The Metropolis-Hastings Sampler in R

Version 21 April 2004

Examples 2 and 3 in the MCMC notes deals with generating a Metropolis-Hastings sampler for a scaled inverse χ^2 distribution. Here we show how to code this in **R**. The reader may find it helpful to also have these MCMC notes in front of them while reading this.

Recall that we can write the probability distribution for a scaled inverse χ^2 as

$$p(x) = \text{constant} \cdot x^{-n/2} \cdot \exp\left(\frac{-a}{2x}\right)$$

where n is the degrees of freedom and a the scaling parameter. (Recall that the inverse χ^2 distribution is a special case of the inverse gamma distribution and arises as a natural prior for the variance in a normal likelihood, see Bayesian notes). We will assume $n = 5$ and $a = 4$ in the following notes. First, we need to define a function **`f(x,n,a)`** in **R**,

```
> f <- function(x,n,a) x^(-n/2)*exp(-a/(2x))
```

which is the density function (up to the constant) for our inverse χ^2 distribution.

Metropolis Sampler in R

To obtain a Metropolis Sampler, recall (MCMC notes) the various stages in the sampler:

(i) Start with some initial value, x_0

(ii) Given this initial value, draw a candidate value x^* from some proposal distribution

(iii) Compute the ratio α of the density at the candidate and initial points, $\alpha = f(x^*)/f(x_0)$

(iv) With probability min(α,1), accept the candidate point, else retain the current point

(v) Return to step (ii)

Now let's implement these steps. As in Example 2 (MCMC notes), we will use a uniform distribution over $(0, 100)$ as the proposal distribution. First, we need to specific a vector for the chain. Lets call it **`met`** and give it length 5000

```
> met <- numeric(5000)
```

Lets now start the chain, say at value 1. We will call the current value in the chain **`last`**,

```
> last <- 1
```

Now let's run the chain 5000 times. To do so, recall that in **R** we can obtain a single random variable from uniform distribution over (a, b) with the code **`runif(1,a,b)`**. The default case is over $(0, 1)$, so that generating a single uniform random variable over $(0, 1)$ can be coded as either **`runif(1,0,1)`** or (more compactly) as **`runif(1)`**. [If we wished (say) 5 uniform $(0, 1)$, we would use **`runif(5)`**.]

We now have all the pieces for the code, which is

```
> for (i in 1:5000) {
cand<-runif(1,0,100)
alpha <- f(cand,5,4)/f(last,5,4)
if  (runif(1) < min(alpha, 1))  last <- cand
met[i]<- last}
```

Note that the starting and ending parentheses ({, }) in the **for** statement are critical!

The command **cand<-runif(1,0,100)** generates a single uniform random variable over $(0, 100)$ and assigns this value to the variable **cand**. This is step (ii).

We compute the ratio of the two inverse χ^2 (with $n = 5$ degrees of freedom and scale parameter $a = 4$) using **alpha <- f(cand,5,4)/f(last,5,4)**. This accomplishes step (iii).

To accomplish step (iv), accept the candidate with probability min(α, 1), we use the command **if (runif(1) < min(alpha, 1)) last <- cand** , which says that if a uniform $(0, 1)$ random variable is less than the probability of acceptance, we accept, otherwise the value of **last** remains unchanged.

We can view that histogram for this sample using

```
> hist(met)
```

Likewise, the current value of the chain as a function of the iteration (the **time series** for the chain) can be obtained using the **plot** command,

```
> plot(met)
```

Notice that we did not "burn-in" the sampler. This could easily be done by not storing the first k values (where k is our burn-in value), and then generating 5000 additional values.

Metropolis-Hastings in R

The implementation of the Metropolis-Hastings sampler is almost identical to the strict Metropolis sampler, except that the proposal distribution need no longer be symmetric. For example, if $\Pr(x_1 \to x_2) = \Pr(x_2 \to x_1)$ for all values of x_1 and x_2, then the proposal distribution is symmetric and Metropolis can be used. This is satisfied for a uniform (a, b) distribution as $\Pr(x_1 \to x_2) = \Pr(x_2) = 1/(b - a)$. Since the probability is independent of both x_1 and x_2, it is symmetric. [Note that $\Pr(x_1 \to x_2) = \Pr(x_2)$, the probability that we are at x_1 and we move to x_2, is independent of where we start, i.e., x_1, as we are drawing from a set distribution that does not depend on our current value x_1, but rather only on the value x_2 draw.]

Suppose, however, that our proposal distribution is to draw from a (normal) χ^2 distribution. Here $\Pr(x_1 \to x_2) = \Pr(x_2)$ where $\Pr(x_2)$ is the χ^2 distribution function evaluated at the value chosen, x_2. Since this clearly depends on the value of x_2, it is not symmetric. The Metropolis-Hastings sampler handles this as follows (again, see MCMC notes). Let $q(x_1, x_2)$ denote the probability of x_2 given the current value is x_1. The step (iii) of the normal Metropolis is modified as follows:

$$\alpha = \frac{f(x^*)\, q(x^*, x_0)}{f(x_0)\, q(x_0, x^*)}$$

Let's now code a Metropolis-Hastings sampler for the above inverse scaled χ^2 distribution using a normal χ^2 distribution with (say) 5 degrees of freedom as our proposal distribution to generate candidates. Recall that the command **`rchisq(1,n)`** generates a single χ^2_n random variable. Likewise, the command **`dchisq(x,n)`** returns the value a χ^2_n probability density evaluated at x.

Using these results, we need only change the two lines in the above Metropolis code from

```
> cand<-runif(1,0,100)
> alpha <- f(cand,5,4)/f(last,5,4)
```

to

```
> cand<-rchisq(1,5)
> alpha <- (f(cand,5,4)/f(last,5,4))*( dchisq(cand,5)/dchisq(last,5))
```

in order to change this into a Metropolis-Hasting sampler using a χ^2_5 proposal distribution.

For fun, run the above code for a Metropolis using the $(0, 100)$ uniform and compare the mixing of the chain (i.e. the time sereis displayed by **`plot`**) with a Metropolis-Hastings using a χ^2_5. You might also try using other degrees of freedom for the proposal χ^2 distribution (e.g. Example 3) to see what happens to the mixing.

Generating Random Variables and Values of Density Functions

Recall that if **`DIST`** represent a particualr distirbution, then the general synthax in **R** to generate a random variabel from that distirbution is

`rDIST(n, parameters)` generates n random variables from DIST(parameters).

Likewise,

`dDIST(x, parameters)` returns the value of the distribution DIST(parameters) at point x.

Distribution	Syntax
Normal	**`rnorm(n)`** Returns n unit normal random variables
	`rnorm(n,mu,sigma)` Returns n normals with mean mu and variance sigma
	`dnorm(x)` Value of unit normal at point x
	`dnorm(x,mu,sigma)` Value of normal with mean mu and variance sigma
Student's t	**`rt(n,df)`** Returns n t values with df degrees of freedom
	`pt(n,df)` Value at point x of t with df degrees of freedom
χ^2	**`rchisq(n,df)`** Returns n central χ^2_{df} random variables
	`pchisq(x,df)` Value of a central χ^2_{df} at x
	`rchisq(n,df,ncp)` Returns n noncentral χ^2_{df} random variables
	`pchisq(x,df,ncp)` Value of a noncentral χ^2_{df} at x
Exponential	**`rexp(n)`** n exponential with rate 1 random variables
	`pexp(x)` Value at x of an exponential with rate 1
	`rexp(n,lam)` n exponential with rate **lam** random variables
	`pexp(x,lam)` Value at x of an exponential with rate **lam**
Beta	**`rbeta(n,shape1,shape2)`** n beta with parameters shape1, shape 2
	`pbeta(x,shape1,shape2)` Value at x for beta with parameters shape1, shape 2
Gamma	**`rgamma(n,shape)`** n gamma with user-specified shape parameter
	`pgamma(x,shape)` Value at x of gamma with user-specified shape parameter
	`rgamma(n,shape, scale)` n gamma with user-specified shape, scale parameters
	`pgamma(x,shape, scale)` Value at x with user-specified shape, scale parameters
Binomial	**`rbinom(k,n,p)`** k Binomial n p random variables
	`pbinom(k,n,p)` Value at k of Binomial with parameters n, p
Geometric	**`rgeom(n,p)`** n Geometric RVs with success parameter p
	`pgeom(k,p)` Value at k of Geometric with parameter p
Poisson	**`rpoism(n,lam)`** n Poisson RVs with parameter **lam**
	`ppoism(k,lam)` Value at k of Poisson with parameter **lam**

Further details on any of the above can be found in **R** using the help command. For example, not sure what the beta shaper parameters are? Type

```
> help("pbeta")
```

and **R** returns a help file with all the required information.