Homework 2

Ian Frankenburg

5/8/2020

Generalized Linear Models

Part (1).

Describe and implement a Metropolis-Hastings algorithm designed to obtain a MC with stationary distribution $p(\beta|Y)$

This MCMC implementation is pretty strightforward. I know my target $p(\boldsymbol{\beta}|\boldsymbol{y})$ is propportional to

$$\Big[\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}\Big]\exp\Big(-\frac{1}{2n}\boldsymbol{\beta}^\top(X^\top X)\boldsymbol{\beta}\Big),\text{ where } p_i=\Phi(x_i^\top\boldsymbol{\beta}).$$

My Metropolis algorithm will make symmetric proposals based around the current iteration of $\boldsymbol{\beta}$ with variance defined to be the inverse sample covariance $n(X^\top X)^{-1}$:

$$\boldsymbol{\beta}^* \sim N(\boldsymbol{\beta}^{(s)}, n(X^\top X)^{-1}).$$

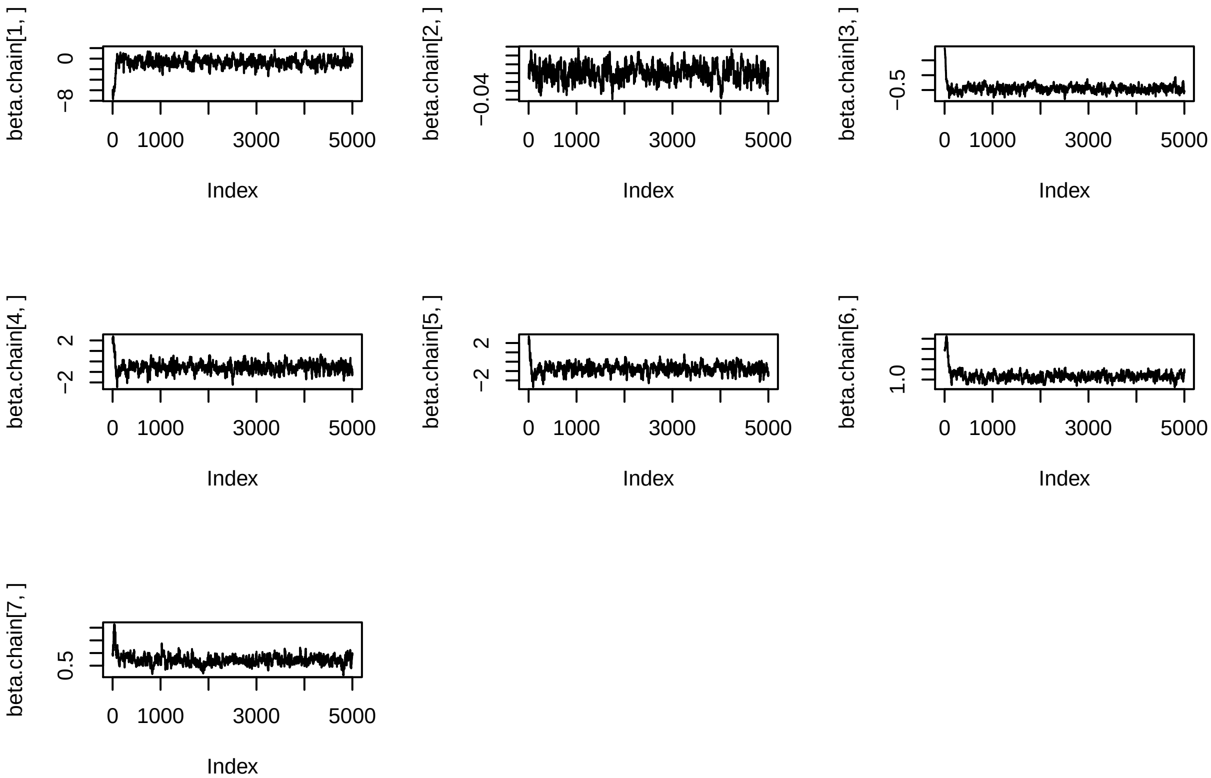
```
require(survival)
```

```
## Loading required package: survival
```

```
data <- infert
target <- function(y, x, xtx, beta, n){
  p <- pnorm(x%*%beta)
  return(sum(y*log(p)+(1-y)*log(1-p))-1/(2*n)*t(beta)%*%xtx%*%beta)
}
samples <- 5000
y <- data$case
n <- length(y)
data$education <- as.numeric(as.factor(data$education))
edu1 <- data$education==2; edu2 <- data$education==3
x <- cbind(rep(1,n),data$age, data$parity, edu1, edu2, data$spontaneous, data$induced)
ncolx <- ncol(x)
xtx <- t(x)%*%x
xtx_inv <- solve(xtx)
beta <- t(rmvnorm(1,rep(0,ncol(x)),xtx_inv*n))
beta.chain <- matrix(0,nrow=ncolx,ncol=samples)
beta.chain[,1] <- beta
for(s in 2:samples){
  beta_new <- t(rmvnorm(1,as.vector(beta), xtx_inv))
  logr <- target(y,x,xtx,beta_new,n)-target(y,x,xtx,beta,n)
  if(log(runif(1)) < logr){beta <- beta_new}
```

```
  beta.chain[,s] <- beta
}
```

```
##        (Intercept)    x[, -1]    x[, -1] x[, -1]edu1 x[, -1]edu2  x[, -1]
## Bayes  -0.7085820 0.02092778 -0.4457638  -0.5614259  -0.7648931 1.176969
## GLM    -0.6413262 0.02056090 -0.4544283  -0.5705585  -0.7987104 1.173780
##          x[, -1]
## Bayes 0.7270190
## GLM   0.7214959
```



Part (2).

Describe and implement a data augmented (DA-MCMC) strategy targeting $p(\beta|Y)$

The original model is

$$P(y_i = 1|x_i, \beta) = \Phi(x_i^\top \beta).$$

This is equivalent to the model $P(y_i = 1|x_i, \beta) = P(\mathbf{1}(z_i > 0) = 1) = P(z_i > 0)$, where $z_i \overset{\text{iid}}{\sim} N(x_i^\top \beta, 1)$. This follows immediately since

$$\int_{-\infty}^{x_i^\top \beta} N(t;\ 0, 1)dt = \int_{-\infty}^{0} N(z_i; x_i^\top \beta, 1)dz_i$$

by a change of variables $z_i := t - x_i^\top \beta$. Thus $\Phi(x_i^\top \beta) = P(z_i > 0)$.

In defining the latent model, the full conditionals $p(\beta|\boldsymbol{y}, z)$, $p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\beta})$ become tractible, so I can use a Gibbs sampler.

I'll start with $p(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{z})$

$$\begin{aligned} p(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{z}) &\propto p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\beta})p(\boldsymbol{\beta}) \\ &= \cancel{p(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{z})}p(\boldsymbol{z}|\boldsymbol{\beta})p(\boldsymbol{\beta}) = p(\boldsymbol{z}|\boldsymbol{\beta})p(\boldsymbol{\beta}) \\ &= N(\boldsymbol{z}; X\boldsymbol{\beta}, \boldsymbol{I})N(\boldsymbol{\beta}; \mathbf{0}, n(X^\top X)^{-1}) \\ &= N(\frac{n}{n+1}(X^\top X)^{-1}X^\top \boldsymbol{y}, \frac{n}{n+1}(X^\top X)^{-1}) \end{aligned}$$

Now for $p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\beta})$.

$$\begin{aligned} p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\beta}) &\propto p(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{z}) \\ &\propto p(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{z})p(\boldsymbol{z}|\boldsymbol{\beta})\cancel{p(\boldsymbol{\beta})} \\ &\propto \prod_{i=1}^{n}[\mathbf{1}(y_i = 1)\mathbf{1}(z_i > 0) + \mathbf{1}(y_i = 0)\mathbf{1}(z_i < 0)]N(\boldsymbol{z}; X\boldsymbol{\beta}, \boldsymbol{I}) \end{aligned}$$

Since our sampling model assumes the y's are independent, so are the z's and I can sample the full conditionals independently, i.e.

$$\begin{aligned} p(z_i|\boldsymbol{y}, \boldsymbol{\beta}) &\propto p(\boldsymbol{y}, \boldsymbol{\beta}, z_i) \propto p(y|\boldsymbol{\beta}, z_i)p(z_i|\beta) \\ &\propto 1(y_i = 1)\mathbf{1}(z_i > 0) + \mathbf{1}(y_i = 0)\mathbf{1}(z_i < 0)]N(z_i; x_i^\top \boldsymbol{\beta}, 1) \\ &= \begin{cases} N(z_i; x_i^\top \boldsymbol{\beta}, 1) * \mathbf{1}_{[0,\infty)}(z_i) & \text{if } y_i = 1 \\ N(z_i; x_i^\top \boldsymbol{\beta}, 1) * \mathbf{1}_{(-\infty,0)}(z_i) & \text{if } y_i = 0 \end{cases} \end{aligned}$$

Now I can implement a Gibbs sampler to iteratively draw from these conditionals.

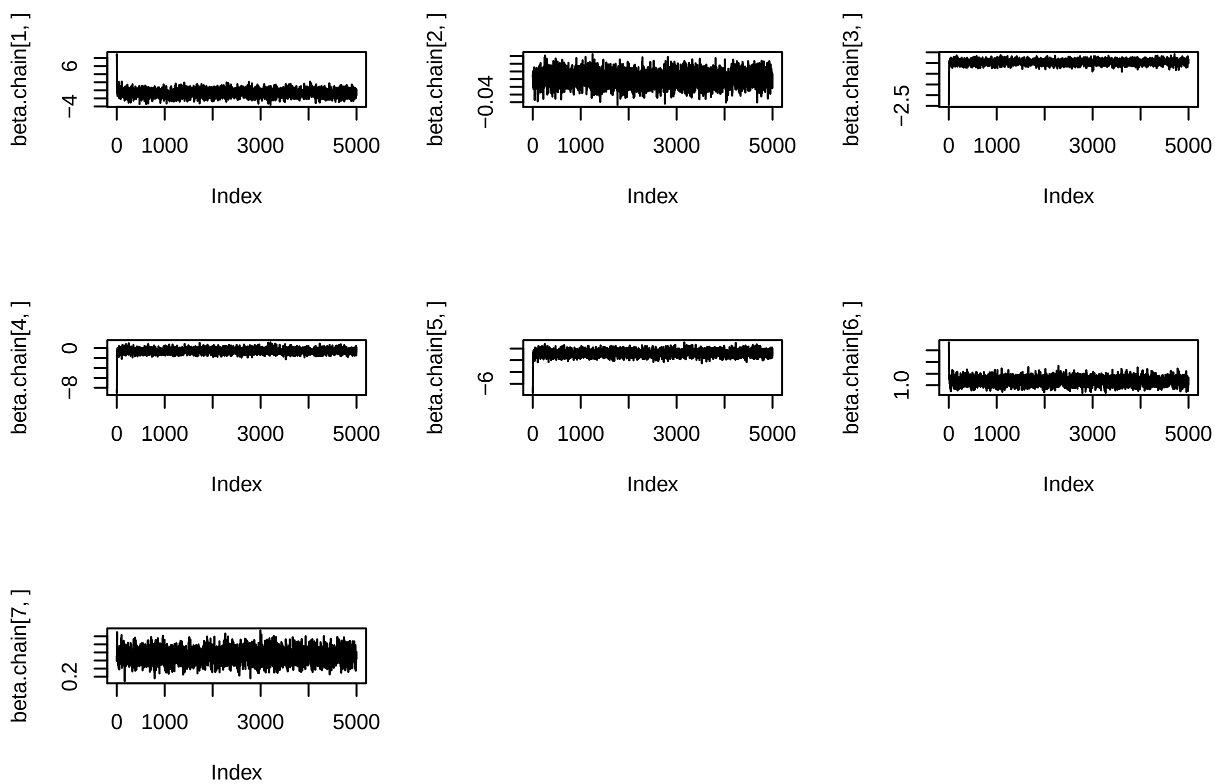
```
require(truncnorm)
```

```
## Loading required package: truncnorm
```

```
samples <- 5000
data <- infert
y <- data$case
n <- length(y)
data$education <- as.numeric(as.factor(data$education))
edu1 <- data$education==2; edu2 <- data$education==3
x <- cbind(rep(1,n),data$age, data$parity, edu1, edu2, data$spontaneous, data$induced)
p <- ncol(x)
xtx <- t(x)%*%x
```

```r
xtx_inv <- solve(xtx)
beta <- t(rmvnorm(1,rep(0,ncol(x)),xtx_inv*n))
beta.chain <- matrix(0,nrow=p,ncol=samples)
beta.chain[,1] <- beta
z <- t(rmvnorm(1,x%*%beta,sigma=diag(n)))
for(s in 2:samples){
  beta <- t(rmvnorm(1, n/(n+1)*xtx_inv%*%t(x)%*%z, n/(n+1)*xtx_inv))
  # truncated normal
  for(i in 1:n){
    if(y[i]==1){
      z[i] <- rtruncnorm(1, a = 0, b = Inf, mean = matrix(x[i,],ncol=p)%*%beta, sd=1)
    }else{
      z[i] <- rtruncnorm(1, a = -Inf, b = 0, mean = matrix(x[i,],ncol=p)%*%beta, sd=1)
    }
  }
  beta.chain[,s] <- beta
}
```

```
##        (Intercept)    x[, -1]    x[, -1] x[, -1]edu1 x[, -1]edu2 x[, -1]
## Bayes  -0.6664524 0.02116593 -0.4601014  -0.5598672  -0.7926758 1.18323
## GLM    -0.6413262 0.02056090 -0.4544283  -0.5705585  -0.7987104 1.17378
##          x[, -1]
## Bayes 0.7268029
## GLM   0.7214959
```



Part (3).

Describe and implement a parameter expanded-data augmentation (PX-DA MCMC) algorithm targeting $p(\beta|Y)$.

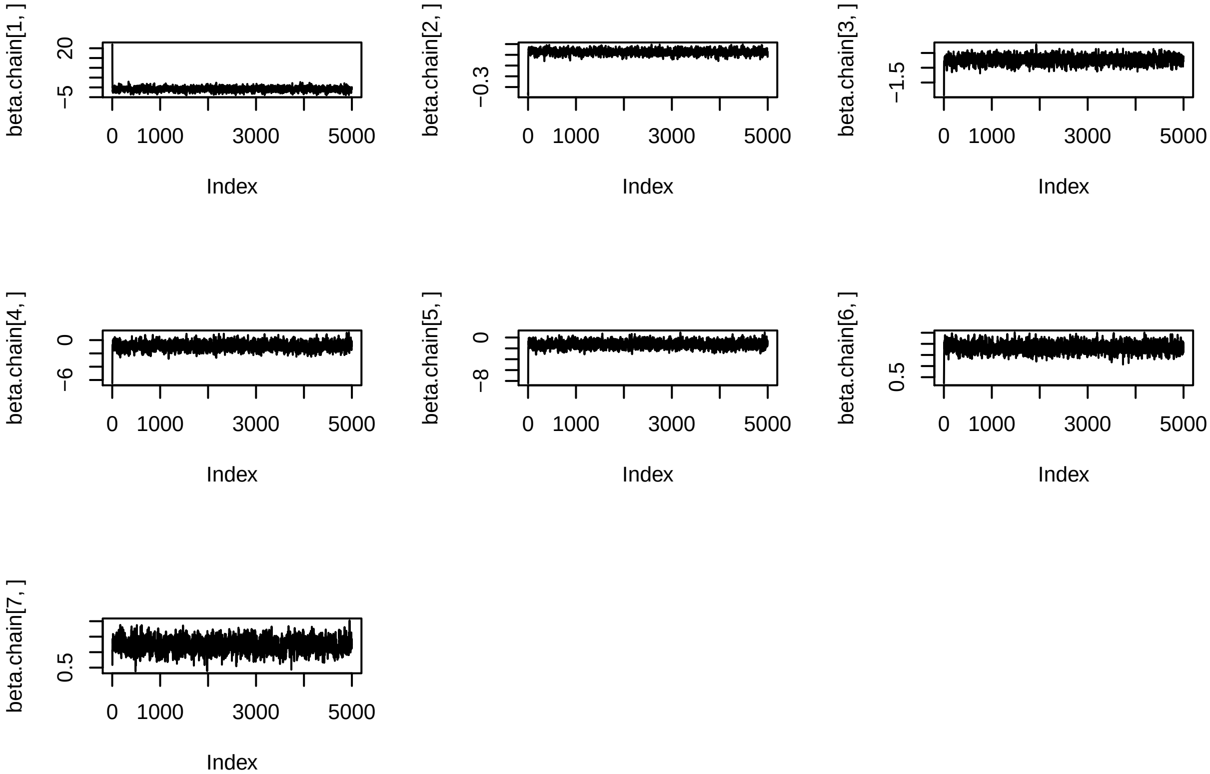
For this parameter-expanded model, I'll introduce another parameter $\alpha^2 \sim IG(a,b)$ and I'll consider the transformation $w_i := \alpha z_i$. Then I'll use a Gibbs sampler to sample iteratively from the conditionals

$$
\begin{aligned}
p(\boldsymbol{\beta}|\boldsymbol{y},\boldsymbol{w},\alpha) &\propto \prod_{i=1}^{n} N(w_i;\alpha x_i^\top \boldsymbol{\beta},\alpha^2) N(\boldsymbol{\beta};0,n(X^\top X)^{-1}) \\
&= N(Mm, M), \text{ where } M = \frac{n}{n+1}(X^\top X)^{-1} \text{ and } m = MX^\top \boldsymbol{w}/\alpha \\
p(w_i|\boldsymbol{y},\boldsymbol{\beta},\alpha) &= \begin{cases} N(w_i;\alpha x_i^\top \boldsymbol{\beta},\alpha^2) * \mathbf{1}_{[0,\infty)}(w_i) & \text{if } y_i = 1 \\ N(w_i;\alpha x_i^\top \boldsymbol{\beta},\alpha^2) * \mathbf{1}_{(-\infty,0)}(w_i) & \text{if } y_i = 0 \end{cases} \\
p(\alpha|\boldsymbol{y},\boldsymbol{\beta},\boldsymbol{z}) &= \prod_{i=1}^{n} N(w_i;\alpha x_i^\top \boldsymbol{\beta},\alpha^2) IG(\alpha^2;a,b)
\end{aligned}
$$

```
require(truncnorm)
samples <- 5000
data <- infert
y <- data$case
data$education <- as.numeric(as.factor(data$education))
edu1 <- data$education==2; edu2 <- data$education==3
x <- cbind(rep(1,n),data$age, data$parity, edu1, edu2, data$spontaneous, data$induced)
n <- nrow(x); p <- ncol(x)
xtx <- t(x)%*%x
xtx_inv <- solve(xtx)
beta <- t(rmvnorm(1,rep(0,ncol(x)),xtx_inv*n))
beta.chain <- matrix(0,nrow=p,ncol=samples)
beta.chain[,1] <- beta
alpha2 <- 1
z <- t(rmvnorm(1,x%*%beta,sigma=alpha2 * diag(n)))
for(s in 2:samples){
  beta <- t(rmvnorm(1, n/(n+1)*xtx_inv%*%t(x)%*%z%*%solve(sqrt(alpha2)), n/(n+1)*xtx_inv))
  # truncated normal
  for(i in 1:n){
    if(y[i]==1){
      z[i] <- rtruncnorm(1, a = 0, b = Inf, mean = matrix(x[i,],ncol=p)%*%beta, sd=1)
    }else{
      z[i] <- rtruncnorm(1, a = -Inf, b = 0, mean = matrix(x[i,],ncol=p)%*%beta, sd=1)
    }
  }
  rss <- t((z-x%*%beta))%*%(z-x%*%beta)/2
  d <- rchisq(1, df=n)
  alpha2 <- sqrt(rss/d)
  beta.chain[,s] <- beta
}
```

```
##       (Intercept)    x[, -1]    x[, -1] x[, -1]edu1 x[, -1]edu2  x[, -1]
## Bayes  -0.8661613 0.02697548 -0.7355791  -0.8758183  -1.2404586 1.847926
## GLM    -0.6413262 0.02056090 -0.4544283  -0.5705585  -0.7987104 1.173780
##          x[, -1]
```

```
## Bayes 1.2268688
## GLM   0.7214959
```



Part (6).

For logit model, describe and a random walk MH targeting $p(\beta|Y)$

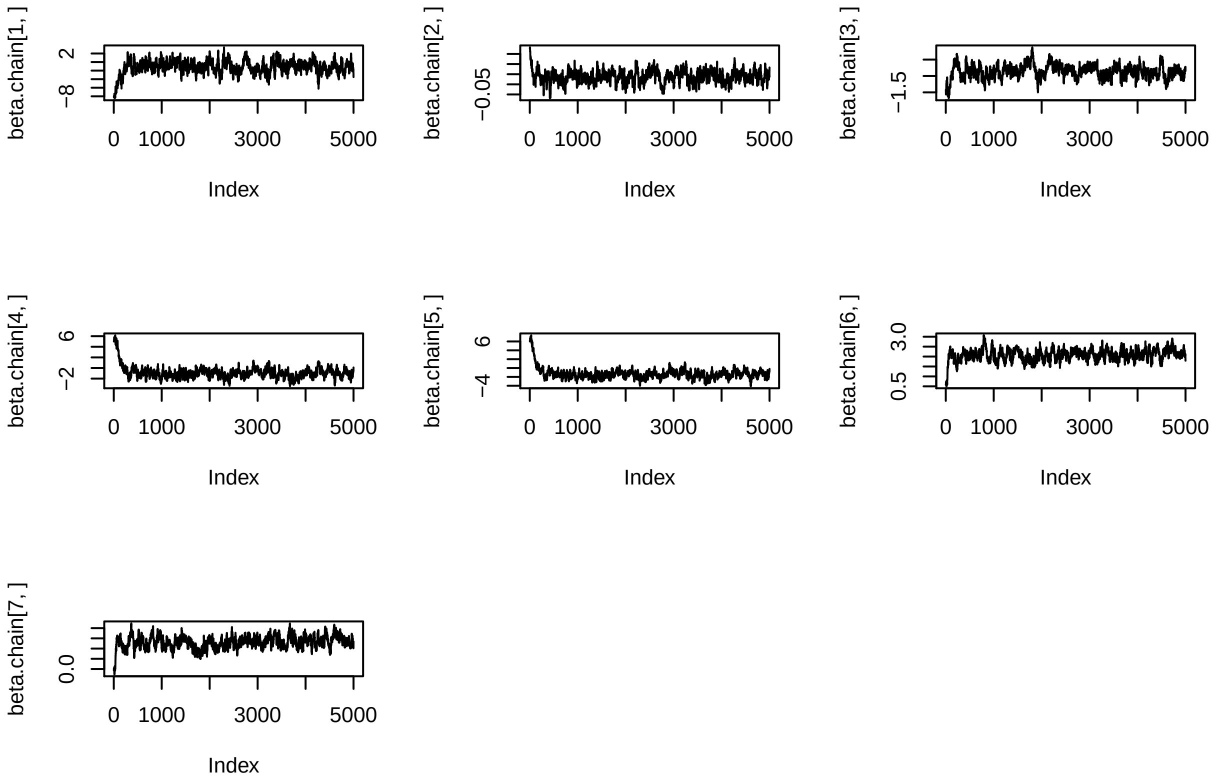
Similar to the probit case, the posterior is intractible but can be written as

$$
\begin{aligned}
p(\beta|\boldsymbol{y}) &\propto \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} p(\beta) \\
&= \prod_{i=1}^{n} \left[\frac{e^{x_i^\top \beta}}{1+e^{x_i^\top \beta}}\right]^{y_i} \left[1-\frac{e^{x_i^\top \beta}}{1+e^{x_i^\top \beta}}\right]^{1-y_i} N(\boldsymbol{\beta}; 0, n(X^\top X)^{-1}) \\
&= \prod_{i=1}^{n} \frac{(e^{x_i^\top \beta})^{y_i}}{1+e^{x_i^\top \beta}} N(\boldsymbol{\beta}; 0, n(X^\top X)^{-1})
\end{aligned}
$$

This will be my target in the random walk MH.

```
data <- infert
y <- data$case
data$education <- as.numeric(as.factor(data$education))
edu1 <- data$education==2; edu2 <- data$education==3
x <- cbind(rep(1,n),data$age, data$parity, edu1, edu2, data$spontaneous, data$induced)
target <- function(y, x, xtx, beta, n){
  p <- exp(x%*%beta)/(1+exp(x%*%beta))
  return(sum(y*log(p)+(1-y)*log(1-p))-1/(2*n)*t(beta)%*%xtx%*%beta)
}
samples <- 5000
n <- nrow(x); ncolx <- ncol(x)
xtx <- t(x)%*%x
xtx_inv <- solve(xtx)
beta <- t(rmvnorm(1,rep(0,ncol(x)),xtx_inv*n))
beta.chain <- matrix(0,nrow=ncolx,ncol=samples)
beta.chain[,1] <- beta
for(s in 2:samples){
  beta_new <- t(rmvnorm(1,as.vector(beta), xtx_inv))
  logr <- target(y,x,xtx,beta_new,n)-target(y,x,xtx,beta,n)
  if(log(runif(1)) < logr){beta <- beta_new}
  beta.chain[,s] <- beta
}
```

```
##       (Intercept)    x[, -1]    x[, -1] x[, -1]edu1 x[, -1]edu2  x[, -1]
## Bayes   -1.175232 0.03751771 -0.8646845  -0.8969633   -1.304649 2.077206
## GLM     -1.149237 0.03958200 -0.8282774  -1.0442436   -1.403205 2.045905
##        x[, -1]
## Bayes 1.303842
## GLM   1.288757
```


beta.chain[1,]
2
−8
beta.chain[2,]
−0.05
beta.chain[3,]
−1.5
beta.chain[4,]
6
−2
beta.chain[5,]
6
−4
beta.chain[6,]
3.0
0.5
beta.chain[7,]
0.0
0
1000
3000
5000
Index

Part (7).

Describe and implement a Langevin-Hastings algorithm designed to obtain a MC with stationary distribution $p(\beta|Y)$

The Langevin-Hastings algorithm will work by utilizing the a 2^{nd} order Taylor approximation of the target distribution. That way I can try to match the proposal to the target.

If I was proposing for β at iteration s, I'd use something like

$$\boldsymbol{\beta}^* \sim N\Big(\boldsymbol{\beta}^{(s)} + \frac{1}{2}\sigma^2 \nabla \log\big(p(\boldsymbol{y}|\boldsymbol{\beta}^{(s)})p(\boldsymbol{\beta}^{(s)})\big), \sigma^2 \boldsymbol{I}\Big)$$

Earlier I showed the poserior is proportional to

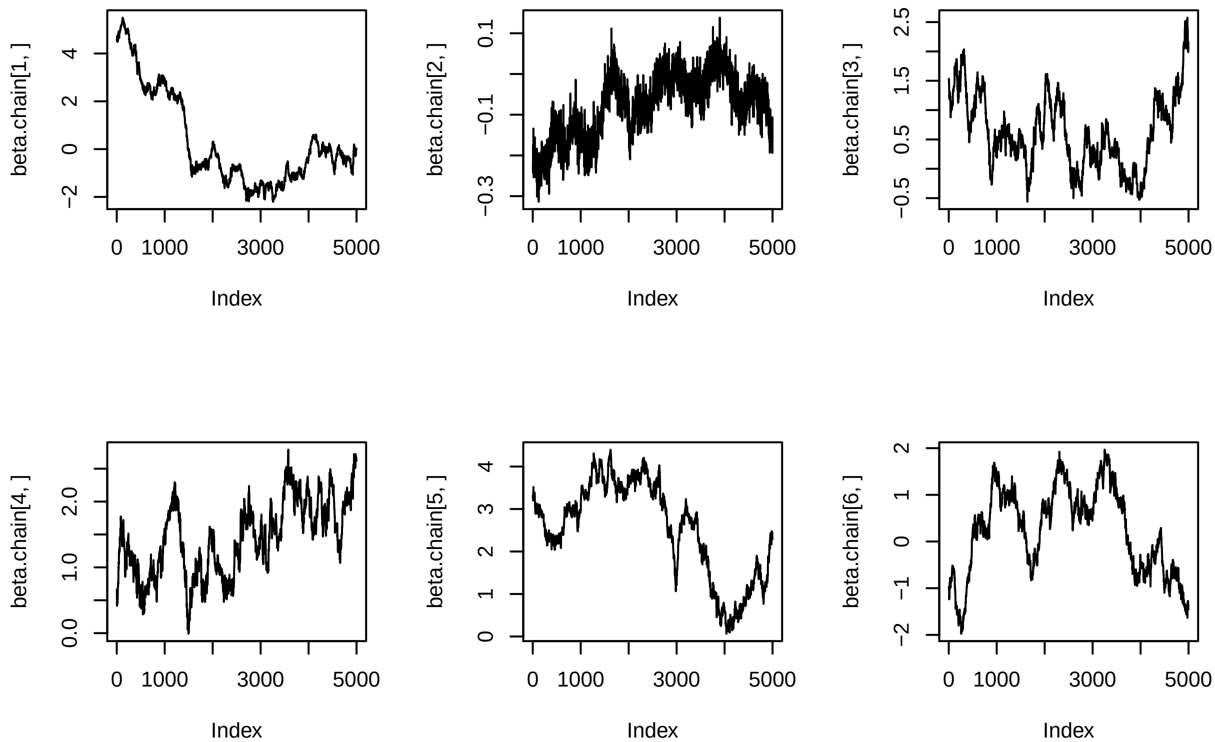
$$\prod_{i=1}^{n} \frac{(e^{x_i^\top \beta})^{y_i}}{1+e^{x_i^\top \beta}} N(\boldsymbol{\beta}; 0, n(X^\top X)^{-1})$$

so the log-posterior is

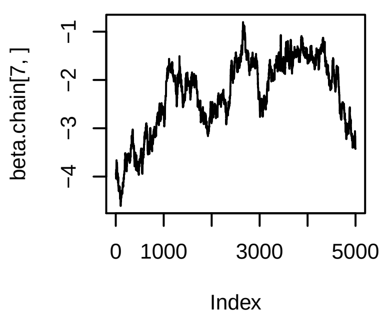
$$\begin{aligned}
&\sum_{i=1}^{n} y_i x_i^\top \boldsymbol{\beta} - \log(1+e^{x_i^\top \boldsymbol{\beta}}) - \frac{1}{2n}\beta^\top (X^\top X)\beta \\
\Rightarrow &\nabla\Big\{\sum_{i=1}^{n} y_i x_i^\top \boldsymbol{\beta} - \log(1+e^{x_i^\top \boldsymbol{\beta}}) - \frac{1}{2n}\boldsymbol{\beta}^\top (X^\top X)\beta\Big\} \\
= &\sum_{i=1}^{n} [y_i - \frac{e^{x_i^\top \boldsymbol{\beta}}}{1+e^{x_i^\top \boldsymbol{\beta}}}] x_i^\top - \frac{1}{n}\boldsymbol{\beta}^\top (X^\top X)
\end{aligned}$$

```
gradient <- function(y, x, xtx, beta, p){
  n <- length(y)
  return(t(y-p)%*%x-1/n*t(beta)%*%xtx)
}
target <- function(y, x, xtx, beta, n){
  return(sum(y*log(p)+(1-y)*log(1-p))-1/(2*n)*t(beta)%*%xtx%*%beta)
}
data <- infert
y <- data$case
data$education <- as.numeric(as.factor(data$education))
edu1 <- data$education==2; edu2 <- data$education==3
x <- cbind(rep(1,n),data$age, data$parity, edu1, edu2, data$spontaneous, data$induced)
samples <- 5000
n <- nrow(x); ncolx <- ncol(x)
xtx <- t(x)%*%x
xtx_inv <- solve(xtx)
beta <- t(rmvnorm(1,rep(0,ncolx),xtx_inv*n))
beta.chain <- matrix(0,nrow=ncolx,ncol=samples)
beta.chain[,1] <- beta
c <- 0.005
num_accepted <- accept_ratio <- 0
for(s in 2:samples){
  p <- exp(x%*%beta)/(1+exp(x%*%beta))
  grad <- gradient(y, x, xtx, beta, p)
  beta_new <- t(rmvnorm(1,as.vector(beta) + c^2*as.vector(grad), c*diag(ncolx)))
  logr <- target(y,x,xtx,beta_new,n)-target(y,x,xtx,beta,n)
  if(log(runif(1)) < logr){
```

```
    beta <- beta_new
    num_accepted <- num_accepted+1
  }
  accept_ratio <- num_accepted/s
  beta.chain[,s] <- beta
}
```




```
##        (Intercept)     x[, -1]    x[, -1] x[, -1]edu1 x[, -1]edu2   x[, -1]
## Bayes   0.3153487 -0.07770807  0.6285502   1.4391816   2.5141209 0.2328441
## GLM    -0.6413262  0.02056090 -0.4544283  -0.5705585  -0.7987104 1.1737802
##           x[, -1]
## Bayes -2.3075979
## GLM    0.7214959
```



These chains look crazy, but I'm not sure what's happening. I think the tuning parameter c is the culprit.

Part (7).

Describe and implement an adaptive Metropolis-Hastings algorithm designed to obtain a MC with stationary distribution $p(\beta|Y)$

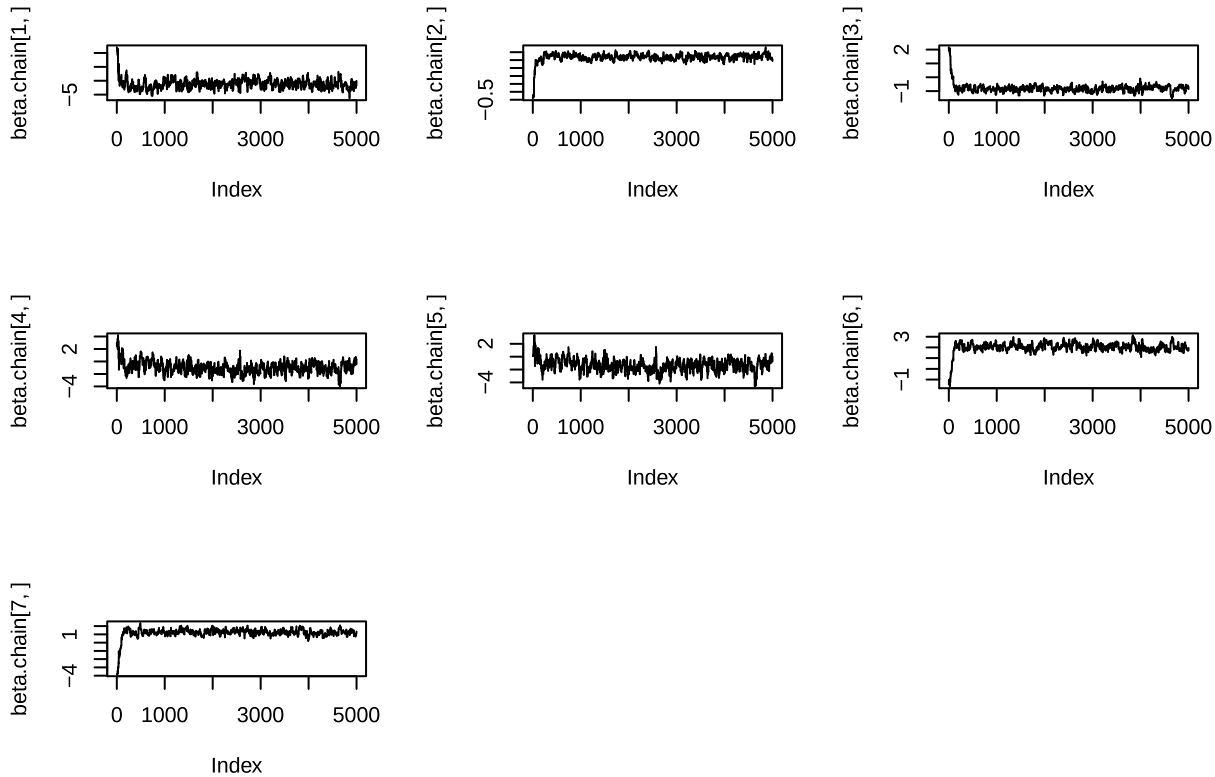
I think in this in this problem, I'll make proposals at step s such as

$$\beta^* \sim \beta^{(s)} + N(0, c\Sigma^{(s)})$$

and adaptively change c based on my acceptance ratio.

```
data <- infert
y <- data$case
data$education <- as.numeric(as.factor(data$education))
edu1 <- data$education==2; edu2 <- data$education==3
x <- cbind(rep(1,n),data$age, data$parity, edu1, edu2, data$spontaneous, data$induced)
target <- function(y, x, xtx, beta, n){
  p <- exp(x%*%beta)/(1+exp(x%*%beta))
  return(sum(y*log(p)+(1-y)*log(1-p))-1/(2*n)*t(beta)%*%xtx%*%beta)
}
samples <- 5000
n <- nrow(x); ncolx <- ncol(x)
xtx <- t(x)%*%x
xtx_inv <- solve(xtx)
beta <- t(rmvnorm(1,rep(0,ncol(x)),xtx_inv*n))
beta.chain <- matrix(0,nrow=ncolx,ncol=samples)
beta.chain[,1] <- beta
num_accepted <- accept_ratio <- 0
c <- 10
for(s in 2:samples){
  beta_new <- t(rmvnorm(1,as.vector(beta), c*xtx_inv))
  logr <- target(y,x,xtx,beta_new,n)-target(y,x,xtx,beta,n)
  if(log(runif(1)) < logr){
    beta <- beta_new
    num_accepted <- num_accepted+1
  }
  accept_ratio <- num_accepted/s
  if(accept_ratio > 0.4){
    c <- c+1
  }else{
    c <- c/2
  }
  beta.chain[,s] <- beta
}
```

```
##       (Intercept)    x[, -1]    x[, -1] x[, -1]edu1 x[, -1]edu2  x[, -1]
## Bayes   -1.137631 0.03744315 -0.7925801  -0.9782338   -1.341475 2.014576
## GLM     -1.149237 0.03958200 -0.8282774  -1.0442436   -1.403205 2.045905
##        x[, -1]
## Bayes 1.203220
## GLM   1.288757
```


beta.chain[1,]
−5
beta.chain[2,]
−0.5
beta.chain[3,]
2
−1
beta.chain[4,]
2
−4
beta.chain[5,]
2
−4
beta.chain[6,]
3
−1
beta.chain[7,]
1
−4
0 1000 3000 5000
Index