Biostat276 Project 3

Zhaodong Wu

3/3/2022

Contents


```r
rm(list = ls())
library(ggplot2)
library(tidyverse)
library(lme4)
library(MCMCpack)
library(reshape2)
library(kableExtra)
```

Bayesian Mixed-Effects Model

Consider the dataset `sleepstudy` available from the `R` package `lme4`. These data, reports a longitudinal study of reaction times after sleep deprivation. Let y_{ij} be the reaction of subject i after t_{ij} days of sleep deprivation. We assume:

$$y_{ij} \sim N(\mu_{ij}, \sigma^2)$$

with $\mu_{ij} = \beta_0 + \beta_1 t_{ij} + b_{i0} + b_{i1} t_{ij}, b_{i0} \sim N(0, \alpha_0)$ independent of $b_{i1} \sim N(0, \alpha_1)$ for all i. The model is completed with the following priors:

$$\begin{aligned}
\beta_0 &\sim N(0, 100.0) \\
\beta_1 &\sim N(0, 100.0) \\
\alpha_0 &\sim IG(1.0, 1.0) \\
\alpha_1 &\sim IG(1.0, 1.0) \\
\sigma^2 &\sim IG(0.01, 0.01)
\end{aligned}$$

where all IG priors use the shape, scale parametrization.

(1) Describe and implement a Gibbs sampling strategy for MCMC simulation from the posterior distribution.

$$p(\beta_0, \beta_1, \alpha_0, \alpha_1, \sigma^2 | y)$$

Derive posterior summaries for all population level parameters, including posterior means, posterior SDs and, 95% credible intervals.

(2) Implement an HMC sampler for MCMC simulation for the posterior distribution in (1). Compare all posterior summaries with the estimates obtained using Gibbs sampling.

(3) Compare convergence and mixing associated with the posterior simulations algorithms in (1) and (2).

Solutions:

(1)

```
# load the data
data("sleepstudy")
dat <- sleepstudy
str(dat)
```

```
## 'data.frame':    180 obs. of  3 variables:
##  $ Reaction: num  250 259 251 321 357 ...
##  $ Days    : num  0 1 2 3 4 5 6 7 8 9 ...
##  $ Subject : Factor w/ 18 levels "308","309","310",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Then we should calculate posteriors for all population level parameters:

Let $m =$ number of rows, $n =$ number of columns.

For β_0 :

$$
\begin{aligned}
p(\beta_0|\beta_1, b_{i0}, b_{i1}, \sigma^2, y) &\propto \prod_{i=1}^{m}\prod_{j=1}^{n} p(y_{ij}|\beta_0, \beta_1, b_{i0}, b_{i1}, \sigma^2) \cdot p(\beta_0) \\
&\propto \exp\left\{-\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})^2}{2\sigma^2} - \frac{\beta_0^2}{200})\right\} \\
&\propto \exp\left\{-\frac{(\sigma^2+100mn)\beta_0^2 - 2\beta_0\cdot 100\sum_{i=1}^{m}\sum_{j=1}^{n}(y_{ij}-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})}{200\sigma^2}\right\} \\
&= \exp\left\{-\frac{(\frac{\sigma^2+100mn}{100\sigma^2})\beta_0^2 - 2\beta_0\frac{100\sum_{i=1}^{m}\sum_{j=1}^{n}(y_{ij}-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})}{100\sigma^2}}{2}\right\}
\end{aligned}
$$

We get that the posterior of β_0 follows a normal kernel and by completing the square, the full conditional distribution of β_0 is normal, i.e.

$$
\beta_0|\beta_1, b_{i0}, b_{i1}, \sigma^2, y \sim N\left(\frac{100\sum_{i=1}^{m}\sum_{j=1}^{n}(y_{ij}-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})}{\sigma^2+100mn}, \frac{100\sigma^2}{\sigma^2+100mn}\right)
$$

For β_1 :

$$
\begin{aligned}
p(\beta_1|\beta_0, b_{i0}, b_{i1}, \sigma^2, y) &\propto \prod_{i=1}^{m}\prod_{j=1}^{n} p(y_{ij}|\beta_0, \beta_1, b_{i0}, b_{i1}, \sigma^2) \cdot p(\beta_1) \\
&\propto \exp\left\{ -\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(y_{ij} - \beta_0 - \beta_1 t_{ij} - b_{i0} - b_{i1}t_{ij})^2}{2\sigma^2} - \frac{\beta_1^2}{200}) \right\} \\
&\propto \exp\left\{ -\frac{(\sigma^2 + 100\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}^2)\beta_1^2 - 2\beta_1 \cdot 100\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}(y_{ij} - \beta_0 - b_{i0} - b_{i1}t_{ij})}{200\sigma^2} \right\} \\
&= \exp\left\{ -\frac{(\frac{\sigma^2 + 100\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}^2}{100\sigma^2})\beta_1^2 - 2\beta_1 \frac{100\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}(y_{ij} - \beta_0 - b_{i0} - b_{i1}t_{ij})}{100\sigma^2}}{2} \right\}
\end{aligned}
$$

We get that the posterior of β_1 follows a normal kernel and by completing the square, the full conditional distribution of β_1 is normal, i.e.

$$
\beta_1|\beta_0, b_{i0}, b_{i1}, \sigma^2, y \sim N\left(\frac{100\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}(y_{ij} - \beta_0 - b_{i0} - b_{i1}t_{ij})}{\sigma^2 + 100\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}^2}, \frac{100\sigma^2}{\sigma^2 + 100\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}^2} \right)
$$

For α_0 :

$$
\begin{aligned}
p(\alpha_0|b_{i0}) &\propto p(b_{i0}|\alpha_0)p(\alpha_0) \\
&= (\frac{1}{\sqrt{2\pi\alpha_0}})^m \exp\left\{ -\frac{\sum_{i=1}^{m} b_{i0}^2}{2\alpha_0} \right\} \cdot (\frac{1}{\alpha_0})^{1+1} \exp\left\{ -\frac{1}{\alpha_0} \right\} \\
&= (\frac{1}{\alpha_0})^{\frac{m}{2}+1+1} \exp\left\{ -\frac{\frac{\sum_{i=1}^{m} b_{i0}^2}{2} + 1}{\alpha_0} \right\} \\
&\sim IG(\frac{m}{2} + 1, \frac{\sum_{i=1}^{m} b_{i0}^2}{2} + 1)
\end{aligned}
$$

The full conditional distribution of α_0 follows an inverse gamma distribution, i.e.

$$
IG\left(\frac{m}{2} + 1, \frac{\sum_{i=1}^{m} b_{i0}^2}{2} + 1 \right)
$$

For α_1 :

$$
\begin{aligned}
p(\alpha_1|b_{i1}) &\propto p(b_{i1}|\alpha_1)p(\alpha_1) \\
&= (\frac{1}{\sqrt{2\pi\alpha_1}})^m \exp\left\{ -\frac{\sum_{i=1}^{m} b_{i1}^2}{2\alpha_1} \right\} \cdot (\frac{1}{\alpha_1})^{1+1} \exp\left\{ -\frac{1}{\alpha_1} \right\} \\
&= (\frac{1}{\alpha_1})^{\frac{m}{2}+1+1} \exp\left\{ -\frac{\frac{\sum_{i=1}^{m} b_{i1}^2}{2} + 1}{\alpha_1} \right\} \\
&\sim IG(\frac{m}{2} + 1, \frac{\sum_{i=1}^{m} b_{i1}^2}{2} + 1)
\end{aligned}
$$

The full conditional distribution of α_1 follows an inverse gamma distribution, i.e.

$$IG\left(\frac{m}{2}+1, \frac{\sum_{i=1}^m b_{i1}^2}{2}+1\right)$$

For σ^2 :

$$\begin{aligned}
p(\sigma^2|\beta_0,\beta_1,b_{i0},b_{i1},y) &\propto \prod_{i=1}^m\prod_{j=1}^n p(y_{ij}|\beta_0,\beta_1,b_{i0},b_{i1},\sigma^2)\cdot p(\sigma^2) \\
&\propto (\frac{1}{\sqrt{2\pi\sigma^2}})^{mn}\exp\left\{-\frac{\sum_{i=1}^m\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})^2}{2\sigma^2}\right\}\cdot(\frac{1}{\sigma^2})^{0.01+1}\exp\left\{-\frac{0.01}{\sigma^2}\right\} \\
&= (\frac{1}{\sigma^2})^{1+0.01+\frac{mn}{2}}\exp\left\{-\frac{\sum_{i=1}^m\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})^2+0.02}{2\sigma^2}\right\} \\
&\sim IG(0.01+\frac{mn}{2}, \frac{\sum_{i=1}^m\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})^2+0.02}{2})
\end{aligned}$$

The full conditional distribution of σ^2 follows an inverse gamma distribution, i.e.

$$\sigma^2|\beta_0,\beta_1,b_{i0},b_{i1},y \sim IG\left(0.01+\frac{mn}{2}, \frac{\sum_{i=1}^m\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})^2+0.02}{2}\right)$$

For b_{i0} :

$$\begin{aligned}
p(b_{i0}|y_i,\beta_0,\beta_1,b_{i1},\sigma^2,\alpha_0) &\propto \prod_{j=1}^n p(y_{ij}|\beta_0,\beta_1,b_{i0},b_{i1},\sigma^2)\cdot p(b_{i0}|\alpha_0) \\
&\propto \exp\left\{-\frac{\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})^2}{2\sigma^2}-\frac{b_{i0}^2}{2\alpha_0}\right\} \\
&\propto \exp\left\{-\frac{\sum_{j=1}^n[b_{i0}^2-2b_{i0}(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i1}t_{ij})]}{2\sigma^2}-\frac{b_{i0}^2}{2\alpha_0}\right\} \\
&= \exp\left\{-\frac{(\sigma^2+n\alpha_0)b_{i0}^2-2b_{i0}\alpha_0\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i1}t_{ij})}{2\alpha_0\sigma^2}\right\} \\
&\sim N\left(\frac{\alpha_0\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i1}t_{ij})}{\sigma^2+n\alpha_0}, \frac{\alpha_0\sigma^2}{\sigma^2+n\alpha_0}\right)
\end{aligned}$$

The full conditional distribution of b_{i0} follows a normal distribution, i.e.

$$b_{i0}|y_i,\beta_0,\beta_1,b_{i1},\sigma^2,\alpha_0 \sim N\left(\frac{\alpha_0\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i1}t_{ij})}{\sigma^2+n\alpha_0}, \frac{\alpha_0\sigma^2}{\sigma^2+n\alpha_0}\right)$$

For b_{i1} :

$$
\begin{aligned}
p(b_{i1}|y_i,\beta_0,\beta_1,b_{i0},\sigma^2,\alpha_1) &\propto \prod_{j=1}^{n} p(y_{ij}|\beta_0,\beta_1,b_{i0},b_{i1},\sigma^2)\cdot p(b_{i1}|\alpha_1) \\
&\propto \exp\left\{-\frac{\sum_{j=1}^{n}(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})^2}{2\sigma^2}-\frac{b_{i1}^2}{2\alpha_1}\right\} \\
&\propto \exp\left\{-\frac{\sum_{j=1}^{n}[t_{ij}^2 b_{i1}^2-2b_{i1}t_{ij}(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0})]}{2\sigma^2}-\frac{b_{i1}^2}{2\alpha_1}\right\} \\
&= \exp\left\{-\frac{(\sigma^2+\alpha_1\sum_{j=1}^{n}t_{ij}^2)b_{i1}^2-2b_{i1}\alpha_1\sum_{j=1}^{n}t_{ij}(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0})}{2\alpha_1\sigma^2}\right\} \\
&\sim N\left(\frac{\alpha_1\sum_{j=1}^{n}t_{ij}(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0})}{\sigma^2+\alpha_1\sum_{j=1}^{n}t_{ij}^2},\frac{\alpha_1\sigma^2}{\sigma^2+\alpha_1\sum_{j=1}^{n}t_{ij}^2}\right)
\end{aligned}
$$

The full conditional distribution of b_{i1} follows a normal distribution, i.e.

$$
b_{i1}|y_i,\beta_0,\beta_1,b_{i0},\sigma^2,\alpha_1 \sim N\left(\frac{\alpha_1\sum_{j=1}^{n}t_{ij}(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0})}{\sigma^2+\alpha_1\sum_{j=1}^{n}t_{ij}^2},\frac{\alpha_1\sigma^2}{\sigma^2+\alpha_1\sum_{j=1}^{n}t_{ij}^2}\right)
$$

Based on these full conditional distributions, we can conduct a Gibbs sampling here:

```
# function for sum of day and reaction time for each subject
sum_y_day_sub <- function(y = dat[,1], day = dat[, 2], b_1) {
  a <- vector()
  for (i in 1:(dim(dat)[1]/10)) {
    a[i] = sum(y[(10*i-9):(10*i)] * day[(10*i-9):(10*i)])
  }
  add = sum(b_1 * a)
  return(add)
}

sum_y_sub <- function(y = dat[, 1], b_0) {
  a <- vector()
  for (i in 1:(dim(dat)[1]/10)) {
    a[i] = sum(y[(10*i-9):(10*i)])
  }
  add2 = sum(b_0 * a)
  return(add2)
}
```

```
gibbs_Q1 <- function(nsim = 10000, burn = 0.1, # chain parameters
                     seed = 1998,
                     y = dat[, 1],
                     day = dat[, 2]) {

  set.seed(1998)
  # initialization----------------------------------
  nsim1 <- nsim * (1 + burn)
  burni <- nsim * burn

  m <- 18 # number of subjects
```

```r
n <- 10 # number of days for each subject

beta_0 <- 0
beta_0.ch <- vector()
beta_1 <- 0
beta_1.ch <- vector()
sigma_sq <- 1
sigma_sq.ch <- vector()
alpha_0 <- 1
alpha_0.ch <- vector()
alpha_1 <- 1
alpha_1.ch <- vector()
b_0 <- matrix(data = rep(0, m), nrow = 1)
b_0.ch <- matrix(data = NA, nrow = nsim, ncol = m)
b_1 <- matrix(data = rep(0, m), nrow = 1)
b_1.ch <- matrix(data = NA, nrow = nsim, ncol = m)

for (i in 1:nsim1) {
# beta_0
  beta0_var <- 100 * sigma_sq / (sigma_sq + 100 * m * n)
  beta0_mean <- 100 * (sum(y) - beta_1 * sum(day) - n * sum(b_0)
                       - sum(day[1:10]) * sum(b_1))/ (sigma_sq + 100 * m * n)
  beta_0 <- rnorm(1, mean = beta0_mean, sd = sqrt(beta0_var))

# beta_1
  beta1_var <- 100 * sigma_sq / (sigma_sq + 100 * sum(day^2))
  beta1_mean <- 100 * (sum(day * y) - beta_0 * sum(day)
                       - sum(day[1:10]) * sum(b_0)
                       - sum((day[1:10])^2) * sum(b_1)) /
    (sigma_sq + 100 * sum(day^2))
  beta_1 <- rnorm(1, mean = beta1_mean, sd = sqrt(beta1_var))

# alpha_0
  a0_shape <- m / 2 + 1
  a0_scale <- sum(b_0^2) / 2 + 1
  alpha_0 <- rinvgamma(1, shape = a0_shape, scale = a0_scale)

# alpha_1
  a1_shape <- m / 2 + 1
  a1_scale <- sum(b_1^2) / 2 + 1
  alpha_1 <- rinvgamma(1, shape = a1_shape, scale = a1_scale)

# sigma_sq
  sigma_sq_shape <- 0.01 + m * n / 2
  sigma_sq_rate <- (0.02 + sum(y^2) - 2*beta_0*sum(y) - 2*beta_1*sum(day*y) -
                      2*sum_y_sub(y, b_0) -
                      2*sum_y_day_sub(y, day, b_1) + m*n*beta_0^2 +
                      2*beta_0*beta_1*sum(day) + 2*beta_0*n*sum(b_0) +
                      2*beta_0*sum(day[1:10])*sum(b_1) +
                      (beta_1^2)*sum(day^2) +
                      2*beta_1*sum(day[1:10])*sum(b_0) +
                      2*beta_1*sum((day[1:10])^2)*sum(b_1) +
                      n*sum((b_0)^2) +
```

```
                        2*sum(day[1:10])*sum(b_0*b_1) +
                        sum((b_1)^2)*sum((day[1:10])^2)
                      ) / 2
    sigma_sq <- rinvgamma(1, shape = sigma_sq_shape, scale = sigma_sq_rate)

  # b_0
    for (j in 1:m) {
      b_0j_var <- alpha_0 * sigma_sq / (sigma_sq + n * alpha_0)
      b_0j_mean <- alpha_0 * (sum(y[(10 * j - 9):(10 * j)])
                              - n*beta_0
                              - beta_1*sum(day[(10 * j - 9):(10 * j)])
                              - b_1[j]*sum(day[(10 * j - 9):(10 * j)])) /
        (sigma_sq + n * alpha_0)
      b_0[j] <- rnorm(1, mean = b_0j_mean, sd = sqrt(b_0j_var))
  # b_1
      b_1j_var <- alpha_1 * sigma_sq / (sigma_sq + alpha_1 *
                                    sum((day[(10 * j - 9):(10 * j)])^2))
      b_1j_mean <- alpha_1 * (sum(y[(10 * j - 9):(10 * j)]
                                  * day[(10 * j - 9):(10 * j)])
                              - beta_0 * sum(day[(10 * j - 9):(10 * j)])
                              - beta_1 * sum((day[(10 * j - 9):(10 * j)])^2)
                              - b_0[j] * sum(day[(10 * j - 9):(10 * j)])) /
        (sigma_sq + alpha_1 * sum((day[(10 * j - 9):(10 * j)]^2)))
      b_1[j] <- rnorm(1, mean = b_1j_mean, sd = sqrt(b_1j_var))
    }

     # Store Chain after burn-----------------------------------------------
    if (i > burni) {
      i1 <- i - burni
      beta_0.ch[i1] <- beta_0
      beta_1.ch[i1] <- beta_1
      alpha_0.ch[i1] <- alpha_0
      alpha_1.ch[i1] <- alpha_1
      sigma_sq.ch[i1] <- sigma_sq
      b_0.ch[i1, ] <- b_0
      b_1.ch[i1, ] <- b_1
    }

  }
  return(list(beta0 = beta_0.ch, beta1 = beta_1.ch,
              alpha0 = alpha_0.ch, alpha1 = alpha_1.ch,
              sigmasq = sigma_sq.ch))
}
```

```
gibbs <- gibbs_Q1(nsim = 30000, burn = 0.2, seed = 123)
```

```
# All population level parameters
para <- Reduce("cbind", gibbs) %>% as.data.frame(.)
colnames(para) <- names(gibbs)

# mean
gibbs_mean <- apply(para, 2, function(x) {
  quantile(x, c(0.025, 0.5, 0.975))}) %>%
  round(., 2) %>%
```

Table 1: Summary Table (Gibbs)

	Mean (95% CI)	SD
beta0	7.41 (-11.88, 26.5)	9.88
beta1	10.39 (7.33, 13.21)	1.49
alpha0	55956.95 (30837.62, 113850.24)	22037.98
alpha1	29.56 (12.46, 70.92)	15.36
sigmasq	664.67 (531.05, 853.36)	82.32

```
  apply(., 2, function(x) {
    paste0(x[2], " (", x[1], ", ", x[3], ")")
    })

# SD
gibbs_sd <- apply(para, 2, sd) %>%
  round(., 2)

cbind(gibbs_mean, gibbs_sd) %>%
  kable(
    caption = "Summary Table (Gibbs)",
    col.names = c("Mean (95% CI)","SD")) %>%
  kable_classic(full_width = F, html_font = "Cambria")
```

(2)
First we let the target posteriors be written as $\pi(x) \propto \exp\{-U(x)\}$, then compute $\nabla U(x)$ to apply Hamiltonian dynamic:

From previous posterior calculations, we know that:

For β_0 :

$$\begin{aligned} p(\beta_0|\beta_1, b_{i0}, b_{i1}, \sigma^2, y) &\propto \exp\left\{-\frac{(\sigma^2 + 100mn)\beta_0^2 - 2\beta_0 \cdot 100 \sum_{i=1}^m \sum_{j=1}^n (y_{ij} - \beta_1 t_{ij} - b_{i0} - b_{i1} t_{ij})}{200\sigma^2}\right\} \\ &= \exp\{-U(\beta_0|\beta_1, b_{i0}, b_{i1}, \sigma^2, y)\} \end{aligned}$$

Therefore

$$\nabla U(\beta_0|\beta_1, b_{i0}, b_{i1}, \sigma^2, y) = (\frac{\sigma^2 + 100mn}{100\sigma^2})\beta_0 - \frac{\sum_{i=1}^m \sum_{j=1}^n (y_{ij} - \beta_1 t_{ij} - b_{i0} - b_{i1} t_{ij})}{\sigma^2}$$

For β_1 :

$$\begin{aligned} p(\beta_1|\beta_0, b_{i0}, b_{i1}, \sigma^2, y) &\propto \exp\left\{-\frac{(\sigma^2 + 100 \sum_{i=1}^m \sum_{j=1}^n t_{ij}^2)\beta_1^2 - 2\beta_1 \cdot 100 \sum_{i=1}^m \sum_{j=1}^n t_{ij}(y_{ij} - \beta_0 - b_{i0} - b_{i1} t_{ij})}{200\sigma^2}\right\} \\ &= \exp\{-U(\beta_1|\beta_0, b_{i0}, b_{i1}, \sigma^2, y)\} \end{aligned}$$

Therefore

$$\nabla U(\beta_1|\beta_0, b_{i0}, b_{i1}, \sigma^2, y) = (\frac{\sigma^2 + 100 \sum_{i=1}^m \sum_{j=1}^n t_{ij}^2}{100\sigma^2})\beta_1 - \frac{\sum_{i=1}^m \sum_{j=1}^n t_{ij}(y_{ij} - \beta_0 - b_{i0} - b_{i1} t_{ij})}{\sigma^2}$$

For α_0 :

$$
\begin{aligned}
p(\alpha_0|b_{i0}) &\propto (\frac{1}{\alpha_0})^{\frac{m}{2}+1+1}\exp\left\{-\frac{\frac{\sum_{i=1}^m b_{i0}^2}{2}+1}{\alpha_0}\right\} \\
&= \exp\left\{-\left[\frac{\frac{\sum_{i=1}^m b_{i0}^2}{2}+1}{\alpha_0}+\left(\frac{m}{2}+2\right)\log(\alpha_0)\right]\right\} \\
&= \exp\left\{-U(\alpha_0|b_{i0})\right\}
\end{aligned}
$$

Therefore

$$
\nabla U(\alpha_0|b_{i0}) = \frac{\frac{m}{2}+2}{\alpha_0} - \frac{\frac{\sum_{i=1}^m b_{i0}^2}{2}+1}{\alpha_0^2}
$$

Similarly, for α_1 :

$$
\nabla U(\alpha_1|b_{i1}) = \frac{\frac{m}{2}+2}{\alpha_1} - \frac{\frac{\sum_{i=1}^m b_{i1}^2}{2}+1}{\alpha_1^2}
$$

For σ^2 :

$$
\begin{aligned}
p(\sigma^2|\beta_0,\beta_1,b_{i0},b_{i1},y) &\propto (\frac{1}{\sigma^2})^{1+0.01+\frac{mn}{2}}\exp\left\{-\frac{\sum_{i=1}^m\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})^2+0.02}{2\sigma^2}\right\} \\
&= \exp\left\{-\left[\frac{\sum_{i=1}^m\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})^2+0.02}{2\sigma^2}+\left(1.01+\frac{mn}{2})\log(\sigma^2\right)\right]\right\} \\
&= \exp\left\{-U(\sigma^2|\beta_0,\beta_1,b_{i0},b_{i1},y)\right\}
\end{aligned}
$$

Therefore

$$
\nabla U(\sigma^2|\beta_0,\beta_1,b_{i0},b_{i1},y) = \frac{1.01+\frac{mn}{2}}{\sigma^2} - \frac{\sum_{i=1}^m\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0}-b_{i1}t_{ij})^2+0.02}{2(\sigma^2)^2}
$$

For b_{i0} :

$$
\begin{aligned}
p(b_{i0}|y_i,\beta_0,\beta_1,b_{i1},\sigma^2,\alpha_0) &\propto \exp\left\{-\frac{(\sigma^2+n\alpha_0)b_{i0}^2-2b_{i0}\alpha_0\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i1}t_{ij})}{2\alpha_0\sigma^2}\right\} \\
&= \exp\left\{-U(b_{i0}|y_i,\beta_0,\beta_1,b_{i1},\sigma^2,\alpha_0)\right\}
\end{aligned}
$$

Therefore

$$
\nabla U(b_{i0}|y_i,\beta_0,\beta_1,b_{i1},\sigma^2,\alpha_0) = (\frac{1}{\alpha_0}+\frac{n}{\sigma^2})b_{i0} - \frac{\sum_{j=1}^n(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i1}t_{ij})}{\sigma^2}
$$

For b_{i1} :

$$
\begin{aligned}
p(b_{i1}|y_i,\beta_0,\beta_1,b_{i0},\sigma^2,\alpha_1) &\propto \exp\left\{-\frac{(\sigma^2+\alpha_1\sum_{j=1}^n t_{ij}^2)b_{i1}^2-2b_{i1}\alpha_1\sum_{j=1}^n t_{ij}(y_{ij}-\beta_0-\beta_1 t_{ij}-b_{i0})}{2\alpha_1\sigma^2}\right\} \\
&= \exp\left\{-U(b_{i1}|y_i,\beta_0,\beta_1,b_{i0},\sigma^2,\alpha_1)\right\}
\end{aligned}
$$

Therefore

$$\nabla U(b_{i1}|y_i, \beta_0, \beta_1, b_{i0}, \sigma^2, \alpha_1) = (\frac{1}{\alpha_1} + \frac{\sum_{j=1}^{n} t_{ij}^2}{\sigma^2})b_{i1} - \frac{\sum_{j=1}^{n} t_{ij}(y_{ij} - \beta_0 - \beta_1 t_{ij} - b_{i0})}{\sigma^2}$$

We define these gradient functions here:

```
partials <- function(input) {
  for(i in names(input)) assign(i, input[[i]])
# beta_0
  dbeta0 <- beta_0 * (sigma_sq + 100*m*n)/(100*sigma_sq) -
    (sum(y) - beta_1 * sum(day) - n * sum(b_0) - sum(day[1:10]) * sum(b_1)) /
    sigma_sq
# beta_1
  dbeta1 <- beta_1 * (sigma_sq + 100 * sum(day^2))/(100*sigma_sq) -
    ((sum(day * y) - beta_0 * sum(day) - sum(day[1:10]) * sum(b_0) -
        sum((day[1:10])^2) * sum(b_1))) / sigma_sq
# alpha_0
  dalpha0 <- (m/2 + 2)/alpha_0 - (sum(b_0^2) / 2 + 1)/(alpha_0^2)

# alpha_1
  dalpha1 <- (m/2 + 2)/alpha_1 - (sum(b_1^2) / 2 + 1)/(alpha_1^2)

# sigma^2
  dsigmasq <- (1.01 + m*n/2)/(sigma_sq) -
    (0.02 + sum(y^2) - 2*beta_0*sum(y) - 2*beta_1*sum(day*y) -
       2*sum_y_sub(y, b_0) -
       2*sum_y_day_sub(y, day, b_1) + m*n*beta_0^2 +
       2*beta_0*beta_1*sum(day) + 2*beta_0*n*sum(b_0) +
       2*beta_0*sum(day[1:10])*sum(b_1) +
       (beta_1^2)*sum(day^2) +
       2*beta_1*sum(day[1:10])*sum(b_0) +
       2*beta_1*sum((day[1:10])^2)*sum(b_1) +
       n*sum((b_0)^2) +
       2*sum(day[1:10])*sum(b_0*b_1) +
       sum((b_1)^2)*sum((day[1:10])^2)
     ) / (2*(sigma_sq)^2)
# b_0
  db0 <- vector()
  for (i in 1:m) {
    db0[i] <- b_0[i]*(1/alpha_0 + n/(sigma_sq)) -
      (sum(y[(10 * i - 9):(10 * i)]) -
         n*beta_0 -
         beta_1*sum(day[(10 * i - 9):(10 * i)]) -
         b_1[i]*sum(day[(10 * i - 9):(10 * i)])) / (sigma_sq)
  }
# b_1
  db1 <- vector()
  for (i in 1:m) {
  db1[i] <- b_1[i]*(1/alpha_1 +sum((day[(10*i - 9):(10*i)]^2))/(sigma_sq)) -
    (sum(y[(10 * i - 9):(10 * i)] * day[(10 * i - 9):(10 * i)])-
       beta_0 * sum(day[(10 * i - 9):(10 * i)]) -
       beta_1 * sum((day[(10 * i - 9):(10 * i)])^2) -
       b_0[i] * sum(day[(10 * i - 9):(10 * i)])) / (sigma_sq)
  }
```

```
  return(list(dbeta0, dbeta1, dalpha0, dalpha1, dsigmasq, db0, db1))
}
```

Use leapfrog steps here:

```
leap <- function(vars, d0, epsilon = 1, nn = 1, M = rep(1, 7)){
  xx <- as.list(1:nn)
  dd <- as.list(1:nn)
  xx[[1]] <- vars
  dd[[1]] <- d0
  for(i in 1:(nn - 1)){
     #i=1
     temp <- lapply(partials(xx[[i]]), function(x) 0.5*epsilon*x)
     dd[[i+1]] <- lapply(c(1:7) %>% as.list,function(x){dd[[i]][[x]]-temp[[x]]})

     xx[[i+1]] <- lapply(c(1:7) %>% as.list, function(x){
       xx[[i]][[x]] + epsilon*dd[[i+1]][[x]]/(2*M[x]^2)})
     names(xx[[i+1]]) <- names(vars)

     temp <- lapply(partials(xx[[i+1]]), function(x) 0.5*epsilon*x)
     dd[[i+1]] <- lapply(c(1:7) %>% as.list,function(x){dd[[i+1]][[x]]-
         temp[[x]]})
  }
  return(xx[[nn]])
}
```

```
hmc.gibbs <- function(nsim = 10000, burn = 0.1, # chain parameters
                      seed = 1998) {
  set.seed(1998)
  # initialization----------------------------------
  nsim1 <- nsim * (1 + burn)
  burni <- nsim * burn

  beta_0 <- 0
  beta_0.ch <- vector()
  beta_1 <- 0
  beta_1.ch <- vector()
  sigma_sq <- 1
  sigma_sq.ch <- vector()
  alpha_0 <- 1
  alpha_0.ch <- vector()
  alpha_1 <- 1
  alpha_1.ch <- vector()
  b_0 <- matrix(data = rep(0, m), nrow = 1)
  b_0.ch <- matrix(data = NA, nrow = nsim, ncol = m)
  b_1 <- matrix(data = rep(0, m), nrow = 1)
  b_1.ch <- matrix(data = NA, nrow = nsim, ncol = m)

  for (i in 1:nsim1) {
  # beta_0
    beta0_var <- 100 * sigma_sq / (sigma_sq + 100 * m * n)
    beta0_mean <- 100 * (sum(y) - beta_1 * sum(day) - n * sum(b_0)
                     - sum(day[1:10]) * sum(b_1))/ (sigma_sq + 100 * m * n)
    beta_0 <- rnorm(1, mean = beta0_mean, sd = sqrt(beta0_var))
```

```r
# beta_1
  beta1_var <- 100 * sigma_sq / (sigma_sq + 100 * sum(day^2))
  beta1_mean <- 100 * (sum(day * y) - beta_0 * sum(day)
                      - sum(day[1:10]) * sum(b_0)
                      - sum((day[1:10])^2) * sum(b_1)) /
    (sigma_sq + 100 * sum(day^2))
  beta_1 <- rnorm(1, mean = beta1_mean, sd = sqrt(beta1_var))

# alpha_0
  a0_shape <- m / 2 + 1
  a0_scale <- sum(b_0^2) / 2 + 1
  alpha_0 <- rinvgamma(1, shape = a0_shape, scale = a0_scale)

# alpha_1
  a1_shape <- m / 2 + 1
  a1_scale <- sum(b_1^2) / 2 + 1
  alpha_1 <- rinvgamma(1, shape = a1_shape, scale = a1_scale)

# sigma_sq
  sigma_sq_shape <- 0.01 + m * n / 2
  sigma_sq_rate <- (0.02 + sum(y^2) - 2*beta_0*sum(y) - 2*beta_1*sum(day*y) -
                    2*sum_y_sub(y, b_0) -
                    2*sum_y_day_sub(y, day, b_1) + m*n*beta_0^2 +
                    2*beta_0*beta_1*sum(day) + 2*beta_0*n*sum(b_0) +
                    2*beta_0*sum(day[1:10])*sum(b_1) +
                    (beta_1^2)*sum(day^2) +
                    2*beta_1*sum(day[1:10])*sum(b_0) +
                    2*beta_1*sum((day[1:10])^2)*sum(b_1) +
                    n*sum((b_0)^2) +
                    2*sum(day[1:10])*sum(b_0*b_1) +
                    sum((b_1)^2)*sum((day[1:10])^2)
                  ) / 2
  sigma_sq <- rinvgamma(1, shape = sigma_sq_shape, scale = sigma_sq_rate)

# b_0
  for (j in 1:m) {
    b_0j_var <- alpha_0 * sigma_sq / (sigma_sq + n * alpha_0)
    b_0j_mean <- alpha_0 * (sum(y[(10 * j - 9):(10 * j)])
                          - n*beta_0
                          - beta_1*sum(day[(10 * j - 9):(10 * j)])
                          - b_1[j]*sum(day[(10 * j - 9):(10 * j)])) /
      (sigma_sq + n * alpha_0)
    b_0[j] <- rnorm(1, mean = b_0j_mean, sd = sqrt(b_0j_var))
# b_1
    b_1j_var <- alpha_1 * sigma_sq / (sigma_sq + alpha_1 *
                                      sum((day[(10 * j - 9):(10 * j)])^2))
    b_1j_mean <- alpha_1 * (sum(y[(10 * j - 9):(10 * j)]
                              * day[(10 * j - 9):(10 * j)])
                          - beta_0 * sum(day[(10 * j - 9):(10 * j)])
                          - beta_1 * sum((day[(10 * j - 9):(10 * j)])^2)
                          - b_0[j] * sum(day[(10 * j - 9):(10 * j)])) /
      (sigma_sq + alpha_1 * sum((day[(10 * j - 9):(10 * j)]^2)))
    b_1[j] <- rnorm(1, mean = b_1j_mean, sd = sqrt(b_1j_var))
```

```r
    }

    # Leapfrog Path
    vars <- list(beta_0 = beta_0, beta_1 = beta_1, alpha_0 = alpha_0,
                 alpha_1 = alpha_1, sigma_sq = sigma_sq, b_0 = b_0, b_1 = b_1)

    d0 <- lapply(vars, function(x){rnorm(n = length(x), mean = 0, sd = M)})

    leap_result <- leap(vars, d0 = d0, epsilon = epsilon, nn = nn, M = M)

    names(leap_result) <- names(vars)

    for(names in names(leap_result)) {
      assign(names, leap_result[[names]])
    }

    # Store Chain after burn-----------------------------------------------
    if (i > burni) {
      i1 <- i - burni
      beta_0.ch[i1] <- beta_0
      beta_1.ch[i1] <- beta_1
      alpha_0.ch[i1] <- alpha_0
      alpha_1.ch[i1] <- alpha_1
      sigma_sq.ch[i1] <- sigma_sq
      b_0.ch[i1, ] <- b_0
      b_1.ch[i1, ] <- b_1
    }
  }
  return(list(beta0 = beta_0.ch, beta1 = beta_1.ch,
              alpha0 = alpha_0.ch, alpha1 = alpha_1.ch,
              sigmasq = sigma_sq.ch))
}
```

```r
# initial value setting
y = dat[, 1]
day = dat[, 2]
m  = 18
n = 10
nsim <- 30000
M <- c(500, rep(100, 6))
epsilon <- 2
nn <- 15

HMC <- hmc.gibbs(nsim = nsim, burn = 0.2, seed = 123)
```

```r
# All population level parameters
para2 <- Reduce("cbind", HMC) %>% as.data.frame(.)
colnames(para2) <- names(HMC)

# mean
HMC_mean <- apply(para2, 2, function(x) {
  quantile(x, c(0.025, 0.5, 0.975))}) %>%
  round(., 2) %>%
  apply(., 2, function(x) {
```

Table 2: Summary Table (HMC)

	Mean (95% CI)	SD
beta0	7.09 (-20.58, 36.56)	14.51
beta1	10.35 (6.58, 14.15)	1.9
alpha0	55915.19 (29930.69, 117001.05)	22665.15
alpha1	30.57 (12.84, 72.97)	15.83
sigmasq	668.58 (534.7, 855.74)	82.08

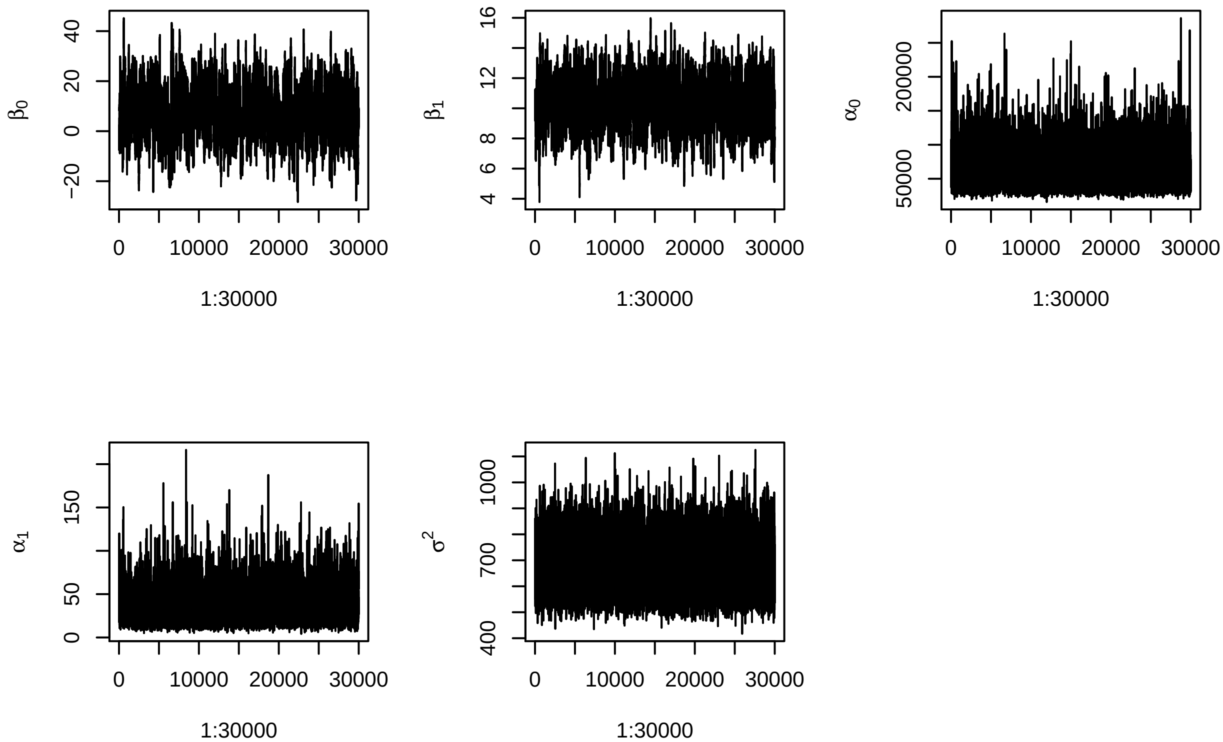
```
    paste0(x[2], " (", x[1], ", ", x[3], ")")
    })

# SD
HMC_sd <- apply(para2, 2, sd) %>%
  round(., 2)

cbind(HMC_mean, HMC_sd) %>%
  kable(
    caption = "Summary Table (HMC)",
    col.names = c("Mean (95% CI)","SD")) %>%
  kable_classic(full_width = F, html_font = "Cambria")
```

Two algorithms showed similar summary statistics with 30000 simulations and 6000 burn-in iterations. The posterior mean and standard deviation of $\beta_0, \beta_1, \alpha_0, \alpha_1$ obtained from Hamiltonian MC Sampler were larger. However, for σ^2, the posterior mean and standard deviation obtained from Gibbs Sampler was larger. The length of 95 % credible confidence interval of five population-level parameters got from two algorithms were not obviously different.

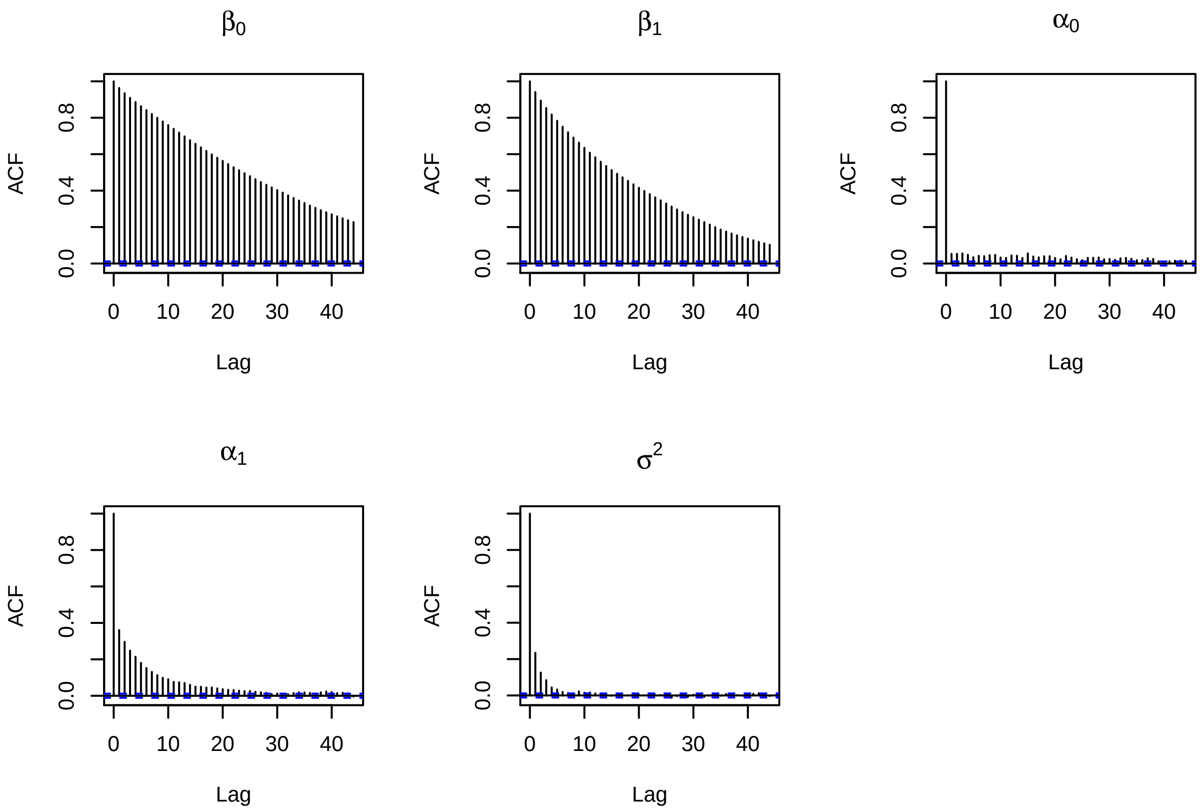
(3)

Gibbs Sampler:

```
par(mfrow = c(2, 3))
plot(1:30000, gibbs$beta0, ylab = expression(beta[0]), type = "l")
plot(1:30000, gibbs$beta1, ylab = expression(beta[1]), type = "l")
plot(1:30000, gibbs$alpha0, ylab = expression(alpha[0]), type = "l")
plot(1:30000, gibbs$alpha1, ylab = expression(alpha[1]), type = "l")
plot(1:30000, gibbs$sigmasq, ylab = expression(sigma^{2}), type = "l")
```

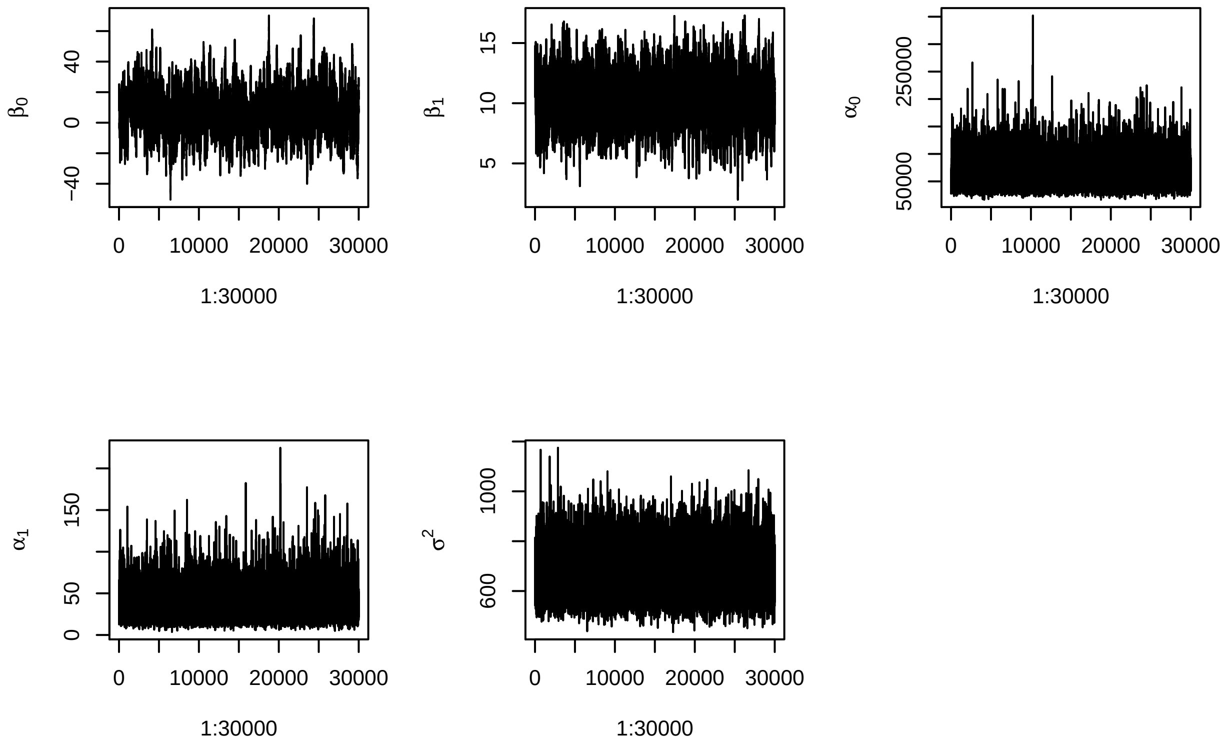



```
par(mfrow = c(2, 3))
acf(gibbs$beta0, main = expression(beta[0]))
acf(gibbs$beta1, main = expression(beta[1]))
acf(gibbs$alpha0, main = expression(alpha[0]))
acf(gibbs$alpha1, main = expression(alpha[1]))
acf(gibbs$sigmasq, main = expression(sigma^{2}))
```

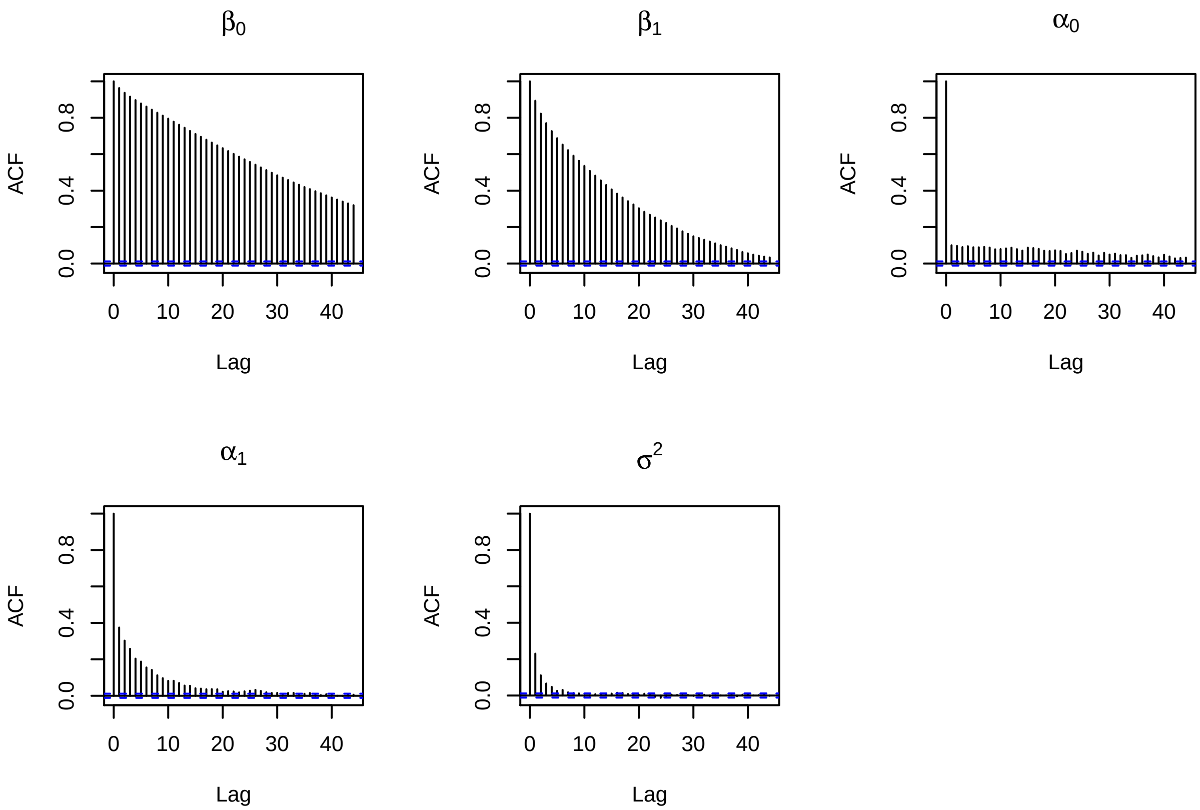


Gibbs Sampler with Hamiltonian MC Sampler:

```
par(mfrow = c(2, 3))
plot(1:30000, HMC$beta0, ylab = expression(beta[0]), type = "l")
plot(1:30000, HMC$beta1, ylab = expression(beta[1]), type = "l")
plot(1:30000, HMC$alpha0, ylab = expression(alpha[0]), type = "l")
plot(1:30000, HMC$alpha1, ylab = expression(alpha[1]), type = "l")
plot(1:30000, HMC$sigmasq, ylab = expression(sigma^{2}), type = "l")
```




```
par(mfrow = c(2, 3))
acf(HMC$beta0, main = expression(beta[0]))
acf(HMC$beta1, main = expression(beta[1]))
acf(HMC$alpha0, main = expression(alpha[0]))
acf(HMC$alpha1, main = expression(alpha[1]))
acf(HMC$sigmasq, main = expression(sigma^{2}))
```



Two algorithms converged to similar convergence with 30000 simulations and 6000 burn-in iterations. For β_0, the Gibbs sampler with Hamiltonian MC Sampler had worse mixture compared to Gibbs Sampler. However, the mixture of β_1 would be better if we used Hamiltonian MC Sampler. The mixture of other three parameters are similar.

Hamiltonian MC Sampler relies on the initial momentum vector δ_0 and mass parameter m, we normally can get better result from Hamiltonian MC Sampler (not too much in this project), but Gibbs Sampler process can be understood more easily and the code complexity is lower.