Random Walk Metropolis-Hastings Algorithm

Ying-Ting Lin

April 1, 2016

Goal:

Build a Random Walk Metropolis-Hastings function to derive the posterior marginal densities of parameters in a Gamma distribution with Jeffery's prior.

Jeffery's prior for $Gamma(\alpha, \beta)$ distribution:

$$\pi_J(\theta) \propto \sqrt{det(I(\theta))} = \sqrt{trigamma(\alpha)\frac{\alpha}{\beta^2} - [\frac{1}{\beta}]^2} = \frac{\sqrt{\alpha * trigamma(\alpha) - 1}}{\beta}$$

Random Walk Metropolis-Hastings:

Step 1: generate some observed data
Step 2: choose the starting values alpha & beta
Step 3: use Gibbs Sampling to generate a new alpha & beta
Step 4: compute acceptance probability
Step 5: accept or not

```
GammaMHExample <- function(n.sim, n.burnin){
  library(MASS)
  alpha <- 3; beta <-3 #pick the parameter values to generate a random "observed" data
  N <- 30 #set observed sample size
  x <- rgamma(N, shape = alpha, scale = 1/beta)#generate observed data of size N

  #defines a function that returns unnormalized posterior density
  post_den <-function(alpha, beta, x, N){
    #Jeffery's prior
    prior <- sqrt(alpha*trigamma(alpha)-1)/beta
    #gamma distr
    likelihood <- prod(x^(alpha-1))*prod(exp(-beta*x))*((beta^alpha)/gamma(alpha))^N
    # unnormalized target distr
    working_cond_den <- prior*likelihood
  }

  theta.mh <- matrix(NA, nrow = n.sim, ncol = 2) #empty matrix to store sampled parameters
  rho <- 0.05
  #generate inital parameters
  theta.current <- mvrnorm(n=1, mu = c(3,3), Sigma = matrix(c(0.3,rho,rho,0.6), nrow=2, ncol=2))
  theta.update <- function(index, theta.current,x,N,i, rho) {
    if (index == 1){ #if it's sample for alpha, update the first element of the parameter vector
      #sample a new parameter value from Gaussian
      theta.star <- rnorm(n = 1, mean = theta.current[index], sd = sqrt(0.3-rho^2))
      #resample is drawn value is not positive
      while(theta.star <=0){
        theta.star <- rnorm(n = 1, mean = theta.current[index], sd = sqrt(0.6-rho^2))
        }
```

```
      theta.temp <- c(theta.star, theta.current[2])

    }
    else {
      theta.star <- rnorm(n = 1, mean = theta.current[index]+
                            rho*(theta.current[1]-theta.mh[i, 1]), sd = sqrt(1-rho^2))
      #resample is drawn value is not positive
      while(theta.star <=0){
        theta.star <- rnorm(n = 1, mean = theta.current[index]+
                              rho*(theta.current[1]-theta.mh[i, 1]), sd = sqrt(1-rho^2))
        }
      theta.temp <- c(theta.current[1], theta.star)#for beta, update the second element
      }
      #compute MH ratio
      r <- post_den(theta.temp[1],theta.temp[2],x,N)/
                                      post_den(theta.current[1],theta.current[2],x,N)
      r <- min(r, 1, na.rm = TRUE) # r is the accpetance probability, NA values are ignored
      if (runif(1) < r)
      #runif(1) generates a uniform random number bewteen 0 and 1
      #if runif(1) < r, accept the new value; else reject the new value and keep the old value
        theta.star
      else theta.current[index]

  }
    for (i in 1:n.sim) {
    #iteration for alpha
    theta.current[1] <- theta.mh[i, 1] <- theta.update(1, theta.current,x, N, i,rho)
    #iteration for beta
    theta.current[2] <- theta.mh[i, 2] <- theta.update(2, theta.current,x, N, i,rho)
  }
  theta.mh <- theta.mh[(n.burnin + 1):n.sim, ] #discard burn-in
}

mh.draws <- GammaMHExample(n.sim = 10000, n.burnin = 1000)
```

Summary Statistics

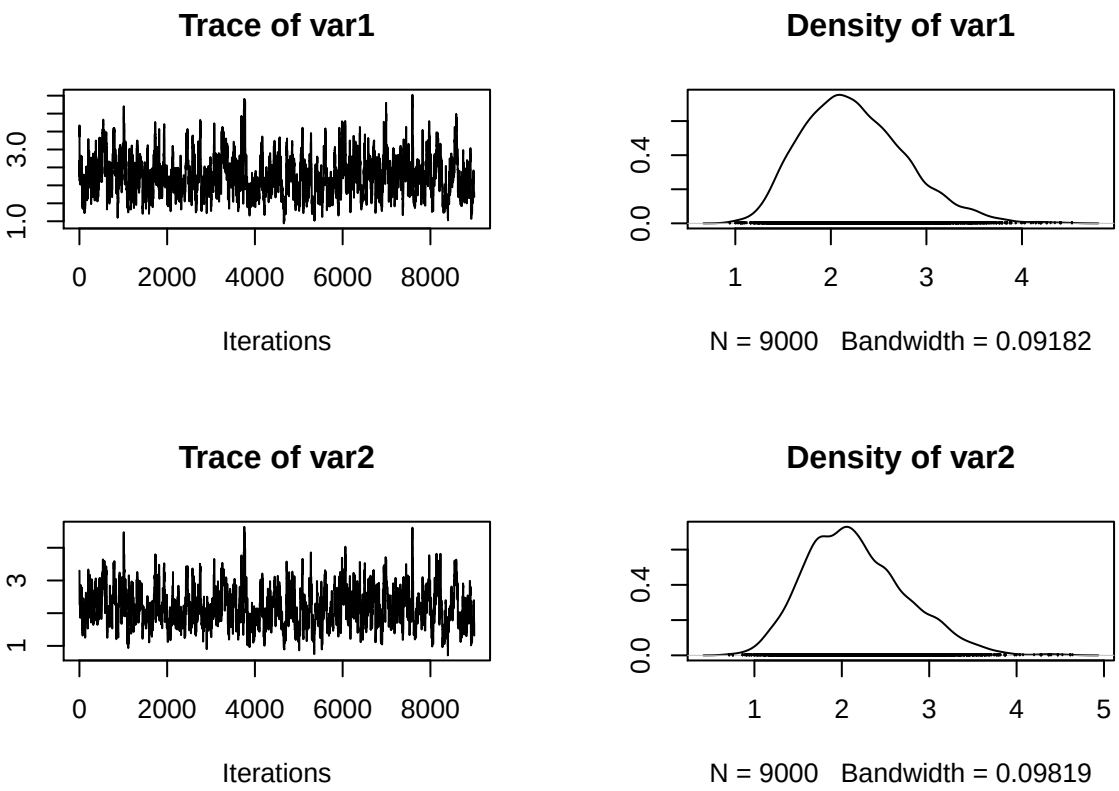
```
library(coda)
mh.draws <- mcmc(mh.draws)
summary(mh.draws)
```

```
##
## Iterations = 1:9000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 9000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##       Mean     SD Naive SE Time-series SE
```

```
## [1,] 2.246 0.5352 0.005641         0.03574
## [2,] 2.164 0.5723 0.006033         0.03596
##
## 2. Quantiles for each variable:
##
##       2.5%   25%   50%   75% 97.5%
## var1 1.374 1.857 2.195 2.586 3.454
## var2 1.200 1.742 2.099 2.513 3.426
```

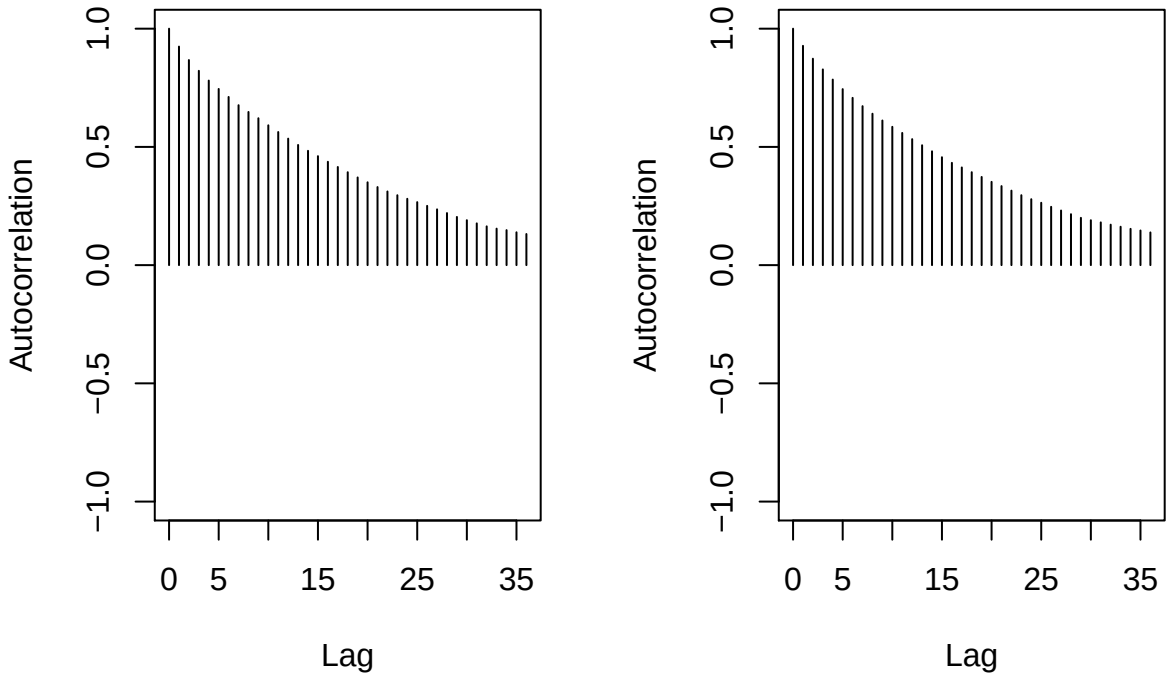
Diagnostics

```
#Traceplot
plot(mh.draws)
```



The traceplots suggest a good mixing, since the sample draws stay within a reasonable range of the parameter space and do not stuck in any particular area.

```
#Autocorrelation function plots
autocorr.plot(mh.draws)
```

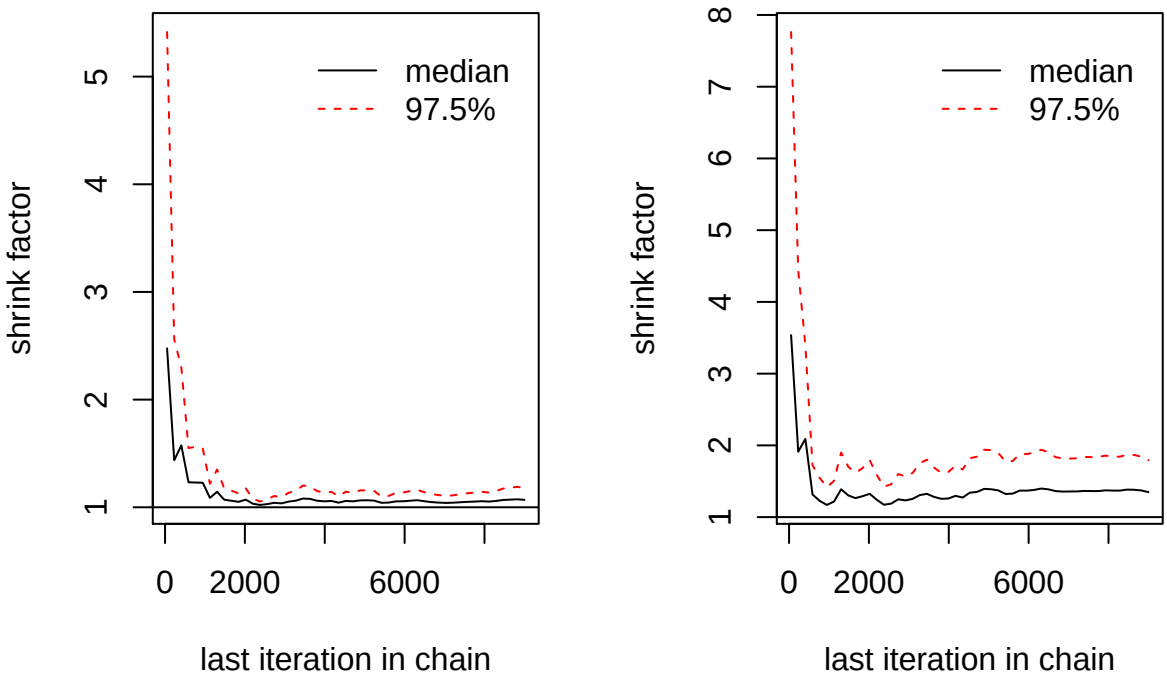


The autocorrleation plots show that the autocorrelation is decreasing but still stays positive for a long lag.

```
#Gelman and Rubin
mh.draws1 <- mcmc(GammaMHExample(n.sim = 10000, n.burnin = 1000))
mh.draws2 <- mcmc(GammaMHExample(n.sim = 10000, n.burnin = 1000))
mh.draws3 <- mcmc(GammaMHExample(n.sim = 10000, n.burnin = 1000))
mh.draws4 <- mcmc(GammaMHExample(n.sim = 10000, n.burnin = 1000))
mh.draws5 <- mcmc(GammaMHExample(n.sim = 10000, n.burnin = 1000))
mh.list <- mcmc.list(list(mh.draws1, mh.draws2, mh.draws3,mh.draws4, mh.draws5))
gelman.diag(mh.list)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.07       1.18
## [2,]       1.35       1.79
##
## Multivariate psrf
##
## 2.02
```

```
gelman.plot(mh.list)
```



The scale reduction factors reduces greatly after the first 1,000 iterations, and the median scale reduction factors are not far away from 1. The results suggest that between-chain variance is not much greater than within-chain variance and that the chains have converged to the stationary distribution with 10,000 sample draws.

```
#Geweke
geweke.diag(mh.draws)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##  var1  var2
## 1.305 1.275
```

The Z-scores are low, and hence we cannot reject the null hypothesis that fractions in the two windows are from the same distribution.

```
#Raftery and Lewis
raftery.diag(mh.draws,q=0.025,r=0.005,s=0.95)
```

```
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##  Burn-in  Total Lower bound  Dependence
##  (M)      (N)   (Nmin)       factor (I)
##  23       24843 3746         6.63
##  29       31539 3746         8.42
```

Dependence factors exceed the benchmark 5, which may be caused from influential starting values, high correlations between parameters, or poor mixing.

```
#Heidelberg and Welch
heidel.diag(mh.draws)
```

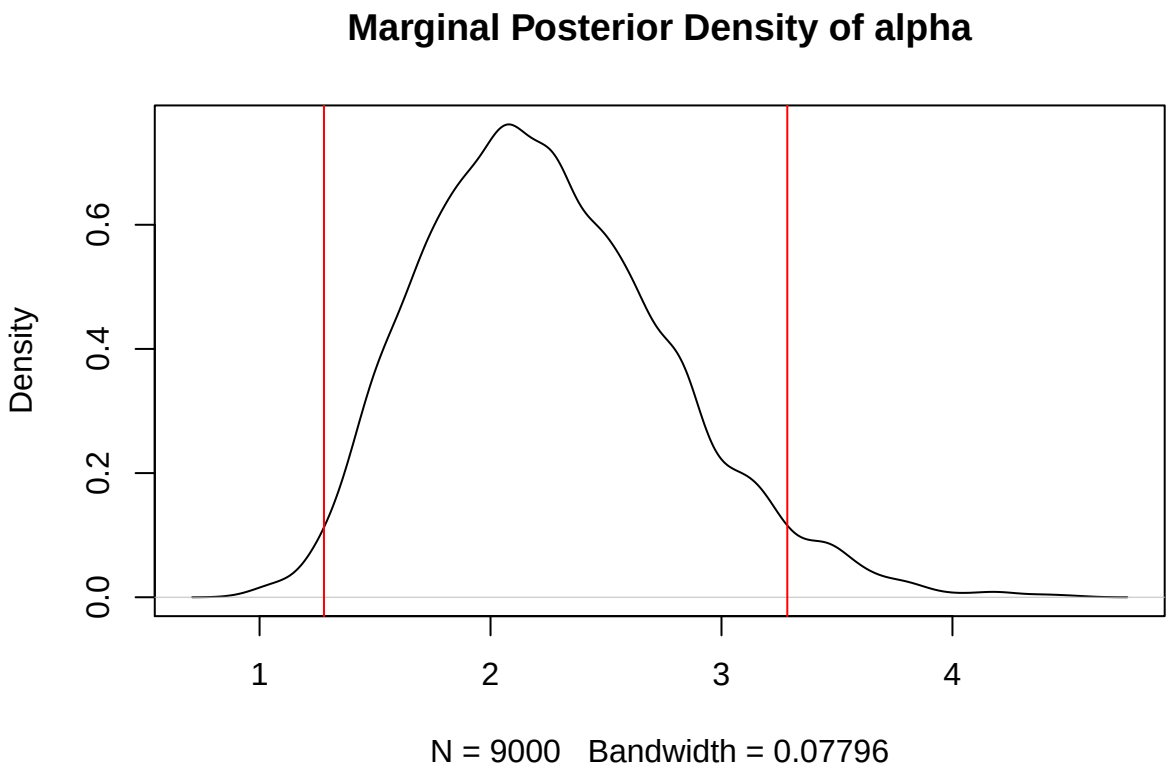
```
##
##      Stationarity start     p-value
##      test         iteration
## var1 passed       1         0.585
## var2 passed       1         0.551
##
##      Halfwidth Mean Halfwidth
##      test
## var1 passed    2.25 0.0701
## var2 passed    2.16 0.0705
```

The chains pass the test of stationarity.

Marginal posterior density

Plot the MCMC estimate of the marginal posterior density for each parameter with an 95% HPD interval for each parameter

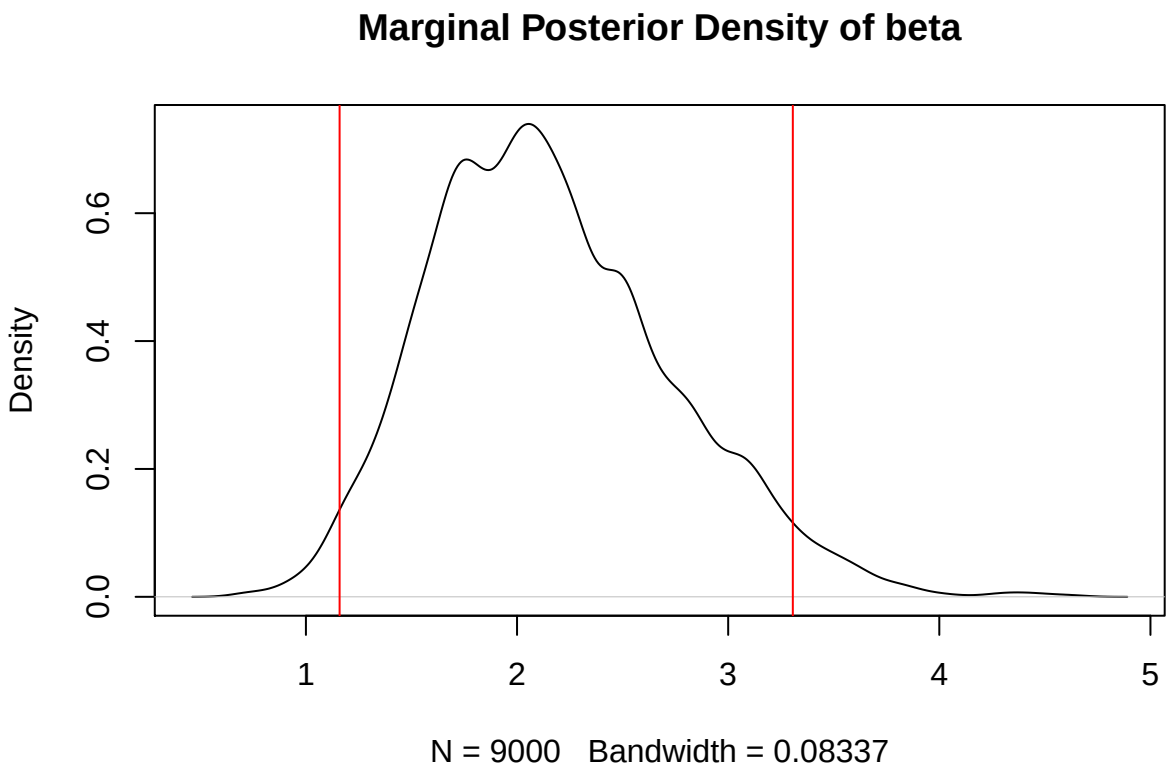
```
library(TeachingDemos)
d1 <- density(mh.draws[,1])
plot(d1, main="Marginal Posterior Density of alpha")
abline(v=emp.hpd(mh.draws[,1], conf=0.95), col = "red")
```



```r
emp.hpd(mh.draws[,1], conf=0.95) #95% HPD interval for alpha
```

```
## [1] 1.278886 3.284908
```

```r
d2 <- density(mh.draws[,2])
plot(d2, main="Marginal Posterior Density of beta")
abline(v=emp.hpd(mh.draws[,2], conf=0.95), col = "red")
```




```r
emp.hpd(mh.draws[,2], conf=0.95) #95% HPD interval for beta
```

```
## [1] 1.159514 3.306168
```