5

Introduction to Bayesian Computation

5.1 Introduction

In the previous two chapters, two types of strategies were used in the summarization of posterior distributions. If the sampling density has a familiar functional form, such as a member of an exponential family, and a conjugate prior is chosen for the parameter, then the posterior distribution often is expressible in terms of familiar probability distributions. In this case, we can simulate parameters directly by using the R collection of random variate functions (such as `rnorm, rbeta,` and `rgamma`), and we can summarize the posterior using computations on this simulated sample. A second type of computing strategy is what we called the "brute-force" method. In the case where the posterior distribution is not a familiar functional form, then one simply computes values of the posterior on a grid of points and then approximates the continuous posterior by a discrete posterior that is concentrated on the values of the grid. This brute-force method can be generally applied for one- and two-parameter problems such as those illustrated in Chapters 3 and 4.

In this chapter, we describe the Bayesian computational problem and introduce some of the more sophisticated computational methods that will be employed in later chapters. One general approach is based on the behavior of the posterior distribution about its mode. This gives a multivariate normal approximation to the posterior that serves as a good first approximation in the development of more exact methods. We then provide a general introduction to the use of simulation in computing summaries of the posterior distribution. When one can directly simulate samples from the posterior distribution, then the Monte Carlo algorithm gives an estimate and associated standard error for the posterior mean for any function of the parameters of interest. In the situation where the posterior distribution is not a standard functional form, rejection sampling with a suitable choice of proposal density provides an alternative method for producing draws from the posterior. Importance sampling and sampling importance resampling (SIR) algorithms are alternative general methods for computing integrals and simulating from a general posterior

J. Albert, *Bayesian Computation with R*, Use R, DOI 10.1007/978-0-387-92298-0_5,

distribution. The SIR algorithm is especially useful when one wishes to investigate the sensitivity of a posterior distribution with respect to changes in the prior and likelihood functions.

5.2 Computing Integrals

The Bayesian recipe for inference is conceptually simple. If we observe data y from a sampling density $f(y|\theta)$, where θ is a vector of parameters and one assigns θ a prior $g(\theta)$, then the posterior density of θ is proportional to

$$g(\theta|y) \propto g(\theta)f(y|\theta).$$

The computational problem is to summarize this multivariate probability distribution to perform inference about functions of θ.

Many of the posterior summaries are expressible in terms of integrals. Suppose we are interested in the posterior mean of a function $h(\theta)$. This mean is expressible as a ratio of integrals,

$$E(h(\theta)|y) = \frac{\int h(\theta)g(\theta)f(y|\theta)d\theta}{\int g(\theta)f(y|\theta)d\theta}.$$

If we are interested in the posterior probability that $h(\theta)$ falls in a set A, we wish to compute

$$P(h(\theta) \in A|y) = \frac{\int_{h(\theta)\in A} g(\theta)f(y|\theta)d\theta}{\int g(\theta)f(y|\theta)d\theta}.$$

Integrals are also involved when we are interested in obtaining marginal densities of parameters of interest. Suppose we have the parameter $\theta = (\theta_1, \theta_2)$, where θ_1 are the parameters of interest and θ_2 are so-called nuisance parameters. One obtains the marginal posterior density of θ_1 by integrating out the nuisance parameters from the joint posterior:

$$g(\theta_1|y) \propto \int g(\theta_1, \theta_2|y)d\theta_2.$$

In the common situation where one needs to evaluate these integrals numerically, there are a number of quadrature methods available. However, these quadrature methods have limited use for Bayesian integration problems. First, the choice of quadrature method depends on the location and shape of the posterior distribution. Second, for a typical quadrature method, the number of evaluations of the posterior density grows exponentially as a function of the number of components of θ. In this chapter, we focus on the use of computational methods for computing integrals that are applicable to high-dimensional Bayesian problems.

5.3 Setting Up a Problem in R

Before we describe some general summarization methods, we first describe setting up a Bayesian problem in R. Suppose one is able to write an explicit expression for the joint posterior density. In writing this expression, it is not necessary to include any normalizing constants that don't involve the parameters. Next, for the algorithms described in this book, it is helpful to reparameterize all parameters so that they are all real-valued. If one has a positive parameter such as a variance, then transform using a log function. If one has a proportion parameter p, then it can be transformed to the real line by the logit function $\text{logit}(p) = \log(p/(1-p))$.

After the posterior density has been expressed in terms of transformed parameters, the first step in summarizing this density is to write an R function defining the logarithm of the joint posterior density.

The general structure of this R function is

```
mylogposterior=function(theta,data)
{
[statements that compute the log density]
return(val)
}
```

To apply the functions described in this chapter, `theta` is assumed to be a vector with k components; that is, $\theta = (\theta_1, ..., \theta_k)$. The input `data` is a vector of observed values or a list of data values and other model specifications such as the values of prior hyperparameters. The function returns a single value of the log posterior density.

One common situation is where one observes a random sample $y_1, ..., y_n$ from a sampling density $f(y|\theta)$ and one assigns θ the prior density $g(\theta)$. The logarithm of the posterior density of θ is given, up to an additive constant, by

$$\log g(\theta|y) = \log g(\theta) + \sum_{i=1}^{n} \log f(y_i|\theta).$$

Suppose we are sampling from a normal distribution with mean μ and standard deviation σ, the parameter vector $\theta = (\mu, \log\sigma)$, and we place an $N(10, 20)$ prior on μ and a flat prior on $\log\sigma$. The log posterior would have the form

$$\log g(\theta|y) = \log \phi(\mu; 10, 20) + \sum_{i=1}^{n} \log \phi(y_i; \mu, \sigma),$$

where $\phi(y; \mu, \sigma)$ is the normal density with mean μ and standard deviation σ. To program this function, we first write the simple function that evaluates the log likelihood of (μ, σ) for a component of y:

```
logf = function(y, mu, sigma)
  dnorm(y,mean=mu,sd=sigma,log=TRUE)
```

Note that we use the `log = TRUE` option in `dnorm` to compute the logarithm of the density. Then, if `data` represents the vector of observations $y_1, ..., y_n$, one can evaluate the sum of log likelihood terms $\sum_{i=1}^{n} \log \phi(y_i; \mu, \sigma)$ using the `sum` command:

```
sum(logf(data,mu,sigma))
```

The function `mylogposterior` defining the log posterior would in this case be written as follows.

```
mylogposterior=function(theta,data)
{
n=length(data)
mu=theta[1];  sigma=exp(theta[2])
logf = function(y, mu, sigma)
  dnorm(y,mean=mu,sd=sigma,log=TRUE)
val=dnorm(mu, mean=10, sd=20,log=TRUE)+sum(logf(data,mu,sigma))
return(val)
}
```

5.4 A Beta-Binomial Model for Overdispersion

Tsutakawa et al. (1985) describe the problem of simultaneously estimating the rates of death from stomach cancer for males at risk in the age bracket 45–64 for the largest cities in Missouri. Table 5.1 displays the mortality rates for 20 of these cities, where a cell contains the number n_j at risk and the number of cancer deaths y_j for a given city.

Table 5.1. Cancer mortality data. Each ordered pair represents the number of cancer deaths y_j and the number at risk n_j for an individual city in Missouri.

(0, 1083)	(0, 855)	(2, 3461)	(0, 657)	(1, 1208)	(1, 1025)
(0, 527)	(2, 1668)	(1, 583)	(3, 582)	(0, 917)	(1, 857)
(1, 680)	(1, 917)	(54, 53637)	(0, 874)	(0, 395)	(1, 581)
(3, 588)	(0, 383)				

A first modeling attempt might assume that the $\{y_j\}$ represent independent binomial samples with sample sizes $\{n_j\}$ and common probability of death p. But it can be shown that these data are overdispersed in the sense that the counts $\{y_j\}$ display more variation than would be predicted under a binomial model with a constant probability p. A better-fitting model assumes that y_j is distributed from a beta-binomial model with mean η and precision K:

$$f(y_j|\eta, K) = \binom{n_j}{y_j} \frac{B(K\eta + y_j, K(1-\eta) + n_j - y_j)}{B(K\eta, K(1-\eta))}.$$

Suppose we assign the parameters the vague prior proportional to

$$g(\eta, K) \propto \frac{1}{\eta(1-\eta)} \frac{1}{(1+K)^2}.$$

Then the posterior density of (η, K) is given, up to a proportionality constant, by

$$g(\eta, K|\text{data}) \propto \frac{1}{\eta(1-\eta)} \frac{1}{(1+K)^2} \prod_{j=1}^{20} \frac{B(K\eta + y_j, K(1-\eta) + n_j - y_j)}{B(K\eta, K(1-\eta))},$$

where $0 < \eta < 1$ and $K > 0$.

We write a short function `betabinexch0` to compute the logarithm of the posterior density. The inputs to the function are `theta`, a vector containing the values of η and K, and `data`, a matrix having as columns the vector of counts $\{y_j\}$ and the vector of sample sizes $\{n_j\}$.

```
betabinexch0=function (theta, data)
{
    eta = theta[1]
    K = theta[2]
    y = data[, 1]
    n = data[, 2]
    N = length(y)
    logf = function(y, n, K, eta) lbeta(K * eta + y, K * (1 -
        eta) + n - y) - lbeta(K * eta, K * (1 - eta))
    val = sum(logf(y, n, K, eta))
    val = val - 2 * log(1 + K) - log(eta) - log(1 - eta)
    return(val)
}
```

We read in the dataset `cancermortality` and use the function `mycontour` together with the log density function `betabinexch0` to display a contour plot of the posterior density of (η, K) (see Figure 5.1).

```
> data(cancermortality)
> mycontour(betabinexch0,c(.0001,.003,1,20000),cancermortality,
+    xlab="eta",ylab="K")
```

Note the strong skewness in the density, especially toward large values of the precision parameter K. This right-skewness is a common characteristic of the likelihood function of a precision or variance parameter.

Following the general guidance in Section 5.3, suppose we transform each parameter to the real line by using the reexpressions

$$\theta_1 = \text{logit}(\eta) = \log\left(\frac{\eta}{1-\eta}\right), \ \theta_2 = \log(K).$$

The posterior density of (θ_1, θ_2) is given by

$$g_1(\theta_1, \theta_2|\text{data}) = g\Big(\frac{e^{\theta_1}}{1+e^{\theta_1}}, e^{\theta_2}\Big)\frac{e^{\theta_1+\theta_2}}{(1+e^{\theta_1})^2},$$

where the right term in the product is the Jacobian term in the transformation. The log posterior density of the transformed parameters is programmed in the function `betabinexch`. Note the change in the next-to-last line of the function that accounts for the logarithm of the Jacobian term.

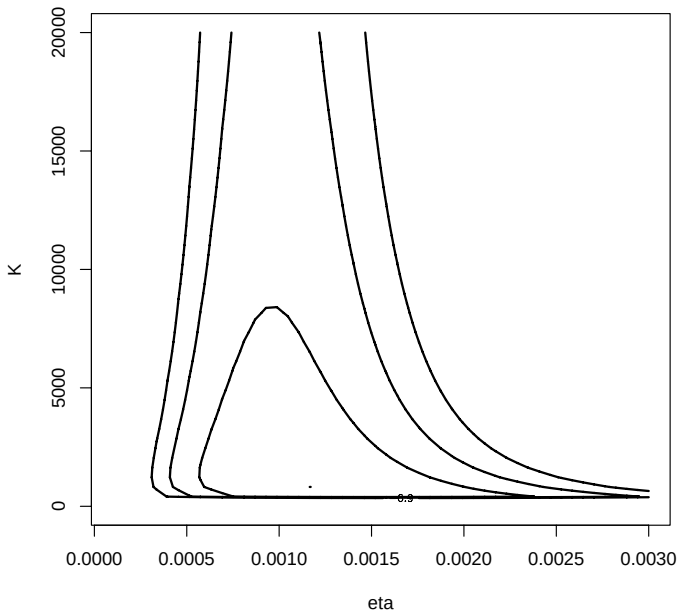

Fig. 5.1. Contour plot of parameters η and K in the beta-binomial model problem.

```
betabinexch=function (theta, data)
{
    eta = exp(theta[1])/(1 + exp(theta[1]))
    K = exp(theta[2])
    y = data[, 1]
    n = data[, 2]
    N = length(y)
    logf = function(y, n, K, eta) lbeta(K * eta + y, K * (1 -
        eta) + n - y) - lbeta(K * eta, K * (1 - eta))
```

```
    val = sum(logf(y, n, K, eta))
    val = val + theta[2] - 2 * log(1 + exp(theta[2]))
    return(val)
}
```

Figure 5.2 displays a contour plot of the posterior of (θ_1, θ_2) using the `mycontour` function.

```
> mycontour(betabinexch,c(-8,-4.5,3,16.5),cancermortality,
+   xlab="logit eta",ylab="log K")
```

Although the density has an unusual shape, the strong skewness has been reduced and the distribution is more amenable to the computational methods described in this and the following chapters.

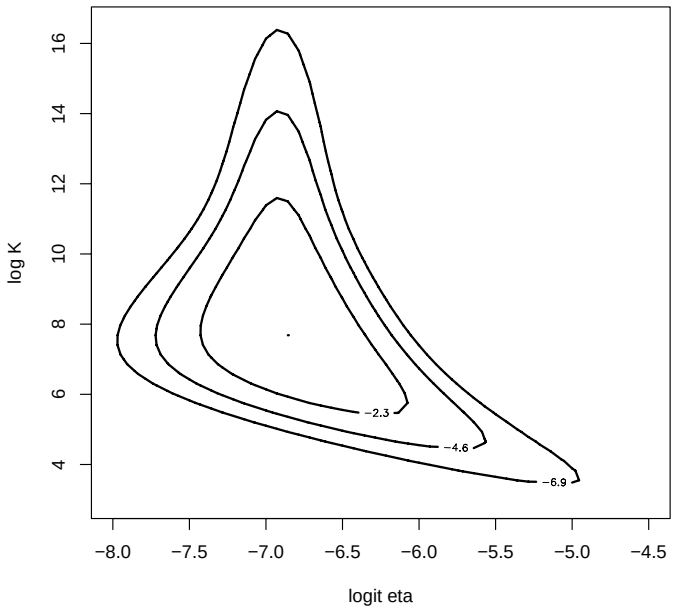

Fig. 5.2. Contour plot of transformed parameters $\text{logit}(\eta)$ and $\log K$ in the beta-binomial model problem.

5.5 Approximations Based on Posterior Modes

One method of summarizing a multivariate posterior distribution is based on the behavior of the density about its mode. Let θ be a vector-valued parameter with prior density $g(\theta)$. If we observe data y with sampling density $f(y|\theta)$, then consider the logarithm of the joint density of θ and y,

$$h(\theta, y) = \log(g(\theta)f(y|\theta)).$$

In the following, we write this log density as $h(\theta)$ since after the data are observed θ is the only random quantity. Denoting the posterior mode of θ by $\hat{\theta}$, we expand the log density in a second-order Taylor series about $\hat{\theta}$. This gives the approximation

$$h(\theta) \approx h(\hat{\theta}) + (\theta - \hat{\theta})' h''(\hat{\theta})(\theta - \hat{\theta})/2,$$

where $h''(\hat{\theta})$ is the Hessian of the log density evaluated at the mode. Using this expansion, the posterior density is approximated by a multivariate normal density with mean $\hat{\theta}$ and variance-covariance matrix

$$V = (-h''(\hat{\theta}))^{-1}.$$

In addition, this approximation allows one to analytically integrate out θ from the joint density and obtain the following approximation to the prior predictive density,

$$f(y) \approx (2\pi)^{d/2} g(\hat{\theta}) f(y|\hat{\theta}) | - h''(\hat{\theta})|^{1/2},$$

where d is the dimension of θ.

To apply this approximation, one needs to find the mode of the posterior density of θ. One general-purpose optimization algorithm for finding this mode is provided by Newton's method. Suppose one has a guess at the posterior mode θ^0. If θ^{t-1} is the estimate at the mode at the $t-1$ iteration of the algorithm, then the next iterate is given by

$$\theta^t = \theta^{t-1} - [h''(\theta^{t-1})]^{-1} h'(\theta^{t-1}),$$

where $h'(\theta^{t-1})$ and $h''(\theta^{t-1})$ are the gradient and Hessian of the log density evaluated at the current guess at the mode. One continues these iterations until convergence. There are many alternative algorithms available for finding the posterior mode. In the following, we will use the Nelder-Mead algorithm, which is the default method in the R function `optim` in the R base package. This algorithm is an iterative method based on the evaluation of the objective function over vertices of a simplex (a triangle for two variables). For the examples described in this book, the Nelder-Mead algorithm appears to be preferable to Newton's method since it is less sensitive to the choice of starting value.

After one writes an R function to evaluate the log posterior density, the R function `laplace` in the `LearnBayes` package finds the joint posterior mode by using `optim` and the default Nelder-Mead algorithm. The inputs to `laplace` are the function defining the joint posterior, an intelligent guess at the posterior mode, and data and parameters used in the definition of the log posterior. The choice of "intelligent guess" can be important since the algorithm may fail to converge with a poor choice of starting value. Suppose that a suitable starting value is used and `laplace` is successful in finding the posterior mode. The output of `laplace` is a list with four components. The component `mode` gives the value of the posterior mode $\hat{\theta}$, the component `var` is the associated variance-covariance matrix V, the component `int` is the approximation to the logarithm of the prior predictive density, and `converge` indicates if the algorithm converged.

5.6 The Example

We illustrate the use of the function `laplace` for our beta-binomial modeling example. Based on our contour plot, we start the Nelder-Mead method with the initial guess $(\text{logit}(\eta), \log K) = (-7, 6)$.

```
> fit=laplace(betabinexch,c(-7,6),cancermortality)
> fit

$mode
[1] -6.819793  7.576111

$var
            [,1]       [,2]
[1,]  0.07896568 -0.1485087
[2,] -0.14850874  1.3483208

$int
[1] -570.7743

$converge
[1] TRUE
```

We find the posterior mode to be $(-6.82, 7.58)$. From the output of `laplace`, we have the approximation that $(\text{logit}(\eta), \log K)$ is approximately bivariate normal with mean vector `fit$mode` and variance-covariance matrix `fit$var`. Using the `mycontour` function with the log bivariate normal function `lbinorm`, Figure 5.3 displays the contours of the approximate normal density. Comparing Figure 5.2 and Figure 5.3, we see significant differences between the exact and approximate normal posteriors.

```
> npar=list(m=fit$mode,v=fit$var)
> mycontour(lbinorm,c(-8,-4.5,3,16.5),npar,
+     xlab="logit eta", ylab="log K")
```

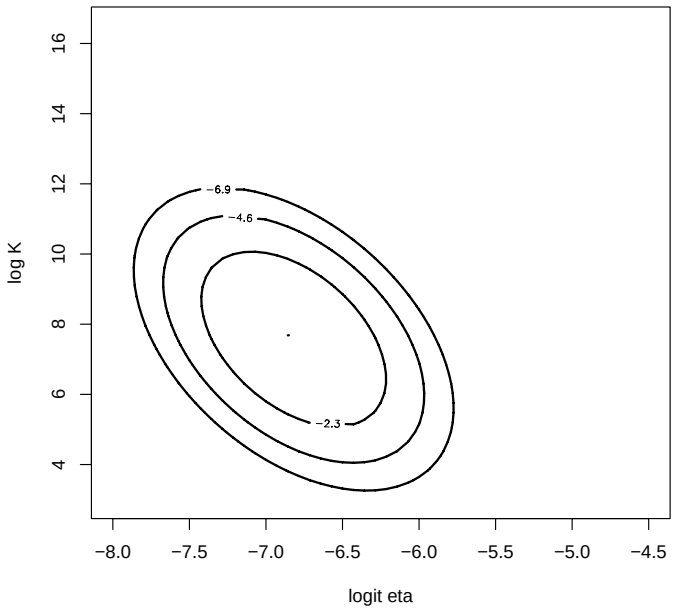

Fig. 5.3. Contour plot of normal approximation of $\text{logit}(\eta)$ and $\log K$ in the beta-binomial model problem.

One advantage of this algorithm is that one obtains quick summaries of the parameters by using the multivariate normal approximation. By using the diagonal elements of the variance-covariance matrix, one can construct approximate probability intervals for $\text{logit}(\eta)$ and $\log K$. For example, the following code constructs 90% probability intervals for the parameters:

```
> se=sqrt(diag(fit$var))
> fit$mode-1.645*se

[1] -7.282052  5.665982

> fit$mode+1.645*se

[1] -6.357535  9.486239
```

So a 90% interval estimate for logit(η) is $(-7.28, -6.36)$, and a 90% interval estimate for $\log K$ is (5.67, 9.49).

5.7 Monte Carlo Method for Computing Integrals

A second general approach for summarizing a posterior distribution is based on simulation. Suppose that θ has a posterior density $g(\theta|y)$ and we are interested in learning about a particular function of the parameters $h(\theta)$. The posterior mean of $h(\theta)$ is given by

$$E(h(\theta)|y) = \int h(\theta)g(\theta|y)d\theta.$$

Suppose we are able to simulate an independent sample $\theta^1, ..., \theta^m$ from the posterior density. Then the Monte Carlo estimate at the posterior mean is given by the sample mean

$$\bar{h} = \frac{\sum_{j=1}^{m} h(\theta^j)}{m}.$$

The associated simulation standard error of this estimate is estimated by

$$se_{\bar{h}} = \sqrt{\frac{\sum_{j=1}^{m}(h(\theta^j) - \bar{h})^2}{(m-1)m}}.$$

The Monte Carlo approach is an effective method for summarizing a posterior distribution when simulated samples are available from the exact posterior distribution. For a simple illustration of the Monte Carlo method, return to Section 2.4, where we were interested in the proportion of heavy sleepers p at a college. With the use of a beta prior, the posterior distribution for p was beta(14.26, 23.19). Suppose we are interested in the posterior mean of p^2. (This is the predictive probability that two students in a future sample will be heavy sleepers.) We simulate 1000 draws from the beta posterior distribution. If $\{p^j\}$ represent the simulated sample, the Monte Carlo estimate at this posterior mean will be the mean of the $\{(p^j)^2\}$, and the simulated standard error is the standard deviation of the $\{(p^j)^2\}$ divided by the square root of the simulation sample size.

```
> p=rbeta(1000, 14.26, 23.19)
> est=mean(p^2)
> se=sd(p^2)/sqrt(1000)
> c(est,se)

[1] 0.149122267 0.001885676
```

The Monte Carlo estimate at $E(p^2|\text{data})$ is 0.149, with an associated simulation standard error of 0.002.

5.8 Rejection Sampling

In the examples of Chapters 2, 3, and 4, we were able to produce simulated samples directly from the posterior distribution since the distributions were familiar functional forms. Then we would be able to obtain Monte Carlo estimates of the posterior mean for any function of the parameters of interest. But in many situations, such as the beta-binomial example of this chapter, the posterior does not have a familiar form and we need to use an alternative algorithm for producing a simulated sample.

A general-purpose algorithm for simulating random draws from a given probability distribution is rejection sampling. In this setting, suppose we wish to produce an independent sample from a posterior density $g(\theta|y)$ where the normalizing constant may not be known. The first step in rejection sampling is to find another probability density $p(\theta)$ such that:

- It is easy to simulate draws from p.
- The density p resembles the posterior density of interest g in terms of location and spread.
- For all θ and a constant c, $g(\theta|y) \leq cp(\theta)$.

Suppose we are able to find a density p with these properties. Then one obtains draws from g using the following accept/reject algorithm:

1. Independently simulate θ from p and a uniform random variable U on the unit interval.
2. If $U \leq g(\theta|y)/(cp(\theta))$, then accept θ as a draw from the density g; otherwise reject θ.
3. Continue steps 1 and 2 of the algorithm until one has collected a sufficient number of "accepted" θ.

Rejection sampling is one of the most useful methods for simulating draws from a variety of distributions, and standard methods for simulating from standard probability distributions such as normal, gamma, and beta are typically based on rejection algorithms. The main task in designing a rejection sampling algorithm is finding a suitable proposal density p and constant value c. At step 2 of the algorithm, the probability of accepting a candidate draw is given by $g(\theta|y)/(cp(\theta))$. One can monitor the algorithm by computing the proportion of draws of p that are accepted; an efficient rejection sampling algorithm has a high acceptance rate.

We consider the use of rejection sampling to simulate draws of $\theta = (\text{logit}(\eta), \log K)$ in the beta-binomial example. We wish to find a proposal density of a simple functional form that, when multiplied by an appropriate constant, covers the posterior density of interest. One choice for p would be a bivariate normal density with mean and variance given as outputs of the function `laplace`. Although this density does resemble the posterior density, the normal density has relatively sharp tails and the ratio $g(\theta|y)/p(\theta)$ likely would not be bounded. A better choice for a covering density is a multivariate t with

mean and scale matrix chosen to match the posterior density and a small number of degrees of freedom. The small number of degrees of freedom gives the density heavy tails and one is more likely to find bounds for the ratio $g(\theta|y)/p(\theta)$.

In our earlier work, we found approximations to the posterior mean and variance-covariance matrix of $\theta = (\text{logit}(\eta), \log K)$ based on the Laplace method. If the output variable of `laplace` is `fit`, then `fit$mode` is the posterior mode and `fit$var` the associated variance-covariance matrix. Suppose we decide to use a multivariate t density with location `fit$mode`, scale matrix `2 fit$var`, and 4 degrees of freedom. These choices are made to mimic the posterior density and ensure that the ratio $g(\theta|y)/p(\theta)$ is bounded from above.

To set up the rejection algorithm, we need to find the value of the bounding constant. We want to find the constant c such that

$$g(\theta|y) \le cp(\theta) \text{ for all } \theta.$$

Equivalently, since g is programmed on the log scale, we want to find the constant $d = \log c$ such that

$$\log g(\theta|y) - \log p(\theta) \le d \text{ for all } \theta.$$

Basically we wish to maximize the function $\log g(\theta|y) - \log p(\theta)$ over all θ. A convenient way to perform this maximization is by using the `laplace` function. We write a new function `betabinT` that computes values of this difference function. There are two inputs, the parameter `theta` and a list `datapar` with components `data`, the data matrix, and `par`, a list with the parameters of the t proposal density (mean, scale matrix, and degrees of freedom).

```
betabinT=function(theta,datapar)
{
data=datapar$data
tpar=datapar$par
d=betabinexch(theta,data)-dmt(theta,mean=c(tpar$m),
  S=tpar$var,df=tpar$df,log=TRUE)
return(d)
}
```

For our problem, we define the parameters of the t proposal density and the list `datapar`:

```
> tpar=list(m=fit$mode,var=2*fit$var,df=4)
> datapar=list(data=cancermortality,par=tpar)
```

We run the function `laplace` with this new function and using an "intelligent" starting value.

```
> start=c(-6.9,12.4)
> fit1=laplace(betabinT,start,datapar)
> fit1$mode
```

```
[1] -6.888963 12.421993
```

We find that the maximum value d occurs at the value $\theta = (-6.889, 12.422)$. We note that this θ value is not at the extreme portion of the space of simulated draws, which indicates that we indeed have found an approximate maximum. The value of d is found by evaluating the function at the modal value.

```
> betabinT(fit1$mode,datapar)
```

```
[1] -569.2829
```

We implement rejection sampling using the function `rejectsampling`. The inputs are the function `logf` defining the log posterior, the parameters of the t covering density `tpar`, the maximum value of d denoted by `dmax`, the number of candidate values simulated `n`, and the data for the log posterior function `data`. In this function, we simulate a vector of θ from the proposal density, compute the values of $\log g$ and $\log f$ on these simulated draws, compute the acceptance probabilities, and return only the simulated values of θ where the uniform draws are smaller than the acceptance probabilities. In the function `rejectsampling`, these four steps are accomplished by the commands

```
theta=rmt(n,mean=c(tpar$m),S=tpar$var,df=tpar$df)
lf=logf(theta,data)
lg=dmt(theta,mean=c(tpar$m),S=tpar$var,df=tpar$df,log=TRUE)
prob=exp(lf-lg-dmax)
theta[runif(n)<prob,]
```

We run the function `rejectsampling` using the constant value of d found earlier and simulate 10,000 draws from the proposal density. We see that the output value `theta` has only 2406 rows, so the acceptance rate of this algorithm is 2406/10,000 = .24. This is a relatively inefficient algorithm since it has a small acceptance rate, but the proposal density was found without too much effort.

```
> theta=rejectsampling(betabinexch,tpar,-569.2813,10000,
+   cancermortality)
> dim(theta)
```

```
[1] 2406    2
```

We plot the simulated draws from rejection sampling on the contour plot of the log posterior density in Figure 5.4. As expected, most of the draws fall within the inner contour of the exact density.

```
> mycontour(betabinexch,c(-8,-4.5,3,16.5),cancermortality,
+  xlab="logit eta",ylab="log K")
> points(theta[,1],theta[,2])
```

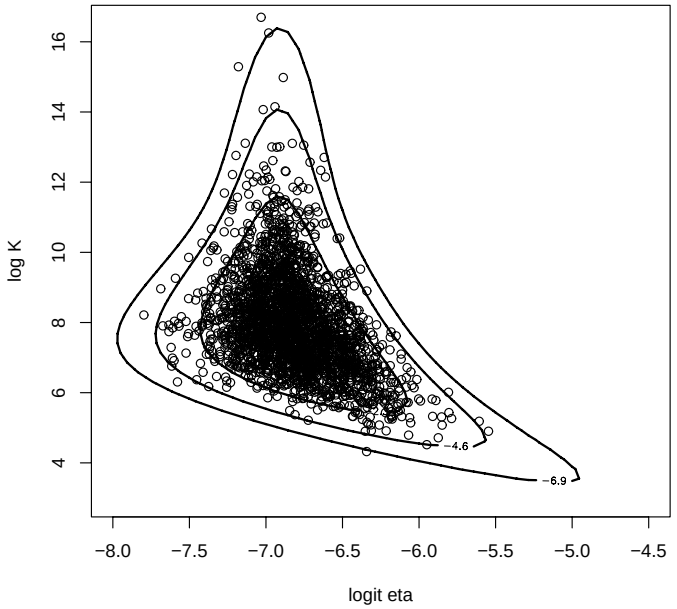

Fig. 5.4. Contour plot of logit(η) and $\log K$ in the beta-binomial model problem together with simulated draws from the rejection algorithm.

5.9 Importance Sampling

5.9.1 Introduction

Let us return to the basic problem of computing an integral in Bayesian inference. In many situations, the normalizing constant of the posterior density $g(\theta|y)$ will be unknown, so the posterior mean of the function $h(\theta)$ will be given by the ratio of integrals

$$E(h(\theta)|y) = \frac{\int h(\theta)g(\theta)f(y|\theta)d\theta}{\int g(\theta)f(y|\theta)|d\theta},$$

where $g(\theta)$ is the prior and $f(y|\theta)$ is the likelihood function. If we were able to simulate a sample $\{\theta^j\}$ directly from the posterior density g, then we could approximate this expectation by a Monte Carlo estimate. In the case where we are not able to generate a sample directly from g, suppose instead that we can construct a probability density p that we can simulate and that approximates the posterior density g. We rewrite the posterior mean as

$$E(h(\theta)|y) = \frac{\int h(\theta)\frac{g(\theta)f(y|\theta)}{p(\theta)}p(\theta)d\theta}{\int \frac{g(\theta)f(y|\theta)}{p(\theta)}p(\theta)d\theta} = \frac{\int h(\theta)w(\theta)p(\theta)d\theta}{\int w(\theta)p(\theta)d\theta},$$

where $w(\theta) = g(\theta)f(y|\theta)/p(\theta)$ is the weight function. If $\theta^1, ..., \theta^m$ are a simulated sample from the approximation density p, then the importance sampling estimate of the posterior mean is

$$\bar{h}_{IS} = \frac{\sum_{j=1}^m h(\theta^j)w(\theta^j)}{\sum_{j=1}^m w(\theta^j)}.$$

This is called an *importance sampling estimate* because we are sampling values of θ that are important in computing the integrals in the numerator and denominator. The simulation standard error of an importance sampling estimate is estimated by

$$se_{\bar{h}_{IS}} = \frac{\sqrt{\sum_{j=1}^m ((h(\theta^j) - \bar{h}_{IS})w(\theta^j))^2}}{\sum_{j=1}^m w(\theta^j)}.$$

As in rejection sampling, the main issue in designing a good importance sampling estimate is finding a suitable sampling density p. This density should be of a familiar functional form so simulated draws are available. The density should mimic the posterior density g and have relatively flat tails so that the weight function $w(\theta)$ is bounded from above. One can monitor the choice of p by inspecting the values of the simulated weights $w(\theta^j)$. If there are no unusually large weights, then it is likely that the weight function is bounded and the importance sampler is providing a suitable estimate.

To illustrate the use of different proposal densities in importance sampling in our example, consider the problem of estimating the posterior mean of a function of $\theta_2 = \log K$ conditional on a value of $\theta_1 = \text{logit}(\eta)$. The posterior density of θ_2, conditional on θ_1 is given by

$$g_1(\theta_2|\text{data}, \theta_1) \propto \frac{K}{(1+K)^2}\prod_{j=1}^{20}\frac{B(K\eta + y_j, K(1-\eta) + n_j - y_j)}{B(K\eta, K(1-\eta))},$$

where $\eta = \exp(\theta_1)/(1+\exp(\theta_1))$ and $K = \exp(\theta_2)$. In the following, we write the function `betabinexch.cond` to compute this posterior density conditional on the value $\theta_1 = -6.818793$. This function is written to allow the input of a vector of values of $\theta_2 = \log K$. Also, unlike the other functions in this chapter, the function `betabinexch.cond` returns the value of the density rather than the value of the log density.

```
betabinexch.cond=function (log.K, data)
{
    eta = exp(-6.818793)/(1 + exp(-6.818793))
    K = exp(log.K)
    y = data[, 1]; n = data[, 2]; N = length(y)
    logf=0*log.K
    for (j in 1:length(y))
      logf = logf + lbeta(K * eta + y[j], K * (1 -
        eta) + n[j] - y[j]) - lbeta(K * eta, K * (1 - eta))
    val = logf + log.K - 2 * log(1 + K)
    return(exp(val-max(val)))
}
```

To compute the mean of $\log K$ for the cancer mortality data, suppose we let the proposal density p be normal with mean 8 and standard deviation 2. In the R code below, we use the `integrate` function to find the normalizing constant of the posterior density of $\log K$. Then, using the `curve` function, we display the conditional posterior density of $\log K$ and the normal proposal density in the top left graph of Figure 5.5. The top right graph displays the weight function, the ratio of the posterior density to the proposal density.

```
> I=integrate(betabinexch.cond,2,16,cancermortality)
> par(mfrow=c(2,1))
> curve(betabinexch.cond(x,cancermortality)/I$value,from=3,to=16,
+  ylab="Density", xlab="log K",lwd=3, main="Densities")
> curve(dnorm(x,8,2),add=TRUE)
> legend("topright",legend=c("Exact","Normal"),lwd=c(3,1))
> curve(betabinexch.cond(x,cancermortality)/I$value/
+        dnorm(x,8,2),from=3,to=16, ylab="Weight",xlab="log K",
+        main="Weight = g/p")
```

Although the normal proposal density resembles the posterior density with respect to location and spread, the posterior density has a flatter right tail than the proposal and the weight function is unbounded for large $\log K$. Suppose instead that we let the proposal density have the t functional form with location 8, scale 2, and 2 degrees of freedom. Using a similar set of R commands, the bottom graphs of Figure 5.5 display the posterior and proposal densities and the weight function. Here the t proposal density has flatter tails than the posterior density and the weight function is bounded. Here the t functional form is a better proposal for importance sampling.

5.9.2 Using a Multivariate t as a Proposal Density

For a posterior density of a vector of real-valued parameters, a convenient choice of sampler p is a multivariate t density. The R function `impsampling` will implement importance sampling for an arbitrary posterior density when

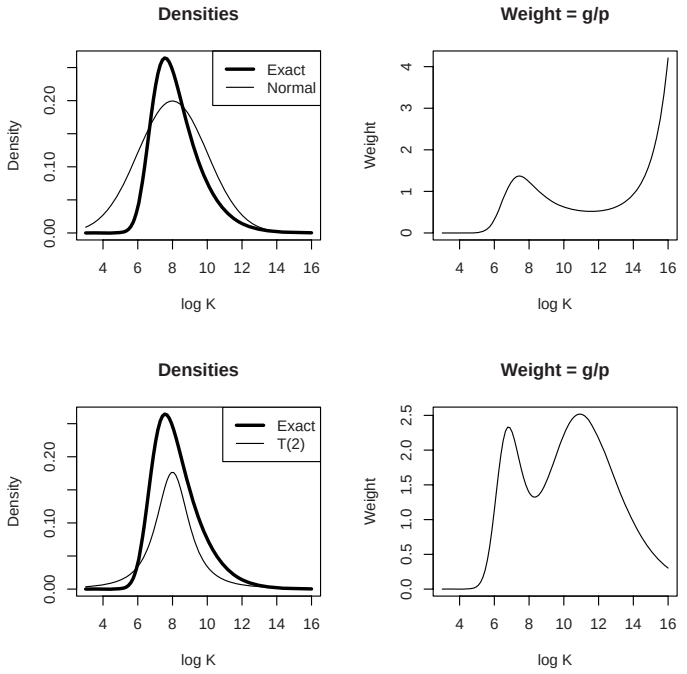

Fig. 5.5. Graph of the posterior density of $\log K$ and weight function using a normal proposal density (top) and a t(2) proposal density (bottom). By using a t proposal density, the weight function appears to be bounded from above.

p is a t density. There are five inputs to this function: `logf` is the function defining the logarithm of the posterior, `tpar` is a list of parameter values of the t density, `h` is a function defining the function $h(\theta)$ of interest, `n` is the size of the simulated sample, and `data` is the vector or list used in the definition of `logf`. In the function `impsampling`, the functions `rmt` and `dmt` from the `mnormt` library are used to simulate and compute values of the t density. In the following portion of R code from `impsampling`, we simulate draws from the sampling density, compute values of the log sampling density and the log posterior density at the simulated draws, and compute the weights and importance sampler estimate.

```
theta = rmt(n, mean = c(tpar$m), S = tpar$var, df = tpar$df)
lf = matrix(0, c(dim(theta)[1], 1))
lp = dmt(theta, mean = c(tpar$m), S = tpar$var, df = tpar$df,
      log = TRUE)
md = max(lf - lp)
wt = exp(lf - lp - md)
est = sum(wt * H)/sum(wt)
```

Note that the value `md` is the maximum value of the difference of logs of the posterior and proposal density – this value is used in the computation of the weights to prevent possible overflow. The output of `impsampling` is a list with four components: `est` is the importance sampling estimate, `se` is the corresponding simulation standard error, `theta` is a matrix of simulated draws from the proposal density p, and `wt` is a vector of the corresponding weights.

To illustrate importance sampling, let us return to our beta-binomial example and consider the problem of estimating the posterior mean of $\log K$. For this example, the proposal density used in the development of a rejection algorithm seems to be a good choice for importance sampling. We choose a t density where the location is the posterior mode (found from `laplace`), the scale matrix is twice the estimated variance-covariance matrix, and the number of degrees of freedom is 4. This choice for p will resemble the posterior density and have flat tails that we hope will result in bounded weights. We define a short function `myfunc` to compute the function h. Since we are interested in the posterior mean of $\log K$, we define the function to be the second component of the vector θ. We are now ready to run `impsampling`.

```
> tpar=list(m=fit$mode,var=2*fit$var,df=4)
> myfunc=function(theta)
+   return(theta[2])
> s=impsampling(betabinexch,tpar,myfunc,10000,cancermortality)
> cbind(s$est,s$se)

          [,1]       [,2]
[1,] 7.957802 0.01967276
```

We see from the output that the importance sampling estimate of the mean of $\log K$ is 7.958 with an associated standard error of 0.020. To check if the weight function is bounded, we compute a histogram of the simulated weights (not shown here) and note that there are no extreme weights.

5.10 Sampling Importance Resampling

In rejection sampling, we simulated draws from a proposal density p and accepted a subset of these values to be distributed according to the posterior density of interest $g(\theta|y)$. There is an alternative method of obtaining a simulated sample from the posterior density g motivated by the importance sampling algorithm.

As before, we simulate m draws from the proposal density p denoted by $\theta^1, ..., \theta^m$ and compute the weights $\{w(\theta^j) = g(\theta^j|y)/p(\theta^j)\}$. Convert the weights to probabilities bu using the formula

$$p^j = \frac{w(\theta^j)}{\sum_{j=1}^m w(\theta^j)}.$$

Suppose we take a new sample $\theta^{*1}, ..., \theta^{*m}$ from the discrete distribution over $\theta^1, ..., \theta^m$ with respective probabilities $p^1, ..., p^m$. Then the $\{\theta^{*j}\}$ will be approximately distributed according to the posterior distribution g. This method, called sampling importance resampling, or SIR for short, is a weighted bootstrap procedure where we sample with replacement from the sample $\{\theta^j\}$ with unequal sampling probabilities.

This sampling algorithm is straightforward to implement in R using the `sample` command. Suppose we wish to obtain a simulated sample of size `n`. As in importance sampling, we first simulate from the proposal density which in this situation is a multivariate t distribution, and then compute the importance sampling weights stored in the vector `wt`.

```
theta = rmt(n, mean = c(tpar$m), S = tpar$var, df = tpar$df)
lf = logf(theta, data)
lp = dmt(theta, mean = c(tpar$m), S = tpar$var, df = tpar$df,
      log = TRUE)
md = max(lf - lp)
wt = exp(lf - lp - md)
```

To implement the SIR algorithm, we first convert the weights to probabilities and store them in the vector `probs`. Next we use `sample` to take a sample with replacement from the indices 1, ..., `n`, where the sampling probabilities are contained in the vector `probs`; the simulated indices are stored in the vector `indices`.

```
probs=wt/sum(wt)
indices=sample(1:n,size=n,prob=probs,replace=TRUE)
```

Finally, we use the random indices in `indices` to select the rows of `theta` and assign them to the matrix `theta.s`. The matrix `theta.s` contains the simulated draws from the posterior.

```
theta.s=theta[indices,]
```

The function `sir` implements this algorithm for a multivariate t proposal density. The inputs to this function are the function defining the log posterior `logf`, the list `tpar` of parameters of the multivariate proposal density, the number `n` of simulated draws, and the `data` used in the log posterior function. The output is a matrix of simulated draws from the posterior. In the beta-binomial modeling example, we implement the SIR algorithm using the command

```
> theta.s=sir(betabinexch,tpar,10000,cancermortality)
```

We have illustrated the use of the SIR algorithm in converting simulated draws from a proposal density to draws from the posterior density. But this algorithm can be used to convert simulated draws from one probability density to a second probability density. To show the power of this method, suppose we wish to perform a Bayesian sensitivity analysis with respect to the individual

observations in the dataset. Suppose we focus on posterior inference about the log precision parameter $\log K$ and question how the inference would change if we removed individual observations from the likelihood. Let $g(\theta|y)$ denote the posterior density from the full dataset and $g(\theta|y_{(-i)})$ denote the posterior density with the ith observation removed. Let $\{\theta^j\}$ represent a simulated sample from the full dataset. We can obtain a simulated sample from $g(\theta|y_{(-i)})$ by resampling from $\{\theta^j\}$, where the sampling probabilities are proportional to the weights

$$\begin{aligned} w(\theta) &= \frac{g(\theta|y_{(-i)})}{g(\theta|y)} \\ &= \frac{1}{f(y_i|\theta)} \\ &= \frac{B(K\eta, K(1-\eta))}{B(K\eta + y_i, K(1-\eta) + n_i - y_i)}. \end{aligned}$$

Suppose that the inference of interest is a 90% probability interval for the log precision $\log K$. The R code for this resampling for the "leave observation i out" follows. One first computes the sampling weights and the sampling probabilities. Then the `sample` command is used to do the resampling from `theta` and the simulated draws from the "leave one out" posterior are stored in the variable `theta.s`. We summarize the simulated values of $\log K$ by the 5th, 50th, and 95th quantiles.

```
weight=exp(lbeta(K*eta,K*(1-eta))-
    lbeta(K*eta+y[i],K*(1-eta)+n[i]-y[i]))
probs=weight/sum(weight)
indices=sample(1:m,size=m,prob=probs,replace=TRUE)
theta.s=theta[indices,]
summary.obs[i,]=quantile(theta.s[,2],c(.05,.5,.95))
```

The function `bayes.influence` computes probability intervals for $\log K$ for the complete dataset and "leave one out" datasets using the SIR algorithm. We assume one already has simulated a sample of values from the complete data posterior, and the draws are stored in the matrix variable `theta.s`. The inputs to `bayes.influence` are `theta.s` and the dataset `data`. In this case, suppose we have just implemented the SIR algorithm, and the posterior draws are stored in the matrix `theta.s`. Then the form of the function would be

```
> S=bayes.influence(theta.s,cancermortality)
```

The output of this function is a list `S`; `S$summary` is a vector containing the 5th, 50th, and 95th percentiles, and `S$summary.obs` is a matrix where the ith row gives the percentiles for the posterior with the ith observation removed.

Figure 5.6 is a graphical display of the sensitivity of the posterior inference about $\log K$ with respect to the individual observations. The bold line shows the posterior median and 90% probability interval for the complete dataset,

and the remaining lines show the inference with each possible observation removed. Note that if observation number 15 is removed ($(y_i, n_i) = (54, 53637)$), then the location of $\log K$ is shifted toward smaller values. Also, if either observation 10 or observation 19 is removed, $\log K$ is shifted toward larger values. These two observations are notable since each city experienced three deaths and had relatively high mortality rates.

```
> plot(c(0,0,0),S$summary,type="b",lwd=3,xlim=c(-1,21),
+  ylim=c(5,11), xlab="Observation removed",ylab="log K")
> for (i in 1:20)
+  lines(c(i,i,i),S$summary.obs[i,],type="b")
```

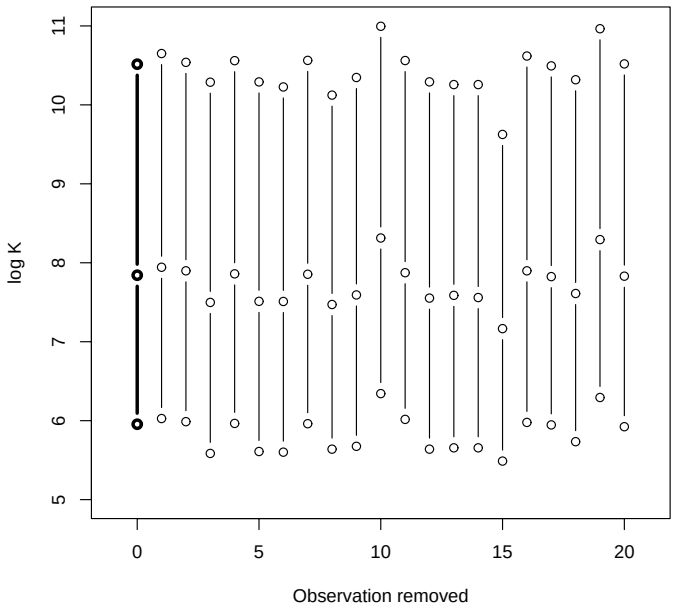

Fig. 5.6. Ninety percent interval estimates for $\log K$ for the full dataset (thick line) and interval estimates for datasets with each individual observation removed.

5.11 Further Reading

Rejection sampling is a general method used in simulating probability distributions; rejection sampling for statistical problems is described in Givens

and Hoeting (2005), Monahan (2001), and Robert and Casella (2004). Tanner (1996) introduces normal approximations to posterior distributions in Chapter 2 and Monte Carlo methods in Chapter 3. Robert and Casella (2004) in Chapter 3 describe different aspects of Monte Carlo integration. Smith and Gelfand (1992) introduce the use of rejection sampling and the SIR algorithm in simulating from the posterior distribution.

5.12 Summary of R Functions

`bayes.influence` – computes probability intervals for the log precision parameter K in a beta-binomial model for all "leave one out" models using sampling importance resampling
Usage: `bayes.influence(theta,data)`
Arguments: `theta`, matrix of simulated draws from the posterior of (logit eta, log K) for a beta-binomial model; `data`, matrix with columns of counts and sample sizes
Value: `summary`, vector of 5th, 50th and 95th percentiles of log K for the posterior of complete sample; `summary.obs`, matrix where the ith row contains the 5th, 50th and 95th percentiles of log K for the posterior when the ith observation is removed

`betabinexch0` – computes the logarithm of the posterior for the parameters (mean and precision) in a beta-binomial model
Usage: `betabinexch0(theta,data)`
Arguments: `theta`, vector of parameter values (eta, K); `data`, matrix with columns of counts and sample sizes
Value: value of the log posterior

`betabinexch` – computes the logarithm of the posterior for the parameters (logit mean and log precision) in a beta-binomial model
Usage: `betabinexch(theta,data)`
Arguments: `theta`, vector of parameter values (logit eta, log K); `data`, matrix with columns of counts and sample sizes
Value: value of the log posterior

`impsampling` – implements importance sampling to compute the posterior mean of a function using a multivariate t proposal density,
Usage: `impsampling(logf,tpar,h,n,data)`
Arguments: `logf`, function defining the log density; `tpar`, list of parameters of a multivariate t proposal density including the mean `m`, the scale matrix `var`, and the degrees of freedom `df`; `h`, function that defines h(theta); `n`, number of simulated draws from the proposal density; `data`, data and or parameters used in the function `logf`
Value: `est`, estimate at the posterior mean; `se`, simulation standard error of the estimate; `theta`, matrix of simulated draws from proposal density; `wt`, vector of importance sampling weights

`laplace` – for a general posterior density, computes the posterior mode, the associated variance-covariance matrix, and an estimate of the logarithm of the normalizing constant
Usage: `laplace(logpost,mode,par)`
Arguments: `logpost`, function that defines the logarithm of the posterior density; `mode`, vector that is a guess at the posterior mode; `par`, vector or list of parameters associated with the function `logpost`
Value: `mode`, current estimate of the posterior mode; `var`, current estimate of the associated variance-covariance matrix; `int`, estimate of the logarithm of the normalizing constant; `converge`, indication (TRUE or FALSE) if the algorithm converged

`lbinorm` – computes the logarithm of a bivariate normal density
Usage: `lbinorm(xy,par)`
Arguments: `xy`, vector consisting of two variables `x` and `y`; `par`, list containing `m`, a vector of means, and `v`, a variance-covariance matrix
Value: value of the kernel of the log density function

`rejectsampling` – implements a rejection sampling algorithm for a probability density using a multivariate t proposal density
Usage: `rejectsampling(logf,tpar,dmax,n,data)`
Arguments: `logf`, function that defines the logarithm of the density of interest; `tpar`, list of parameters of a multivariate t proposal density, including the mean `m`, the scale matrix `var`, and the degrees of freedom `df`; `dmax`, logarithm of the rejection sampling constant; `n`, number of simulated draws from the proposal density; `data`, data and/or parameters used in the function `logf`
Value: matrix of simulated draws from density of interest

`sir` – implements the sampling importance resampling algorithm for a multivariate t proposal density
Usage: `sir(logf,tpar,n,data)`
Arguments: `logf`, function defining logarithm of density of interest; `tpar`, list of parameters of a multivariate t proposal density including the mean `m`, the scale matrix `var`, and the degrees of freedom `df`; `n`, number of simulated draws from the posterior; `data`, data and parameters used in the function `logf`
Value: matrix of simulated draws from the posterior, where each row corresponds to a single draw

5.13 Exercises

1. **Estimating a log-odds with a normal prior**
   Suppose y has a binomial distribution with parameters n and p, and we are interested in the log-odds value $\theta = \log(p/(1-p))$. Our prior for θ is that $\theta \sim N(\mu, \sigma)$. It follows that the posterior density of θ is given, up to a proportionality constant, by

$$g(\theta|y) \propto \frac{\exp(y\theta)}{(1+\exp(\theta))^n} \exp\left[\frac{-(\theta-\mu)^2}{2\sigma^2}\right].$$

More concretely, suppose we are interested in learning about the probability that a special coin lands heads when tossed. A priori we believe that the coin is fair, so we assign θ an $N(0, .25)$ prior. We toss the coin $n = 5$ times and obtain $y = 5$ heads.

a) Using a normal approximation to the posterior density, compute the probability that the coin is biased toward heads (i.e., that θ is positive).
b) Using the prior density as a proposal density, design a rejection algorithm for sampling from the posterior distribution. Using simulated draws from your algorithm, approximate the probability that the coin is biased toward heads.
c) Using the prior density as a proposal density, simulate values from the posterior distribution using the SIR algorithm. Approximate the probability that the coin is biased toward heads.

2. **Genetic linkage model from Rao (2002)**
Suppose 197 animals are distributed into four categories with the following frequencies:

Category	1	2	3	4
Frequency	125	18	20	34

Assume that the probabilities of the four categories are given by the vector

$$\left(\frac{1}{2}+\frac{\theta}{4}, \frac{1}{4}(1-\theta), \frac{1}{4}(1-\theta), \frac{\theta}{4}\right),$$

where θ is an unknown parameter between 0 and 1. If θ is assigned a uniform prior, then the posterior density of θ is given by

$$g(\theta|\text{data}) \propto \left(\frac{1}{2}+\frac{\theta}{4}\right)^{125} \left(\frac{1}{4}(1-\theta)\right)^{18} \left(\frac{1}{4}(1-\theta)\right)^{20} \left(\frac{\theta}{4}\right)^{34},$$

where $0 < \theta < 1$. If θ is transformed to the real-valued logit $\eta = \log(\theta/(1-\theta))$, then the posterior density of η can be written as

$$f(\eta|\text{data}) \propto \left(2+\frac{e^\eta}{1+e^\eta}\right)^{125} \frac{1}{(1+e^\eta)^{39}} \left(\frac{e^\eta}{1+e^\eta}\right)^{35}, -\infty < \eta < \infty.$$

a) Use a normal approximation to find a 95% probability interval for η. Transform this interval to obtain a 95% probability interval for the original parameter of interest θ.
b) Design a rejection sampling algorithm for simulating from the posterior density of η. Use a t proposal density using a small number of degrees of freedom and mean and scale parameters given by the normal approximation.

3. **Estimation for the two-parameter exponential distribution**
Martz and Waller (1982) describe the analysis of a "type I/time-truncated" life testing experiment. Fifteen reciprocating pumps were tested for a prespecified time and any failed pumps were replaced. One assumes that the failure times follow the two-parameter exponential distribution

$$f(y|\beta,\mu) = \frac{1}{\beta} e^{-(y-\mu)/\beta},\ y \geq \mu.$$

Suppose one places a uniform prior on (μ, β). Then Martz and Waller show that the posterior density is given by

$$g(\beta,\mu|\text{data}) \propto \frac{1}{\beta^s} \exp\{-(t-n\mu)/\beta\},\ \mu \leq t_1,$$

where n is the number of items placed on test, t is the total time on test, t_1 is the smallest failure time, and s is the observed number of failures in a sample of size n. In the example, data were reported in cycles to failure; $n = 15$ pumps were tested for a total time of $t = 15962989$. Eight failures ($s = 8$) were observed, and the smallest failure time was $t_1 = 237217$.

a) Suppose one transforms the parameters to the real line using the transformations $\theta_1 = \log\beta, \theta_2 = \log(t_1 - \mu)$. Write down the posterior density of (θ_1, θ_2).
b) Construct an R function that computes the log posterior density of (θ_1, θ_2).
c) Use the `laplace` function to approximate the posterior density.
d) Use a multivariate t proposal density and the SIR algorithm to simulate a sample of 1000 draws from the posterior distribution.
e) Suppose one is interested in estimating the reliability at time t_0, defined by

$$R(t_0) = e^{-(t_0-\mu)/\beta}.$$

Using your simulated values from the posterior, find the posterior mean and posterior standard deviation of $R(t_0)$ when $t_0 = 10^6$ cycles.

4. **Poisson regression**
Haberman (1978) considers an experiment involving subjects reporting one stressful event. The collected data are $y_1, ..., y_{18}$, where y_i is the number of events recalled i months before the interview. Suppose y_i is Poisson distributed with mean λ_i, where the $\{\lambda_i\}$ satisfy the loglinear regression model

$$\log\lambda_i = \beta_0 + \beta_1 i.$$

The data are shown in Table 5.2. If (β_0, β_1) is assigned a uniform prior, then the logarithm of the posterior density is given, up to an additive constant, by

$$\log g(\beta_0,\beta_1|\text{data}) = \sum_{i=1}^{18} \Big[y_i(\beta_0 + \beta_1 i) - \exp(\beta_0 + \beta_1 i) \Big].$$

Table 5.2. Numbers of subjects recalling one stressful event.

Months	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
y_i	15	11	14	17	5	11	10	4	8	10	7	9	11	3	6	1	1	4

a) Write an R function to compute the logarithm of the posterior density of (β_0, β_1).
b) Suppose we are interested in estimating the posterior mean and standard deviation for the slope β_1. Approximate these moments by a normal approximation about the posterior mode (function `laplace`).
c) Use a multivariate t proposal density and the SIR algorithm to simulate 1000 draws from the posterior density. Use this sample to estimate the posterior mean and standard deviation of the slope β_1. Compare your estimates with the estimates using the normal approximation.

5. **Grouped Poisson data**
Hartley (1958) fits a Poisson model to the following grouped data:

Number of Events	0	1	2	3+	Total
Group Frequency	11	37	64	128	240

Suppose the mean Poisson parameter is λ, and the frequency of observations with j events is $n_j, j = 0, 1, 2$, and n_3 is the frequency of observations with at least three events. If the standard noninformative prior $g(\lambda) = 1/\lambda$ is assigned, then the posterior density is given by

$$g(\lambda|\text{data}) \propto e^{-\lambda(n_0+n_1+n_2)}\lambda^{n_1+2n_2-1}\left[1 - e^{-\lambda}\left(1 + \lambda + \frac{\lambda^2}{2}\right)\right]^{n_3}.$$

- Write an R function to compute the logarithm of the posterior density of λ.
- Use the function `laplace` to find a normal approximation to the posterior density of the transformed parameter $\theta = \log\lambda$.
- Use a t proposal density and the SIR algorithm to simulate 1000 draws from the posterior. Use the simulated sample to estimate the posterior mean and standard deviation of λ. Compare the estimates with the normal approximation estimates found in part (a).

6. **Mixture of exponential data**
Suppose a company obtains boxes of electronic parts from a particular supplier. It is known that 80% of the lots are acceptable and the lifetimes of the "acceptable" parts follow an exponential distribution with mean λ_A. Unfortunately, 20% of the lots are unacceptable and the lifetimes of the "bad" parts are exponential with mean λ_B, where $\lambda_A > \lambda_B$. Suppose $y_1, ..., y_n$ are the lifetimes of n inspected parts that can come from either acceptable or unacceptable lots. The y_is are a random sample from the mixture distribution

$$f(y|\lambda_A, \lambda_B) = p\frac{1}{\lambda_A}\exp(-y/\lambda_A) + (1-p)\frac{1}{\lambda_B}\exp(-y/\lambda_B),$$

where $p = .8$. Suppose (λ_A, λ_B) are assigned the noninformative prior proportional to $1/(\lambda_A\lambda_B)$.
The following function `log.exponential.mix` computes the log posterior density of the transformed parameters $\theta = (\theta_A, \theta_B) = (\log\lambda_A, \log\lambda_B)$:

```
log.exponential.mix=function(theta, y)
{
lambda.A=exp(theta[1]);  lambda.B=exp(theta[2])
sum(log(.8*dexp(y,1/lambda.A)+(1-.8)*dexp(y,1/lambda.B)))
}
```

The following lifetimes are observed from a sample of 30 parts:

```
 9.3  4.9  3.5 26.0  0.6  1.0  3.5 26.9
 2.6 20.4  1.0 10.0  1.7 11.3  7.7 14.1
24.8  3.8  8.4  1.1 24.5 90.7 16.4 30.7
 8.5  5.9 14.7  0.5 99.5 35.2
```

a) Construct a contour plot of (θ_A, θ_B) over the grid (1,4, −2, 8).
b) Using the function `laplace`, search for the posterior mode with a starting guess of $(\theta_A, \theta_B) = (3, 0)$.
c) Search for the posterior mode with a starting guess of $(\theta_A, \theta_B) = (2, 4)$.
d) Explain why you obtain different estimates of the posterior mode in parts (a) and (b).

7. **Variance components model**
Box and Tiao (1973) analyze data concerning batch-to-batch variation in yields of dyestuff. The following data arise from a balanced experiment whereby the total product yield was determined for five samples from each of six randomly chosen batches of raw material.

Batch	Batch Yield (in grams)
1	1545 1440 1440 1520 1580
2	1540 1555 1490 1560 1495
3	1595 1550 1605 1510 1560
4	1445 1440 1595 1465 1545
5	1595 1630 1515 1635 1625
6	1520 1455 1450 1480 1445

Let y_{ij} denote the jth observation in batch i. To determine the relative importance of between-batch variation versus sampling variation, the following multilevel model is applied (N denotes the number of batches and n denotes the number of observations per batch).

- y_{ij} is $N(\mu + b_i, \sigma_y), i = 1, ..., N, j = 1, ..., n$.
- $b_1, ..., b_N$ are a random sample from $N(0, \sigma_b)$.
- (σ_y^2, σ_b^2) is assigned a uniform prior.

In this situation, the focus is on the marginal posterior distribution of the variance components. It is possible to analytically integrate out the random effects $b_1, ..., b_N$, resulting in the marginal posterior density of $(\mu, \sigma_y^2, \sigma_b^2)$ given, up to a proportionality constant, by

$$\prod_{i=1}^{N} \left[\phi\left(\bar{y}_i | \mu, \sqrt{\sigma_y^2/n + \sigma_b^2}\right) f_G\left(S_i | (n-1)/2, 1/(2\sigma_y^2)\right) \right],$$

where $\bar{y}_i$ and S_i are respectively the mean yield and the "within sum of squares" of the ith batch, $\phi(y|\mu, \sigma)$ is the normal density with mean μ and standard deviation σ, and $f_G(y|a, b)$ is the gamma density proportional to $y^{a-1} \exp(-by)$. The posterior density of $\theta = (\mu, \log \sigma_y, \log \sigma_b)$ is programmed in the following R function `log.post.var.comp`. The input `y` in the function is a matrix with N rows and n columns, where a row contains the measurements for a particular batch.

```
log.post.var.comp=function(theta,y)
{
mu = theta[1]; sigma.y = exp(theta[2]); sigma.b = exp(theta[3])
Y=apply(y,1,mean); n=dim(y)[2]
S=apply(y,1,var)*(n-1)
loglike=sum(dnorm(Y,mu,sqrt(sigma.y^2/n+sigma.b^2),log=TRUE)+
       dgamma(S,shape=(n-1)/2,rate=1/(2*sigma.y^2),log=TRUE))
return(loglike+theta[2]+theta[3])
}
```

a) Using the function `laplace`, find the posterior mode of θ using the starting value $\theta = (1500, 3, 3)$. Try the alternative starting values of $\theta = (1500, 1, 1)$ and $\theta = (1500, 10, 10)$ to assess the sensitivity of the Nelder-Mead algorithm to the starting value.
b) Use the normal approximation to find 90% interval estimates for the logarithms of the standard deviations $\log \sigma_b$ and $\log \sigma_y$.
c) Using the results from part (b), find 90% interval estimates for the variance components σ_b^2 and σ_y^2.