Bertil Wegmann
Dept of Computer and Information Science
Linköping University

How to code up the Random Walk Metropolis Algorithm in R

This little note will help you with coding up the random walk Metropolis algorithm so that the same function, let's call it `RWMSampler`, can be applied to simulate from the posterior of the parameters in *any* model. The trick is to use *function objects* in `R`. Note the following:

1. One of the input arguments of your `RWMSampler` function should be `logPostFunc` (or some other suitable name). `logPostFunc` is a *function object* that computes the log posterior density at any value of the parameter vector. This is needed when you compute the acceptance probability of the Metropolis algorithm. I suggest **always** to program the *log* posterior density, since logs are more stable and avoids problems with too small or large numbers (overflow). Note that the ratio of posterior densities in the Metropolis acceptance probability can be written as

$$\frac{p(\theta_p|\mathbf{y})}{p(\theta^{(i-1)}|\mathbf{y})} = \exp\left[\log p(\theta_p|\mathbf{y}) - \log p(\theta^{(i-1)}|\mathbf{y})\right]$$

   This is handy because common multiplicative factors in $p(\theta_p|\mathbf{y})$ and $p(\theta^{(i-1)}|\mathbf{y})$ cancel out before we evaluate the exponential function (which can otherwise overflow).

2. The first argument of your (log) posterior function should be `theta`, the vector of parameters for which the posterior density is evaluated. You can of course use some other name for the variable, but it must be the *first* argument of your posterior density function.

Example:

```
# This is the log posterior density of the beta(s+a,f+b) density
LogPostBernBeta <- function(theta, s, f, a, b){
 logPost <- (s+a-1)*log(theta) + (f+b-1)*log(1-theta)
 return(logPost)
}

# Testing if the log posterior function works
s <- 8;f <- 2;a <- 1;b <- 1
logPost <- LogPostBernBeta(theta = 0.1, s, f, a, b)
print(logPost)
```