Biostat276 Project 2

Zhaodong Wu

2/16/2022

Contents

Bayesian Probit Regression

In `R` load the `package (survival)` and consider the analysis of the data-set `(infert)`. Ignoring dependence due to matching, consider a Bayesian analysis for a logistic regression model relating case status to: age, parity, education, spontaneous and induced. More precisely, assume case status y_i has density $y_i \sim_{ind} Bern(p_i)$, $p_i = \Phi(X_i'\beta)$, where $\Phi(\cdot)$ is the standard Gaussian cdf. Consider a prior $\beta \sim N(0, 10^2(X'X)^{-1})$. We are interested in $p(\beta|Y)$.

```
rm(list = ls())
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.1.0     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(mvtnorm)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
library(truncnorm)
```

```
# load the data
library(survival)
```

```r
data("infert")

# change the data type of `education`
inferthw <- infert
inferthw$education <- as.numeric(inferthw$education) - 1
dat <- inferthw %>%
  dplyr::select(c("education", "age", "parity", "induced", "case",
                  "spontaneous"))
str(dat) # all variables are numeric
```

```
## 'data.frame':    248 obs. of  6 variables:
##  $ education  : num  0 0 0 0 1 1 1 1 1 1 ...
##  $ age        : num  26 42 39 34 35 36 23 32 21 28 ...
##  $ parity     : num  6 1 6 4 3 4 1 2 1 2 ...
##  $ induced    : num  1 1 2 2 1 2 0 0 0 0 ...
##  $ case       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ spontaneous: num  2 0 0 0 1 1 0 0 1 0 ...
```

Frequentist Method (Not related to the questions, just a try)

Before answering the questions, we can directly compute the MLE of the model parameters using `glm()` function with a probit link function.

```r
freq <- glm(case ~ age + parity + education + spontaneous + induced,
            data = dat, family = binomial(link = probit))
```

```r
# MLE estimator
freq$coefficients
```

```
## (Intercept)         age      parity   education spontaneous     induced
## -0.91127035  0.02073028 -0.43967798 -0.28754507  1.16488031  0.71911970
```

```r
# Deviance
freq$deviance # the deviance is large
```

```
## [1] 259.6669
```

Question 1

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

The full posterior distribution for the Bayesian binary probit model can be computed as follows:

$$
\begin{aligned}
\pi(\beta|Y, X) &\propto \pi(\beta) \cdot \pi(Y, X|\beta) \\
&= \pi(\beta) \cdot \prod_{i=1}^{n} p(y_i, X_i|\beta) \\
&= \pi(\beta) \cdot \prod_{i=1}^{n} \Phi(X_i'\beta)^{y_i}[1 - \Phi(X_i'\beta)]^{1-y_i} \\
&\propto \exp[-\frac{\beta'(X'X)\beta}{200}] \cdot \prod_{i=1}^{n} \Phi(X_i'\beta)^{y_i}[1 - \Phi(X_i'\beta)]^{1-y_i}
\end{aligned}
$$

It's obvious that $\pi(\beta)$ is not a conjugate prior by the fact that no conjugate prior $\pi(\beta)$ exists for the parameters of the probit regression model.

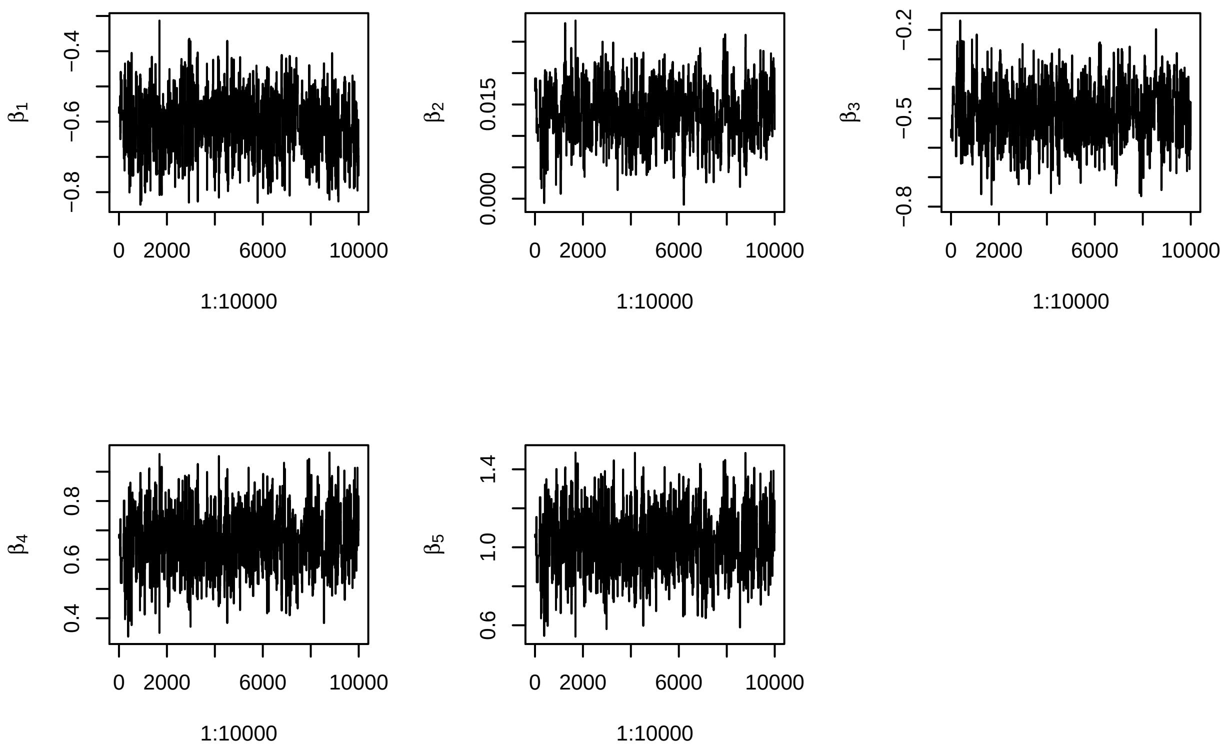
We compute the posterior distribution first:

```
# Calculation of posterior
beta_prior <- 100 * solve(as.matrix(t(dat[, -5])) %*% as.matrix(dat[, -5]))

posterior <- function(beta) {
  pi <- pnorm(as.matrix(dat[, -5]) %*% t(beta))
  data <- data.frame(pi = pi, y = dat[, 5])
  post <- apply(data, 1, function(x) {ifelse(x[2] == 1, x[1], 1 - x[1])})
  post <- log(post) %>%
    sum(.) - 1/2 * beta %*% solve(beta_prior) %*% t(beta)
}
```

```
# Adaptive Random Walk-Metropolis Hastings
mh.Q1 <- function(nsim = 10000, burn = 0.2, # chain parameters
                  delta = 0.75, c = 1,       # set c = 1 (c > 0)
                  seed = 1998) {
  # initialization----------------------------------
  set.seed(seed)
  nsim1 <- nsim * (1 + burn)
  burni <- nsim * burn

  beta_num <- dim(dat)[2] - 1

  beta <- matrix(data = rep(0, beta_num), nrow = 1)
  beta.ch <- matrix(data = NA, nrow = nsim, ncol = beta_num)
  betavar <- diag(beta_num) * (10^(-14)) # error term \epsilon
  deltaset <- rbinom(nsim1, 1, delta)
  # run chain---------------------------------------

  for (i in 1:nsim1) {
    # here we use adaptive MH for the first 2200 iterations
    if (i <= 2200) {
    # \Sigma^{\tilde}_t = \Sigma_t + \epsilon*I
    betavar <- (betavar * (i - 1) + t(beta) %*% beta)/i +
      diag(beta_num) * (10^(-14))
    delta_tm <- deltaset[i] # delta = 0.75 during adaptive
    } else {delta_tm <- 1} # delta is fixed after adaptive
    # use the last variance of beta from adaptive for the remaining iterations
    beta_tm1 <- rmvnorm(n = 1, mean = beta, sigma = c * betavar)
    beta_tm2 <- rmvnorm(n = 1, mean = beta, sigma = beta_prior)
    beta_tm <- beta_tm1 * delta_tm + beta_tm2 * (1 - delta_tm)

    P0 <- posterior(beta)
    P1 <- posterior(beta_tm)
    ratio <- P1 - P0
    if (log(runif(1)) < ratio) {
      beta <- beta_tm
    }
    # Store Chain after burn
    if (i > burni) {
      i1 <- i - burni
```

```r
      beta.ch[i1, ] <- beta
      }
    }
  return(list(beta = beta.ch))
}

# Simulation
arw_mh <- mh.Q1(nsim = 10000, burn = 0.2, delta = 0.75, c = 1)
```

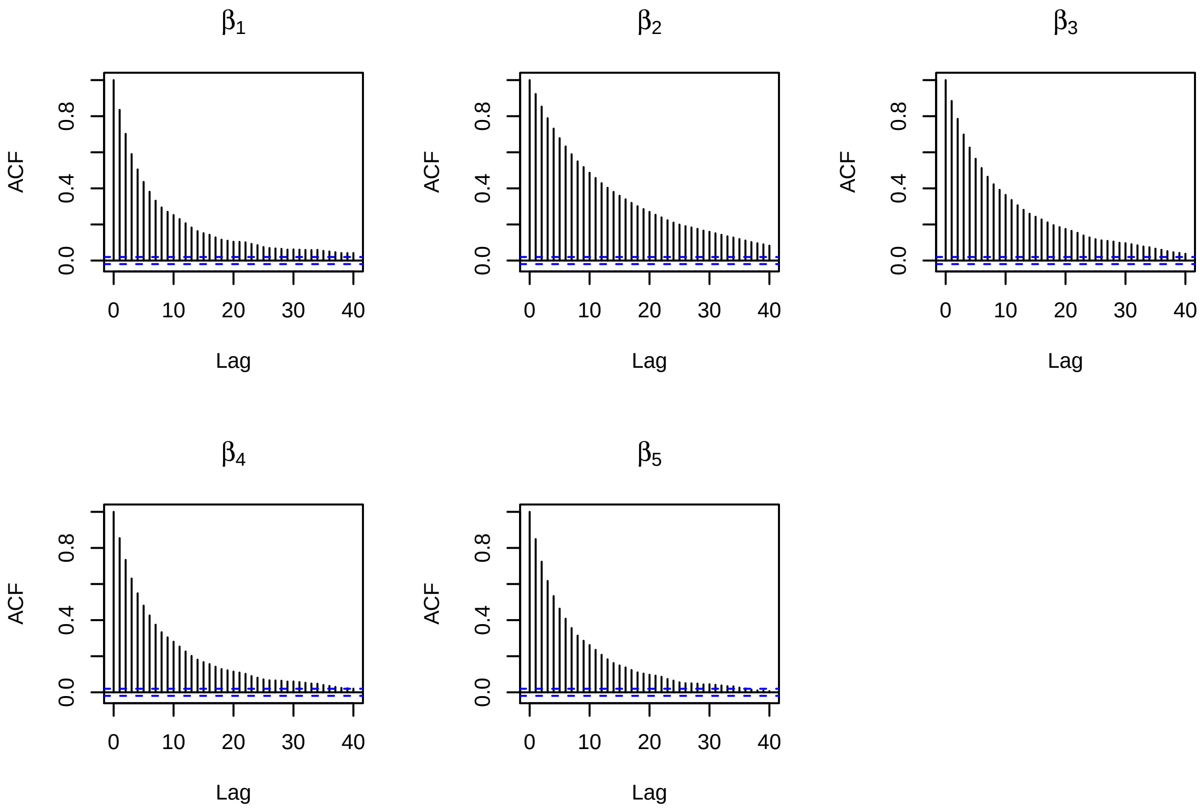
```r
# convergence checking
par(mfrow = c(2, 3))
for (i in 1:5) {
  plot(1:10000, arw_mh$beta[, i], type = "l",
       ylab = substitute(beta[x], list(x = i)))
}
```




```r
# Autocorrelation plot
par(mfrow = c(2, 3))
for (i in 1:5) {
  acf(arw_mh$beta[, i], main = substitute(beta[x], list(x = i)))
} # autocorrelation plots look good (beta_2 is kind of worse but acceptable)
```

Table 1: Summary Table of Coefficients Using Adaptive MH

	education	age	parity	induced	spontaneous
2.5%	-0.760	0.005	-0.662	0.480	0.751
50%	-0.613	0.014	-0.484	0.667	1.033
97.5%	-0.457	0.022	-0.319	0.850	1.319




```r
# get the result
result_Q1 <- apply(arw_mh$beta, 2,
                   function(x) {quantile(x, c(0.025, 0.5, 0.975))}) %>%
  round(., 3) # 95% credible interval & median
colnames(result_Q1) <- colnames(dat)[-5]

result_Q1 %>%
  kbl(caption = "Summary Table of Coefficients Using Adaptive MH") %>%
  kable_classic(full_width = F, html_font = "Cambria")
```

Question 2

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

Here targeting $p(\beta, z|y)$ could be easier enough.

Let prior of $\beta : \beta \sim \mathcal{N}(0, \Sigma_\beta)$, i.e. $\Sigma_\beta = 10^2(X'X)^{-1})$

Let $Z_i|\beta \sim \mathcal{N}(X_i'\beta, 1)$ and define the sampling model conditionally as $Y_i|Z_i = I(Z_i > 0)$. Then we use the Gibbs Sampling:

For β :

$$
\begin{aligned}
p(\beta|Z_{1:n}, Y_{1:n}) &= p(\beta|Z_{1:n}) \\
&\propto \prod_{i=1}^{n} \exp[-\frac{-(z_i - X_i'\beta)^2}{2}] \cdot \exp(-\frac{\beta^T \Sigma_\beta^{-1} \beta}{2}) \\
&= \exp[-\frac{(Z - X\beta)'(Z - X\beta) + \beta' \Sigma_\beta^{-1}}{\beta} 2] \\
&\propto \exp[\frac{\beta'(X'X + \Sigma_\beta^{-1})\beta - 2\beta' X' Z}{2}]
\end{aligned}
$$

By completing the square, we realize that the density is proportional to a normal kernel, the posterior of β satisfies a normal distribution:

$$p(\beta|Z_{1:n}, Y_{1:n}) \sim \mathcal{N}\{(X'X + \Sigma_\beta^{-1})^{-1} X' Z, (X'X + \Sigma_\beta^{-1})^{-1}\}$$

For Z_i :

$$
\begin{aligned}
p(Z_i|Y_i, \beta) &\propto p(Y_i|Z_i) \cdot p(Z_i|\beta) \\
&= I(Z_i > 0) \cdot \exp[-\frac{-(z_i - X_i'\beta)^2}{2}] \quad (\text{if } y_i = 1) \\
&= I(Z_i \leq 0) \cdot \exp[-\frac{-(z_i - X_i'\beta)^2}{2}] \quad (\text{if } y_i = 0)
\end{aligned}
$$

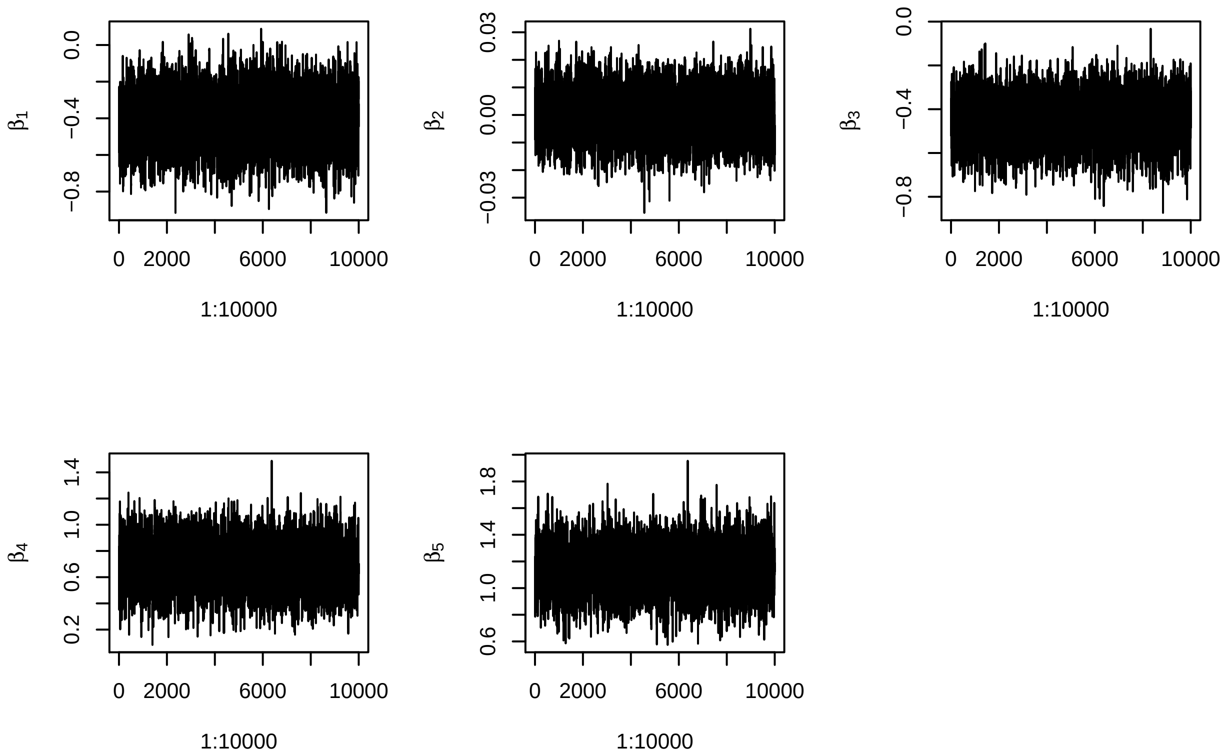
The posterior of Z_i follows a **truncated normal distributon**, i.e.

$$p(Z_i|Y_i, \beta) = \begin{cases} \mathcal{TN}(X_i'\beta, 1, 0, +\infty) & \text{if } y_i = 1 \\ \mathcal{TN}(X_i'\beta, 1, -\infty, 0) & \text{if } y_i = 0 \end{cases}$$

```
DA_Q2 <- function(nsim = 10000, burn = 0.2, # chain parameters
                  seed = 1998,
                  x = as.matrix(dat[-5]), # load the dataset
                  y = as.matrix(dat[5])) {
  # initialization----------------------------------
  set.seed(seed)
  nsim1 <- nsim * (1 + burn)
  burni <- nsim * burn

  beta_num <- dim(dat)[2] - 1

  beta <- matrix(data = rep(0, beta_num), nrow = 1)
  beta.ch <- matrix(data = NA, nrow = nsim, ncol = beta_num)

  # generate data z----------------------------------
  z <- rep(0, length(y))
  z.ch <- matrix(data = NA, nrow = nsim, ncol = length(y))

  range <- cbind(ifelse(y == 1, 0, -Inf),
                 ifelse(y == 1, Inf, 0)
                 )
```

```r
  # run chain--------------------------------------
  for (i in 1:nsim1) {
    # z
    z_comb <- cbind(x %*% t(beta),
                    1,
                    range)
    z <- apply(z_comb, 1, function(comb) {
      rtruncnorm(1, comb[1], sd = comb[2], a = comb[3], b = comb[4])
    })
    # beta
    beta_mean <- solve(t(x) %*% x + solve(beta_prior)) %*% t(x) %*% z
    beta_var <- solve(t(x) %*% x + solve(beta_prior))
    beta <- rmvnorm(1, mean = beta_mean, sigma = beta_var)

    if (i > burni) {
      i1 <- i - burni
      beta.ch[i1, ] <- beta
    }
  }
  return(list(beta = beta.ch))
}

# Simulation
DA_Gibbs <- DA_Q2(nsim = 10000, burn = 0.2)
```

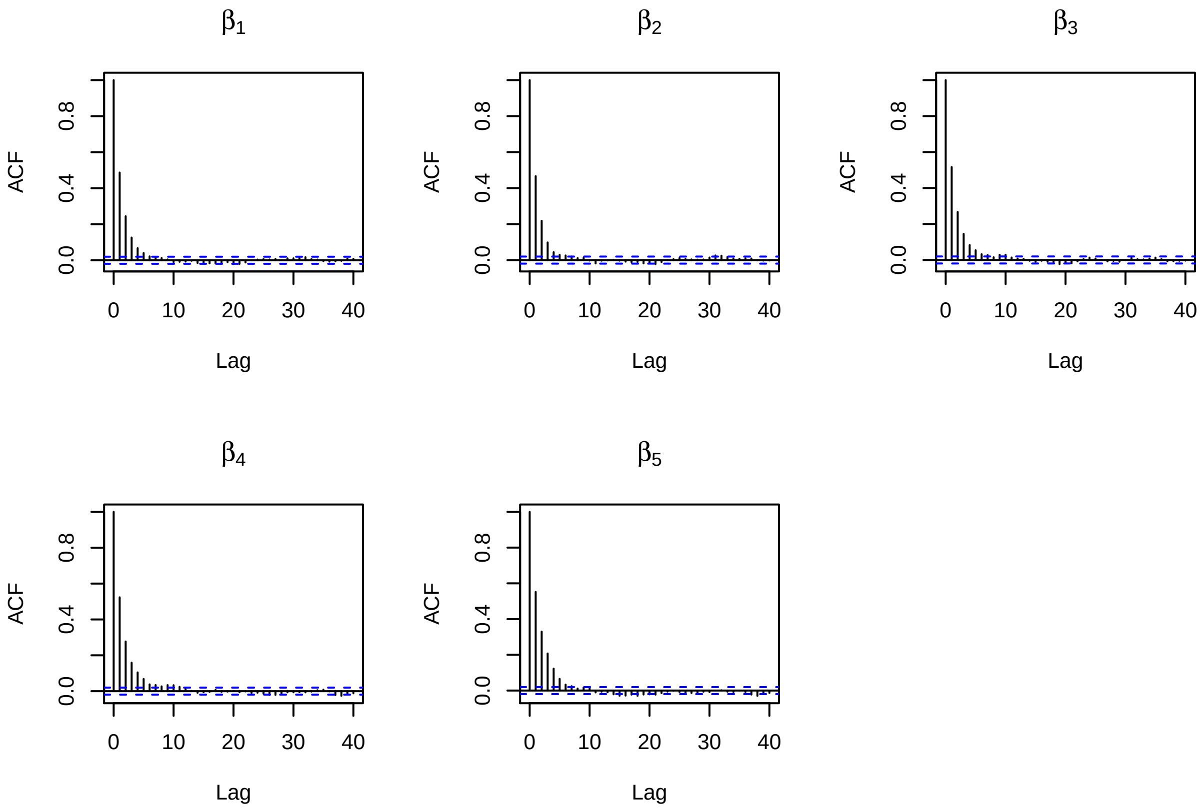
```r
par(mfrow = c(2, 3))
for (i in 1:5) {
  plot(1:10000, DA_Gibbs$beta[, i], type = "l",
       ylab = substitute(beta[x], list(x = i)))
}
```




```r
# Autocorrelation plot
par(mfrow = c(2, 3))
for (i in 1:5) {
  acf(DA_Gibbs$beta[, i], main = substitute(beta[x], list(x = i)))
} # autocorrelation plots look good (beta_2 is kind of worse but acceptable)
```

Table 2: Summary Table of Coefficients using Data Augmentation

	education	age	parity	induced	spontaneous
2.5%	-0.678	-0.016	-0.656	0.372	0.829
50%	-0.416	0.000	-0.456	0.695	1.141
97.5%	-0.148	0.016	-0.260	1.022	1.463




```r
# get the result
result_Q2 <- apply(DA_Gibbs$beta, 2,
                   function(x) {quantile(x, c(0.025, 0.5, 0.975))}) %>%
  round(., 3) # 95% credible interval & median
colnames(result_Q2) <- colnames(dat)[-5]

result_Q2 %>%
  kbl(caption = "Summary Table of Coefficients using Data Augmentation") %>%
  kable_classic(full_width = F, html_font = "Cambria")
```

Question 3

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