Simple MCMC under SIR

Lam ST Ho and Marc A Suchard

2016-03-18

We describe how to set-up and run a simple Metropolis-Hastings-based Markov chain Monte Carlo (MCMC) sampler under the susceptible-infected-removed (SIR) model.

```
library(MultiBD)
```

This example uses the Eyam data that consist the population counts of susceptible, infected and removed individuals across several time points.

```
data(Eyam)
Eyam
```

```
##   time   S  I   R
## 1  0.0 254  7   0
## 2  0.5 235 14  12
## 3  1.0 201 22  38
## 4  1.5 153 29  79
## 5  2.0 121 20 120
## 6  2.5 110  8 143
## 7  3.0  97  8 156
## 8  4.0  83  0 178
```

The log likelihood function is the sum of the log of the transition probabilities between two consecutive observations. Note that, we will use $(\log\alpha, \log\beta)$ as parameters instead of (α, β). The rows and columns of the transition probability matrix returned by *dbd_prob()* correspond to possible values of S (from a to $a0$) and I (from 0 to B) respectively.

```
loglik_sir <- function(param, data) {
  alpha <- exp(param[1]) # Rates must be non-negative
  beta  <- exp(param[2])

  # Set-up SIR model
  drates1 <- function(a, b) { 0 }
  brates2 <- function(a, b) { 0 }
  drates2 <- function(a, b) { alpha * b     }
  trans12 <- function(a, b) { beta  * a * b }

  sum(sapply(1:(nrow(data) - 1), # Sum across all time steps k
             function(k) {
               log(
                 dbd_prob(  # Compute the transition probability matrix
                   t  = data$time[k + 1] - data$time[k], # Time increment
                   a0 = data$S[k], b0 = data$I[k],       # From: S(t_k), I(t_k)
                   drates1, brates2, drates2, trans12,
                   a = data$S[k + 1], B = data$S[k] + data$I[k] - data$S[k + 1],
                   computeMode = 4, nblocks = 80         # Compute using 4 threads
                 )[1, data$I[k + 1] + 1]                 # To: S(t_(k+1)), I(t_(k+1))
               )
             }))
}
```

Here, we choose Normal$(0, 100^2)$ as the prior for both $\log\alpha$ and $\log\beta$.

```
logprior <- function(param) {
  log_alpha <- param[1]
  log_beta <- param[2]

  dnorm(log_alpha, mean = 0, sd = 100, log = TRUE) +
    dnorm(log_beta, mean = 0, sd = 100, log = TRUE)
}
```

We will use the random walk Metropolis algorithm implemented in the function *MCMCmetrop1R()* (**MCMCpack** package) to explore the posterior distribution. So, we first need to install the package and its dependencies.

```
source("http://bioconductor.org/biocLite.R")
biocLite("graph")
biocLite("Rgraphviz")
install.packages("MCMCpack", repos = 'http://cran.us.r-project.org')
library(MCMCpack)
```

The starting point of our Markov chain is the estimated value of (α, β) from Raggett (1982).

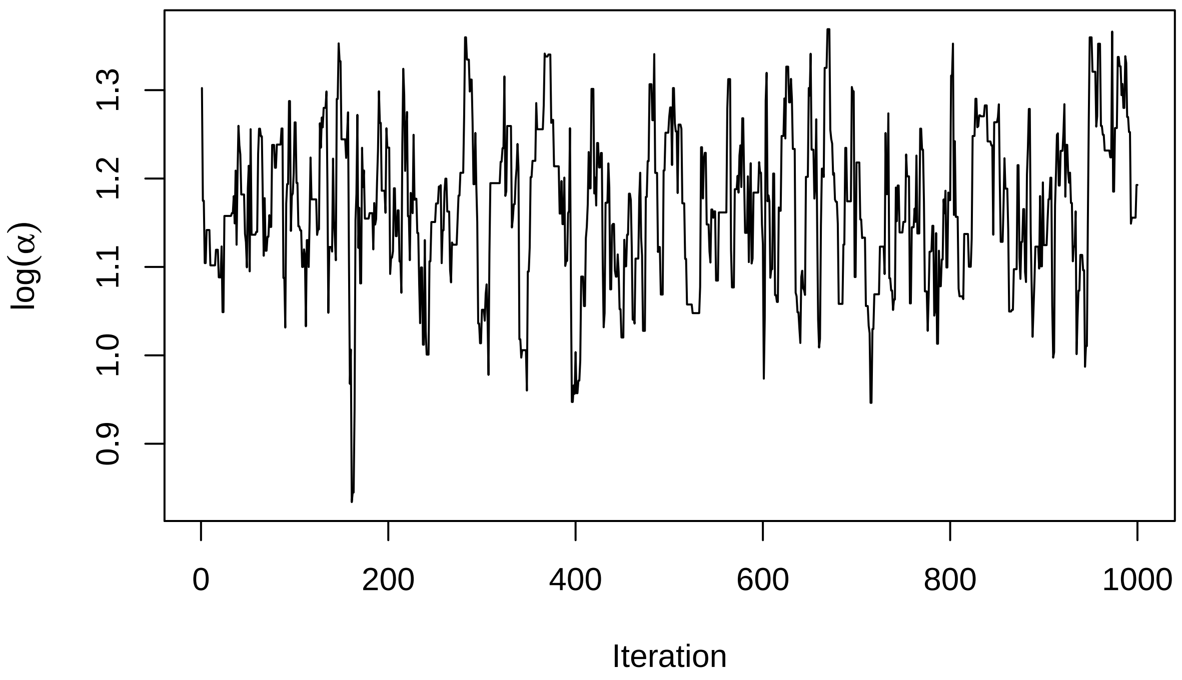
```
alpha0 <- 3.39
beta0  <- 0.0212
```

We discard the first 200 iterations and keep the next 1000 iterations of the chain.

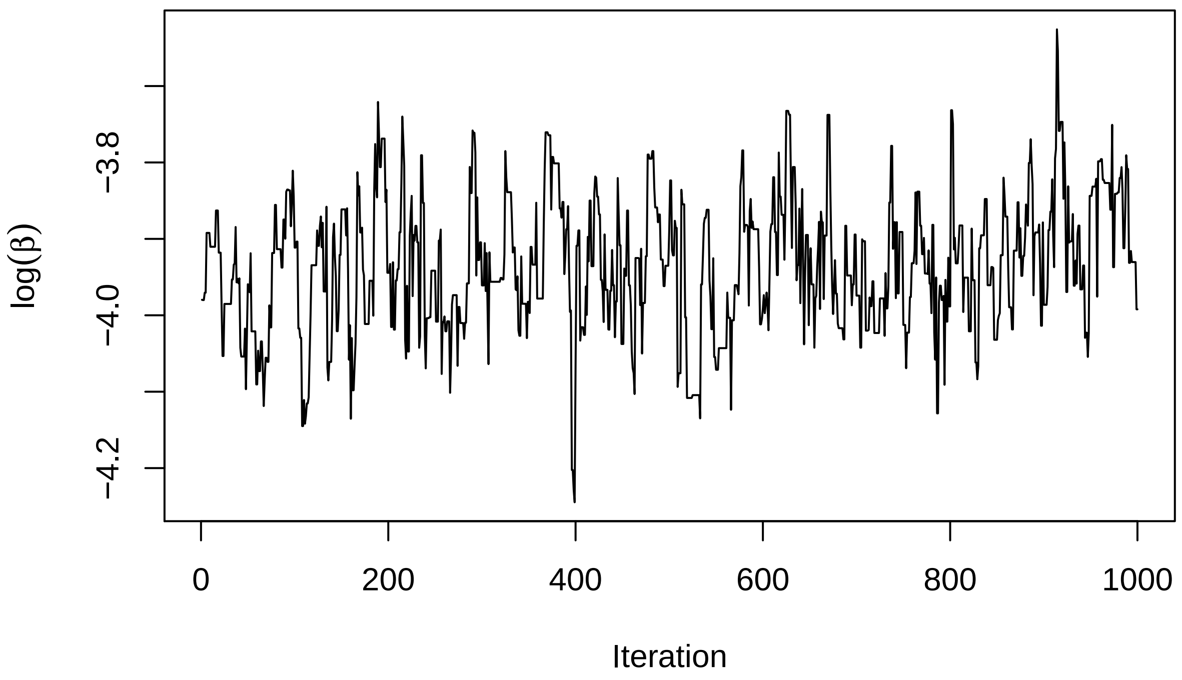
```
post_sample <- MCMCmetrop1R(fun = function(param) { loglik_sir(param, Eyam) + logprior(param) },
                            theta.init = log(c(alpha0, beta0)),
                            mcmc = 1000, burnin = 200)
```

The trace plots of both $\log\alpha$ and $\log\beta$ look good.

```
plot(as.vector(post_sample[,1]), type = "l", xlab = "Iteration", ylab = expression(log(alpha)))
```

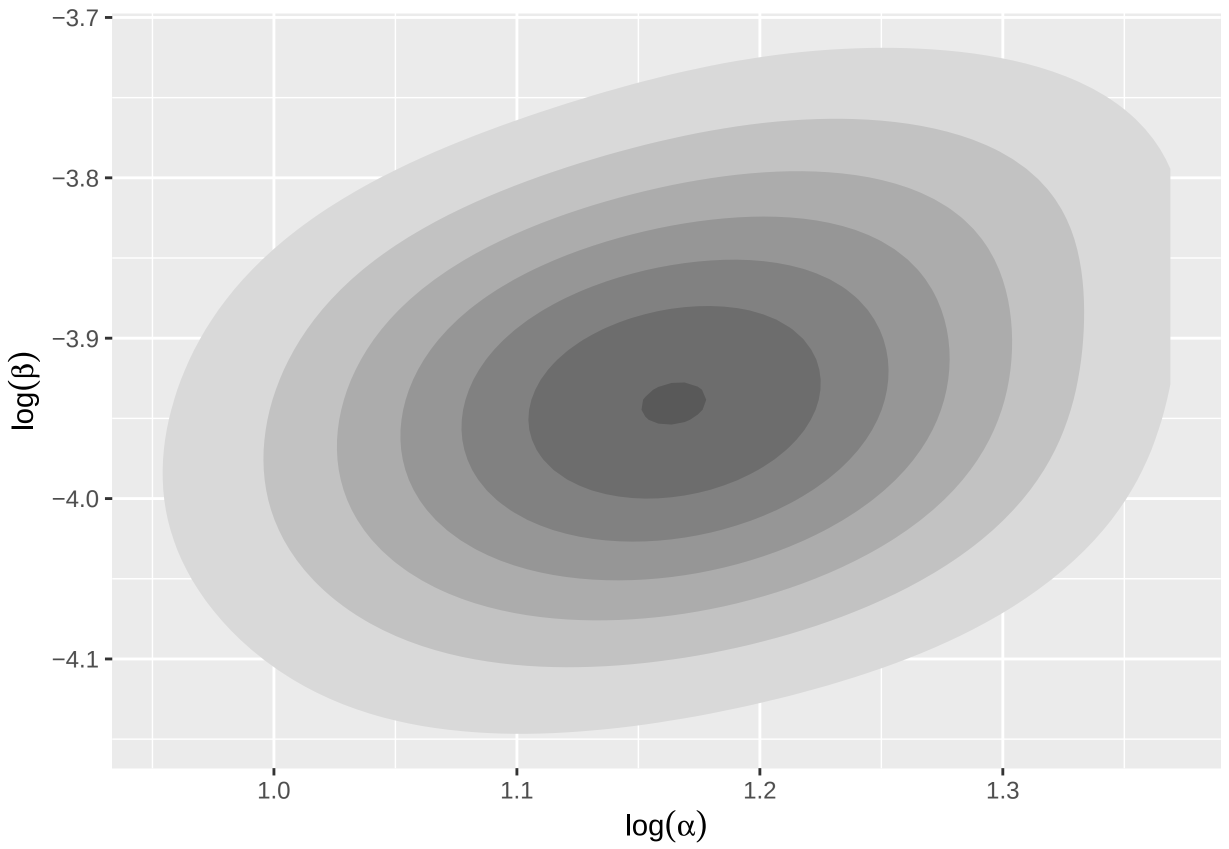


```r
plot(as.vector(post_sample[,2]), type = "l", xlab = "Iteration", ylab = expression(log(beta)))
```



We can visualize the joint posterior distribution of $\log \alpha$ and $\log \beta$ using the **ggplot2** package.

```r
library(ggplot2)
x = as.vector(post_sample[,1])
y = as.vector(post_sample[,2])
df <- data.frame(x, y)
ggplot(df,aes(x = x,y = y)) +
  stat_density2d(aes(fill = ..level..), geom = "polygon", h = 0.26) +
  scale_fill_gradient(low = "grey85", high = "grey35", guide = FALSE) +
  xlab(expression(log(alpha))) +
  ylab(expression(log(beta)))
```



We can also construct the 95% Bayesian credible intervals for α and β.

```
quantile(exp(post_sample[,1]), probs = c(0.025,0.975))
```

```
##     2.5%    97.5%
## 2.721921 3.809780
```

```
quantile(exp(post_sample[,2]), probs = c(0.025,0.975))
```

```
##       2.5%      97.5%
## 0.01649866 0.02327176
```