LESSON 1 AN INTRODUCTION TO MCMC SAMPLING METHODS

Recommended Texts
Unfortunately it is difficult to recommend a single book that satisfactorily covers all the material in the course. The following two are recommended:
Ntzoufras, I (2008) Bayesian Modeling Using WinBUGS. Wiley
Hoff P (2009) A First Course in Bayesian Statistical Methods. Springer

For Lecture 1, useful supplementary reading is
Hoff P (2009) A First Course in Bayesian Statistical Methods, Chapters 4 and 10

1.1 Introduction

Let y denote the observations or data, and let θ denote the parameter or set of parameters by which the data are to be summarised. Bayesian methods combine prior evidence on the parameters contained in the density $p(\theta)$ with the likelihood $p(y|\theta)$ to produce the entire posterior density $p(\theta|y)$ of θ. From the posterior density one may extract any information – not simply "the most likely value" of a parameter, as with maximum likelihood (ML) estimators. However, until the advent of Monte Carlo Markov Chain methods it was not straightforward to sample from the posterior density, except in cases where it was analytically defined. Monte Carlo Markov Chain (MCMC) methods are iterative sampling methods that allow sampling from $p(\theta|y)$.

Although MCMC methods can be encompassed within the broad class of Monte Carlo methods, they must be distinguished from conventional Monte Carlo methods that generate independent simulations $\{u^{(1)}, u^{(2)}, \ldots u^{(T)}\}$ from a target density $\pi(u)$. From such simulations the expectation of a function $g(u)$ under $\pi(u)$, namely

$$E_\pi[g(u)] = \int g(u)\pi(u)du,$$

is estimated by taking the average of the function $g(u)$ evaluated at the sampled $u^{(t)}$, namely

$$\hat{g} = \sum_{t=1}^{T} g(u^{(t)})/T.$$

Under independent sampling the Monte Carlo estimator $\hat{g}$ tends to $E_\pi[g(u)]$ as $T \to \infty$. However, suppose we were to take $\pi(\theta) = p(\theta|y)$ as the density we wanted to find expectations from. We cannot use conventional independent Monte Carlo sampling, as this form of sampling from a posterior density $p(\theta|y)$ is not usually feasible.

When suitably implemented, MCMC methods offer an effective way to generate samples from the joint posterior distribution, $p(\theta|y)$. The "target density" for MCMC samples is the posterior density $\pi(\theta) = p(\theta|y)$, and MCMC sampling is especially relevant when the posterior cannot be stated exactly in analytic form, e.g. when the prior density assumed for θ is not conjugate with the likelihood $p(y|\theta)$. Suppose we seek the expectation μ of a function $g(\theta)$ under $\pi(\theta) = p(\theta|y)$, namely

$$\mu = E_\pi[g(\theta)] = \int g(\theta)p\,(\theta|y)\,d\theta.$$

Like Monte Carlo methods in general, MCMC methods when suitably defined satisfy the ergodic property. So the average of the values of $g(\theta)$ evaluated at MCMC samples $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(T)}\}$, namely

$$\hat{\mu} = \sum_{t=1}^{T} g(\theta^{(t)})/T,$$

tends to $\mu = E_\pi[g(\theta)]$ as $T \to \infty$. However, MCMC methods differ from conventional Monte Carlo methods in that successive sampled parameters are dependent or autocorrelated. This dependence means that larger samples are needed to obtain a given precision.

Assume a preset initial parameter value $\theta^{(0)}$. Then MCMC methods involve generating a correlated sequence of sampled values $\theta^{(t)}$ (t = 1, 2, 3, ..), where updated values $\theta^{(t)}$ are drawn from a transition distribution

$$K(\theta^{(t)}|\theta^{(0)}, .., \theta^{(t-1)}) = K(\theta^{(t)}|\theta^{(t-1)})$$

that is Markovian in the sense of depending only on $\theta^{(t-1)}$. The transition distribution $K(t|t-1)$ is chosen to satisfy additional conditions ensuring that the sequence has the joint posterior density $p(\theta|y)$ as its stationary distribution. These conditions typically reduce to requirements on the proposal distribution and acceptance rule used to generate new parameter samples (see sections 1.2 and 1.5). The proposal distribution must be specified in a way that guarantees irreducibility and positive recurrence; see, for example, Andrieu & Moulines (2006). Under such conditions, the sampled parameters $\theta^{(t)}$ $\{t = B, B+1, ..., T\}$ beyond a certain burn-in phase in the sampling (of length B) can be viewed as a random sample from the target density $\pi(\theta) = p(\theta|y)$.

In practice MCMC methods may be applied separately to individual parameters or groups ("blocks") of more than one parameter. So if there are P parameters, the number of blocks C may well be less than P. Parameters in a particular block may be updated jointly (a so-called "block update"). Different MCMC methods may be applied to different parameters or blocks. So assuming θ contains more than one parameter, and consists of C components or "blocks" $\{\theta_1, .., \theta_C\}$, different updating methods may be used for each component. Note that for simplicity, we will often represent the MCMC algorithms as involving a generic parameter θ, even though several parameters are typically involved in real examples.

1.2 Metropolis Sampling

We now consider some strategies for generating $\theta^{(t)}$ in a MCMC sampling sequence. Let $p(y|\theta)$ denote the likelihood, and $p(\theta)$ denote the prior density for θ, or more accurately the prior densities $p(\theta_1), ..., p(\theta_C)$ that are adopted on the components of θ. Then the earliest MCMC method is known as the Metropolis algorithm (Metropolis et al, 1953) and involves a symmetric proposal density (e.g. a Normal, Student t or uniform density)

$$q(\theta_{cand}|\theta^{(t)})$$

(Chib and Greenberg, 1995) for generating candidate parameter values θ_{cand}. The Metropolis sampling algorithm is a special case of a broader class of Metropolis-Hastings algorithms (section 1.5).

The acceptance probability (the probability of accepting the candidate value θ_{cand} as a replacement for the current value $\theta^{(t)}$) may be obtained as

$$\alpha = min(1, \frac{\pi(\theta_{cand})}{\pi(\theta^{(t)})}) = min(1, \frac{p(\theta_{cand}|y)}{p(\theta^{(t)}|y)}) = min(1, \frac{p(y|\theta_{cand})p(\theta_{cand})}{p(y|\theta^{(t)})p(\theta^{(t)})}). \qquad (1.1)$$

So one compares the (likelihood * prior), namely $p(y|\theta)p(\theta)$, for the candidate and existing parameter values. If the (likelihood * prior) is higher for the candidate value, it is automatically accepted and $\theta^{(t+1)} = \theta_{cand}$. However, even if the (likelihood * prior) is lower for the candidate value, such that α is less than 1, the candidate value may still be accepted. This involves random sampling from a uniform density, $U^{(t)}$, and the candidate value is accepted subject to $\alpha \geq U^{(t)}$.

The third equality in (1.1) follows because the marginal likelihood $M = p(y)$ in the Bayesian formula

$$p(\theta|y) = p(y|\theta)p(\theta)/p(y) = p(y|\theta)p(\theta)/M$$

cancels out, as it is a constant. Stated more completely and formally, it is not necessary to know the normalized target distribution, namely the posterior density, $\pi(\theta) = p(\theta|y)$. It is enough to know the target density up to a constant factor. Note that the (likelihood * prior), namely $p(y|\theta)p(\theta)$, is sometimes called the un-normalized posterior density.

So the Metropolis algorithm *can* be implemented by using the full posterior distribution

$$\pi(\theta) = p(\theta|y) = p(y|\theta)p(\theta)/M \qquad (1.2)$$

as the target distribution. However, for updating values on a particular parameter θ_j, it is not just M that cancels out in the ratio

$$\pi(\theta_{cand})/\pi(\theta^{(t)}) = \frac{p(y|\theta_{cand})p(\theta_{cand})}{p(y|\theta^{(t)})p(\theta^{(t)})}$$

but any parts of the likelihood $p(y|\theta)$ or prior $p(\theta)$ not involving θ_j. These parts relate to parameters other than θ_j, and can be viewed as constant when θ_j is being updated.

When those parts of the likelihood or prior not relevant to θ_j are abstracted out, the remaining part of $p(\theta|y) = p(y|\theta)p(\theta)/M$, the part relevant to updating θ_j, is known as the full conditional density for θ_j (Gilks et al, 1996, p 11; Gilks, 1996). One may denote the full conditional density for θ_j as

$$\pi_j(\theta_j) \propto p(\theta_j|\theta_{[j]}, y)p(\theta_j) \qquad (1.3)$$

where $\theta_{[j]}$ consists of all parameters apart from θ_j, namely $\theta_{[j]} = (\theta_1, .., \theta_{j-1,}\theta_{j+1}, ..., \theta_C)$. In (1.3), the prior on θ_j is $p(\theta_j)$ while $p(\theta_j|\theta_{[j]}, y)$ denotes the likelihood as a function of θ_j only. How to obtain the full conditional in practice is discussed in section 1.4.

So the probability α for updating a particular parameter θ_j within the full parameter set can be obtained *either* by comparing the full posterior (known up to a constant M), namely

$$\alpha = min(1, \frac{\pi(\theta_{j,cand})}{\pi(\theta_j^{(t)})}) = min(1, \frac{p(y|\theta_{j,cand})p(\theta_{j,cand})}{p(y|\theta_j^{(t)})p(\theta_j^{(t)})})$$

or by comparing values of the full conditional for the j^{th} parameter at values $\theta_{j,cand}$ and $\theta_j^{(t)}$, namely

$\alpha = min(1, \frac{\pi_j(\theta_{j,cand})}{\pi_j(\theta_j^{(t)})})$.

If the candidate value provides a higher value of the un-normalized posterior density or the full conditional density - that is, the ratio $\frac{p(y|\theta_{j,cand})p(\theta_{j,cand})}{p(y|\theta_j^{(t)})p(\theta_j^{(t)})}$ or $\frac{\pi_j(\theta_{j,cand})}{\pi_j(\theta_j^{(t)})}$ exceeds one - then the candidate value is accepted automatically. If the ratio is less than one, one sets

$\theta_j^{(t+1)} = \theta_{j,cand}$

with probability α, or keeps the current value

$\theta_j^{(t+1)} = \theta_j^{(t)}$

otherwise. To determine whether the candidate value is accepted when the ratio is not decisive, a uniform number $U^{(t)}$ is generated and the transition $\theta_j^{(t)} \rightarrow \theta_{j,cand}$ occurs only if $U^{(t)} \leq \alpha$.

1.3 Choice of Proposal Density

There is some flexibility in choice of proposal density q, but the chosen density and the parameters incorporated in it are relevant to successful MCMC updating and convergence. A standard recommendation is that the proposal density for a particular parameter θ_j should approximate the posterior density $p(\theta_j|y)$ of that parameter. In some cases one may have an idea (e.g. from a classical statistical analysis) of what the posterior density is, or what the main defining parameters are. Since posterior densities are very often (though not always) approximately normal, then an estimate of the posterior mean and variance will often suffice to construct a normal proposal density. Albert (2007) actually applies a Laplace approximation technique to estimate the posterior mode and uses the mean and variance parameters to define the proposal densities used in a subsequent stage of Metropolis-Hastings sampling. This approach may become difficult in problems with many parameters and in fact, ad hoc initial approximations, such as a N(0,1) proposal density (a normal with mean 0 and variance 1), may be used initially and subsequently improved on using the MCMC acceptance rates.

The rate at which a proposal generated by $q(\theta_{cand}|\theta^{(t)})$ is accepted (the acceptance rate) depends on how close the candidate value θ_{cand} is to $\theta^{(t)}$, and this depends on both the mean μ_q and variance $V_q = \sigma_q^2$ of the proposal density, but especially the latter. A higher acceptance rate would typically follow from reducing V_q, but with the risk that the posterior density will take longer to explore. If the acceptance rate is too high, then autocorrelation in sampled values will be excessive (since the chain tends to move in a restricted space), while a too low acceptance rate leads to the same problem, since the chain then gets locked at particular values. One possibility is to use a variance or dispersion estimate V_m of θ from a maximum likelihood or other mode finding analysis (which approximates the posterior variance), and then scale V_m by a constant $k > 1$, so that the proposal density variance is $V_q = kV_m$. Values of k in the range 2-10 are typical; for more theoretical results see Roberts et al (1997) and Roberts & Rosenthal (2004).

For an optimal MCMC sequence (one that "mixes well" and moves effectively through the posterior density of the parameter) working rules are for an acceptance rate of 0.4 when a parameter is updated singly (e.g. by separate univariate normal proposals), and 0.2 when a group of parameters are updated simultaneously as a block (e.g. by a multivariate normal proposal). Lynch (2007, p 141) states that "a rate somewhere between 25% and 75% is often acceptable".

Typical Metropolis updating schemes use uniform, standard normal or standard Student t variables W_t. A Normal proposal density $q(\theta_{cand}|\theta^{(t)})$ involves samples $G_t \sim N(0,1)$, with candidate values

$\theta_{cand} = \theta^{(t)} + \sigma_q G_t$,

where σ_q determines the size of the potential jump from current to future value (and the acceptance rate). This is equivalent to

$\theta_{cand} = \theta^{(t)} + G_t^*$,

where $G_t^* \sim N(0, \sigma_q^2)$. A uniform random walk samples $H_t \sim Unif(-1, 1)$, and scales this to form a proposal

$\theta_{cand} = \theta^{(t)} + \kappa H_t$,

with the value of κ determining the acceptance rate. This is equivalent to

$\theta_{cand} = \theta^{(t)} + H_t^*$,

where $H_t^* \sim Unif(-\kappa, \kappa)$. As noted above, it is desirable that the proposal density approximately matches the shape of the target density $p(\theta|y)$. The Langevin random walk scheme (not considered in this tutorial) is an example of a scheme including information about the shape of $p(\theta|y)$ in the proposal, as in

$\theta_{cand} = \theta^{(t)} + \sigma[G_t + 0.5\nabla log(p(\theta^{(t)}|y)]$

where ∇ denotes the gradient function (Roberts & Tweedie, 1996).

Sometimes one may sample a transformed version of a parameter, for example, one may use normal sampling for a log variance, rather than direct sampling of values for the variance (which has to be restricted to positive values). In this case an appropriate Jacobean adjustment $J^{(t)}$ must be included in the likelihood, so that

$\alpha = min(1, \frac{\pi(\theta_{j,cand})J_{cand}}{\pi(\theta_j^{(t)})J^{(t)}}) = min(1, \frac{p(y|\theta_{j,cand})p(\theta_{j,cand})J_{cand}}{p(y|\theta_j^{(t)})p(\theta_j^{(t)})J^{(t)}})$.

The extended worked example in Appendix 2 illustrates this, as well as Example 1.2 below.

1.4 Obtaining Full Conditional Densities

As noted above, Metropolis sampling may be based on the full conditional density when a particular parameter θ_j is being updated. These full conditionals are particularly central in Gibbs sampling (see section 1.5 below). The full conditional densities may be obtained from the joint density $p(\theta, y) = p(y|\theta)p(\theta)$ and in many cases reduce to standard densities (Normal, exponential, gamma, etc) from which direct sampling is straightforward.

Full conditional densities for θ_j are derived by abstracting out from the joint model density $p(y|\theta)p(\theta)$ (namely the likelihood * prior) only those elements including θ_j and treating other components as constants (Gilks, 1996; George et al, 1993, p 149).

Consider a conjugate model for Poisson count data $\{y_i, i = 1, ..., n\}$ with means $\mu_i h_i$ where h_i are exposures, and the μ_i are themselves gamma distributed. For example, one might consider the number y_i of days with thunder over a span h_i of days. Assuming the μ_i are themselves gamma distributed provides a form of hierarchical model appropriate for overdispersed count data. Overdispersed count data show actual variability $var(y)$ exceeding that assumed (namely the average $\bar{y}$) under the Poisson model. So at the first (data likelihood) stage one has

$y_i \sim Poisson(\mu_i h_i)$.

Suppose at the second stage $\mu_i \sim Ga(\alpha, \beta)$, namely

$p(\mu_i|\alpha, \beta) = \mu_i^{\alpha-1} e^{-\beta\mu_i} \beta^\alpha / \Gamma(\alpha)$

where $Ga(\alpha, \beta)$ denotes a gamma with mean α/β and variance α/β^2. Further assume that

$\alpha|A_0 \sim E(A_0)$,

namely that the "hyperparameter" α is exponential with parameter A_0, and that

$\beta|B_0, C_0 \sim Ga(B_0, C_0)$

where A_0, B_0 and C_0 are *preset* constants. Often one would find the terminology that A_0, B_0 and C_0 are *known*.

So the full posterior density $p(\theta|y)$ of $\theta = (\mu_1, ..\mu_n, \alpha, \beta)$ given y is proportional to

$$e^{-A_0\alpha} \beta^{B_0-1} e^{-C_0\beta} [\beta^\alpha / \Gamma(\alpha)]^n \prod_{i=1}^{n} e^{-\mu_i h_i} \mu_i^{y_i} \{\mu_i^{\alpha-1} e^{-\beta\mu_i}\} \qquad (1.4)$$

where all constants are combined in the proportionality constant. These constants are:

the denominator $y_i!$ in the Poisson likelihood;

the term $h_i^{y_i}$ in the Poisson likelihood;

the multiplier A_0 in the exponential prior density for α, $p(\alpha|A_0) = A_0 \exp(-A_0\alpha)$;

the term $C_0^{B_0}/\Gamma(B_0)$ in the prior density for β, namely $p(\beta|B_0, C_0) = \beta^{B_0-1} e^{-C_0\beta} C_0^{B_0}/\Gamma(B_0)$

It is apparent from inspecting (1.4) that the full conditional densities of μ_i and β are also gamma, namely

$\mu_i \sim Ga(y_i + \alpha, \beta + h_i)$

and

$$\beta \sim Ga(B_0 + n\alpha, C_0 + \sum_{i=1}^{n} \mu_i)$$

respectively. The full conditional density of α, also obtained from inspecting (1.4), is

$$p(\alpha|y, \beta, \mu) \propto e^{-A_0\alpha} [\beta^\alpha / \Gamma(\alpha)]^n \prod_{i=1}^{n} \mu_i^{\alpha-1}$$

This density is non-standard and cannot be sampled directly (as can the gamma densities for μ_i and β). Hence a Metropolis or Metropolis-Hastings step can be used for updating it.

Example 1.1 Estimating normal parameters via Metropolis.

Consider $n = 1000$ values x_i generated randomly from a $N(3, 25)$ distribution, a Normal with mean $\mu = 3$ and variance $\sigma^2 = 25$. Using the generated x we seek to estimate the mean and variance, now treating them as unknowns. The sampling scheme used in the R code below actually updates σ rather than σ^2.

Setting the vector of unknowns as $\theta = (\mu, \sigma^2)$, the likelihood is

$$p(y|\theta) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x_i-\mu)^2}{2\sigma^2}) \ .$$

Assume a flat prior for μ (uniform between -∞ and +∞), and a prior

$$p(\sigma) \propto \frac{1}{\sigma}$$

on σ; this is a standard noninformative prior on σ, in fact called the Jeffreys prior (see Albert, 2007, p 109 and http://en.wikipedia.org/wiki/Jeffreys_prior). Then the full posterior density $p(\theta|y) = p(y|\theta)p(\theta)/M$ is proportional to

$$\frac{1}{\sigma}\prod_{i=1}^{n} \left\{ \frac{1}{\sigma} exp(-\frac{(x_i-\mu)^2}{2\sigma^2}) \right\} = \frac{1}{\sigma^{n+1}} \prod_{i=1}^{n} exp(-\frac{(x_i-\mu)^2}{2\sigma^2}).$$

(An aside: can you say what is the proportionality constant here?)

Here we use separate updating schemes for μ and for σ (rather than, say, a bivariate normal update for μ and $log(\sigma)$ jointly). Specifically, assume uniform $U(-\kappa, \kappa)$ proposal densities around the current parameter values $\mu^{(t)}$ and $\sigma^{(t)}$, with $\kappa = 0.5$. For generating candidate values σ_{cand} of the standard deviation, the absolute value of $\sigma^{(t)} + U(-\kappa, \kappa)$ is used. Note that varying the lower and upper limit of the uniform sampling (e.g. taking $\kappa = 1$, or $\kappa = 0.25$) may considerably affect the acceptance rates. In the acceptance step, the log of the ratio $\frac{p(y|\theta_{cand})p(\theta_{cand})}{p(y|\theta^{(t)})p(\theta^{(t)})}$ is compared to the log of a random uniform value to avoid computer over/underflow.

We use R code to illustrate the analysis. Note that with this code and subsequent examples, one can simply paste the whole code into the R Console page. Useful guides to features of R are provided at a number of sites, including:

http://www.statmethods.net/

http://cran.r-project.org/doc/manuals/R-intro.html#R-as-a-set-of-statistical-tables

The R syntax for sampling from major densities is summarised in Appendix 1.

The R code for $\kappa = 0.5$ is as follows, and uses the full posterior density (1.2) as the target density for assessing candidate values. The R code runif(1,-0.5,0.5) means one random uniform sample from a $U(-0.5, 0.5)$. The number of iterations is set at $T = 10000$, with $B = 1000$ burn-in. Thus we first generate the data (n=1000 values) and then re-estimate the underlying parameters (as if they were unknowns) as follows. Note that the simulated data may well have a mean (and sd) that vary a bit from 3 (and 5) respectively, so I have included a line to summarise the simulated data.

```
# generate data
x = rnorm(1000,3,5);
mean(x); sd(x)
# initial vector setting and initial parameter values
```

```
T <- 10000; B <- 1000; u.mu <- runif(T); u.sig <- runif(T)
mu <- sig <- numeric(T); mu[1] <- 3; sig[1] <- 5
# counters for rejections of proposals
REJmu <- 0; REJsig <- 0
# log posterior density (up to a constant)
logpost = function(x,mu,sig){
loglike = sum(dnorm(x,mu,sig,log=TRUE))
# -log(sig) is log of Jeffreys prior
return(loglike - log(sig))}
# MCMC sampling loop
for (t in 2:T) { mut <- mu[t-1]; sigt <- sig[t-1]
    mucand <- mut + runif(1,-0.5,0.5)
    sigcand <- abs(sigt + runif(1,-0.5,0.5))
# accept (or not) proposal for mu
log.alph.mu = logpost(x,mucand,sigt)-logpost(x,mut,sigt)
if (log(u.mu[t]) < log.alph.mu) mu[t] <- mucand
else { mu[t] <- mut; REJmu <- REJmu+1 }
# accept (or not) proposal for sigma
log.alph.sig = logpost(x,mu[t],sigcand)-logpost(x,mu[t],sigt)
if (log(u.sig[t]) < log.alph.sig) sig[t] <- sigcand
else { sig[t] <- sigt; REJsig <- REJsig+1 }}
# MCMC Rejection rates
REJratemu <- REJmu/T; REJratesig <- REJsig/T;
# posterior plots and summaries
samp <- 1001:10000; musamp <- mu[samp]; sigsamp <- sig[samp]
hist(musamp,50) # mu posterior marginal histogram plot
mu.den <- density(musamp)
plot(mu.den) # mu posterior marginal kernel plot
plot(musamp,sigsamp) # covariance plot of sampled values
summary(musamp)
quantile(musamp, probs=c(.025,0.5,0.975))
summary(sigsamp)
quantile(sigsamp, probs=c(.025,0.5,0.975))
cat("Rejection Rate mu = ",REJratemu,"\n")
cat("Rejection Rate sigma = ",REJratesig,"\n")
```

Note that the function is returning the log of the normal likelihood accumulated over all cases plus the log of $1/\sigma$. Except for extraneuous constants (that cancel out in Metropolis comparisons) this is equivalent to the posterior density discussed above.

Taking all T=10000 iterations (rather than say the last 9000 iterations with B=1000 regarded as burn-in) leads to rejection rates of 51% and 63% (i.e. acceptance rates of 49%

and 37%) on μ and σ respectively. The rates obtained in any particular run will fluctuate a bit around these values (especially when T is not that large) due to random factors. For example, my own repeat run with this code gave rejection rates of 52% and 66%. With T=100,000 there would probably be less fluctuation between MCMC runs.

Exercise 1.1 In the above code for Example 1.1, obtain the acceptance rates after iteration B=1000, and with $\kappa = 1$ and $\kappa = 0.25$ on both parameters. You will need to insert an if statement in the above code (and change the denominator in REJratemu <- REJmu/T and REJratesig <- REJsig/T) in order to obtain the acceptance rate after iteration B=1000. Which value of κ is preferred for better MCMC acceptance (the higher of $\kappa = 0.25$ or $\kappa = 1$, the lower, or neither)?

Exercise 1.2 In the above code for Example 1.1, replace the uniform $U(-0.5, 0.5)$ proposal densities on μ and σ by $N(0, 1)$ proposal densities, and find what the acceptance rates are. Note that, as in the uniform proposal scheme in Example 1.1, one will actually use the absolute values of σ_{cand}. Note also that in R a single random sample

rnorm(1,mu,sigma)

from a normal $N(\mu, \sigma^2)$ density specifies the standard deviation σ rather than the variance.

Exercise 1.3 Consider how we might tune the normal variances on both mu and sig proposals to improve the acceptance rates. This involves a "trial and error" procedure here (although formal adaptive procedures to tune the proposal variance have been suggested in the literature). Specifically obtain the acceptance rates on μ and σ with $N(0, 0.5)$ densities on both parameters, and with $N(0, 2)$ densities on both parameters, i.e. with variances 0.5 and 2. Be careful with regard to the value you plug into rnorm(1,mu,sigma) as sigma is the standard deviation. Which variance option is preferable for MCMC acceptance rates?

Exercise 1.4. The set up for data and likelihood is as in Example 1.1, but we consider the posterior density under an alternative prior. We keep a flat prior for μ, but now adopt the prior

$p(\sigma^2) \propto \frac{1}{\sigma^2}$

on the variance $V = \sigma^2$. Reframe the R code to account for the new posterior density, and also so that the uniform proposal density is for V rather than for σ. As in the proposal scheme in Example 1.1, one will actually use the absolute values of V_{cand}.

In defining the posterior density function it is convenient to reparameterize so that

```
logpost = function(x,mu,V){sig=?(V)
loglike = sum(dnorm(x,mu,sig,log=TRUE))
return(loglike - log(?))}
```

Since $V = \sigma^2$, it is likely that a wider uniform interval will be needed for V than for σ. So retain the $U(-0.5, 0.5)$ proposal density on μ, but apply a $U(-2.5, 2.5)$ density for V. With $T = 10000$, what are the rejection rates for candidate values of μ and V for iterations $t > B$

(where $B = 1000$).

Example 1.2 Unknown Binomial Probability
N=100 binomial observations are generated with probability $\pi = 0.4$. The relevant R code is

```
n =rpois(100, 100)
y = rbinom(100, n, 0.4)
```

We then take $\{n_i, y_i\}$ as the observations and seek to estimate π as an unknown. Thus for numbers n_i at risk, and with $\theta = \pi$, one has a likelihood $p(y|\theta)$

$y_i \sim Bin(n_i, \pi)$.

We assume a beta prior, $\pi \sim Beta(\alpha, \beta)$ for the single unknown, with α and β known (i.e. take preset values such as $\alpha = \beta = 1$).

The full posterior $p(\pi|y)$ is then proportional to

$$\pi^{\alpha-1}(1-\pi)^{\beta-1}\prod_i \pi^{y_i}(1-\pi)^{n_i-y_i} \qquad (1.5)$$

with the constant of proportionality encompassing the $\begin{pmatrix} n_i \\ y_i \end{pmatrix}$ in the likelihood, and the ratio $\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$ in the prior.

As one possible prior, one might assume $\alpha = \beta = 0.5$ (equivalent to a prior sample size of $\alpha + \beta = 1$), namely

$\pi \sim Beta(0.5, 0.5)$.

A non-symmetric proposal density q (e.g. a beta density) might be used to generate candidate values for π. However, to enable use of the Metropolis algorithm, a normal proposal with $\sigma_q = 0.025$ is used and applied in conjunction with the transformed parameter $\eta = log(\pi)$. The posterior must be adjusted for the transformation. Since

$\frac{\partial \pi}{\partial \eta} = e^{\eta} = \pi$,

one now has, after adjusting (1.5) to take account of the transformation, that

$$p(\eta|y) \propto \pi^{\alpha}(1-\pi)^{\beta-1}\prod_i \pi^{y_i}(1-\pi)^{n_i-y_i}.$$

In a R analysis, T=100000 iterations are used, with B=1000, and initial value $\eta_1 = -1$ for the unknown parameter $\eta = log(\pi)$. The sampled π is taken as the minimum of $exp(\eta)$ and 1 since (especially in early sampling), candidate values of η greater than 0 (i.e. $\pi > 1$) may be generated. The data and R code are

```
# generate data
 n =rpois(100, 100);  y = rbinom(100, n, 0.4)
# re-estimate
  f = function(eta) {pi <- min(exp(eta),1)
  loglike <- y*log(pi)+(n-y)*log(1-pi)
  logpriorpi <- alpha*log(pi)+(beta-1)*log(1-pi)
  f <- sum(loglike)+logpriorpi}
   # Set initial values and vectors
 T = 100000; B <- 1000; eta <- numeric(T); eta[1] <- -1;
 etarej = 0; u <- runif(T); alpha=0.5; beta=0.5
  # metropolis proposal standard devn
sd <- 0.025
for (t in 2:T) { etacand <- eta[t-1]+sd*rnorm(1,0,1);
  tcand <- f(etacand); tcurr <- f(eta[t-1])
  if (log(u[t]) <= tcand-tcurr) eta[t] <- etacand else {
 eta[t] <- eta[t-1]; if(t > B) etarej <- etarej+1}}
  # posterior summary
  summary(eta[(B+1):T])
# acceptance rate
 1-etarej/(T-B)
```

with the posterior for η having mean and median -0.92. The acceptance rate is 0.51.

Exercise 1.5 Program this problem using a uniform proposal density for π with sampled values less than 0 or above 1 rejected. This is likely to involve using both the abs() and min() commands. Use a $U(-0.025, 0.025)$ proposal density. Note that there is no Jacobean under this sampling scheme. With $T = 100000$ and $B = 1000$ find the posterior mean for π and the acceptance rate. Comment on what happens to the acceptance rate as the uniform sampling interval is widened to $U(-0.05, 0.05)$ and $U(-0.1, 0.1)$.

Exercise 1.6 In Example 1.2, consider normal sampling of proposals using the transformed parameter

$\eta = log[\pi/(1-\pi)]$.

State $\frac{\partial\pi}{\partial\eta}$ under this transformation. Hence state the form for $p(\eta|y)$. With T=100000 and B=1000, obtain the posterior median and mean for both η and π for iterations $t > B$, and the acceptance rate with a Metropolis proposal standard deviation of 0.1.

1.5 Metropolis-Hastings Sampling

The Metropolis-Hastings algorithm is a generalisation of Metropolis sampling, and provides the baseline for MCMC schemes that simulate a Markov chain $\theta^{(t)}$ with $p(\theta|y)$ as its stationary distribution. Following Hastings (1970), the chain is updated from $\theta^{(t)}$ to θ_{cand} with probability

$\alpha(\theta_{cand}|\theta^{(t)}) = min\left(1, \frac{p(\theta_{cand}|y)q(\theta^{(t)}|\theta_{cand})}{p(\theta^{(t)}|y)q(\theta_{cand}|\theta^{(t)})}\right)$

where the proposal density q may now be non-symmetric, so that $q(\theta_{cand}|\theta^{(t)})$ does not necessarily equal $q(\theta^{(t)}|\theta_{cand})$. $q(\theta_{cand}|\theta^{(t)})$ is the probability (or density ordinate) of θ_{cand} for a density centred at $\theta^{(t)}$, while $q(\theta^{(t)}|\theta_{cand})$ is the probability (or density) for moving back from θ_{cand} to the current value $\theta^{(t)}$. The transition kernel is

$K(\theta^{(t)}|\theta_{cand}) = \alpha(\theta_{cand}|\theta^{(t)})q(\theta_{cand}|\theta^{(t)})$

for $\theta_{cand} \neq \theta^{(t)}$, with a nonzero probability of staying in the current state, namely

$$K(\theta^{(t)}|\theta^{(t)}) = 1 - \int \alpha(\theta_{cand}|\theta^{(t)})q(\theta_{cand}|\theta^{(t)})d\theta_{cand}.$$

Conformity of M-H sampling to the Markov chain requirements is considered by Mengersen & Tweedie (1996) and Roberts & Rosenthal (2004).

If the proposed new value θ_{cand} is accepted, then $\theta^{(t+1)} = \theta_{cand}$, while if it rejected the next state is the same as the current state, i.e. $\theta^{(t+1)}=\theta^{(t)}$. The target density $p(\theta|y)$ appears in ratio form, so, as for Metropolis sampling, it is not necessary to know the normalising constant M. If the proposal density is symmetric, with $q(\theta_{cand}|\theta^{(t)}) = q(\theta^{(t)}|\theta_{cand})$, then the Metropolis-Hastings algorithm reduces to the Metropolis algorithm discussed above. If the proposal density has the form

$$q(\theta_{cand}|\theta^{(t)}) = q(\theta^{(t)} - \theta_{cand}),$$

then a random walk Metropolis scheme is obtained (Gelman et al, 2004; Albert, 2007, p 105). Another option is independence sampling, when the density $q(\theta_{cand})$ for sampling candidate values is independent of the current value $\theta^{(t)}$.

While it is possible for the target density to relate to the entire parameter set, it is often computationally simpler in multi-parameter problems to divide θ into C blocks or components, and use the full conditional densities in componentwise updating. Consider the update for the j^{th} parameter or parameter block. At step j of iteration $t+1$ the preceding $j-1$ parameter blocks are already updated via the M-H algorithm, while $\{\theta_{j+1}, ..., \theta_C\}$ are still at their iteration t values (Chib & Greenberg, 1995).

Let $\theta_j^{(t)}$ be the current value for θ_j, with the vector of remaining partially updated parameters denoted

$$\theta_{[j]} = (\theta_1^{(t+1)}, \theta_2^{(t+1)}, .., \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, .., \theta_C^{(t)}).$$

The candidate value for θ_j is generated from the j^{th} proposal density, denoted $q_j(\theta_{j,cand}|\theta_j^{(t)})$. Also governing the acceptance of a proposal are full conditional densities

$$\pi_j(\theta_j) \propto p(\theta_j|\theta_{[j]}, y)p(\theta_j)$$

specifying the density of θ_j conditional on other parameters $\theta_{[j]}$. The candidate value θ is then accepted with probability

$$\alpha = min\left[1, \frac{p(\theta_{cand}|\theta_{[j]},y)p(\theta_{cand})q(\theta_j^{(t)}|\theta_{cand})}{p(\theta_j^{(t)}|\theta_{[j]},y)p(\theta_j^{(t)})q(\theta_{cand}|\theta_j^{(t)})}\right] \qquad (1.6).$$

Example 1.3 M-H sampling from a Rayleigh Density

The Rayleigh density has the form

$$f_R(x|\sigma) = x\ exp(-x^2/2\sigma^2)/\sigma^2$$

for positive x in $[0, \infty)$. This density has mean

$$\sigma\sqrt{\pi/2}$$

and median

$$\sigma\sqrt{log(4)}$$

(see http://en.wikipedia.org/wiki/Rayleigh_distribution). We consider M-H sampling to generate random values from a particular density. In particular this Example uses a chi-square proposal density (i.e. a nonsymmetric proposal density $q(x_{cand}|x^{(t-1)})$ for sampling random values from a Rayleigh density, $f_R(x|\sigma)$ as above, with a preset scale parameter $\sigma = 4$.

The code firstly defines the density by a function command. The sampled values of $x^{(t)}$ have to be positive and the initial $x^{(1)}$ is drawn with a $\chi^2(1)$ density. Another possibility for the initial x candidate would be a draw from a $\chi^2(\sigma\sqrt{\pi/2}\,)$ density. Later candidate values are drawn with a $\chi^2(x^{(t-1)})$ proposal density with mean $x^{(t-1)}$.

```
# Metropolis-Hastings sampling from Rayleigh Density
   f <- function(x, sig) { return((x / sig^2) * exp(-x^2 / (2*sig^2)))}
   T <- 100000; x <- numeric(T); u <- runif(T)
   sig <- 4
# generate initial value from chi-square with 1 df
   x[1] <- rchisq(1, df=1); Rej <- 0
# Sampling loop
   for (t in 2:T) { xt <- x[t-1];
   xcand <- rchisq(1, df = xt)
   num <- f(xcand, sig) * dchisq(xt, df = xcand)
   den <- f(xt, sig) * dchisq(xcand, df = xt)
   if (u[t] <= num/den) x[t] <- xcand
   else {x[t] <- xt; Rej <- Rej+1}} # proposal x.cand is rejected
   RejR <- Rej/T; AccepR <- 1-RejR
   cat("Acceptance Rate",AccepR,"\n")
# posterior summary
   samp <- 5001:10000; xsamp <- x[samp]
   summary(xsamp)
# plot samples 5001-10000
   plot(samp, xsamp, type="l", main="Plot of Samples", ylab="x",xlab="Iteration")
# kernel plot
   den.x <- density(xsamp) # returns the density data
   plot(den.x)
```

Because the Rayleigh density is positive, the proposal density q must be confined to positive values, as the chi-square density is above. One could use a symmetric density, in the sense that $q(\theta_{cand}|\theta^{(t)}) = q(\theta^{(t)}|\theta_{cand})$, and revert to Metropolis sampling. For example, a truncated normal, after first loading library(msm), could be used in the above. The relevant lines in the code become

```
   x[1] <- rtnorm(1, 1,1,0,Inf); Rej <- 0
   # Sampling loop
   for (t in 2:T) { xt <- x[t-1];
```

```
xcand <- rtnorm (1, xt,1,0,Inf)
num <- f(xcand, sig)/den <- f(xt, sig)
......
```

Exercise 1.7 Adapt the code in Example 1.3 to sample random x values from a gamma density $Ga(x|\alpha,\beta)$, but still using a chi-square proposal density for candidate x values. The gamma parameters are $\alpha = 2$, $\beta = 0.5$. For the gamma density one has

$Ga(x|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$,

with mean α/β, and variance α/β^2. For T=100,000 samples compare the mean and variance of the sampled x values with the "true" values of 4 and 8. Monte Carlo error will mean there may be a bit of deviation from the true value.

Exercise 1.8 Adapt the code in Example 1.3 to sample from the Rayleigh density $f(x|4)$, but now using a gamma proposal density,

$q(y|x) = Ga(y|\kappa, \frac{\kappa}{x}) = \frac{\kappa^\kappa}{x^\kappa \Gamma(\kappa)} y^{\kappa-1} e^{-\kappa y/x}$

where κ has a role as a precision parameter that can be tuned to provide improved acceptance rates. Larger values of κ mean that the sampled candidate values

$x_{cand} \sim Ga(\kappa, \kappa/x_{curr})$

will be more closely clustered about the current parameter value x_{curr}. Compare acceptance rates obtained with $\kappa = 2$ and $\kappa = 1$; which value leads to an acceptance rate closer to 0.4?

1.5 Gibbs Sampling

The Gibbs sampler (Gelfand & Smith, 1990; Gilks et al, 1993; Casella and George, 1992) is a special component-wise M-H algorithm whereby the proposal density for updating θ_j equals the full conditional $f_j(\theta_j|\theta_{[j]}) = p(y|\theta_j)p(\theta_j)$. It follows from (1.6) that proposals are accepted with probability 1.

If it is possible to update all blocks this way, then the Gibbs sampler involves parameter by parameter (or block by block) updating which when completed forms the transition from $\theta^{(t)} = (\theta_1^{(t)}, ..., \theta_C^{(t)})$ to $\theta^{(t+1)} = (\theta_1^{(t+1)}, ..., \theta_C^{(t+1)})$:

1.$\theta_1^{(t+1)} \sim f_1(\theta_1|\theta_2^{(t)}, \theta_3^{(t)}, \ldots, \theta_C^{(t)})$;

2.$\theta_2^{(t+1)} \sim f_2(\theta_2|\theta_1^{(t+1)}, \theta_3^{(t)}, \ldots, \theta_C^{(t)})$;

.

$C.\theta_C^{(t+1)} \sim f_C(\theta_C|\theta_1^{(t+1)}, \theta_2^{(t+1)}, \ldots, \theta_{C-1}^{(t+1)})$.

Example 1.4 Gibbs Sampling Example Schools Data Meta Analysis

Consider the schools data from Gelman et al (2004, p 138), consisting of point estimates y_j ($j = 1, .., J$) of unknown effects θ_j, where each y_j has a known design variance σ_j^2 (though the listed data provides σ_j not σ_j^2). The terminology "known" means that the σ_j^2 have known values and hence are not unknown parameters. Our model here is a simple hierarchical model, a form of meta-analysis (the normal-normal model) well suited to analysis by Bayesian techniques. Lynch (2007, pp 241-246) sets out Gibbs sampling routines for another form of hierarchical model, the random intercept model for nested data.

The first stage of the hierarchical normal-normal model assumes there are underlying "true" means θ_j for school j, namely

$y_j \sim N(\theta_j, \sigma_j^2)$.

The second stage of the hierarchical prior specifies a normal model for the latent θ_j,

$\theta_j \sim N(\mu, \tau^2)$.

The final stage relates to the "hyperparameters" μ and τ^2. We assume a flat prior on μ (as in Gelman et al, 2004; Sinharay & Stern, 2003), and that the inverse variance (or precision) $1/\tau^2$ has a $Ga(a, b)$ gamma prior.

The full conditionals for the latent effects θ_j, namely $p(\theta_j|y, \mu, \tau^2)$ are then as specified by Gelman et al (2004, p 135), namely

$\theta_j \backsim N(\frac{1}{Dj}[y_j/\sigma^2 + \mu/\tau^2], 1/D_j)$

where $D_j = 1/\sigma_j^2 + 1/\tau^2$. The full conditional for μ is

$\mu \backsim N(\bar{\theta}, \tau^2/J)$,

and that for $1/\tau^2$ is gamma, namely

$1/\tau^2 \backsim Ga(J/2 + a, \frac{\sum_j (\theta_j - \mu)^2}{2} + b)$.

There are $J + 2$ unknowns in the R code (N.B. the σ_j^2 are not unknowns) for implementing these Gibbs updates. There are $T = 20000$ MCMC samples to be accumulated in the array MCMCsamp.

With $a = b = 0.1$ in the prior for $1/\tau^2$, and remembering that the normal density in R uses the standard deviation, one then has

```
y=c(28,8,-3,7,-1,1,18,12); sigma=c(15,10,16,11,9,11,10,18)
J <- 8; T <- 20000; sigma2 <- sigma^2
MCMCsamp <- matrix(0, nrow=T, ncol=J+2)
# starting values for unknown hyperparameters mu and tau2
mu=mean(y); tau2=median(sigma2)
# main sampling loop
for (t in 1:T) {D=1/sigma2+1/tau2
th.mu=(y/sigma2+mu/tau2)/D
th.sd=sqrt(1/D)
theta=rnorm(J,th.mu,th.sd)
mu=rnorm(1,mean(theta),sqrt(tau2/J))
invtau2=rgamma(1,J/2+0.1,sum((theta-mu)^2)/2+0.1)
tau2 <- 1/invtau2; tau <- sqrt(tau2)
MCMCsamp[t,1] =mu; MCMCsamp[t,2] =tau
MCMCsamp[t,3:(2+J)] = theta}
```

```
write.table(MCMCsamp, file="MCMCsamp.txt")
```

Posterior means and standard deviations obtained in subequent spreadsheet analysis of MCMCsamp.txt are as follows:

	μ	τ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8
Mean	8.0	2.5	9.0	8.0	7.6	8.0	7.1	7.5	8.8	8.1
St devn	4.4	2.8	5.6	4.9	5.4	5.1	5.0	5.2	5.2	5.4

Exercise 1.9 R code for informative prior on μ.
In Example 1.4, instead of assuming a flat prior for μ, assume instead a normal prior with

$\mu \sim N(m_\mu, V_\mu)$,

where m_μ and V_μ are known (i.e. m_μ and V_μ take predefined values, and so are not extra parameters). The full posterior conditional for μ now involves a precision weighted average of m_μ, and of the average $\bar{\theta}$ of the θ_j. The precision for m_μ is $1/V_\mu$ and that for $\bar{\theta}$ is J/τ^2, so one has

$\mu \backsim N(\frac{\bar{\theta} J V_\mu + m_\mu \tau^2}{J V_\mu + \tau^2}, \frac{\tau^2 V_\mu}{J V_\mu + \tau^2})$.

For a prior $\mu \sim N(8,1)$ (i.e. relatively informative as opposed to the flat prior assumed in Example 1.4), amend the code in Example 1.4 to include extra lines specifying m_μ and V_μ, and the amended full conditional update for μ. What implications does the new prior have for inferences on μ. A revised code might follow the template (omitting lines that do not need to be changed)

```
y=c(28,8,-3,7,-1,1,18,12); sigma=c(15,10,16,11,9,11,10,18)
J <- 8; T <- 20000; sigma2 <- sigma^2
# settings for Normal N(8,1) prior on mu
mmu <- ?; Vmu <- ?
...
# main sampling loop
for (t in 1:T) {D=1/sigma2+1/tau2
th.mu=(y/sigma2+mu/tau2)/D
th.sd=sqrt(1/D)
theta=rnorm(J,th.mu,th.sd)
thmn <- ?; Pr.thmn <- ?; Pr.mmu <- ?
mu.mean <- ?
mu=rnorm(1,?,?)
invtau2=rgamma(1,J/2+0.1,sum((theta-mu)^2)/2+0.1)
tau2 <- 1/invtau2; tau <- sqrt(tau2)
MCMCsamp[t,1] =mu; MCMCsamp[t,2] =tau
MCMCsamp[t,3:(2+J)] = theta}
write.table(MCMCsamp, file="MCMCsamp.txt")
```

What is the effect of the revised prior for μ on its own posterior standard deviation, previously obtained as 4.4.

Example 1.5 Poisson-Gamma Model for Pumps Data
As mentioned above, one does not necessarily apply the same sampling procedure for each parameter. Here we exemplify this, and apply a Metropolis sampling step within Gibbs sampling to the overdispersed count data Poisson-gamma model set out in section 1.4. Thus the likelihood and priors are

$y_i \sim Poisson(\mu_i h_i); \mu_i \sim Ga(\alpha, \beta), \qquad i = 1, ..., n$

$\alpha \sim E(A_0)$; $\beta \sim Ga(B_0, C_0)$

where A_0, B_0 and C_0 are preset constants. So the posterior density for $\theta = (\mu_1, ..\mu_n, \alpha, \beta)$ given y is proportional to

$$e^{-A_0\alpha}\beta^{B_0-1}e^{-C_0\beta}[\beta^\alpha/\Gamma(\alpha)]^n \prod_{i=1}^{n} e^{-\mu_i h_i}\mu_i^{y_i}\{\mu_i^{\alpha-1}e^{-\beta\mu_i}\}$$

The full conditional densities of μ_i and β are gamma, namely

$\mu_i \sim Ga(y_i + \alpha, \beta + h_i)$

$\beta \sim Ga(b_1 + n\alpha, b_2 + \sum_{i=1}^{n}\mu_i)$

but the full conditional density of α is non-standard, namely

$$p(\alpha|y, \beta, \mu) \propto e^{-a_0\alpha}[\beta^\alpha/\Gamma(\alpha)]^n \prod_{i=1}^{n} \mu_i^{\alpha-1} \qquad (1.7).$$

The pumps data from the WINBUGS14 examples fits this template. In this data, h_i denotes the length of operation time of the pump (in 1000s of hours) and y_i is the number of failures. We use the full conditional (1.7) for α in a Metropolis sampling step. Preset parameters are $B_0 = 0.1, C_0 = 1$, and $A_0 = 1$. A uniform proposal $U(-\kappa, \kappa)$ with $\kappa = 0.5$, and with a restriction to positive values of α_{cand}, is adopted to sample new values of α. In the R code the restriction is imposed via the if statement

```
if(alphcand > 0).
```

Thus

```
# pumps data
h = c(94.3, 15.7, 62.9, 126, 5.24, 31.4, 1.05, 1.05, 2.1, 10.5)
y = c( 5, 1,5,14, 3,19, 1, 1, 4,22)
# settings
n <- 10; B=1000;T=10000; B0=0.1; C0=1; A0=1; kap=0.5
alph <- rep(0,T+1); beta<- rep(0,T+1); mu <- matrix(0, ncol=n, nrow=T+1)
alph[1] <-1; beta[1] <- 1; ACC <- 0
# sampling loop
for (t in 2:(T+1)) {# sample mu from gamma posterior conditional
mu[t,] <- rgamma(n, shape=alph[t-1]+y, rate= beta[t-1] + h)
beta[t] <- rgamma(1, shape=alph[t-1]*n+B0, rate=sum(mu[t,])+C0)
alphcand <- runif(1,alph[t-1]-kap, alph[t-1]+kap)
# logRnum and logRden are logs of the full conditional for alpha
if(alphcand > 0){logRnum<- n*alphcand*log(beta[t])-n*lgamma(alphcand)+
alphcand*sum(log(mu[t,])) - alphcand*A0
logRden<- n*alph[t-1]*log(beta[t])-n*lgamma(alph[t-1])+
alph[t-1]*sum(log(mu[t,])) - alph[t-1]*A0
u <- runif(1)
if(u < exp(logRnum - logRden)){ alph[t]<-alphcand; ACC<-ACC+1}
else{ alph[t]<-alph[t-1]}}
else{ alph[t]<-alph[t-1]}}
# acceptance rate, alpha
print(ACC/T);
# summary alpha,beta
summary(alph[B:T]); summary(beta[B:T])
```

Note that the multiplier for sum(log(mu[t,])) is alphcand and alph[t-1], rather than (alphcand-1) and alph[t-1]-1, because sum(log(mu[t,])) is not a function of alpha. So the term

-sum(log(mu[t,]))

will cancel out in comparing logRnum and logRden.

Exercise 1.10 Obtain 90% credible intervals for α and β using the above code. One possibility is to use the quantile function, e.g. quantile(alph[B:T], probs=..)

Exercise 1.11 Amend the above code to re-estimate the model under an alternative gamma prior $Ga(A_1, A_2)$ prior for α, with $A_1 = 0.1$,$A_2 = 0.1$. You will need to consider how the posterior density changes, and so how logRnum and logRden will change. How does the new prior affect 90% credible intervals for α and β.

Exercise 1.12 Revert to the original exponential prior for α, namely $\alpha \sim E(A_0)$ with $A_0 = 1$. Obtain the acceptance rate subsequent to iteration 1000 under a uniform proposal for α, but now with $\kappa = 0.75$. Suggest what would happen if we set $\kappa = 1$.

Exercise 1.13 Suggest how, using the function for an absolute value in R, one might achieve restriction to positive values of α_{cand} under the uniform proposal $U(-\kappa, \kappa)$.

1.6 MCMC Convergence and Ways of Improving Convergence

It is necessary to decide how many iterations to use to

a) accurately represent the posterior density and

(b) to ensure that the sampling process has converged.

As mentioned above MCMC iterations are not independent and successive samples are correlated. Nonvanishing autocorrelations at high lags (e.g. lag 5 or 10) between successive samples mean that less information about the posterior distribution is provided by each iterate, and a higher sample size is necessary to cover the parameter space. Autocorrelation will be reduced by "thinning", namely retaining only sample that are $S > 1$ steps apart $\{\theta_h^{(t)}, \theta_h^{(t+S)}, \theta_h^{(t+2S)}, \ldots\}$ that more closely approximate independent samples; however, this results in a loss of precision in the posterior estimates. The autocorrelation present in MCMC samples may depend on the form of parameterisation, the complexity of the model, and the form of sampling (e.g. block or univariate sampling for collections of random effects).

Also useful for assessing the efficiency of MCMC sampling is the Monte Carlo standard error (obtainable from WINBUGS output), which is an estimate of the standard deviation of the difference between the true posterior mean $E(\theta_h|y) = \int \theta_h p(\theta|y) d\theta$ and the simulation based estimate

$$\bar{\theta}_h = \frac{1}{T} \sum_{t=B+1}^{T+B} \theta_h^{(t)}.$$

The ratio of the posterior variance in a parameter, namely

$$Var(\theta_h) = \frac{1}{T} \sum_{t=B+1}^{T+B} (\theta_h^{(t)} - \bar{\theta}_h)^2$$

to its Monte Carlo variance is a measure of the efficiency of the Markov chain sampling (Roberts, 1996). Both quantities are routine outputs from WINBUGS (Lesson 2). It is sometimes suggested that the MC standard error should be less than 5% of the posterior standard deviation of a parameter (Toft et al, 2007).

As to the second issue mentioned above, there is no guarantee that sampling from an MCMC algorithm will converge to the posterior distribution, despite obtaining a high number of iterations. This is especially so when sampling is inefficient (high autocorrelations, or model not well identified). Convergence can be informally assessed by examining the time series or trace plots of parameters. Slow convergence will show in trace plots (see the "trace" command in the WINBUGS "sample monitor tool") that wander, and that exhibit short term trends, rather than fluctuating rapidly around a stable mean. Failure to converge is typically partial in terms of the model parameters; for example, fixed regression effects in a general linear mixed model may show convergence, but not the parameters relating to the random components. Often measures of overall fit (e.g. model deviance) may converge while particular parameters in a model do not.

Problems of convergence in MCMC sampling may reflect problems in model identifiability, either formal nonidentification as in multiple random effects models, or poor empirical identifiability when an overly complex model is applied to a small sample ('over-fitting'). Choice of diffuse priors tends to increase the chance that models are poorly identified, especially in complex models for small data samples (Gelfand and Sahu, 1999). Elicitation of more informative priors and/or application of parameter constraints may assist identification and convergence. Another source of poor convergence is suboptimal parameterisation or data form; for example, convergence is improved by centering independent variables in regression applications (Zuur et al, 2002).

1.7 Assessing Convergence under Multiple Chain Approaches

Many practitioners prefer to use two or more parallel chains with diverse starting values to ensure full coverage of the sample space of the parameters (Gelman & Rubin, 1996; Toft et al, 2007). Diverse starting values are more likely to reveal convergence or identification problems. One possibility is to take the 5% and 95% posterior points from an exploratory run to provide initial values for two chains; or one might take the means, 5% and 95% posterior points from an exploratory run to provide initial values for three chains. On line monitoring of sampled parameter values $\{\theta_j^{(t)}, t = 1,..,T\}$ from chains $j = 1,..,J$ assists in diagnosing lack of model identifiability. Single runs may still be be adequate for straightforward problems, and single chain convergence diagnostics (Cowles & Carlin, 1996) may be applied in this case. Single runs are often useful for exploring the posterior density and as a preliminary to obtain inputs to multiple chains.

Convergence for multiple chains may be assessed using Gelman-Rubin scale reduction factors that measure the convergence of the ratio of the between chain variance in $\theta_j^{(t)}$ to the variance over all chains. These factors converge to 1 if all chains are sampling identical distributions, whereas for poorly identified models variability of sampled parameter values between chains will considerably exceed the variability within any one chain. To apply these criteria one typically allows a short burn-in of B samples while the sampling moves away from the initial values to the region of the posterior. Let $\theta_{jh}^{(t)}$ be sampled values of a parameter θ_h in chain j. The posterior variance matrix $var_{\theta_h|y}$ of θ_h is estimated as

$$var_{\theta_h|y} = B_h + \tfrac{T}{T-1} W_h$$

where

$$W_h = \tfrac{1}{(T-1)J} \sum_{j=1}^{J} \sum_{t=B+1}^{B+T} (\theta_{jh}^{(t)} - \bar{\theta}_{jh})^2$$

denotes variability within chains,

$$B_h = \tfrac{1}{J-1} \sum_{j=1}^{J} (\bar{\theta}_{jh} - \bar{\theta}_h)^2$$

denotes between chain variability, and

$$\bar{\theta}_h = \tfrac{1}{J} \sum_{j=1}^{J} \bar{\theta}_{jh}$$

denotes the pooled average of the posterior means $\bar{\theta}_{jh}$of θ_h in chain j. The potential scale reduction factor compares $var_{\theta|y}$ with the within sample estimate W_h. Specifically the scale factor is $F_h = (var_{\theta_h|y}/W_h)^{0.5}$ with values under 1.2 indicating convergence.

An alternative multiple chain convergence criterion proposed by Brooks and Gelman (1998) avoids reliance on the implicit normality assumptions in the Gelman-Rubin scale reduction factors when assessing mixing of sequences by monitoring means and variances; see the "bgr diag" option in the WINBUGS "Sample Monitor Tool". The normality approximation may be improved by parameter transformation (e.g. log or logit), but problems may still be encountered when posterior densities are skewed or possibly multimodal (Toft et al, 2007). The alternative criterion uses a ratio of parameter interval lengths: for each chain the length of the $100(1-\alpha)\%$ interval for a parameter is obtained, namely the gap between the 0.5α and $(1-0.5\alpha)$ points from T simulated values. This provides J within-chain interval lengths, with mean L_U. From the pooled output of TJ samples, the same interval L_P is also obtained. The ratio L_P/L_U should converge to 1 if there is convergent mixing over the J chains.

Example 1.6 Multiple Chains Using R2WINBUGS
To illustrate use of WINBUGS (followed up in Lesson 2) but retain the advantages of working in *R* (e.g. graphical presentations), one may use the R2WinBUGS facility (Sturtz et al, 2005) which links R to WINBUGS1.4. This includes a simple way of running multiple chains and includes the Brooks and Gelman (1998) convergence statistics. The interface to bugs from R involves specifying data, initial parameter values (inits for short), names of unknown parameters, and the form of the model as well as number of chains J, total iterations T, and length of burn-in B. One option is to save the winbugs model code (and possibly the data also) to the *R* working directory. Another option is to specify the winbugs code in *R* using the cat command. For guidance, one may refer to http://www.stat.columbia.edu/~gelman/bugsR/runningbugs.html.

To illustrate a regression analysis with three chains, consider data relating male life expectancy y_i (in 2003-05) to a health and disability score x_i for 33 London local authorities (expectancy data from http://www.nchod.nhs.uk/; deprivation data from http://www.communities.gov.uk/co

Consider the quadratic model $y_i \sim N(\mu_i, 1/\tau), \mu_i = b_1 + b_2 x_i + b_3 x_i^2$, with N(0,1000) priors for b_j, and a gamma prior for the inverse variance $\tau \sim Ga(1, 0.01)$. One may specify initial values by generating them randomly or by pre-setting them. In the example code below both options are possible.

One needs to have firstly installed the R2WinbUGS package. It is assumed that the WINBUGS14 package is located at c:/Program Files/WinBUGS14/. Using the cat function, and with $J = 3$ chains, one possible command sequence (with data included) is then as follows:

```
library(R2WinBUGS)
y=c(82.1,75.3,78.6,78.3,77.4,78.6,75.7,77.5,76.9,77.1,74.8,75.1,76.8,75.2,78.7,77.8,
77.3,76.2,74.6,82.2,78.5,74.9,75.1,78,74.9,77.5,79.3,75.5,77.9,74.9,75.4,76.3,78.9)
x=c(-0.93,0.59,-0.62,-0.57,0.05,-0.7,0.41,-0.28,0.04,0.02,0.51,0.91,0.25,0.54,-0.53,-0.35,
-0.26,0.08,0.93,-1.09,-0.99,0.58,0.39,-0.77,0.89,-0.13,-1.34,0.47,-0.66,0.98,0.41,0.09,-0.33)
data <- list("x","y")
# winbugs model code in next 4 lines named expecs.bug
cat("model { for (i in 1:33) {y[i] ~dnorm(mu[i],tau)
mu[i] <- b[1]+b[2]*x[i]+b[3]*x[i]*x[i]}
for (j in 1:3) {b[j] ~dnorm(0,0.001)}
tau ~dgamma(1,0.01)}", file="expecs.bug")
# initialise unknowns randomly
# inits <- function(){list(b = rnorm(3, 0, 100), tau=rgamma(1,1)) }
# initialise unknowns via preset values
```

```
inits1 <- list(b=c(70,0,0),tau=1); inits2 <- list(b=c(80,0,0),tau=2)
inits3 <- list(b=c(90,0,0),tau=3); inits <- list(inits1, inits2,inits3)
# name parameters and set total iterations
 pars=c("b","tau"); T <- 10000; B <- 1000
# interface to bugs
expecs.sim <- bugs(data,inits, pars,
model="expecs.bug",n.chains=3,n.iter=T,n.burnin=B,n.thin=1)
# posterior summary
expecs.sim
# plots, etc
attach.all(expecs.sim$sims.list)
summary(b[,1]); summary(b[,2]); summary(b[,3])
# graphical summary of B-G-R convergence statistics and posterior density
plot(expecs.sim)
```

Exercise 1.14
Rerun the analysis in Example 1.6 with five chains and with initial values generated by random sampling. For the b regression coefficients sample from a Normal N(0,9) density (normal with mean 0 and variance 9), and for the inverse variance τ use a gamma(10,10) density. Note that these are not priors but just densities used to generate initial values.

References

Albert J (2007) Bayesian Computation with R. Springer
Andrieu C, Moulines E (2006) On the ergodicity properties of some adaptive MCMC algorithms. Ann. Appl. Probab. 16(3): 1462-1505
Brooks, S, Gelman, A (1998) Alternative methods for monitoring convergence of iterative simulations. J. Comp. Graph. Stats, 7, 434-456
Carlin B, Louis T (2000) Bayes and Empirical Bayes Methods, 2nd ed, Chapman & Hall/CRC
Chib S, Greenberg E (1995) Understanding the Metropolis-Hastings algorithm. The American Statistician 49, 327–335
Cowles M, Carlin B (1996) Markov chain Monte Carlo convergence diagnostics: a comparative review. J. Am. Statist. Ass., 91, 883-904
Gelfand A, Sahu S (1999) Identifiability, improper priors, and Gibbs sampling for generalized linear models. Journal of the American Statistical Association. 94, 247–253
Gelman A, Rubin D (1996) Markov Chain Monte Carlo Methods in Biostatistics. Stat Methods Med Res, 5:339-55.
Gelman A, Carlin J, Stern H, Rubin D (2004) Bayesian Data Analysis, 2nd edition. Chapman&Hall/CRC
George E, Makov U, Smith A (1993) Conjugate likelihood distributions. Scand. J Statist, 20, 147- 156
Gilks W, Richardson S, Spielgelhalter D (1996) Introducing Markov Chain Monte Carlo. In Markov Chain Monte Carlo in Practice, Gilks, W, Richardson, S, Spiegelhalter, D (eds)

Chapman and Hall: London, 1-19
Gilks, W (1996) Full conditional distributions. In Markov Chain Monte Carlo in Practice, Gilks, W, Richardson, S, Spiegelhalter, D (eds) Chapman and Hall: London, 75-88
Hastings, W (1970) Monte-Carlo sampling methods using Markov Chains and their applications. Biometrika, 57, 97–109
Lynch S (2007) Introduction to Applied Bayesian Statistics and Estimation for Social Scientists.. New York: Springer
Metropolis N, Rosenbluth A, Teller A, Teller E (1953) Equations of state calculations by fast computing machines. Journal of Chemical Physics, 21, 1087–1092.
Roberts G, Tweedie R (1996) Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms, Biometrika
Roberts G, Rosenthal J (2004) General state space Markov chains and MCMC algorithms, Probability Surveys, 1, 20-71
Roberts G, Gelman A, Gilks W (1997) Weak convergence and optimal scaling of random walk metropolis algorithms, The Annals of Applied Probability, 7, 110-120
Sinharay S, Stern H (2003) Posterior predictive model checking in hierarchical models, Journal of Statistical Planning and Inference, 111: 209 – 221
Sturtz S, Ligges U, Gelman A (2005) R2WinBUGS: A Package for Running WinBUGS from R. Journal of Statistical Software, 12(3), 1-16
Toft N, Innocent G, Gettinby G, Reid S (2007) Assessing the convergence of Markov Chain Monte Carlo methods: an example from evaluation of diagnostic tests in absence of a gold standard. Preventive Veterinary Medicine, 79: 244-256
Zuur G, Garthwaite P, Fryer R (2002) Practical use of MCMC methods: lessons from a case study, Biometrical Journal, 44, 433 - 455

Appendix 1 **R Distribution Syntax**

Normal

rnorm(n) Generates n standard normal random variables
rnorm(n,mu,sigma) Generates n normals with mean mu and standard deviation sigma
dnorm(x) Value of standard normal at point x
dnorm(x,mu,sigma) Value of normal with mean mu and standard deviation sigma

Uniform

runif(n, min=a, max=b) Generates n Uniform(a,b) random values

Chi-Square χ^2

rchisq(n,df) Generates n central χ^2_{df} random variables
dchisq(x,df) Value of central χ^2_{df} at x
rchisq(n,df,ncp) Generates n noncentral χ^2_{df} random variables
dchisq(x,df,ncp) Value of a noncentral χ^2_{df} at x

Exponential

rexp(n,lam) Generates n exponential variables with rate lam
dexp(x,lam) Value at x of an exponential with rate lam

Beta
rbeta(n,shape1,shape2) Generates n beta variables with parameters shape1, shape 2
dbeta(x,shape1,shape2) Value at x for beta with parameters shape1, shape 2

Gamma(shape=α,scale=β) $\frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x}$
rgamma(n,shape, scale) n gamma random variables with specified shape and scale parameters
dgamma(x,shape, scale) Value at x with specified shape and scale parameters

Binomial
rbinom(k,n,p) Generates k Binomial(n,p) random variables
dbinom(k,n,p) Value at k of Binomial with parameters (n,p)

Geometric
rgeom(n,p) n Geometric RVs with success parameter p
dgeom(k,p) Value at k of Geometric with parameter p

Poisson
rpoism(n,lam) n Poisson RVs with parameter lam
dpoism(k,lam) Value at k of Poisson with parameter lam

Standard Logistic distribution
rlogis(n, location = 0, scale = 1)
dlogis(x, location = 0, scale = 1, log = FALSE)

Appendix 2 Alternative MCMC Strategies for Extended Logistic Regression

Carlin & Louis (2000, p 155) present an extended logistic model for the Bliss binomial mortality data, involving death rates p_i at dose w_i. Thus for deaths y_i one has
$y_i \sim Bin(n_i, p_i)$
$p_i = h(w_i) = [exp(z_i)/(1 + exp(z_i)]^{m_1}$
$z_i = (w_i - \mu)/\sigma$
where m_1 and σ are both positive. To avoid excessive notation we write $V = \sigma^2$.

We consider three options, one involving log transforms of m_1 and V, and separate univariate normal proposals in a Metropolis scheme. The second also involves log transforms of m_1 and V but uses a single multivariate normal proposal in a Metropolis scheme. The third approach reverts to separate sampling of each parameter but adopts a gamma Metropolis-Hastings proposal scheme so that m_1 and V do not have to be transformed.

Under the first option, Jacobian adjustments are needed in the posterior density to account for the two transformed parameters. The full posterior $p(\mu, m_1, V|y)$ is proportional to

$$p(\mu)p(m_1)p(V)\prod_i [h(w_i)]^{y_i}(1-h(w_i)]^{n_i-y_i}$$

where $p(\mu)$, $p(m_1)$ and $p(V)$ are priors for μ, m_1 and V. Suppose the priors $p(\mu), p(m_1)$ and $p(V)$ are as follows:

$\mu \sim N(c_0, d_0^2)$;

$m_1 \sim Ga(a_0, b_0)$ with $a_0 = b_0 = 0.25$ and the gamma has the form

$Ga(x|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1}e^{-\beta x}$;

$V \sim InvGamma(e_0, f_0)$ and the inverse gamma has the form

$Ga(x|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{-(\alpha+1)}e^{-\beta/x}$.

Following Carlin and Louis (2000), the preset parameters (a_0,b_0,c_0,d_0,e_0,f_0) are (a_0=0.25,b_0=0.25,c_0=2,d_0=1
e_0=2.000004,f_0=0.001). The posterior is proportional to

$$(m_1^{a_0-1}e^{-b_0 m_1})\frac{1}{(d_0)^{0.5}}exp(-0.5[\frac{\mu-c_0}{d_0}]^2)V^{-(e_0+1)}e^{-f_0/V}\prod_i [h(w_i)]^{y_i}(1-h(w_i)]^{n_i-y_i}$$

Suppose the likelihood is re-specified in terms of parameters $\theta_1 = \mu, \theta_2 = log(m_1)$ and $\theta_3 = log(V)$. Then the full posterior in terms of the transformed parameters is proportional to

$$(\frac{\partial m_1}{\partial \theta_2})(\frac{\partial V}{\partial \theta_3})p(\mu)p(m_1)p(V)\prod_i [h(w_i)]^{y_i}(1-h(w_i)]^{n_i-y_i}.$$

One has $(\frac{\partial m_1}{\partial \theta_2}) = e^{\theta_2} = m_1$ and $(\frac{\partial V}{\partial \theta_3}) = e^{\theta_3} = V$. So taking account of the parameterisation $(\theta_1, \theta_2, \theta_3)$, the posterior density is proportional to

$$(m_1^{a_0}e^{-b_0 m_1})\frac{1}{(d_0)^{0.5}}exp(-0.5[\frac{\mu-c_0}{d_0}]^2)V^{-e_0}e^{-f_0/V}\prod_i [h(w_i)]^{y_i}(1-h(w_i)]^{n_i-y_i}.$$

In the R code, we assume initial values for $\mu = \theta_1$ of 1.8, for $\theta_2 = log(m_1)$ of 0, and for $\theta_3 = \log(V)$ of 1. The number of iterations is $T = 50000$. We assume normal proposal densities with standard deviations 0.01, 0.1, and 0.1. Metropolis updates involve comparisons of the log (un-normalized) posterior and logged uniform random variables. The R code is then

```
f = function(mu,th2,th3) {V <- exp(th3); m1 <- exp(th2); sig <- sqrt(V)
x <- (w-mu)/sig; xt <- exp(x)/(1+exp(x)); h <- xt^m1;
loglike <- y*log(h)+(n-y)*log(1-h)
logpriorm1 <- a0*th2-m1*b0
logpriorV <- -e0*th3-f0/V
logpriormu <- -0.5*((mu-c0)/d0)^2-0.5*log(d0)
logprior <- logpriormu+logpriorV+logpriorm1
f <- sum(loglike)+logprior}
# Read in data and set initial values and vectors
w = c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839)
n = c(59, 60, 62, 56, 63, 59, 62, 60); y = c(6, 13, 18, 28, 52, 53, 61, 60)
T = 50000; mu <- numeric(T); th3 <- numeric(T); th2 <- numeric(T);
V <- numeric(T); m1 <- numeric(T); samp <- matrix(0, nrow=T, ncol=3);
```

```
pm <- numeric(3)
MCMCsamp <- matrix(0, nrow=T, ncol=3)
# initial parameter values
mu[1] <- 1.8; th2[1] <- 0; th3[1] <- 1;
k1 = 0; k2 = 0; k3 = 0; u1 <- runif(T); u2 <- runif(T); u3 <- runif(T)
a0=0.25; b0=0.25; c0=2; d0=10; e0=2.004; f0=0.001
# metropolis proposal standard devn's
sd1 <- 0.01; sd2 <- 0.1; sd3 <- 0.1
# main MCMC loop
for (i in 2:T) {mucand <- mu[i-1]+sd1*rnorm(1,0,1)
tcand <- f(mucand,th2[i-1],th3[i-1])
tcurr <- f(mu[i-1],th2[i-1],th3[i-1])
if (log(u1[i]) <= tcand-tcurr) mu[i] <- mucand else
{mu[i] <- mu[i-1]; k1 <- k1+1 }
th2cand <- th2[i-1]+sd2*rnorm(1,0,1)
tcand <- f(mu[i],th2cand,th3[i-1])
tcurr <- f(mu[i],th2[i-1],th3[i-1])
if (log(u2[i]) <= tcand-tcurr) th2[i] <- th2cand else
{th2[i] <- th2[i-1]; k2 <- k2+1 }
m1[i] <- exp(th2[i])
th3cand <- th3[i-1]+sd3*rnorm(1,0,1)
tcand <- f(mu[i],th2[i],th3cand)
tcurr <- f(mu[i],th2[i],th3[i-1])
if (log(u3[i]) <= tcand-tcurr) th3[i] <- th3cand else
{th3[i] <- th3[i-1]; k3 <- k3+1}
V[i] <- exp(th3[i])
samp[i-1,1] <- mu[i]; samp[i-1,2] <- m1[i]; samp[i-1,3] <- V[i]}
# output for posterior data analysis
write.table(samp, file="MCMCsamp.txt")
# posterior summary (iterations 1000 to T=10,000)
quantile(mu[1000:T], probs=c(.025,0.5,0.975))
quantile(m1[1000:T], probs=c(.025,0.5,0.975))
quantile(V[1000:T], probs=c(.025,0.5,0.975))
# acceptance rates
1-k1/T; 1-k2/T; 1-k3/T
```

With an initial burn-in of 1000 iterations, the posterior medians for μ, m_1 and V are 1.81, 0.367 and 3.5E-04. Acceptance rates for μ, m_1 and V are 0.41, 0.65, and 0.81, so there is scope for tuning to reduce the acceptance rates for m_1 and V.

We next consider the multivariate normal proposal option, denoting $\theta = (\mu, log(m_1), log(\sigma))$.

Remember that for multivariate normal sampling the R functions use the covariance matrix. Hence the settings adopted for the diagonal elements (variances) in the MVN propsal covariance matrix are important for determining the MCMC acceptance rate. Under the multivariate normal proposal option the R code is

```
f = function(w,n,y,mu,logm1,logsig,a0,b0,c0,d0,e0,f0) {
sig <- exp(logsig); m1 <- exp(logm1); sig2 <- sig^2
x <- (w-mu)/sig; xt <- exp(x)/(1+exp(x))
h <- xt^m1; loglike <- y*log(h)+(n-y)*log(1-h)
logpriorm1 <- a0*logm1-m1/b0
logpriorsig2 <- -2*e0*logsig-1/(f0*sig2)
logpriormu <- -0.5*((mu-c0)/d0)^2-0.5*log(d0)
logprior <- logpriormu+logpriorsig2+logpriorm1
f <- sum(loglike)+logprior}
# Needed for MVN sampling
library(mvtnorm)
# Read in data and set initial values and vectors
w = c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839)
n = c(59, 60, 62, 56, 63, 59, 62, 60); y = c(6, 13, 18, 28, 52, 53, 61, 60)
T = 10000; mu <- numeric(T); logsig <- numeric(T);
mu <- numeric(T); sig <- numeric(T); m1 <- numeric(T);
samp <- matrix(0,T,3); thcand=numeric(3)
sigma <- matrix(c(0.0005,0,0, 0,0.0075,0, 0,0,0.0075), ncol=3)
th <- matrix(0, 3, T)
# initial parameter values
th[1,1] <- 1.8; th[2,1] <- -1; th[3,1] <- -1
k= 0; u <- runif(T);
a0=0.25; b0=4; c0=2; d0=10; e0=2.004; f0=1000
for (i in 2:T) { thcand <- rmvnorm(1,th[,i-1],sigma)
tcand <- f(w,n,y,thcand[1], thcand[2], thcand[3],a0,b0,c0,d0,e0,f0)
tcurr <- f(w,n,y,th[1,i-1],th[2,i-1],th[3,i-1],a0,b0,c0,d0,e0,f0)
if (log(u[i]) <= tcand-tcurr) th[,i] <- thcand[] else
{th[,i] <- th[,i-1]; k <- k+1 }
mu[i] <- th[1,i]; m1[i] <- exp(th[2,i]); sig[i] <- exp(th[3,i])
samp[i-1,1] <- mu[i]; samp[i-1,2] <- m1[i]; samp[i-1,3] <- sig[i]}
write.table(samp, file="MCMCsamp.txt")
# posterior summary
summary(mu[1000:T]); summary(m1[1000:T]);summary(sig[1000:T])
# acceptance rate
1-k/T
```

A final MCMC analysis of the beetle mortality data involves separate sampling of each parameter and a gamma Metropolis-Hastings proposal scheme for m_1 and $V = \sigma^2$. Since parameters do not have to be transformed, there is no need for a Jacobian adjustment in the posterior density. Specifically the proposal

$q(y|x) = Gam(\alpha, \alpha/x) = \frac{\alpha^\alpha}{x^\alpha \Gamma(\alpha)} y^{\alpha-1} e^{-\alpha y/x}$

is used. Note that α has a role as a precision parameter that can be tuned to provide improved acceptance rates: larger values of α mean that the sampled candidate values

$x_{cand} \sim Gam(\alpha, \alpha/x_{curr})$

will be more closely clustered about the current parameter value x_{curr}.

The code is

```
f = function(mu,m1,sig2) {x <- (w-mu)/sqrt(sig2); xt <- exp(x)/(1+exp(x))
h <- xt^m1; loglike <- y*log(h)+(n-y)*log(1-h)
logpriorm1 <- (a0-1)*log(m1)-m1/b0
logpriorsig2 <- -(e0+1)*log(sig2)-1/(f0*sig2)
logpriormu <- -0.5*((mu-c0)/d0)^2-0.5*log(d0)
logprior <- logpriormu+logpriorsig2+logpriorm1
f <- sum(loglike)+logprior}
w = c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839)
n = c(59, 60, 62, 56, 63, 59, 62, 60); y = c(6, 13, 18, 28, 52, 53, 61, 60)
k = 8; T = 10000; mu <- numeric(T); sig2 <- numeric(T); m1 <- numeric(T);
sig <- numeric(T); pm <- numeric(3)
mu[1] <- 1.8; m1[1] <- 0.5; sig2[1] <- 0.001
k1 = 0; k2 = 0; k3 = 0; u1 <- runif(T); u2 <- runif(T); u3 <- runif(T)
a0=0.25; b0=4; c0=2; d0=10; e0=2.004; f0=1000
alphm1=10; alphsig2=10
for (i in 2:T) { mucand <- mu[i-1]+0.01*rnorm(1,0,1)
tcand <- f(mucand,m1[i-1],sig2[i-1])
tcurr <- f(mu[i-1],m1[i-1],sig2[i-1])
if (log(u1[i]) <= tcand-tcurr) mu[i] <- mucand else
{mu[i] <- mu[i-1]; k1 <- k1+1 }
m1cand <- rgamma(1,alphm1,alphm1/m1[i-1])
fcand <- f(mu[i],m1cand,sig2[i-1])
fcurr <- f(mu[i],m1[i-1],sig2[i-1])
tcand <- fcand+log(dgamma(m1[i-1],alphm1,alphm1/m1cand))
tcurr <- fcurr+log(dgamma(m1cand,alphm1,alphm1/m1[i-1]))
if (log(u2[i]) <= tcand-tcurr) m1[i] <- m1cand else
{m1[i] <- m1[i-1]; k2 <- k2+1 }
sig2cand <- rgamma(1,alphsig2, alphsig2/sig2[i-1]);
fcand <- f(mu[i],m1[i],sig2cand)
fcurr <- f(mu[i],m1[i],sig2[i-1])
tcand <- fcand+log(dgamma(sig2[i-1], alphsig2,alphsig2/sig2cand))
tcurr <- fcurr+log(dgamma(sig2cand, alphsig2,alphsig2/sig2[i-1]))
if (log(u3[i]) <= tcand-tcurr) sig2[i] <- sig2cand else
{sig2[i] <- sig2[i-1]; k3 <- k3+1 }
sig[i] <- sqrt(sig2[i])}
# posterior means
pm[1] <- mean(mu[1000:T]); pm[2] <- mean(m1[1000:T])
pm[3] <- mean(sig[1000:T]); pm
# acceptance rates
1-k1/T; 1-k2/T; 1-k3/T
```