MCMC Part 1

Sam Bowyer

2023-04-27

For this task we will be using the Pima Indians Diabetes dataset from the `R` package `mlbench`, containing $n = 768$ input-output pairs $\{(y_i^0, x_i^0)\}_{i=1}^n$ with $y_i^0 \in \{0, 1\}$ indicating whether patient i has diabetes and $x_i^0 \in \mathbb{R}^p$ representing $p = 8$ diagnostic measurements.

```
library(mlbench)
data(PimaIndiansDiabetes)
head(PimaIndiansDiabetes)
```

```
##   pregnant glucose pressure triceps insulin mass pedigree age diabetes
## 1        6     148       72      35       0 33.6    0.627  50      pos
## 2        1      85       66      29       0 26.6    0.351  31      neg
## 3        8     183       64       0       0 23.3    0.672  32      pos
## 4        1      89       66      23      94 28.1    0.167  21      neg
## 5        0     137       40      35     168 43.1    2.288  33      pos
## 6        5     116       74       0       0 25.6    0.201  30      neg
```

```
dim(PimaIndiansDiabetes)
```

```
## [1] 768   9
```

First we will split up the data into inputs and outputs, and we will convert the y_i^0 types into zeroes and ones.

```
p = ncol(PimaIndiansDiabetes)-1
X = PimaIndiansDiabetes[,1:p]
Y = as.numeric(PimaIndiansDiabetes[,p+1] == "pos")

head(X); head(Y)
```

```
##   pregnant glucose pressure triceps insulin mass pedigree age
## 1        6     148       72      35       0 33.6    0.627  50
## 2        1      85       66      29       0 26.6    0.351  31
## 3        8     183       64       0       0 23.3    0.672  32
## 4        1      89       66      23      94 28.1    0.167  21
## 5        0     137       40      35     168 43.1    2.288  33
## 6        5     116       74       0       0 25.6    0.201  30
```

```
## [1] 1 0 1 0 1 0
```

```
dim(X); length(Y)
```

```
## [1] 768   8
```

```
## [1] 768
```

We'll be considering the logistic regression model

$$Pr_{\alpha,\beta}^{(i)}(Y_i^0 = 1) = \frac{1}{1 + e^{-\alpha - \beta^T x_i^0}}, \quad i = 1, ..., n.$$

Then with a prior distribution $\pi(\alpha, \beta)$ on $\mathbb{R}^{p+1}$ and a likelihood function

$$L_n(\alpha, \beta) = \prod_{i=1}^{n} Pr_{\alpha,\beta}^{(i)}(Y_i^0 = y_i^0),$$

we arrive at the posterior distribution over (α, β) given the data:

$$\pi(\alpha, \beta | y^0) \propto L_n(\alpha, \beta)\pi(\alpha, \beta).$$

First we'll implement functions to calculate the log prior and log likelihood. In particular, we'll put standard Gaussian priors on our parameters α and β, so $(\alpha, \beta) \sim \mathcal{N}_{p+1}(\mathbf{0}, \mathbf{I}_{p+1})$.

```
log_prior <- function(params){
  return(sum(dnorm(params, log=TRUE)))
}

log_likelihood <- function(params){
  logits = params[1] + params[-1] %*% t(X)
  return(sum(Y*logits - log(1+exp(logits))))
}

log_posterior <- function(params){
  return(log_prior(params) + log_likelihood(params))
}
```

Next we obtain our initial values for α and β (note that we are referring to the vector $(\alpha, \beta) \in \mathbb{R}^{p+1}$ as `params` in the code).

```
# initial params
params0 = rnorm(p+1)
params0
```

```
## [1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078 -0.8204684  0.4874291
## [8]  0.7383247  0.5757814
```

For our proposal distribution Q we will use $Q(z, dz') = \mathcal{N}_{p+1}(z, c\Sigma_n)$ with a tuning parameter $c > 0$, where

$$\begin{aligned} \Sigma_n &= -(\mathbf{H}_n(\mu_n))^{-1} \\ \mu_n &= \underset{(\alpha,\beta)\in\mathbb{R}^{p+1}}{\operatorname{argmax}} \log \pi(\alpha, \beta | y^0) \\ \mathbf{H}_n(\theta) &= \left(\frac{\partial^2}{\partial\theta_l \partial\theta_l} \log \pi(\theta | y^0) \right)_{j,l=1}^{p+1}, \quad \forall \theta \in \mathbb{R}^{p+1}. \end{aligned}$$

We approximate μ_n using `optim` and calculate the Hessian $\mathbf{H}_n(\mu_n))$ using the `hessian` function from the package `pracma`.

```
mu_n = optim(par = params0, function(x) -log_likelihood(x))$par

library(pracma)
Sigma_n = -solve(hessian(f=log_posterior, x=mu_n))

library(mvtnorm)
proposal <- function(c){
  Q <- list()
  Q$sample <- function(x){
    rmvnorm(1, mean = x, sigma = c*Sigma_n)
  }
  Q$density <- function(x,y){
    dmvnorm(y, mean = x, sigma = c*Sigma_n)
  }
  return(Q)
}
```

Finally we may write our Metropolis-Hastings algorithm.

```
runMH <- function(f, Q, x0, n){
  q = Q$density
  xs = matrix(rep(0, n*p+1), nrow=n)
  x = x0
  acceptanceCount = 0
  for(i in 1:n){
    z = Q$sample(x)
    acceptProb = min(1, f(z)*q(z,x)/(f(x)*q(x,z)))
    if(runif(1)<acceptProb){
      x = z
      acceptanceCount = acceptanceCount + 1
    }
    xs[i,] = x
  }
  return(list(samples = xs, acceptanceRate = acceptanceCount/n))
}
```

First we use $c = 1$ and obtain very good results after generating 100000 samples.

```
n = 100000
c = 1
MH = runMH(function(x) exp(log_posterior(x)), proposal(c), mu_n, n)
```

In particular, we observe an acceptance rate of 0.19538, somewhat close to the theoretical optimal rate of 0.234.

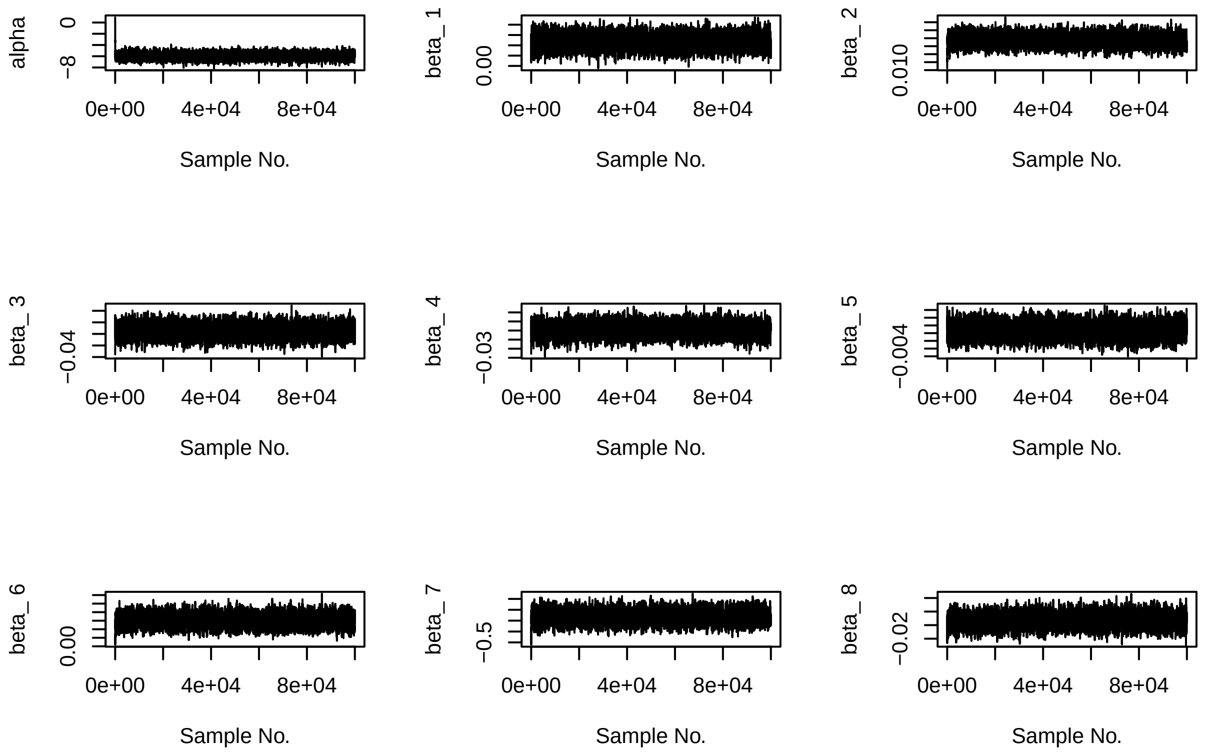
```
MH$acceptanceRate
```

```
## [1] 0.19538
```

Furthermore, looking at the trace plots for the generated samples we observe healthy looking convergence after a small amount of mixing time.

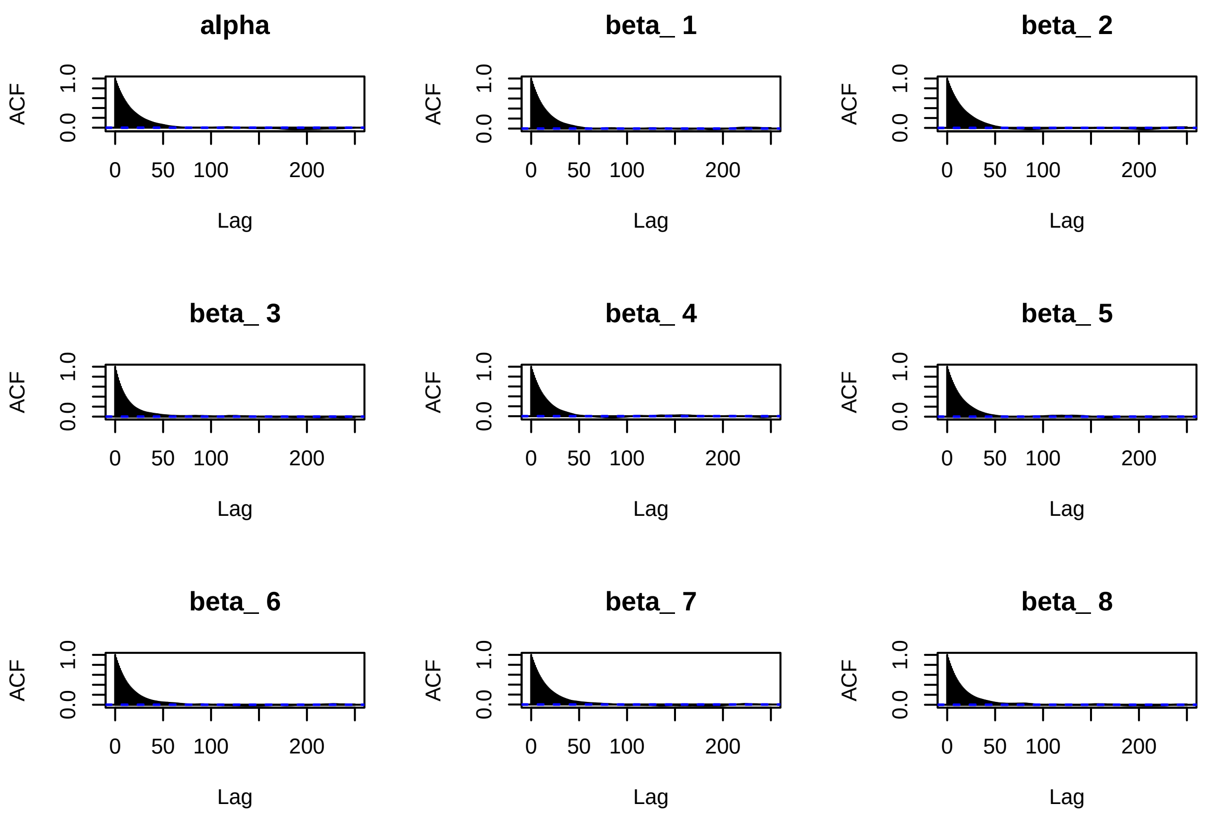
```
var_names = c("alpha")
for (i in 1:9){
  var_names = c(var_names, paste("beta_", i))
}

par(mfrow=c(3,3))
for (i in 1:9){
  plot(1:n, MH$samples[,i], type="l", xlab="Sample No.", ylab=paste(var_names[i]))
}
```



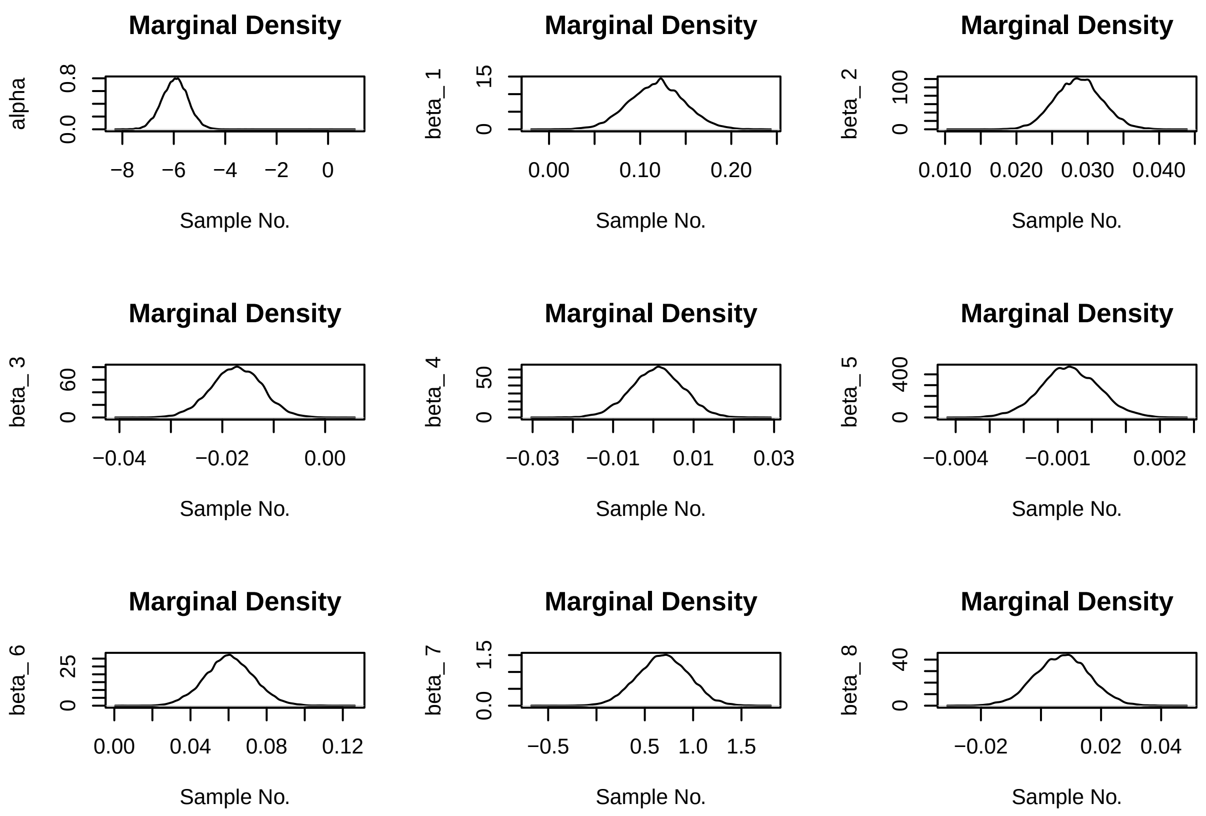
We also see autocorrelation plots that diminish fairly quickly towards 0.

```r
par(mfrow=c(3,3))
for (i in 1:9){
  acf(MH$samples[,i], lag.max=250, main="")
  title(var_names[i])
}
```



This culminates in acceptable-looking plots for estimated marginal posterior distributions for each of the model's nine parameters—each of which are unimodal and vaguely Gaussian.

```r
par(mfrow=c(3,3))
for (i in 1:9){
  plot(density(MH$samples[,i]), xlab="Sample No.", ylab=paste(var_names[i]), main="Marginal Density")
}
```



For comparison against a badly-tuned chain, consider the results when we use $c = 0.001$ (again with 100000 generated samples).

```r
c = 0.001
MH = runMH(function(x) exp(log_posterior(x)), proposal(c), mu_n, n)
```

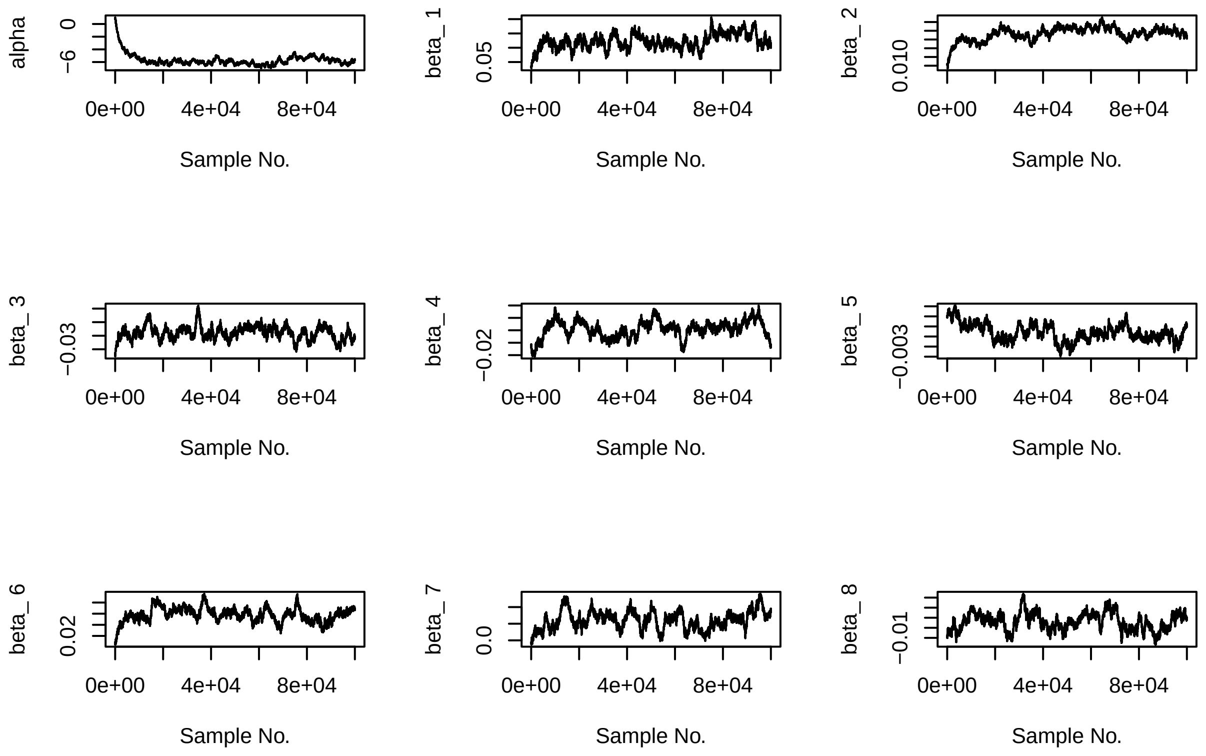
This gives an acceptance rate of 0.96417, far above the optimal value which suggests that the proposal is failing to capture the behaviour of the posterior.

```r
MH$acceptanceRate
```

```
## [1] 0.96417
```

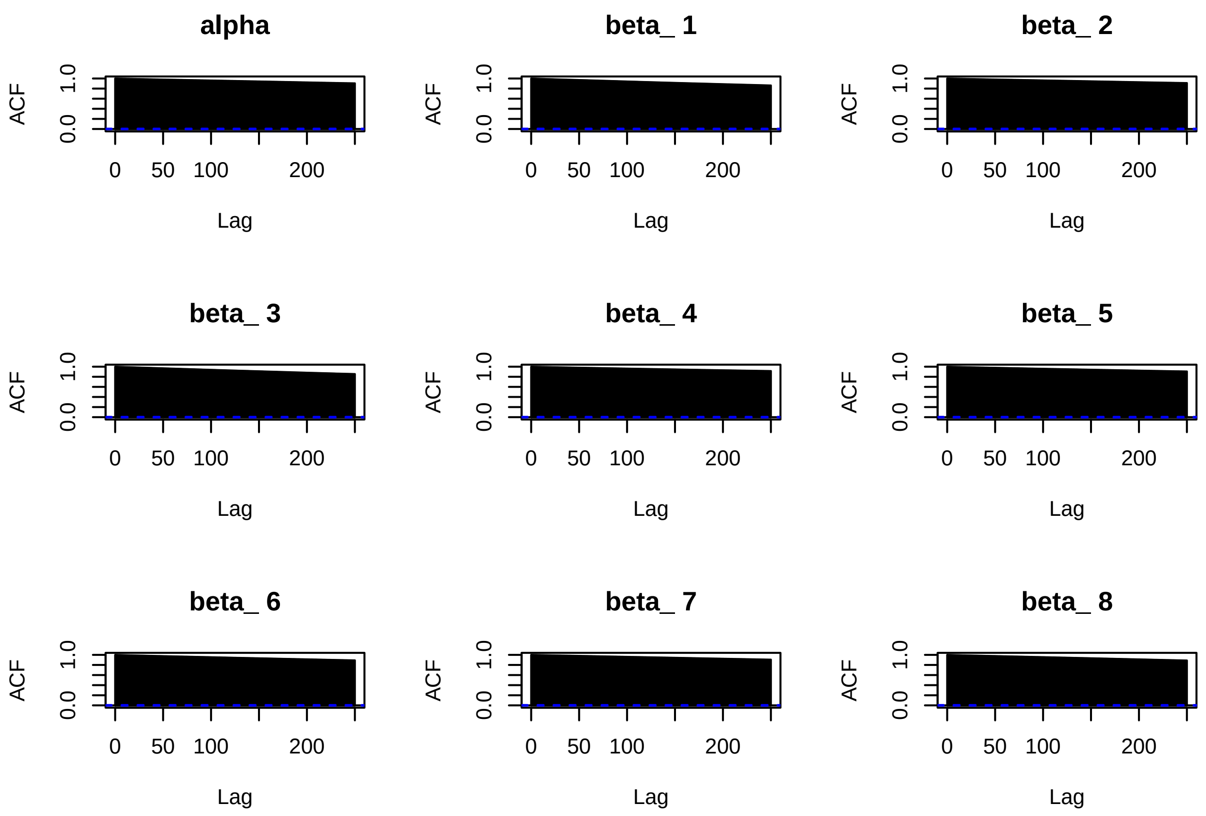
Using these samples we see much worse convergence than before in the trace plots.

```r
par(mfrow=c(3,3))
for (i in 1:9){
  plot(1:n, MH$samples[,i], type="l", xlab="Sample No.", ylab=paste(var_names[i]))
}
```



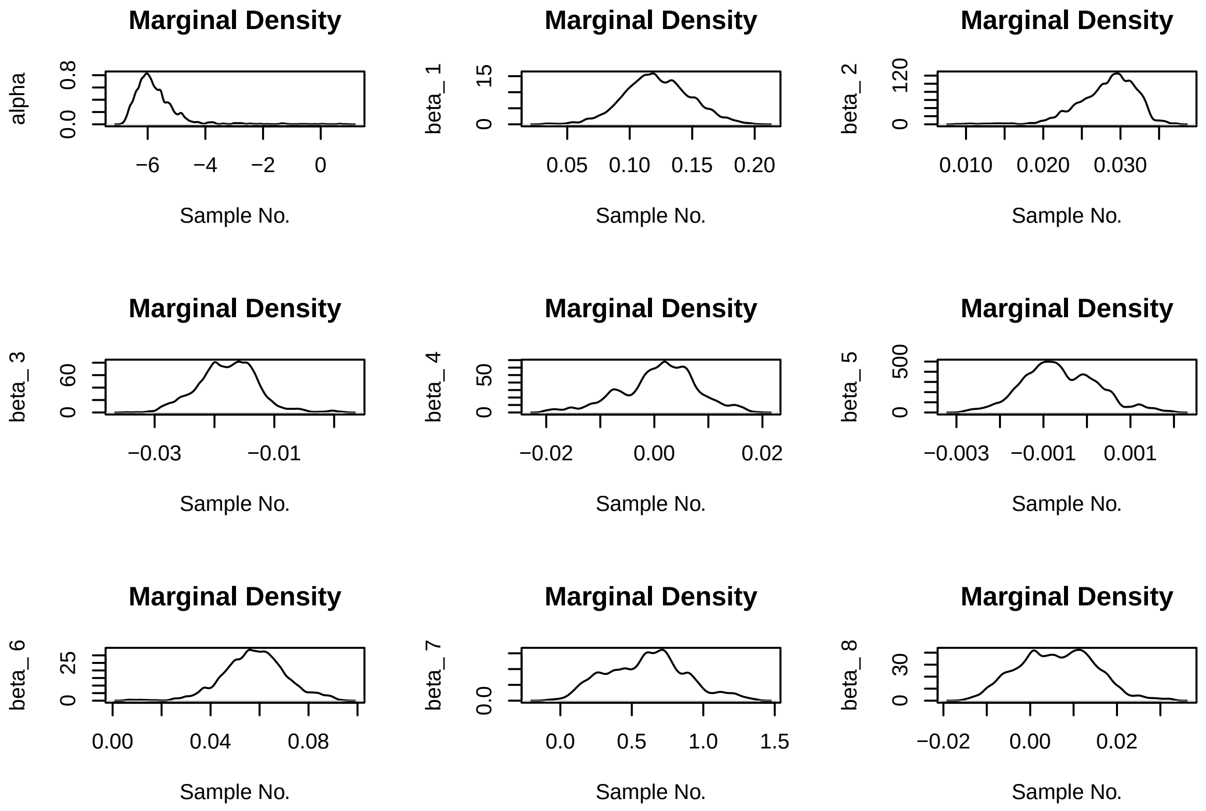
Similarly, the autocorrelation stays relatively large even as we increase the lag, further suggesting that the proposal isn't exploring the posterior space successfully.

```r
par(mfrow=c(3,3))
for (i in 1:9){
  acf(MH$samples[,i], lag.max=250, main="")
  title(var_names[i])
}
```



And finally when we plot the estimated marginal posterior distributions we see plots that looks a lot less Gaussian and unimodal.

```
par(mfrow=c(3,3))
for (i in 1:9){
  plot(density(MH$samples[,i]), xlab="Sample No.", ylab=paste(var_names[i]), main="Marginal Density")
}
```


Marginal Density
alpha
0.0
0.8
−6
−4
−2
0
Sample No.
Marginal Density
beta_1
0
15
0.05
0.10
0.15
0.20
Sample No.
Marginal Density
beta_2
0
120
0.010
0.020
0.030
Sample No.
Marginal Density
beta_3
0
60
−0.03
−0.01
Sample No.
Marginal Density
beta_4
0
50
−0.02
0.00
0.02
Sample No.
Marginal Density
beta_5
0
500
−0.003
−0.001
0.001
Sample No.
Marginal Density
beta_6
0
25
0.00
0.04
0.08
Sample No.
Marginal Density
beta_7
0.0
0.0
0.5
1.0
1.5
Sample No.
Marginal Density
beta_8
0
30
−0.02
0.00
0.02
Sample No.