Chapter 10b

Samuel Eure

11/23/2018

Problem 10.5

The Data

```
data <- read.table("azdiabetes.dat", header = T);
data$diabetes <- as.numeric(data$diabetes)-1 #Making the values binary
Y <- data[, length(data[1,])];
X = data[,c(1,3,5,6,7)];
n <- dim(X)[1];
p <- dim(X)[2];
x.colmeans <- colSums(X)/n;
colMeansMatrix <- matrix(x.colmeans, nrow = n, ncol = p, byrow = T);
X <- X - colMeansMatrix;
stds <- sqrt(apply(X, 2, var));
stdMatrix <- matrix(stds, nrow = n, ncol = p, byrow = T)
X <- X/stdMatrix;
X <- data.matrix(X)
```

Part a

I will be using proposal distributions of

$$\beta_0^\star \sim N(\beta_0^{(s)}, 1/64)$$

$$\beta_j^\star \sim N(\beta_j^{(s)}, 1/64) \quad j > 0$$

$$\gamma_j^\star \sim \text{Bern}\left(\frac{1+2\gamma_j^{(s)}}{4}\right) \quad j > 0$$

###Thus, the proposals will be symmetric, allowing a lot of canceling in the Metropolis-Hastings Algorithm.

Functions I'll Use

```
getLogPY <- function(X,Y, betas, gammas){
  b.g <- betas[-1]*gammas;
  thetas   <- X%*%b.g + betas[1];
  e.thetas <- exp(thetas);
  probs    <- e.thetas/(1+e.thetas);
  probY    <- dbinom(Y, 1, probs);
  tot.log.probY <- sum(log(probY));
  return(tot.log.probY);
}
```

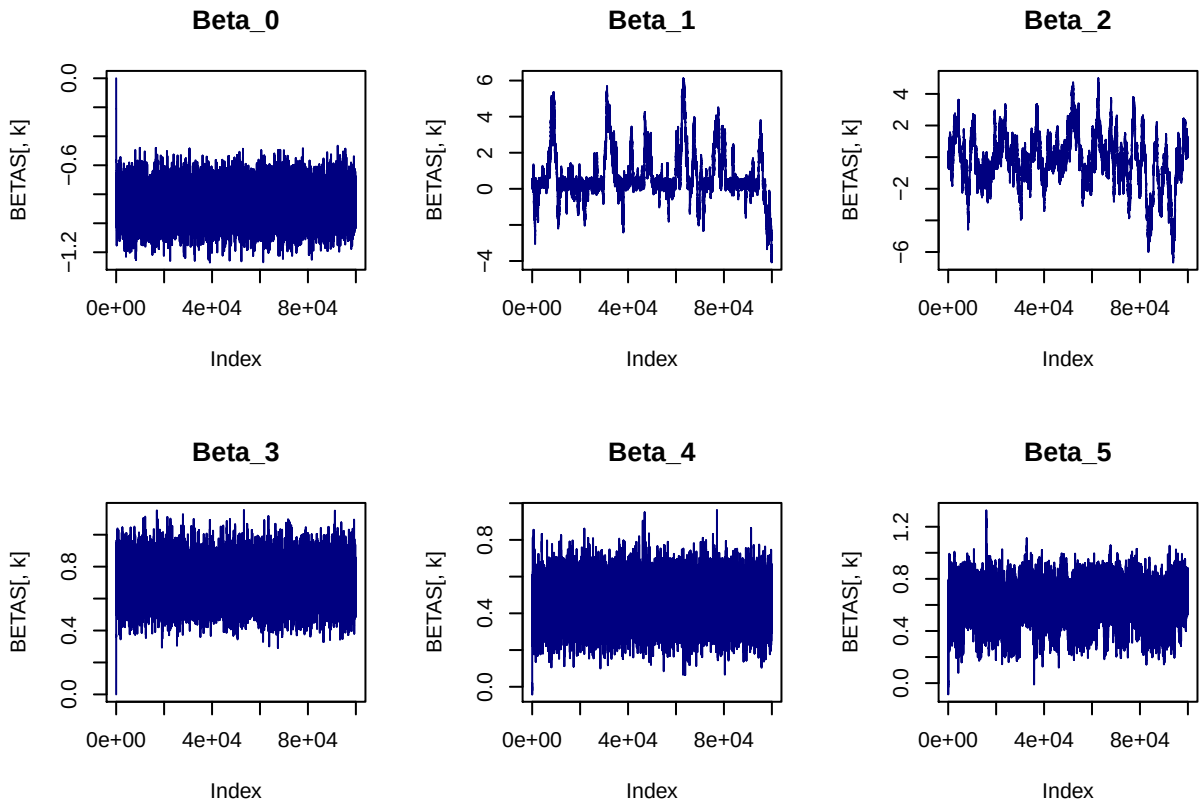
The Metropolis-Hastings Algorithm

```
set.seed(1);
N <- 100000;
BETAS     <- matrix(0, nrow = N, ncol = p+1);
B.MIXED   <- matrix(0, nrow = N, ncol = p+1);
beta.stds <- c(1/8, rep(1/8, p));
GAMMAS  <- matrix(0, nrow = N, ncol = p);
G.MIXED <- matrix(0, nrow = N, ncol = p);
betas   <- rep(0, p+1);
b.mixed <- rep(0, p+1);
gammas  <- rbinom(p,1,1/2);
g.mixed <- rbinom(p,1,1/2);
ARB  <- rep(0, p+1);
ARG  <- rep(0, p);
ARB.m <- rep(0, p+1);
for(i in 2:N){
  adjustStd  <- c(1,g.mixed*(9/10)+1/10);
  newb.mixed <- rnorm(p+1,b.mixed, beta.stds);
  newg.mixed <- rbinom(p,1,(1+2*g.mixed)/4);
  newBetas   <- rnorm(p+1,betas, beta.stds);
  newGammas  <- rbinom(p,1,(1+2*gammas)/4);
  #Betas
  for(j in 1:(p+1)){
    #book betas
    logNewProb  <- getLogPY(X,Y,newBetas,gammas)+log(dnorm(newBetas[j],0, sqrt(4*4^(j==1))));
    logPastProb <- getLogPY(X,Y,betas,gammas)+log(dnorm(betas[j],0, sqrt(4*4^(j==1))));
    #mixed betas
    newPriorStd <- 4*4**(j==1);
    priorStd    <- 4*4**(j==1);
    if(j>1){
      newPriorStd <- newPriorStd*(newg.mixed[j-1]*(9/10)+1/10); #multiply var by 1/10 if gamma = 0
      priorStd    <- priorStd*(g.mixed[j-1]*(9/10)+1/10); #multiply var by 1/10 if gamma = 0
    }
    logNewProb.mixed  <- getLogPY(X,Y,newb.mixed,rep(1,5))+log(dnorm(newb.mixed[j],0, sqrt(newPriorStd)
    logPastProb.mixed <- getLogPY(X,Y,b.mixed,rep(1,5))+log(dnorm(b.mixed[j],0, sqrt(priorStd)));
    #----------
    logR       <- logNewProb-logPastProb;
    logR.m     <- logNewProb.mixed-logPastProb.mixed;
    logU       <- log(runif(1, 0, 1));
    if(logU < logR){
      ARB[j] <- ARB[j]+1;
      betas[j] <- newBetas[j];
    }
    if(logU < logR.m){
      ARB.m[j] <- ARB.m[j]+1;
      b.mixed[j] <- newb.mixed[j];
    }
  }
  BETAS[i,]   <- betas;
  B.MIXED[i,] <- b.mixed;
  #Gammas
  for(j in 1:(p)){
```

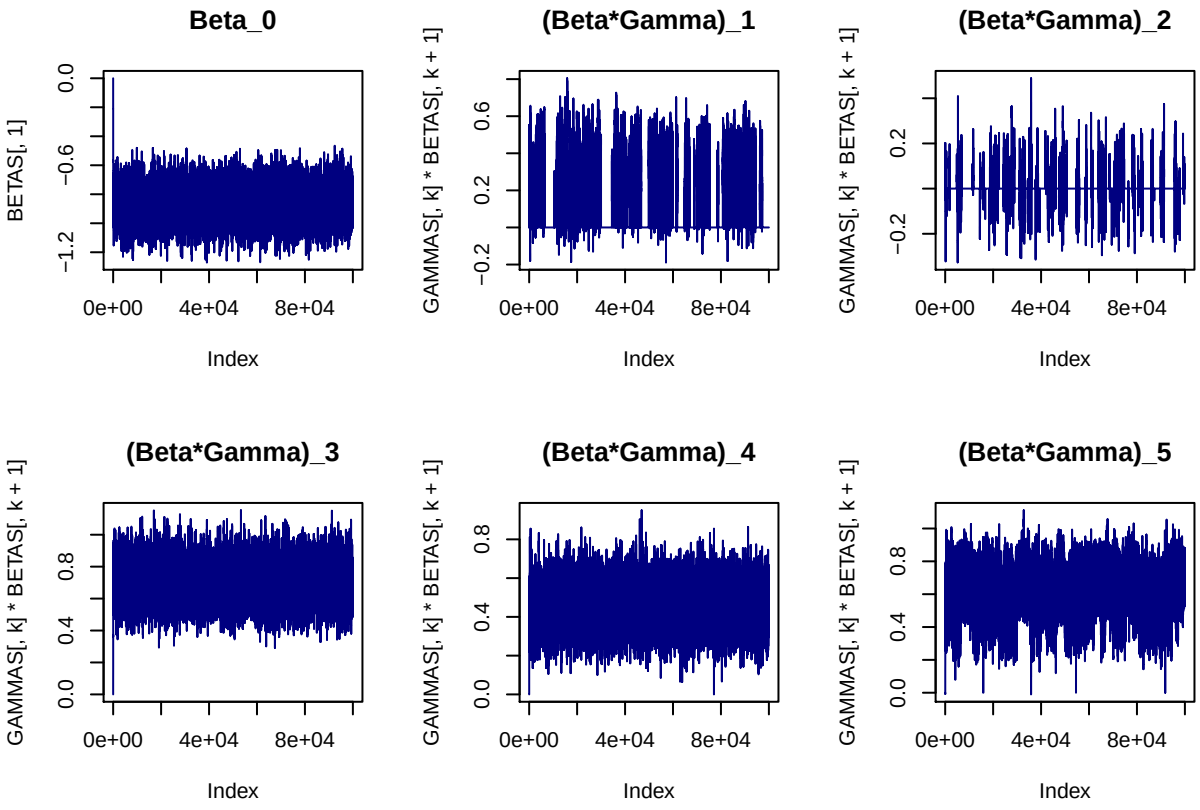
```
    logNewProb    <- getLogPY(X,Y,betas,newGammas); #since P(gamma = 1) = P(gamma=0)
    logPastProb   <- getLogPY(X,Y,betas,gammas);    #since P(gamma = 1) = P(gamma=0)
    logR     <- logNewProb-logPastProb;
    logU     <- log(runif(1, 0, 1));
    if(logU < logR){
      ARG[j] <- ARG[j]+1;
      gammas[j] <- newGammas[j];
    }
    #Since gamma ~ bern(1/2), and gamma plays no role in P(Y|X,B,G)...
    g.mixed[j] <- newg.mixed[j]; #  ... we always accept
  }
  GAMMAS[i,]  <- gammas;
  G.MIXED[i,] <- g.mixed;
}
ARG <- ARG/N;
ARB <- ARB/N;
ARB.m <- ARB.m/N;
```

Analysis for the book's method

```
columnNames <- c("Mean", "2.5%", "97.5%", "Acceptance Rate");
Output <- matrix("n/a", nrow = 3*p+1, ncol = length(columnNames));
colnames(Output) <- columnNames;
betaNames <- c();
par(mfrow = c(2,3));
for(k in 1:(p+1)){
  Output[k, 1]      <- round(mean(BETAS[,k]),5);
  Output[k, c(2,3)] <- round(quantile(BETAS[,k],c(.025, .975)),5);
  Output[k, 4]      <- ARB[k];
  betaNames <- c(betaNames, paste("Beta_",k-1, sep=""));
  plot(BETAS[,k], type = 'l', lwd = 1, main = paste("Beta_", k-1,sep=""), col = 'navy')
}
```




```
gammaOutput <- matrix(0, nrow = p, ncol = length(columnNames));
colnames(gammaOutput) <- columnNames;
gammaNames <- c();
par(mfrow = c(2,3));
plot(BETAS[,1], type = 'l', lwd = 1, main = paste("Beta_", 0,sep=""), col = 'navy')
for(k in 1:p){
  Output[k+p+1, 1]      <- round(mean(GAMMAS[,k]),5);
  Output[k+p+1, c(2,3)] <- round(quantile(GAMMAS[,k],c(.025, .975)),5);
  Output[k+p+1, 4]      <- ARG[k];
  gammaNames <- c(gammaNames, paste("Gamma_",k, sep=""))
  plot(GAMMAS[,k]*BETAS[,k+1],type='l', main = paste("(Beta*Gamma)_", k,sep=""), col="navy")
}
```




```
bgn <- c();
BG <- BETAS[,-1]*GAMMAS
for(i in 1:p){
  bgn[i] <- paste("BetaGamma_",i,sep="");
  Output[(2*p+1+i),1]      <- round(mean(BG[,i]),5);
  Output[(2*p+1+i),c(2,3)] <- round(quantile(BG[,i], c(0.025, 0.975)),5);
}
rownames(Output) <- c(betaNames,gammaNames, bgn);
Output
```

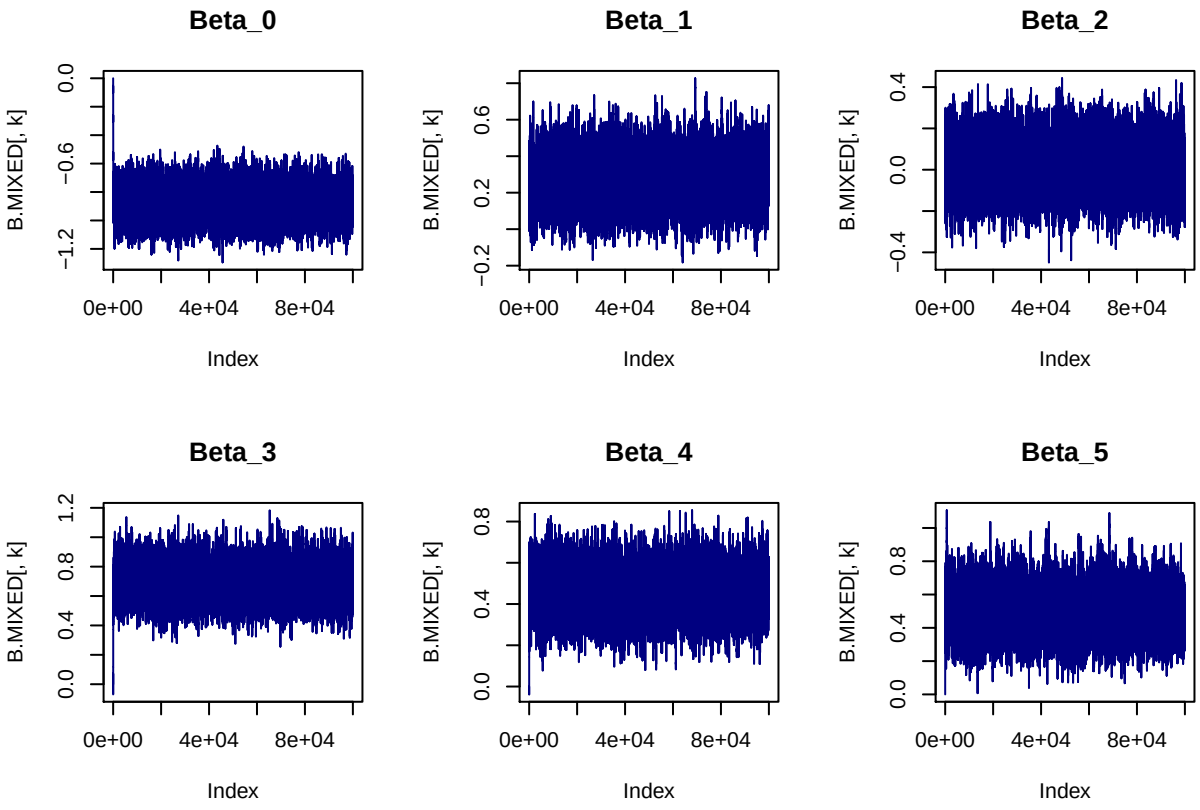
```
##             Mean       2.5%       97.5%      Acceptance Rate
## Beta_0      "-0.86387" "-1.07043" "-0.66137" "0.26179"
## Beta_1      "0.74208"  "-1.62491" "4.34682"  "0.26378"
## Beta_2      "-0.21381" "-4.10618" "2.91929"  "0.26526"
## Beta_3      "0.70197"  "0.49396"  "0.91437"  "0.26523"
## Beta_4      "0.4551"   "0.26508"  "0.65006"  "0.27038"
## Beta_5      "0.60377"  "0.32587"  "0.84659"  "0.27856"
## Gamma_1     "0.486"    "0"        "1"        "0.25706"
## Gamma_2     "0.07011"  "0"        "1"        "0.25986"
## Gamma_3     "0.99999"  "1"        "1"        "0.26129"
## Gamma_4     "0.99981"  "1"        "1"        "0.26102"
## Gamma_5     "0.99859"  "1"        "1"        "0.2613"
## BetaGamma_1 "0.13766"  "0"        "0.48603"  "n/a"
## BetaGamma_2 "0.00036"  "-0.03517" "0.04001"  "n/a"
## BetaGamma_3 "0.70197"  "0.49396"  "0.91437"  "n/a"
## BetaGamma_4 "0.45502"  "0.26498"  "0.65006"  "n/a"
## BetaGamma_5 "0.60272"  "0.32309"  "0.84558"  "n/a"
```

Comments on the mixing of the β chain

Referencing the plots titled $Beta_0, ..., Beta_5$, it appears to me that $\beta_0, \beta_3, \beta_4$, and β_5 have clearly reached stationary distributions given that the path which the beta values take over time seem to remain quite constant (i.e. the average beta value seems to be constant over different intervals of samples); additionally, since the initial beta values for each of these betas was zero, and their expected value over time is quite different from zero (i.e. a value of zero would be an outlier), I would argue that these parameters were able to leave a region of low probability and find a more probable region of higher probability, indicating good mixing. In addition to this, I would argue that my samples for β_1 are also coming from a stationary distribution given the martingale like nature of path which β_1 is taking, and I would make a similar argument for β_2, however the path which β_2 is taking seems a bit furhter from uniform. The movement of these two beta values look like those of Wiener processes, which makes sense if their gamma values are turned off for the majority of time. Nevertheless, the expected value of these two beta values looks like it is around zero – maybe a bit higher for β_1. To justify the seemingly poor mixing of β_1 and β_2, I would like to reference the second set of plots which titles "(Beta*Gamma)_i", which depict the values of $\beta_i\gamma_i$ over time (or as sampling progresses). Firstly, all the beta values which had clearly reached a stationary distribution have plots of"(Beta*Gamma)_i" which are very similar to their corresponding "Beta_i" plots. Next, the "(Beta*Gamma)_1" and "(Beta*Gamma)_2" plots show that the values for β_1 and β_2 experience periods of being continuously turned off (i.e. $\gamma_1 = 0$ or $\gamma_2 = 0$), and brief periods of being turned on (i.e. $\gamma_1 = 1$ or $\gamma_2 = 1$). This supports the idea that these gamma values should be set to zero in our final model since they seem to be getting "stuck" at values of zero. Moreover, since occassionally the values are set back to 1, this indicates that my sampling scheme is mixing well since I am exploring areas of low probability occassionally (i.e. $\gamma_1 = 0$ and $\gamma_2 = 0$) and areas of high probability often (i.e. $\gamma_1 = 1$ and $\gamma_2 = 1$). β_1 and β_2 seem to be experiencing poor mixing is due to the fact that their γ values are usually set to zero, reducing the penalty of exploring low probability regions for these beta values.

Analysis for the George and McCullagh 1993 method

```
Output <- matrix("n/a", nrow = 2*p+1, ncol = length(columnNames));
colnames(Output) <- columnNames;
par(mfrow = c(2,3));
for(k in 1:(p+1)){
  Output[k, 1]      <- round(mean(B.MIXED[,k]),5);
  Output[k, c(2,3)] <- round(quantile(B.MIXED[,k],c(.025, .975)),5);
  Output[k, 4]      <- ARB.m[k];
  plot(B.MIXED[,k], type = 'l', lwd = 1, main = paste("Beta_", k-1,sep=""), col = 'navy')
}
```




```
gammaOutput <- matrix(0, nrow = p, ncol = length(columnNames));
colnames(gammaOutput) <- columnNames;
for(k in 1:p){
  Output[k+p+1, 1]       <- round(mean(G.MIXED[,k]),5);
  Output[k+p+1, c(2,3)] <- round(quantile(G.MIXED[,k],c(.025, .975)),5);
  Output[k+p+1, 4]       <- 1;
}
rownames(Output) <- c(betaNames,gammaNames);
Output
```

```
##          Mean       2.5%       97.5%      Acceptance Rate
## Beta_0  "-0.86788" "-1.0799"  "-0.65974" "0.19221"
## Beta_1  "0.28074"  "0.04205"  "0.51982"  "0.19403"
## Beta_2  "0.01682"  "-0.19958" "0.23194"  "0.19821"
## Beta_3  "0.69865"  "0.48398"  "0.91783"  "0.19801"
## Beta_4  "0.45742"  "0.26215"  "0.65181"  "0.20228"
## Beta_5  "0.50191"  "0.25811"  "0.74706"  "0.20661"
## Gamma_1 "0.50124"  "0"        "1"        "1"
## Gamma_2 "0.50206"  "0"        "1"        "1"
## Gamma_3 "0.50266"  "0"        "1"        "1"
## Gamma_4 "0.50375"  "0"        "1"        "1"
## Gamma_5 "0.49668"  "0"        "1"        "1"
```

I will start by commenting on the γ values. Since we used a bernoulli(1/2) prior for each γ_i, the ratio $\frac{p(\gamma_{new})}{p(\gamma_{old})} = \frac{1/2}{1/2} = 1$. Moreover, since $P(Y \mid X, \beta, \gamma) = P(Y \mid X, \beta)$ in this sampling model and the proposals are symmetric, the value of r (i.e. the probability of accepting a proposed change to any particular γ_i) will always be one, and thus the posterior sampling distribution of γ should be discrete uniform. This is confirmed by the emperical data shown in the table above which shows that each γ_i has an expected value close to .50. Moving onto the β values, the traceplots for $\beta_0, \beta_3, \beta_4$ and β_5 look roughly the same as those obtained from the sampling method proposed in the book, and my interpretation of these traceplots is the same as that provided in the paragraph above. The traceplots for β_1 and β_2 however look quite different from those obtained following the book's method. Here, these traceplots look similar to those of the other beta values, while those obtained from the book's method look more like Wiener processes. Here, the β_1 value seems to have an expected value close to 0.3, whereas it was centered around a value near 0 or 0.1 before. This implies that under this sampling scheme, we predict x_1 to be somewhat indicative of the value of Y. On the contrary, β_2 still seems to be centered around zero, further suggesting that this x_2 does not provide a lot of information in terms of predicting Y.

Part b

First, I'll find the most popular choices of γ

Book's Method

```
countMap = c();
countMap["00000"] = 0;
for(i in 1:N){
  a = paste(GAMMAS[i,], collapse = "");
  if(is.na(countMap[a])){
    countMap[a] = 1;
  }
  else{
    countMap[a] = countMap[a]+1;
  }
}
countMap = sort(countMap, decreasing = T)
CO <- matrix(0, nrow = 5, ncol = 3);
colnames(CO) <- c("Gamma", "count", "Probability")
CO[,1] <- names(countMap[1:5])
CO[,2] <- countMap[1:5]
CO[,3] <- round(countMap[1:5]/sum(countMap),5)
print(CO)
```

```
##      Gamma   count   Probability
## [1,] "00111" "48212" "0.48212"
## [2,] "10111" "44627" "0.44627"
## [3,] "11111" "3843"  "0.03843"
## [4,] "01111" "3165"  "0.03165"
## [5,] "10110" "130"   "0.0013"
```

Here, Gamma = "00111" $\Rightarrow \gamma_1 = \gamma_2 = 0, \gamma_3 = \gamma_4 = \gamma_5 = 1$. As always, $\gamma_0 = 1$, if you would like to think of γ_0 as existing in the process, even though it never changes. In light of belief that my MCMC chain seems to be mixing well, I feel quite confident that these posterior probabilities obtain from the MCMC reflect the true posterior probabilities of each γ vector.

George and McCullagh 1993 Method

```
countMap = c();
countMap["00000"] = 0;
for(i in 1:N){
  a = paste(G.MIXED[i,], collapse = "");
  if(is.na(countMap[a])){
    countMap[a] = 1;
  }
  else{
    countMap[a] = countMap[a]+1;
  }
}
countMap = sort(countMap, decreasing = T)
CO <- matrix(0, nrow = 5, ncol = 3);
colnames(CO) <- c("Gamma", "count", "Probability")
CO[,1] <- names(countMap[1:5])
CO[,2] <- countMap[1:5]
CO[,3] <- round(countMap[1:5]/sum(countMap),5)
print(CO)
```

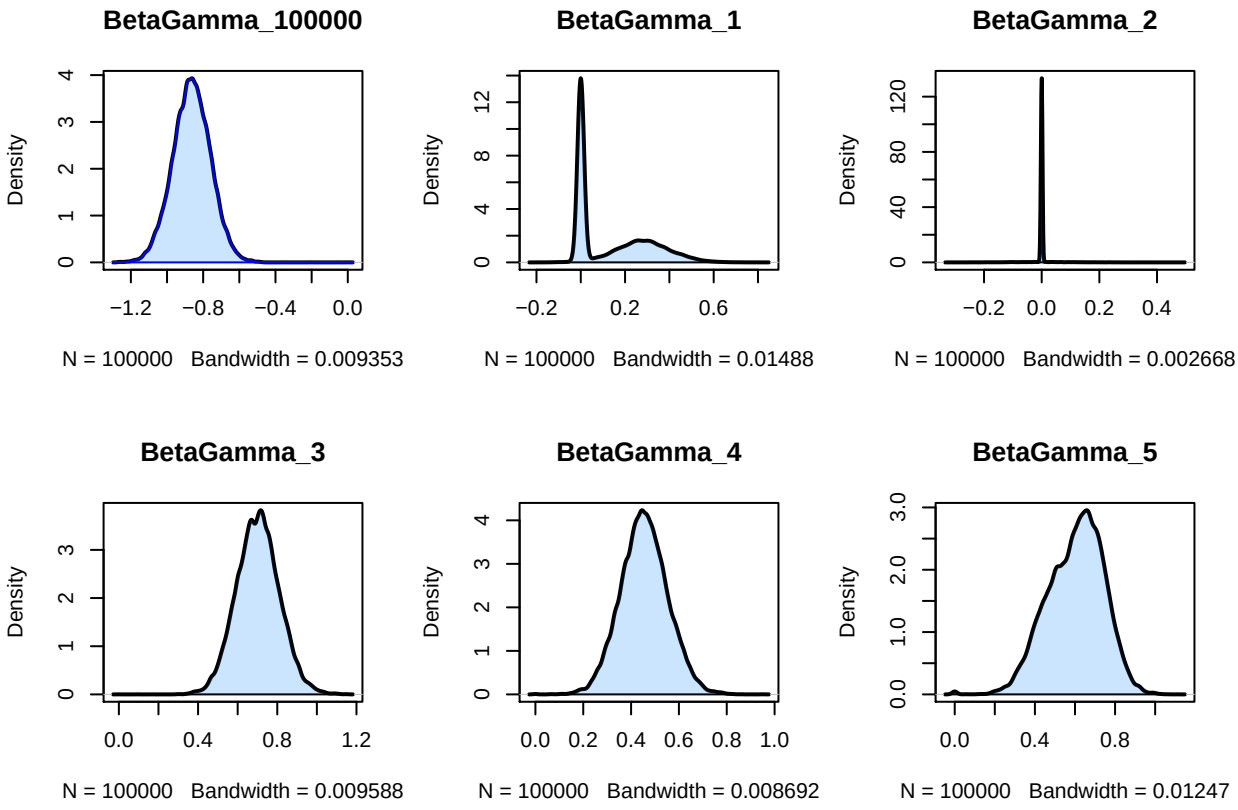
```
##      Gamma   count  Probability
## [1,] "01110" "3262" "0.03262"
## [2,] "11010" "3249" "0.03249"
## [3,] "11110" "3241" "0.03241"
## [4,] "00110" "3220" "0.0322"
## [5,] "10010" "3213" "0.03213"
```

In comparison to the Book's method, the probabilities presented here look quite different. This makes sense since we always accept proposed changes to Gamma, and thus the distribution of γ should be uniform.

Part c

Densities from the Book's Method

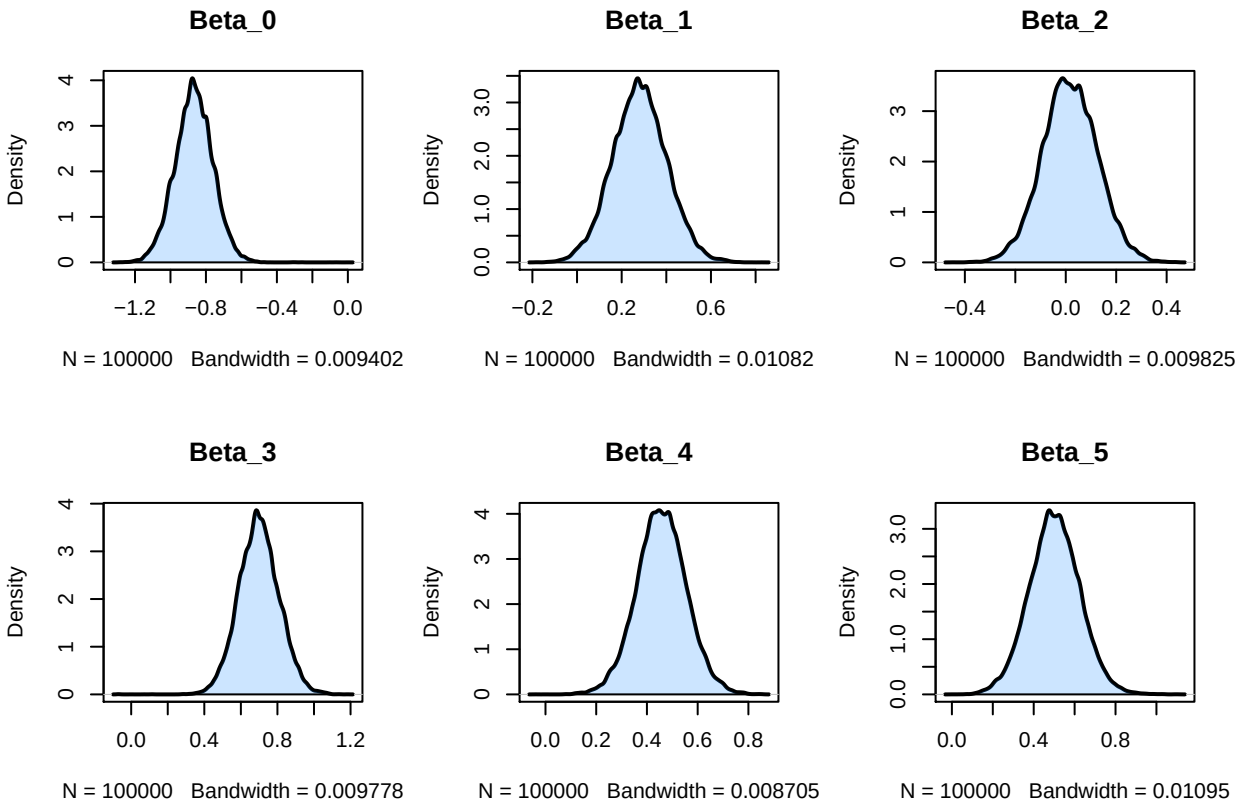
```
par(mfrow = c(2,3))
plot(density(BETAS[,1]), lwd = 2, main = paste("BetaGamma_",i, sep=""))
polygon(density(BETAS[,1]), col=rgb(0,.5,1,.2), border="blue")
for(i in 1:p){
  plot(density(BG[,i]), lwd = 2, main = paste("BetaGamma_",i, sep=""))
  polygon(density(BG[,i]), col=rgb(0,.5,1,.2), border="black")
}
```



###It looks like $\beta_1\gamma_1$ takes a value of around 0.3 when $\gamma_1 = 1$.

Densities from the George and McCullagh 1993 Method

```
par(mfrow = c(2,3))
for(i in 1:(p+1)){
  plot(density(B.MIXED[,i]), lwd = 2, main = paste("Beta_",i-1, sep=""))
  polygon(density(B.MIXED[,i]), col=rgb(0,.5,1,.2), border="black")
}
```



Now that the gamma values do not weaken the influence that beta values have on the conditional probability of Y, it appears that a more natural value for β_1 is roughly normal and centered around 0.3.

The posterior means for $\beta_j\gamma_j$ are in the tables in part a, as well at the $Pr(\gamma_j = 1|x, y)$ values which are simply the mean values of γ_i for each i.