982) Conservatism in human information processing. In *Judgement Under Uncertainty: Heuristics and Biases*, D. Kahneman, P. Slovic, and A. Tversky, ed., Cambridge University Press, New York.

Gelman, A., and Rubin, D. B. (1992) Inference from iterative simulation using multiple sequence (with discussion). *Statistical Science*, **7**, 457–511.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013) *Bayesian Data Analysis*, 3rd ed., Chapman & Hall, London.

Greyserman, A., Jones, D. H., and Strawderman, W. E. (2006) Portfolio selection using hierarchical Bayesian analysis and MCMC methods, *Journal of Banking and Finance*, 30, 669–678.

Kass, R. E., Carlin, B. P., Gelman, A., and Neal, R. (1998) Markov chain Monte Carlo in practice: A roundtable discussion. *American Statistician*, **52**, 93–100.

Kim, S., Shephard, N., and Chib, S. (1998) Stochastic volatility: likelihood inference and comparison with ARCH models.*Review of Economic Studies*, **65**, 361–393.

Ledoit, O., and Wolf, M. (2003) Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, **10**, 603–621.

Lehmann, E. L. (1983) *Theory of Point Estimation*, Wiley, New York.

Lunn, D., Jackson, C., Best, N., Thomas, A., and Spiegelhalter, D. (2013) *The BUGS Book*, Chapman & Hall.

Lunn, D. J., Thomas, A., Best, N., and Spiegelhalter, D. (2000) OpenBUGS—A Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, **10**, 325–337.

Rachev, S. T., Hsu, J. S. J., Bagasheva, B. S., and Fabozzi, F. J. (2008) *Bayesian Methods in Finance*, Wiley, Hoboken, NJ.

Robert, C. P. (2007) *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*, 2nd ed., Springer, New York.

Robert, C. P., and Casella, G. (2005) *Monte Carlo Statistical Methods, 2nd ed.*, Springer, New York.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., and van der Linde, A. (2002) Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society, Series B, Methodological*, **64**, 583–616.

Stein, C. (1956) Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley Symposium on Mathematical and Statistical Probability*, J. Neyman, ed., University of California, Berkeley, pp. 197–206, Volume 1.

van der Vaart, A. W. (1998) *Asymptotic Statistics*, Cambridge University Press, Cambridge.