STA250 Homework 1 Report

Shuo Li

October 27, 2013

1 Question 1

1. Prove that the stationary distribution of the two-component Gibbs sampler is the desired target distribution $p(x_1, x_2)$.

   $\int \pi(x)p(\overrightarrow{x}, \overrightarrow{y})dx$
   $= \iint \pi(x)p(y_1|x_2)p(y_2|y_1)dx_1dx_2$
   $= \iint p(x_1, x_2)dx_1p(y_1|x_2)p(y_2|y_1)dx_2$
   $= \int p(x_2)p(y_1|x_2)p(y_2|y_1)dx_2$
   $= p(y_2|y_1) * \int p(y_1, x_2)$
   $= p(y_1) * p(y_2|y_1)$
   $= p(y_1, y_2)$

   Since $p(y_1, y_2)$ satisfies the conditions for a stationary distribution, which is $\pi(\overrightarrow{y}) = \int \pi(\overrightarrow{x})p(\overrightarrow{x}, \overrightarrow{y})$. And the stationary distribution is unique for a ergodic Markov Chain, so the stationary distribution of the two-component Gibbs sampler is the desired target distribution $p(x_1, x_2)$.

2. Prove that the stationary distribution of the p-component Gibbs sampler is $p(x_1, x_2, ..., x_p)$.

   $\int \pi(x)p(\overrightarrow{x}, \overrightarrow{y})dx$
   $= \int \cdots \int \pi(x)p(y_1|x_2, ..., x_p) * p(y_2|y_1, x_3, ..., x_p) * ... * p(y_p|y_1, ..., y_{p-1})dx_1dx_2...dx_p$
   $= \int \cdots \int p(x_1, ..., x_p)dx_1 * p(y_1|x_2, ..., x_p) * p(y_2|y_1, x_3, ..., x_p) * ... * p(y_p|y_1, ..., y_{p-1})dx_2...dx_p$
   $= \int \cdots \int p(x_2, ..., x_p) * p(y_1|x_2, ..., x_p)dx_2 * p(y_2|y_1, x_3, ..., x_p) * ... * p(y_p|y_1, ..., y_{p-1})dx_3...dx_p$

$= \int \cdots \int p(y_1, x_3, ..., x_p) * p(y_2|y_1, x_3, ..., x_p)dx_3 * p(y_3|y_1, y_2, x_4, ..., x_p) *$
$... * p(y_p|y_1, ..., y_{p-1})dx_4...dx_p$
$= ...$
$= p(y_1, y_2, ..., y_{p-1}) * p(y_p|y_1, ..., y_{p-1})$
$= p(y_1, y_2, ..., y_p)$

Since $p(x_1, x_2, ..., x_p)$ satisfies the conditions for a stationary distribution, which is $\pi(\overrightarrow{y}) = \int \pi(\overrightarrow{x})p(\overrightarrow{x}, \overrightarrow{y})$. And the stationary distribution is unique for a ergodic Markov Chain, so the stationary distribution of the p-component Gibbs sampler is the desired target distribution $p(x_1, x_2, ..., x_p)$.

2 Question 2

1. The analytic form of the posterior distribution, up to proportionality.

   $(y|\beta_0, \beta_1) \sim Bin(m_i, logit^{-1}({x_i}^T\beta))$
   $P(y|\beta_0, \beta_1) \propto \prod_{i=1}^n C_{m_i}^{y_i}(\frac{exp({x_i}^T\beta)}{1+exp({x_i}^T\beta)})^{y_i}(\frac{1}{1+exp({x_i}^T\beta)})^{m_i-y_i}$

   $(\beta_0, \beta_1) \sim N(\mu_0, \Sigma_0)$
   $P(\beta_0, \beta_1) \propto exp(-\frac{1}{2}(\beta-\mu_0)^T{\Sigma_0}^{-1}(\beta-\mu_0))$

   $P(\beta_0, \beta_1|y) \propto P(\beta_0, \beta_1) * P(y|\beta_0, \beta_1)/P(y)$
   $\propto P(\beta_0, \beta_1) * P(y|\beta_0, \beta_1)$

   $P(\beta_0, \beta_1|y) \propto$
   $\prod_{i=1}^n(\frac{exp({x_i}^T\beta)}{1+exp({x_i}^T\beta)})^{y_i}(\frac{1}{1+exp({x_i}^T\beta)})^{m_i-y_i} * exp(-\frac{1}{2}(\beta-\mu_0)^T{\Sigma_0}^{-1}(\beta-\mu_0))$

   $log(P(\beta_0, \beta_1|y)) \propto -\frac{1}{2}(\beta-\mu_0)^T{\Sigma_0}^{-1}(\beta-\mu_0)+\sum_{i=1}^n y_i{x_i}^T\beta-\sum_{i=1}^n m_i log(1+$
   $exp({x_i}^T\beta))$

2. A description of my MCMC algorithm.
   (1) The proposal distribution I used is a bivariate normal distribution.
   $\beta^{(t)} \sim N(\begin{pmatrix} {\beta_0}^{(t-1)} \\ {\beta_1}^{(t-1)} \end{pmatrix}, \begin{pmatrix} v^2 & 0 \\ 0 & v^2 \end{pmatrix})$
   (2) My tunning process is to adjust v^2 in the proposal distribution of β

and calculate the accept rate every 200 retuning period. If the accept rate is too low, which is smaller than 0.3, the v^2 will be adjusted to $v^2 * exp(-0.5)$. If the accept rate is larger than 0.6, the v^2 will be adjusted to $v^2 * exp(0.5)$. If the accept rate is between 0.3 and 0.6, v^2 will be kept the same and the retuning process will end.
(3) The default iterations I run my samplers for is 10000. But in fact, the convergence speed is pretty fast and I think several thousand will be enough.
(4) The default burnin period is set to be 1000. However, based on the trace plot, about 200 to 500 will be enough.
(5) I choose the start value for β_0 and β_1 to be 0 and 0.

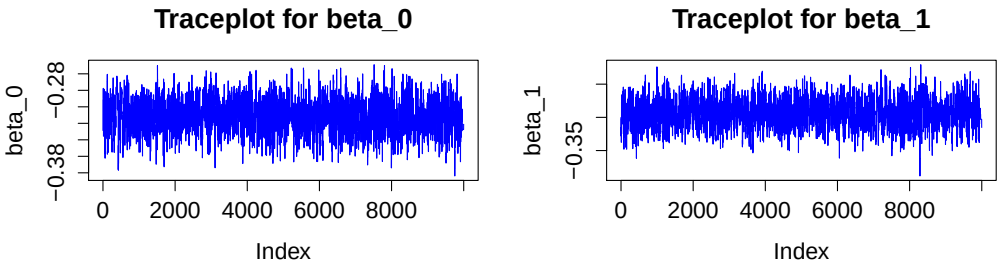

Figure 1:

Figure 1 shows that the markov chain I obtained converges successfully. The β's both fluctuate within a small rage. The effective sample sizes for them are 1388.387 and 1234.167 respectively. Since both of them are larger than 1000, the subsequent analyses based on these samples are reliable.

3. The numerical coverage properties.
   After running my program on Gauss, the result obtained is in next page as Table 1

4. The graph of the coverage properties. (Page 5, Figure 2)

   Since my coverage lines keep fluctuating inside the guiding bands, my

	β_0	β_1
p_01	0.01	0.01
p_05	0.07	0.03
p_10	0.10	0.07
p_25	0.24	0.20
p_50	0.52	0.48
p_75	0.76	0.75
p_90	0.90	0.87
p_95	0.94	0.95
p_99	0.99	0.99

Table 1:

Bayesian Logistic Regression MCMC algorithm is proved to be successful.

3 Question 3

1. Summary of the results of my model.
   In this question, I use the same method with question 2. But the proposal distribution was adjusted to $\beta^{(t)} \sim (\beta^{(t-1)}, v^2 * \Sigma^*)$, where Σ^* is the asymptotic covariance matrix of the MLE of β, which is computed from the logistic regression based on the unscaled cancer data. And we adjust the same v^2 for the whole covariance matrix.

   The retuning process is also the same with above questions, which is to adjust v^2 in the proposal distribution of β and calculate the accept rate every 200 retuning period. In this question, the optimal range of the acceptance rate is from 0.35 to 0.55 since this problem has a higher dimension than the previous one. If the accept rate is too low, which is smaller than 0.35, the v^2 will be adjusted to $v^2 * exp(-0.5)$. If the accept rate is larger than 0.55, the v^2 will be adjusted to $v^2 * exp(0.5)$. If the accept rate is between 0.35 and 0.55, v^2 will be kept the same and the retuning process will end.

   Then I calculate the effective sample size. For each β, it varies from 1.1% to about 8% of the iterations. To obtain enough effective samples,

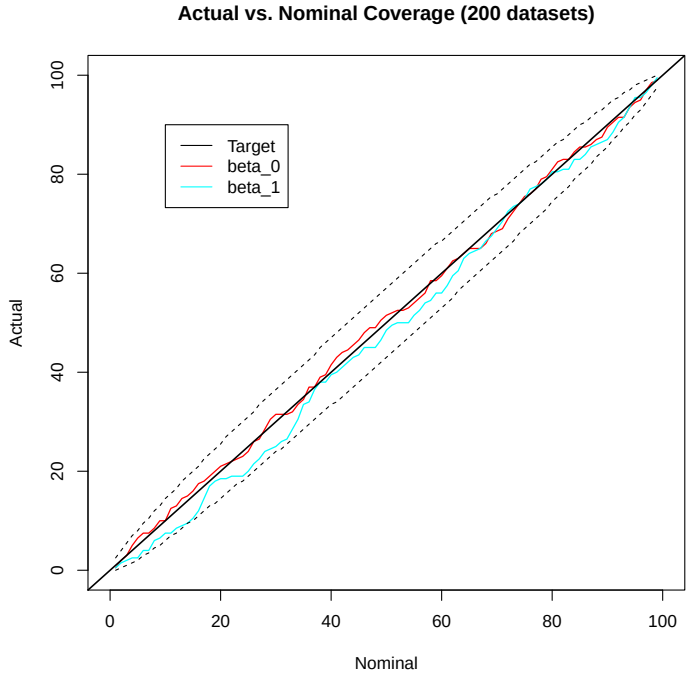

Figure 2:

the iteration is set to be $50,000$ and the burnin period is made to be 1000, since the converge speed of the algorithm is fast.

For the starting point of β's, I choose the estimated value of them from the logistic regression based on the unscaled orignial data.

After the procedure and $50,000$ iterations, the effective sample size for β's are:

	β_0	β_1	β_2	β_3	β_4	β_5
Effective Sample Size	834.6874	800.4857	1069.0173	1376.7414	726.3770	4267.8720
	β_6	β_7	β_8	β_9	β_{10}	
Effective Sample Size	842.2839	804.9720	1285.6891	740.4458	719.6054	

Table 2:

The effective sample size for each β are all above 1000. And from the trace plots for each β it seems that all of them can converge, since they fluctuate within small rages.

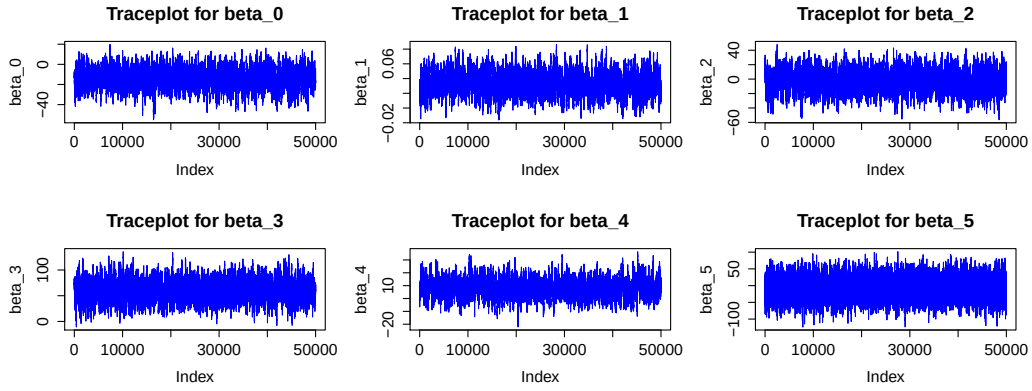

Figure 3:

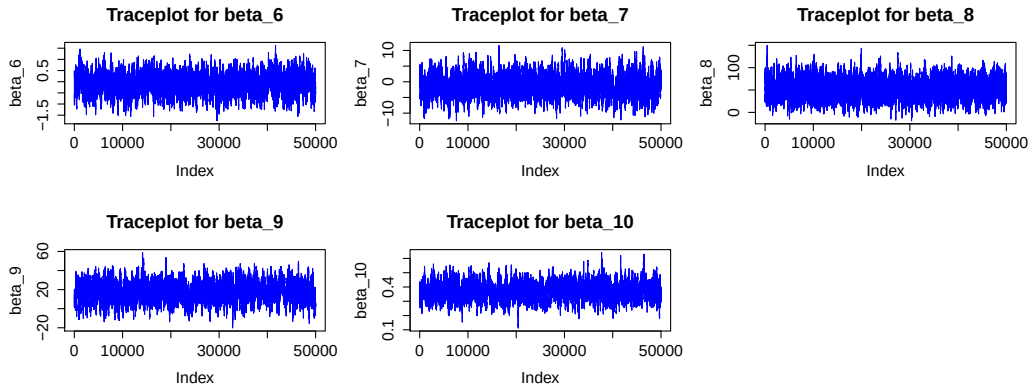

Figure 4:

2. The lag-1 autocorrelation for each component of β
   Since most of the β's are above 0.95, the simulation of β's are not quite effective. And it seems that obtaining more iterations, like $50,000$, is

	β_0	β_1	β_2	β_3	β_4	β_5
lag-1 autocorrelation	0.9592385	0.9684845	0.9534001	0.9454763	0.9667286	0.8379006
	β_6	β_7	β_8	β_9	β_{10}	
lag-1 autocorrelation	0.9668661	0.9683107	0.9430485	0.9708138	0.9704654	

Table 3:

necessary.

3. From the covariance matrix, we can see that between variables and response, the correlation coefficient:
$corr(diagnosis, area) = 0.7090$,
$corr(diagnosis, concavepts) = 0.7766$,
$corr(diagnosis, perimeter) = 0.7426$,
$corr(diagnosis, radius) = 0.7300$,
which are relatively large. So we may assume that covariates "concavepts", "perimeter" and "radius" seems to be related to cancer diagnosis.

   Further, we can construct a 95% Credible Interval for the β's from the posterior distribution and see if 0 is included in the interval. If not, the covariate will be considered to be related to cancer diagnosis. The table of the the credible interval from the simulated β's is in next page, as Table 4.

   From the table we can see that 0 is not included in the 95% quantile of β_1, β_3, β_8 and β_{10}. It seems that covariate "area", "concavepts", "smoothness" and "texture" is important in cancer diagnosis. The first two covariate also meets the results of correlation between response and covariates, since the correlation coefficient for them are significantly large.

   Finally, we can use the model selection technique to find which covariate is in the best fitting model. The covariate in the best fitting model may be regarded to be highly related to cancer diagnosis. Here, I use the BIC criterion to do the model selection on logistic regression.

	$2.5\%quantile$	$97.5\%quantile$
β_0	-33.6679	4.4107
β_1	0.0042	0.0631
β_2	-30.9482	24.6918
β_3	22.2164	98.2521
β_4	-5.6080	22.3637
β_5	-67.2191	48.7185
β_6	-0.9850	0.7576
β_7	-7.6263	5.5553
β_8	14.1278	94.6253
β_9	-2.0066	36.8186
β_{10}	0.2583	0.4934

Table 4:

Results show that the best model contains 3 covariate, containing "area", "concavepts" and "texture", with BIC=-3423. For the top 5 best model with the smallest BIC value, all the models do not contain "compactness ", "concavity", "fracdim" and "symmetry", which indicates that they are not quite useful in cancer diagnosis. This meets the results of Correlation Coefficient & Credible Interval methods above.

In summary, from all three methods, we may say that covariates, including "area", "concavepts", "perimeter", "radius", "smoothness" and "texture" are crucial in diagnosing breast cancer.

4. Perform a posterior predictive check.
The sample of $\beta^{(t)}$'s have been obtained from my posterior simulation function and t is from 1 to $50,000$. Then, for $\beta^{(t)}$ that we sampled each time, we can sample a new data set from $p(y|\beta^{(t)})$, which is the Bernoulli distribution. Thus, we will get $50,000$ predictive datasets. For every datasets, there are 569 simulated observations. There are $50,000$ predictive datasets and one real datasets. Here, I choose three kinds of statistics to compare. They are mean of response, median of response and the 1/0 ratio of response. The simulated responses are ploted using histogram. Meanwhile, the statistics of the real data is showed as the vertical red line. The plot is as follows:

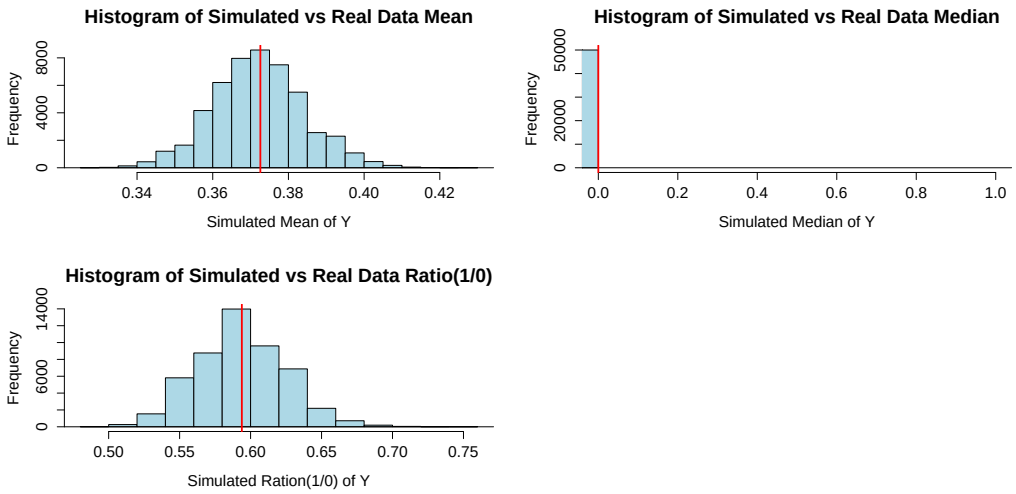

Figure 5:

5. Discussion about the reasonableness of my model.
   From the posterior predictive check, it seems that my model is a good fit. Since the red line is at the center of all three histograms. So, the model here successfully characterizes all three statistics: mean, median and 1/0 ratio of response pretty well. Thus, the algorithm is suited here to simulate the parameters.

6. Discussion about an alternative algorithm.
   I also write the code of an alternative algorithm for this question. In which, the proposal distribution is set to be:
   $\beta^{(t)} \sim (\beta^{(t-1)}, diag({v_1}^2, {v_2}^2, ..., {v_p}^2))$.
   In each time point of the algorithm, we sample one β_i from the proposal distribution, fix the other β's and calculate α to see if this new β_i is accepted. If so, we update β_i, otherwise, we do not update this β_i. Then we continue to sample the β_{i+1} from the proposal distribution until we reach β_p. As a result, in each step, we will try to update all $p\beta$'s and record if each $p\beta$'s is accepted or not. After every 200 tuning period, we will get p accept rate for β_1 to β_p. Then we will adjust ${v_i}^2$

respectively. The tuning method is the same as the above questions. Here, the burnin period is set to be 1000 and iteration is set to be 10000.

However for this algorithm, the results are not quite promising. The trace plot for some β's clearly show some patterns. The convergence is not quite well even for the standardized data. The effective sample size for each β's are:

	β_0	β_1	β_2	β_3	β_4	β_5
Effective Sample Size	32.3469	16.9877	109.3358	270.8587	268.3432	166.2677

	β_6	β_7	β_8	β_9	β_{10}
Effective Sample Size	4.6550	4.6947	282.6907	1067.7583	746.2035

Table 5:

The table shows that for some β's the effective sample size is significantly small, even less than 10, such as β_6 and β_7.

Since the algorithm does not work well on this data, no more detailed results will be stated here. (Code of this algorithm is also attached at the end.)

4 Appendix Code

```
##Problem 2

# Logistic regression
#
# Y_{i} | \beta \sim \textrm{Bin}\left(n_{i}, e^{x_{i}^{T}\beta}/
(1+e^{x_{i}^{T}\beta})\right)
# \beta \sim N\left(\beta_{0},\Sigma_{0}\right)
#
##
library(MASS)
library(mvtnorm)
library(coda)
```

```
library(lattice)
#################################################################
#################################################################
## Handle batch job arguments:

# 1-indexed version is used now.
args <- commandArgs(TRUE)

cat(paste0("Command-line arguments:\n"))
print(args)

####
# sim_start ==>
Lowest simulation number to be analyzed by this particular batch job
###

#########################
sim_start <- 1000
length.datasets <- 200
#########################

if (length(args)==0){
  sinkit <- FALSE
  sim_num <- sim_start + 1
  set.seed(1330931)
} else {
  # Sink output to file?
  sinkit <- TRUE
  # Decide on the job number, usually start at 1000:
  sim_num <- sim_start + as.numeric(args[1])
  # Set a different random seed for every job number!!!
  set.seed(762*sim_num + 1330931)
}

# Simulation datasets numbered 1001-1200

#################################################################
#################################################################
```

```
"bayes1"<- function(m, y, X, beta.star, beta.0, Sigma.0.inv)

{
  p= ncol(X)
  l=nrow(X)
  pi=-0.5*matrix(beta.star-beta.0, 1, p)%*%
    Sigma.0.inv%*%as.matrix(beta.star-beta.0,p,1)+
    matrix(y, 1, l)%*%X%*%matrix(beta.star,p,1)-
    matrix(m, 1, l)%*%log(1+exp(X%*%matrix(beta.star,p,1)))
  return(pi)
}

"bayes.logreg" <- function(m, y, X, beta.0=c(0, 0),
                   Sigma.0.inv=diag(1, 2), niter=10000, burnin=1000,
                   retune=200, verbose=TRUE)
{
  beta_ci=matrix(numeric(2*99), 99, 2)
  v_2=1
  Sigma_new=v_2*diag(1, 2)
  beta=matrix(numeric(2*(burnin+niter+1)), (burnin+niter+1), 2)
  beta[1,]= beta.0
  p= ncol(X)
  U=runif((burnin+niter), 0, 1)
  n=0
  while (verbose)
  {
    for (i in 2: (retune+1))
    {

      beta[i, ]=mvrnorm(1, mu=beta[i-1, ], Sigma=Sigma_new)
      alpha=bayes1(m, y, X, beta[i, ], beta.0, Sigma.0.inv)-
        bayes1(m, y, X, beta[i-1, ], beta.0, Sigma.0.inv)
      if (log(U[i-1])<alpha)
      { beta[i, ]=beta[i, ]
        n=n+1
      }
      else
```

```
    { beta[i, ]=beta[i-1, ] }
  }

  accept_rate=n/retune
  n=0

  if (accept_rate>=0.3 & accept_rate<=0.6)
  {
    verbose = FALSE
    cat("Accept Rate is:",accept_rate,"We take v^2 as:",
        Sigma_new[1, 1], "\n")
    for (i in (2+retune): (burnin+niter+1))
    {
      beta[i, ]=mvrnorm(1, mu=beta[i-1, ], Sigma=Sigma_new)
      alpha=bayes1(m, y, X, beta[i, ], beta.0, Sigma.0.inv)-
        bayes1(m, y, X, beta[i-1, ], beta.0, Sigma.0.inv)
      if (log(U[i-1])<alpha)
      { beta[i, ]=beta[i, ] }
      else
      { beta[i, ]=beta[i-1, ] }
    }
  }
  else if (accept_rate<0.3)
  {
    verbose = TRUE
    Sigma_new=Sigma_new*exp(-0.5)
    cat("Accept Rate is:",accept_rate,"
          we adjust v^2 to:", Sigma_new[1, 1], "\n")
  }
  else
  {
    verbose = TRUE
    Sigma_new=Sigma_new*exp(0.5)
    cat("Accept Rate is:",accept_rate,"we adjust v^2 to:",
        Sigma_new[1, 1], "\n")
  }
  }
beta=beta[((burnin+2):(burnin+niter+1)), ]
```

```
  for (j in 1:2)
  {
    beta_ci[, j]=quantile(beta[, j], probs=seq(0.01, 0.99, 0.01))
  }
  return(beta_ci)
}

#####################################################
# Set up the specifications:

data=read.csv(file=paste('data/blr_data_', as.character(sim_num),
                         '.csv', sep=""), header=TRUE, sep=",")
m=data$n
y=data$y
X=as.matrix(data[, 3:4])
p=ncol(X)
beta.0 = matrix(c(0,0))
Sigma.0.inv = diag(rep(1.0,p))
niter = 10000
beta_cre=bayes.logreg(m, y, X, beta.0, Sigma.0.inv=diag(1, 2),
                      niter=10000, burnin=1000, retune=200, verbose=TRU
write.table(beta_cre, file=paste('results/blr_res_',
            as.character(sim_num), '.csv', sep=""), sep=",",
            row.names = FALSE, col.names = FALSE)

#####################################################
#####################################################

cat("done. :)\n")

##Problem 3
## part a
##Problem 3
## part a
library(MASS)

bayes1= function(y, X, beta.star, beta.0, Sigma.0.inv)
{
```

```
  p= ncol(X)
  pi= -0.5*matrix(beta.star-beta.0, 1, p)%*%
    solve(Sigma.0.inv)%*%as.matrix(beta.star-beta.0,p,1)+
    t(as.matrix(y))%*%data.matrix(X)%*%matrix(beta.star,p,1)-
    sum(log(1+exp(data.matrix(X)%*%matrix(beta.star,p,1))))
  return(pi)
}

bayes.logreg = function(y, X, beta.0, Sigma.0.inv, niter=50000,
                        burnin=1000, retune=200, verbose=TRUE)
{
  p= ncol(X)
  v_2=0.01
  Sigma_new=v_2*Sigma.beta
  beta=matrix(numeric(p*(burnin+niter+1)), (burnin+niter+1), p)
  beta[1,]= mean_start
  U=runif((burnin+niter), 0, 1)
  n=0
  while (verbose)
  {
    for (i in 2: (retune+1))
    {

      beta[i, ]=mvrnorm(1, mu=beta[i-1, ], Sigma=Sigma_new)
      alpha=bayes1(y, X, beta[i, ], beta.0, Sigma.0.inv)-
        bayes1(y, X, beta[i-1, ], beta.0, Sigma.0.inv)
      if (log(U[i-1])<alpha)
      { beta[i, ]=beta[i, ]
        n=n+1
      }
      else
      { beta[i, ]=beta[i-1, ] }
    }

    accept_rate=n/retune
    n=0

    if (accept_rate>=0.35 & accept_rate<=0.55)
```

```
    {
      verbose = FALSE
      cat("Accept Rate is: ",accept_rate,
          " We take v^2 as: ", Sigma_new[1, 1], "\n")
      for (i in (2+retune): (burnin+niter+1))
      {
        beta[i, ]=mvrnorm(1, mu=beta[i-1, ], Sigma=Sigma_new)
        alpha=bayes1(y, X, beta[i, ], beta.0, Sigma.0.inv)-
          bayes1(y, X, beta[i-1, ], beta.0, Sigma.0.inv)
        if (log(U[i-1])<alpha)
        { beta[i, ]=beta[i, ] }
        else
        { beta[i, ]=beta[i-1, ] }
      }
    }
    else if (accept_rate<0.35)
    {
      verbose = TRUE
      Sigma_new=Sigma_new*exp(-0.5)
      cat("Accept Rate is: ",accept_rate,
          " we adjust v^2 to: ", Sigma_new[1, 1], "\n")
    }
    else
    {
      verbose = TRUE
      Sigma_new=Sigma_new*exp(0.5)
      cat("Accept Rate is: ",accept_rate,
          " we adjust v^2 to: ", Sigma_new[1, 1], "\n")
    }
  }
  beta=beta[((burnin+2):(burnin+niter+1)), ]

  return(beta)
}

##data setup
data=read.table(file="breast_cancer.txt", header=TRUE)
index_M=which(data[, 11]=="M")
```

```
diagnosis=numeric(nrow(data))
diagnosis[index_M]=1
data=data[, -11]
data=data.frame(data, diagnosis=diagnosis)
beta.0=matrix(rep(0, 11), 11, 1)
Sigma.0.inv=diag(1000, 11)
mean_start=glm(diagnosis~ . , data , family = "binomial")$coefficients
data=data.frame(intercept=1, data)
y=data$diagnosis
X=data[,-12]
Sigma.beta=summary(glm(diagnosis~ . , data ,
                       family = "binomial"))$cov.unscaled


betas=bayes.logreg(y, X, beta.0, Sigma.0.inv)
par(mfrow = c(2, 3), pty="m")
for (i in 1:11)
{
  plot(betas[,i], type="l", ylab=paste("beta_", i-1, sep=""),
       main=paste("Traceplot for beta_", i-1, sep=""), col="blue")
}
library(coda)
effectiveSize(betas)

## part b
auto_cor=numeric(11)
for (i in 1:11)
{
  auto_cor[i]=acf(betas[,i])[1]
}
unlist(auto_cor)

## part c
cor(data)

quantiles=matrix(numeric(22), 11, 2)
for (i in 1:11)
{
```

```
  quantiles[i, ]=quantile(betas[, i], probs=c(0.025, 0.975))
}

library(BMA)
data=data[,-1]
model_selection = bic.glm(diagnosis ~ ., glm.family="binomial", data=da
summary(model_selection)

## part d & e
p=data.matrix(X)%*%t(betas)
p=exp(p)/(1+exp(p))
p=as.vector(p)
y_simu=sapply(p, function(x) rbinom(1, 1, prob=x))
y_simu=matrix(y_simu, 569, 50000)
simu_mean=colMeans(y_simu)
simu_median=apply(y_simu, 2, median)
ratio=sum(y==1)/sum(y==0)
simu_ratio= apply(y_simu, 2, function(x) sum(x==1)/sum(x==0))

par(mfrow = c(2, 2), pty="m")
hist(simu_mean, col="light blue",
     main="Histogram of Simulated vs Real Data Mean",
     xlab="Simulated Mean of Y")
abline(h=0)
abline(v=mean(y), col="red", lwd=2)

hist(simu_median, xlim=c(0,1), col="light blue",
     main="Histogram of Simulated vs Real Data Median",
     xlab="Simulated Median of Y")
abline(h=0)
abline(v=median(y), col="red", lwd=2)

hist(simu_ratio, col="light blue",
     main="Histogram of Simulated vs Real Data Ratio(1/0)",
     xlab="Simulated Ration(1/0) of Y")
abline(h=0)
abline(v=ratio, col="red", lwd=2)
```

```
### Alternative Algorithm
bayes1= function(y, X, beta.star, beta.0, Sigma.0.inv)
{
  p= ncol(X)
  pi= -0.5*matrix(beta.star-beta.0, 1, p)%*%
    solve(Sigma.0.inv)%*%as.matrix(beta.star-beta.0,p,1)+
    t(as.matrix(y))%*%data.matrix(X)%*%matrix(beta.star,p,1)-
    sum(log(1+exp(data.matrix(X)%*%matrix(beta.star,p,1))))
  return(pi)
}
bayes.logreg = function(y, X, beta.0, Sigma.0.inv, niter=10000,
                        burnin=1000, retune=200, verbose=TRUE)
{
  p= ncol(X)
  v_2=diag(1, p)
  Sigma_new=v_2%*%Sigma.beta
  beta=matrix(numeric(p*(burnin+niter+1)), (burnin+niter+1), p)
  beta[1,]= mean_start
  U=matrix(runif(((burnin+niter)*p), 0, 1), (burnin+niter), p)
  n=rep(0, p)
  while (verbose)
  {
    for (i in 2: (retune+1))
    {
      beta[i, ]=beta[i-1,]
      for (j in 1:p)
      {
        beta.temp=beta[i, ]
        beta[i, j]=rnorm(1, mean=beta[i, j], sd=sqrt(Sigma_new[j, j]))
        alpha=bayes1(y, X, beta[i, ], beta.0, Sigma.0.inv)-
          bayes1(y, X, beta.temp, beta.0, Sigma.0.inv)
        if (log(U[i-1, j])<alpha)
        {
          beta[i, j]=beta[i, j]
          n[j]=n[j]+1
        }
        else
        { beta[i, j]=beta[i-1, j] }
```

```
  }
}
accept_rate=n/retune
n=rep(0, p)

if (all(accept_rate>=0.35) & all(accept_rate<=0.55))
{
  verbose = FALSE
  cat("Accept Rate for each beta is: ",
      accept_rate,"We take v^2 as: ", diag(v_2), "\n")
  for (i in (2+retune): (burnin+niter+1))
  {
    beta[i, ]=beta[i-1,]
    for (j in 1:p)
    {
      beta.temp=beta[i, ]
      beta[i, j]=rnorm(1, mean=beta[i, j], sd=sqrt(Sigma_new[j, j]))
      alpha=bayes1(y, X, beta[i, ], beta.0, Sigma.0.inv)-
        bayes1(y, X, beta.temp, beta.0, Sigma.0.inv)
      if (log(U[i-1, j])<alpha)
      {
        beta[i, j]=beta[i, j]
      }
      else
      {
        beta[i, j]=beta[i-1, j]
      }
    }
  }
}

else
{
  for (j in 1:p)
  {
    if (accept_rate[j]<0.35)
    {
      v_2[j, j]=v_2[j, j]*exp(-0.2)
```

```
        }
        else if(accept_rate[j]>0.55)
        {
          v_2[j, j]=v_2[j, j]*exp(0.2)
        }

      }
      cat("Accept Rate is: ",accept_rate,"
     we adjust v^2 to: ", diag(v_2), "\n")
      Sigma_new=v_2%*%Sigma.beta
    }

  }
  beta=beta[((burnin+2):(burnin+niter+1)), ]
  return(beta)
}

##data set-up
data=read.table(file="breast_cancer.txt", header=TRUE)
index_M=which(data[, 11]=="M")
diagnosis=numeric(nrow(data))
diagnosis[index_M]=1
data=data[, -11]
data=data.frame(data, diagnosis=diagnosis)
beta.0=matrix(rep(0, 11), 11, 1)
Sigma.0.inv=diag(1000, 11)
mean_start=glm(diagnosis~ . , data , family = "binomial")$coefficients
data=data.frame(intercept=1, data)
y=data$diagnosis
X=data[,-12]
Sigma.beta=diag(1, 11)

betas=bayes.logreg(y, X, beta.0, Sigma.0.inv)
effectiveSize(betas)
```