Probit regression for ordinal data

Setting:

- For each observations, we have 3 associated variables

X1: Continuous variable which we will generate from $N(2, 9)$ X2: Nomial variable with 3 possible outcomes {1,2,3} which we will generate from $Mult(1, [0.2, 0.3, 0.5])$ Y: Ordinal variable with 5 levels, {1,2,3,4,5} generated using the following process

$\epsilon_i \sim N(0, 1)$ $z_i = \beta^T X_i + \epsilon$ where $X = [X_1, X_2 == 1, X_2 == 2, X_2 == 3]$ $g(z_i) = Y_i$.

Here $\beta = [5, 1, 4, 7]$ and function g will be the binning function which will bin data into 5 different bins corresponding to 5 possible ordinal outcome.

- We will try to model Y conditioned on other variables (X1, and X2) using probit regression with latent variable Z with rank likelihood on parameter Z.
- We have two unknown parameters β and z_i which will be sampled from the full conditional posterior distribution using blocked gibbs sampling.

Summary on modelling process

$\epsilon_i \sim N(0, 1)$ $z_i = \beta^T X_i + \epsilon$ $z_i \in R(Y)$ where R(Y) = $\{z_i$: $z_i > z_j$ if $Y_i > Y_j$ and $z_i < z_j$ if $Y_i < Y_j\}$

```
# Data generating process
set.seed(0)
n = 300
beta = c(2, -5, 5, -3)
norm = sqrt(sum(beta*beta))
beta = beta/norm

# noise term
epsilon = rnorm(n, mean = 0, sd = 1)

# X0
X0 = rep(1, n)

# X1
X1 = rnorm(n, mean = 0, sd = 1)
#X1 = runif(n, 2, 10)
# X2
X2 = t(rmultinom(n, size = 1, prob = c(0.2, 0.3, 0.5)))
X2 = X2[, 2:3] # Use drop first category
# X
X = cbind(X0, X1, X2)
colnames(X) <- c('X0', 'X1', 'X2_cat2', 'X2_cat3')
# Z
Z = X%*%beta + epsilon

# Cut-off points and Y
g = quantile(Z, probs = c(0.2, 0.4, 0.6, 0.8))
Y = rep(NA, n)
Y[Z<g[1]] = 1
Y[Z>=g[1] & Z<g[2]] = 2
Y[Z>=g[2] & Z<g[3]] = 3
```

```
Y[Z>=g[3] & Z<g[4]] = 4
Y[Z>=g[4]] = 5
```

Prior specifications:

$\beta \sim multiN(0, n(X^T X)^{-1})$

Blocked Gibbs Sampling here consists of two major steps

1. Sample new β from its full conditional

$\beta \mid z, X, Y, z \in R(Y) \sim multiN(\frac{n}{n+1}(X^T X)^{-1} X^T z, \frac{n}{n+1}(X^T X)^{-1})$

2. for each i, using inverse cdf method, sample new z_i from its full conditional

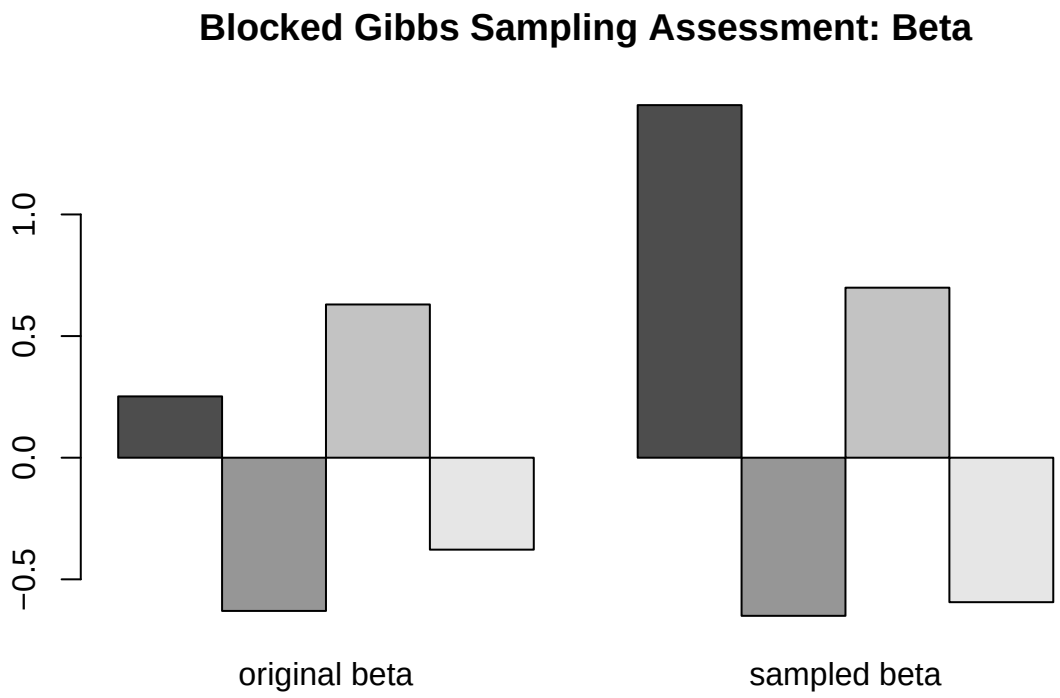
$z_i \mid \beta, X, Y, z_i \in R(Y) \sim N(\beta^T x_i, 1) * I\{z_i \in (a, b)\}$

where a = $\max(z_j \text{ for } Y_j < Y_i)$ b = $\min(z_j \text{ for } Y_j > Y_i)$

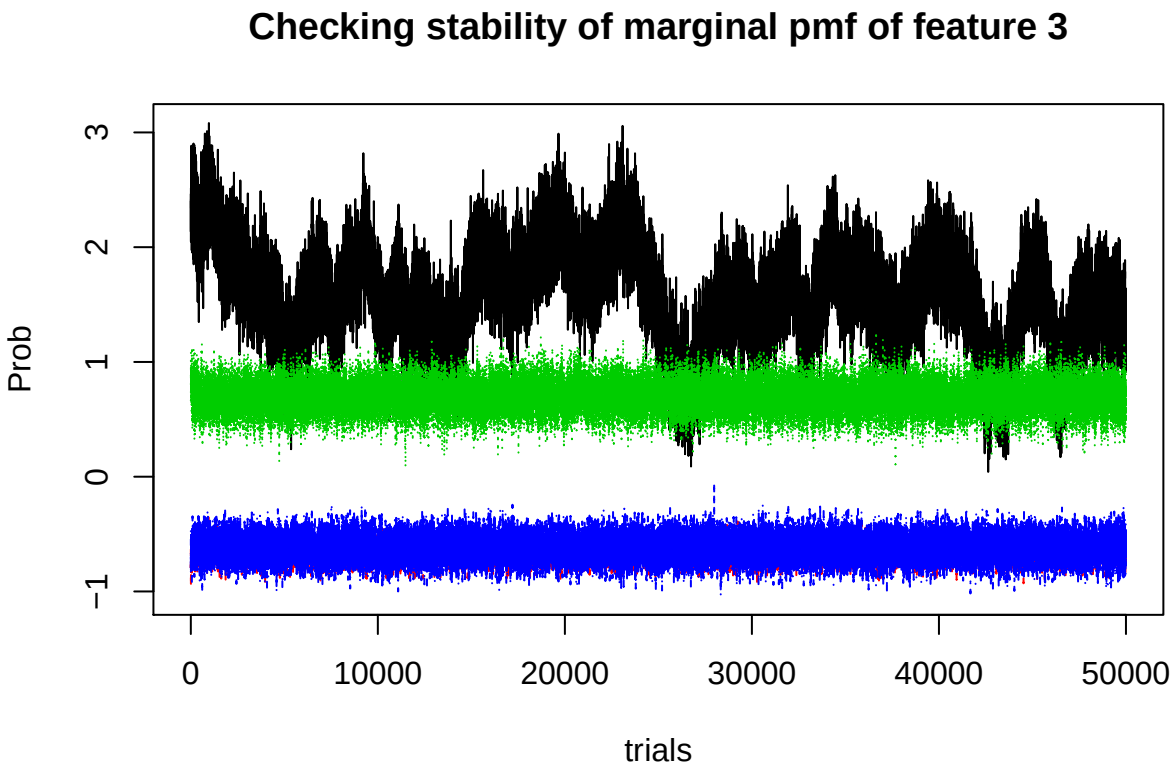
```
# Blocked Gibbs Sampling
set.seed(1)

# Prior Parameter
Z_i = Y - 0.5
variance_beta = (n/(n+1))*solve(t(X)%*%X)
mean_beta_hat = (n/(n+1))*solve(t(X)%*%X)%*%t(X)

# Initialize the sampling matrix
S = 50000
SAMPLED_Z = matrix(nrow=S,ncol=n)
BETA = matrix(nrow=S,ncol=4)

for (round in 1:S) {
  # Step 1: Sample Beta
  mean_beta = mean_beta_hat%*%Z_i
  beta_sampled = dae::rmvnorm(mean = mean_beta,
                              V = variance_beta, method = 'choleski')
  BETA[round,] <- beta_sampled

  # Step 2: Sample Z using inverse cdf appraoch
  for (i in 1:n) {
    # Get the lower and upper bound (a, b) of truncated normal
    a = max(-Inf, Z_i[Y<Y[i]], na.rm = TRUE)
    b = min(Z_i[Y>Y[i]],Inf, na.rm = TRUE)

    # Sample using inverse cdf
    ez = t(beta_sampled)%*%X[i,]
    u = runif(1, pnorm(a - ez), pnorm(b-ez))
    Z_i[i] = ez + qnorm(u)
  }
  SAMPLED_Z[round,] <-  Z_i
}
```

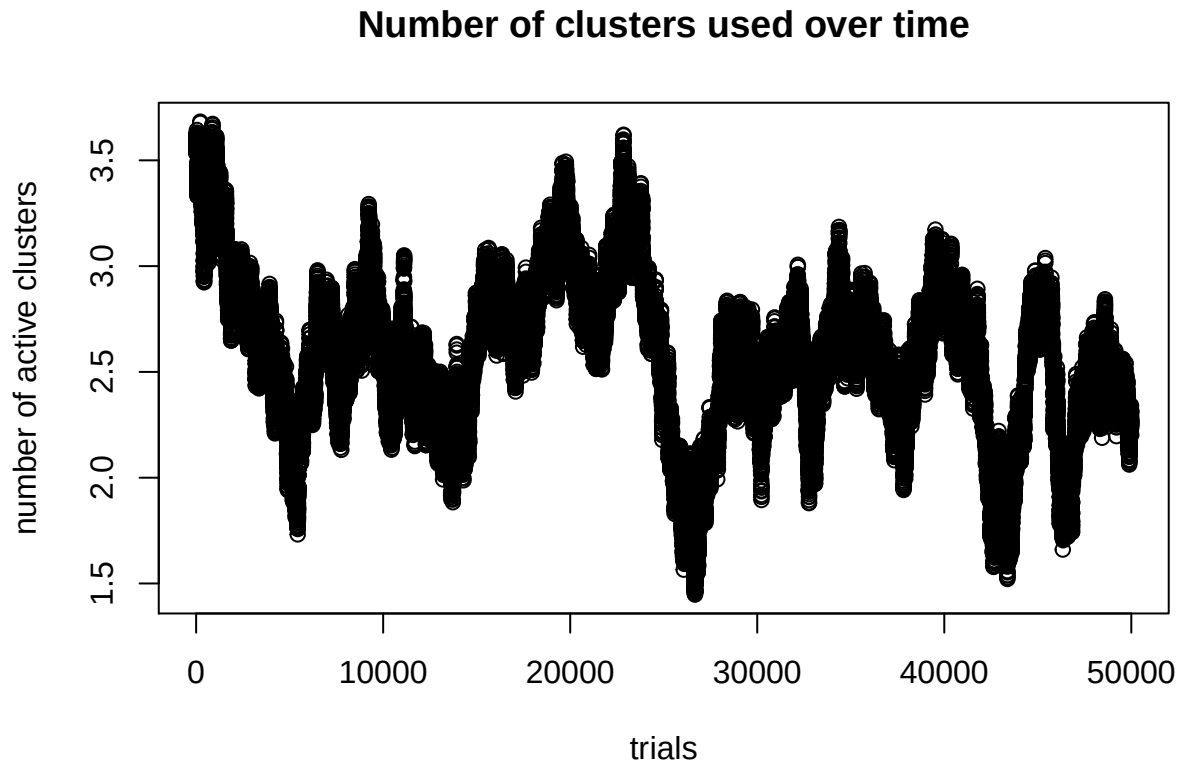
```
# Check posterior expectation of beta
sampling_beta = apply(BETA[seq(30000, dim(BETA)[1], 100),], MARGIN = 2, mean)
df_beta= cbind(beta, sampling_beta)
colnames(df_beta)<- c('original beta', 'sampled beta')
```

```r
barplot(df_beta, beside = TRUE,
        legend = TRUE, main = 'Blocked Gibbs Sampling Assessment: Beta')
```




```r
# Check cut off points
g1 = apply(SAMPLED_Z[, Y == 1], MARGIN = 1, max)
g2 = apply(SAMPLED_Z[, Y == 2], MARGIN = 1, max)
g3 = apply(SAMPLED_Z[, Y == 3], MARGIN = 1, max)
g4 = apply(SAMPLED_Z[, Y == 4], MARGIN = 1, max)
```


Checking stability of marginal pmf of feature 3
Prob
−1
0
1
2
3
0
10000
20000
30000
40000
50000
trials


Number of clusters used over time
number of active clusters
1.5
2.0
2.5
3.0
3.5
0
10000
20000
30000
40000
50000
trials