STA 360: Lab 7

Michael Lin

March 18, 2015

1. Full conditional of β_{0i}:

$$\begin{aligned}
\beta_{0i}|Y_{ij}, \beta_{1i}, \tau, \tau_0, \tau_1 &= \beta_{0i}|Y_{ij} \sim p(\beta_{0i}) \prod_{j=1}^{p} p(Y_{ij}) \\
&= \mathrm{No}(\beta_{0i}|\mu_0, \tau_0) \prod_{j=1}^{p} \mathrm{No}(Y_{ij}|\beta_{0i} + \beta_{1i}x_j, \tau) \\
&\propto \exp\{-\frac{(\beta_{0i} - \mu_0)^2}{2\tau_0}\} \prod_{j=1}^{p} \exp\{-\frac{(Y_{ij} - (\beta_{0i} + \beta_{1i}x_j))^2}{2\tau}\} \\
&= \exp\{-\frac{(\beta_{0i} - \mu_0)^2}{2\tau_0} - \frac{\sum_{j=1}^{n}(Y_{ij} - (\beta_{0i} + \beta_{1i}x_j))^2}{2\tau}\} \\
&\propto \exp\{-\frac{\beta_{0i}^2 - 2\beta_{0i}\mu_0}{2\tau_0} - \frac{p\beta_{0i}^2 + \beta_{0i}\sum_{j=1}^{p}(\beta_{1i}x_j - Y_{ij})}{\tau}\} \\
&= \exp\{-(\frac{1}{2\tau_0} + \frac{p}{2\tau})\beta_{0i}^2 + [\frac{\mu_0}{\tau_0} - \frac{\sum_{j=1}^{p}(\beta_{1i}x_j - Y_{ij})}{\tau}]\beta_{0i}\} \\
&= \exp\{-\frac{1}{2}(\frac{1}{\tau_0} + \frac{p}{\tau})[\beta_{0i}^2 - 2\beta_{0i}\frac{\frac{\mu_0}{\tau_0} - \frac{\sum_{j=1}^{p}(\beta_{1i}x_j - Y_{ij})}{\tau}}{\frac{1}{\tau_0} + \frac{p}{2\tau}}]\} \\
&\propto \mathrm{No}(\beta_{0i}|\mu', \tau')
\end{aligned}$$

where

$$\mu' = \frac{\frac{\mu_0}{\tau_0} + \frac{\sum_{j=1}^{p} Y_{ij}}{\tau}}{\frac{1}{\tau_0} + \frac{p}{2\tau}}$$

$$\tau' = (\frac{1}{\tau_0} + \frac{p}{\tau})^{-1}$$

2. Full conditional of β_{1i}:

$$\begin{aligned}
\beta_{1i}|Y_{ij}, \beta_{0i}, \tau, \tau_0, \tau_1 &= \beta_{1i}|Y_{ij} \sim p(\beta_{1i}) \prod_{j=1}^{p} p(Y_{ij}) \\
&= \mathrm{No}(\beta_{1i}|\mu_1, \tau_1) \prod_{j=1}^{p} \mathrm{No}(Y_{ij}|\beta_{0i} + \beta_{1i}x_j, \tau) \\
&\propto \exp\{-\frac{(\beta_{1i} - \mu_1)^2}{2\tau_1}\} \prod_{j=1}^{p} \exp\{-\frac{(Y_{ij} - (\beta_{0i} + \beta_{1i}x_j))^2}{2\tau}\} \\
&= \exp\{-\frac{(\beta_{1i} - \mu_1)^2}{2\tau_1} - \frac{\sum_{j=1}^{p}(Y_{ij} - (\beta_{0i} + \beta_{1i}x_j))^2}{2\tau}\} \\
&\propto \exp\{-\frac{\beta_{1i}^2 - 2\beta_{1i}\mu_1}{2\tau_1} - \frac{\sum_{j=1}^{p}(\beta_{1i}^2 x_j^2 - 2\beta_{1i}x_j(Y_{ij} - \beta_{0i}))}{2\tau}\} \\
&= \exp\{-\frac{1}{2}(\frac{1}{\tau_1} + \frac{\sum_{j=1}^{p} x_j^2}{\tau})\beta_{1i}^2 + [\frac{\mu_1}{\tau_1} + \frac{\sum_{j=1}^{p} x_j(Y_{ij} - \beta_{0i})}{\tau}]\beta_{1i}\} \\
&= \exp\{-\frac{1}{2}(\frac{1}{\tau_1} + \frac{\sum_{j=1}^{p} x_j^2}{\tau})[\beta_{1i}^2 - 2\beta_{1i}\frac{\tau\mu_1}{\tau + \tau_1\sum_{j=1}^{p} x_j^2}]\} \\
&\propto \mathrm{No}(\beta_{1i}|\mu'', \tau'')
\end{aligned}$$

where

$$\mu'' = \frac{\tau\mu_1}{\tau + \tau_1\sum_{j=1}^{p} x_j^2}$$

$$\tau'' = (\frac{1}{\tau_1} + \frac{\sum_{j=1}^{p} x_j^2}{\tau})^{-1}$$

3. Gibbs sampler (see R code) with 1000 post-burnin draws was implemented. It was found that:

$$\Pr(\beta_{1i} > 0.5) = 0.3349$$

.

R code for Gibbs sampler:

```
##### Initialize Workspace #####
## Load Data ##
data=read.table('hier.txt')
Y = as.matrix(na.omit(data[,2:11]))
library(MCMCpack)

## Define Parameters ##
a = 5; lam = 4;
m0 = 12; s0 = 1; m1 = 1; s1 = 1;
xj = c(-4.5, -3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.5, 4.5)

# (Need only keep track of beta1, beta0) #
## Initialize Vector ##
n = dim(Y)[1]; p = dim(Y)[2];
N = 1000; B = 100;
beta0 = NULL
beta1 = NULL


##### Gibbs Sampler #####
## Priors for tau's ##
tau.vec = rinvgamma(3, shape = a, scale = lam)
tau = tau.vec[1]
tau0 = tau.vec[2]
tau1 = tau.vec[3]

## Sampler ##
for(j in 1:(N+B)){
  beta0.vec = rep(0,n)
  beta1.vec = rep(0,n)

  mu0 = rnorm(1, m0, s0)
  mu1 = rnorm(1, m1, s1)

  for(i in 1:n){
    yj = Y[i,]

    beta0.temp = rnorm(1, (tau*mu0+tau0*sum(yj))/(tau+p*tau0/2),
                       sqrt(tau*tau0/(tau+p*tau0)))
    beta1.temp = rnorm(1, (tau*mu1)/(tau+tau1*sum(xj^2)),
                       sqrt(tau1*tau/(tau+tau1*sum(xj^2))))

    beta0.vec[i] = beta0.temp
    beta1.vec[i] = beta1.temp

  }

  ## Update tau's ##
  summation = 0;
  for(w in 1:n){
    for(z in 1:p){
      summation = summation + (Y[w,z] - (beta0.vec[w] + beta1.vec[w]*xj[z]))^2
    }
```

```
  }

  tau.vec = rinvgamma(3, shape = a + 5*n, scale = lam + 0.5*summation)
  tau = tau.vec[1]
  tau0 = tau.vec[2]
  tau1 = tau.vec[3]

  beta0 = cbind(beta0, beta0.vec)
  beta1 = cbind(beta1, beta1.vec)

}

## Post Burn In Sample ##
int = beta0[,(B+1):(B+N)]
slope = beta1[,(B+1):(B+N)]

## Slope > 0.5 ##
print(mean(slope>0.5))

##### Autocorrelation #####
int.acf = NULL
slope.acf = NULL
for(i in 1:n){
  acf.temp = c(acf(int[i,], plot=F)[[1]])
  int.acf = rbind(int.acf, acf.temp)

  acf.temp = c(acf(slope[i,], plot=F)[[1]])
  slope.acf = rbind(slope.acf, acf.temp)
}
print(mean(int.acf>slope.acf))
```