STA 250 Homework 01

GAO, Qi

October 2013

Contents

1 Correctness of the Gibbs Sampling Algorithm

(a) Since the Markov chain generated by the Gibbs sampler is assumed to be ergodic, the long-run time average of the chain converges to a stationary distribution π that satisfies the following equation:

$$\pi(y) = \int \pi(x)k(x,y)dx, \forall y \tag{1}$$

where $x = (x_1, x_2)$ and $y = (y_1, y_2)$ are two states of the Markov chain and $k(x, y)$ is the transition kernel from state x to state y.

From the algorithm of the Gibbs sampling, we have $k(x, y) = p(y_1|x_2)p(y_2|y_1)$.

$\therefore \int p(x)k(x,y)dx = \int\int p(x_1,x_2)p(y_1|x_2)p(y_2|y_1)dx_1dx_2$

$= \int\int p(x_1,x_2)dx_1p(y_1|x_2)p(y_2|y_1)dx_2 = \int p(x_2)p(y_1|x_2)p(y_2|y_1)dx_2$

$= \int p(y_1,x_2)dx_2p(y_2|y_1) = p(y_1)p(y_2|y_1) = p(y_1,y_2) = p(y)$

Therefore the joint density p also satisfies equation (1).

Since ergodic Markov chain has unique stationary distribution, we come to the conclusion that the stationary distribution π of the two-component Gibbs sampler is the desired target distribution $p(x_1, x_2)$.

(b) Similar to part (a), let $x = (x_1, x_2, ..., x_p)$ and $y = (y_1, y_2, ..., y_p)$, then the transition kernel

$$k(x,y) = p(y_1|x_2,...,x_p)p(y_2|y_1,x_3,...,x_p)\cdots p(y_p|y_1,...,y_{p-1})$$

$\therefore \int p(x)k(x,y)dx$

$= \int\cdots\int p(x_1,...,x_p)dx_1p(y_1|x_2,...,x_p)p(y_2|y_1,x_3,...,x_p)\cdots p(y_p|y_1,...,y_{p-1})dx_2\cdots dx_p$

$= \int\cdots\int p(x_2,...,x_p)p(y_1|x_2,...,x_p)dx_2\cdots p(y_p|y_1,...,y_{p-1})dx_3\cdots dx_p$

$= \int\cdots\int p(y_1,x_2,...,x_p)dx_2p(y_2|y_1,x_3,...,x_p)\cdots p(y_p|y_1,...,y_{p-1})dx_3\cdots dx_p$

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

$= \cdots$

$= \int p(y_1,...,y_{p-2},x_p)p(y_{p-1}|y_1,...,y_{p-2},x_p)p(y_p|y_1,...,y_{p-1})dx_p$

$= \int p(y_1,...,y_{p-1},x_p)dx_pp(y_p|y_1,...,y_{p-1})$

$= p(y_1,y_2,...,y_{p-1})p(y_p|y_1,...,y_{p-1})$

$= p(y_1,y_2,...,y_p) = p(y)$

Again the joint density p satisfies equation (1). Since ergodic Markov chain has unique stationary distribution, we come to the conclusion that the stationary distribution π of the p-component Gibbs sampler is indeed $p(x_1, x_2, ..., x_p)$.

2 Bayesian Logistic Regression MCMC Algorithm

2.1 Analytic form of the posterior distribution up to proportionality

$$
\begin{aligned}
&p(\beta|\mathbf{y}) \propto p(\mathbf{y}|\beta)p(\beta) \\
&\propto \prod_{i=1}^{n} p(y_i|\beta)p(\beta) \\
&\propto \prod_{i=1}^{n} [\text{logit}^{-1}(x_i^T\beta)]^{y_i}[1-\text{logit}^{-1}(x_i^T\beta)]^{(m_i-y_i)}|\Sigma_0|^{-1/2}exp\{-1/2(\beta-\mu_0)^T\Sigma_0^{-1}(\beta-\mu_0)\} \\
&\propto \prod_{i=1}^{n} [\tfrac{exp(x_i^T\beta)}{1+exp(x_i^T\beta)}]^{y_i}[\tfrac{1}{1+exp(x_i^T\beta)}]^{(m_i-y_i)}exp\{-1/2(\beta-\mu_0)^T\Sigma_0^{-1}(\beta-\mu_0)\} \\
&\propto \prod_{i=1}^{n} \tfrac{[exp(x_i^T\beta)]^{y_i}}{[1+exp(x_i^T\beta)]^{m_i}}exp\{-1/2(\beta-\mu_0)^T\Sigma_0^{-1}(\beta-\mu_0)\}.
\end{aligned}
$$

2.2 Description of my MCMC algorithm

In my Metropolis-Hastings algorithm, I used normal distribution as my proposal distribution, i.e. $\beta_{prop} \sim N(\beta_{curr}, V)$ where β_{prop} is the proposed value, β_{curr} is the current value and V is the covariance matrix of the proposal distribution. The value of V was tuned by calculating the acceptance rate within 100 iterations. If the acceptance rate is less than or equal to 0.3, V is decreased to $V/exp(1)$ in the next tuning cycle; if the acceptance rate is greater than or equal to 0.6, V is increased to $V \cdot exp(1)$.

The burnin period is set to be 1000 and the number of iterations used to draw samples is 10000. The starting value of β is simply chosen to be $(0, 0)$. For part d, I chose the 26-th dataset from the 200 generated data files to fit the model. These options work quite well judging from the traceplots as shown in Figutre 1. The effective sample size for β_1 and β_2 are 1426.783 and 1163.738 which are also reasonably large. The highest posterior density intervals for β_1 and β_2 are (-0.7829, -0.6956) and (0.5883, 0.6857) respectively, which contain the true value of β_1 and β_2 (i.e., -0.7229 and 0.6545).

2.3 Numerical coverage properties

The numerical results of the coverage simulation are summarized in Table 1. We can see that the empirical coverage is quite close to the nominal coverage.

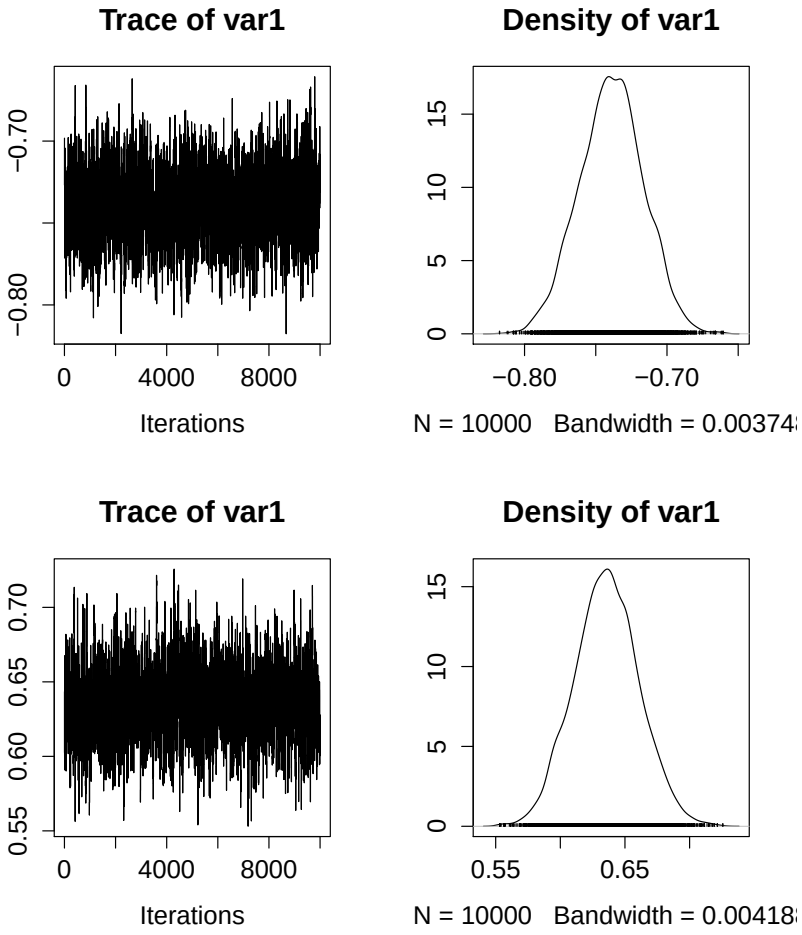

Figure 1: Traceplots of β for Q2 Part d

2.4 Graph of the coverage properties

The coverage plot is shown in Figure 2. The coverage lines approximately stayed inside the guiding bands, which confirms the validation of the algorithm.

3 Breast Cancer Dataset Fitting

(a) Similar to question 2, I used Metropolis-Hastings algorithm to fit the model. The covariance matrix for the proposal distribution is firstly set to be the covariance estimate from the glm fit in R. The covariance matrix is tuned every 200 iterations in order to obtain a more stable estimate of the acceptance rate. The burnin period is also increased to 5000 to obtain more stabilized samples. Otherwise the algorithm is very

	beta_0	beta_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	0.51	0.47
p_75	0.75	0.76
p_90	0.89	0.87
p_95	0.94	0.95
p_99	0.99	0.99

Table 1: Coverage Summaries

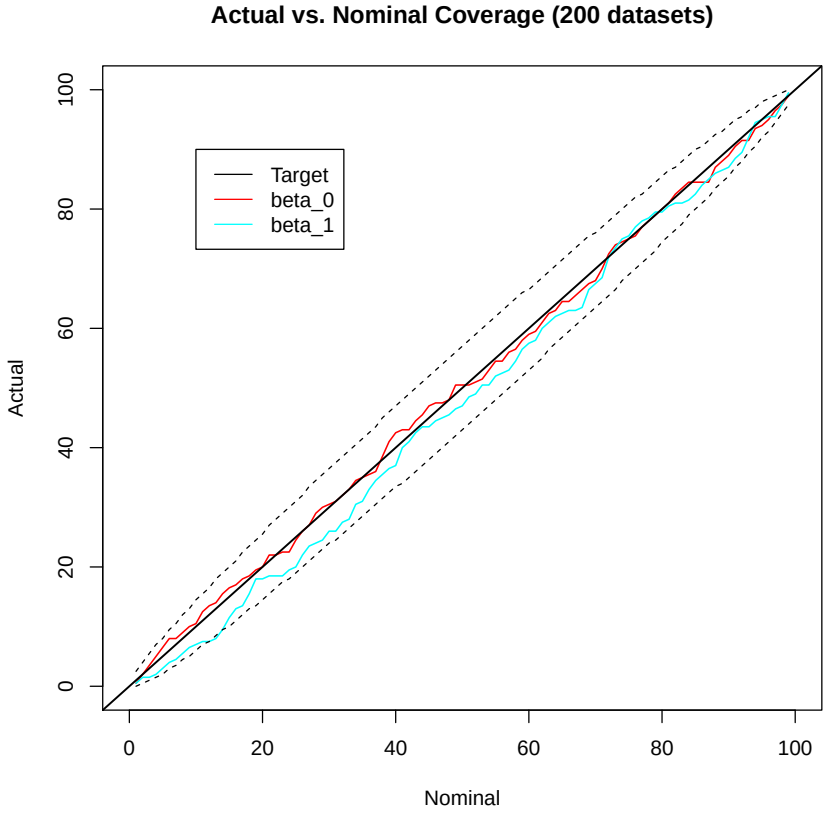

Figure 2: Coverage Plot

similar to question 2 and the data is fitted on the original scale. The traceplots for each component of β are shown in Appendix A. We can see from these traceplots that

the Markov Chain converges reasonably well.

(b) The lag-1 correlation for each component of β is summarized in Table 2.

	lag-1 autocorrelation	95% HPD interval
β_1	0.9554	(-33.7274,4.6562)
β_2	0.9678	(0.0056,0.0652)
β_3	0.9509	(-31.6081,22.4005)
β_4	0.9395	(20.3255,94.2572)
β_5	0.9555	(-4.1334,20.3984)
β_6	0.8315	(-72.5635,44.2105)
β_7	0.9635	(-0.8265,0.8388)
β_8	0.9647	(-7.8068,4.5784)
β_9	0.9321	(16.8699,92.6272)
β_{10}	0.9675	(-0.8294,37.3372)
β_{11}	0.9689	(0.2518,0.4825)

Table 2: Summaries of the Lag-1 Autocorrelations and 95% HPD Intervals

(c) The 95% highest posterior density interval for each component of β is also summarized in Table 2. From the summary we can see that the intervals for β_2, β_4, β_9 and β_{11} do not contain zero. These four β correspond to area, concavepts, smoothness and texture. Since intervals not containing zero indicate the significance of the corresponding β, these four covariates seem to be related to the cancer diagnosis based on the HPD intervals.

(d) I have generated 1000 new datasets and used mean for comparison. The resulting histogram of the mean is shown in Figure 3 and the true mean calculated from the original dataset is also indicated in the plot.

(e) From Figure 3 we can see that the true mean is within the highest sub-interval and around the center of the histogram, which indicates that my model is a reasonable fit to the data.

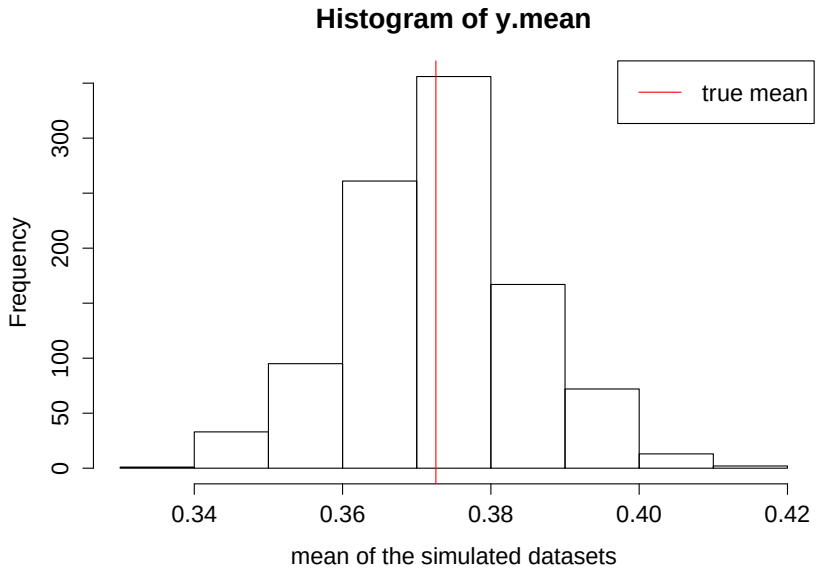

Figure 3: Posterior Predictive Check with Mean

A Traceplots for Q3

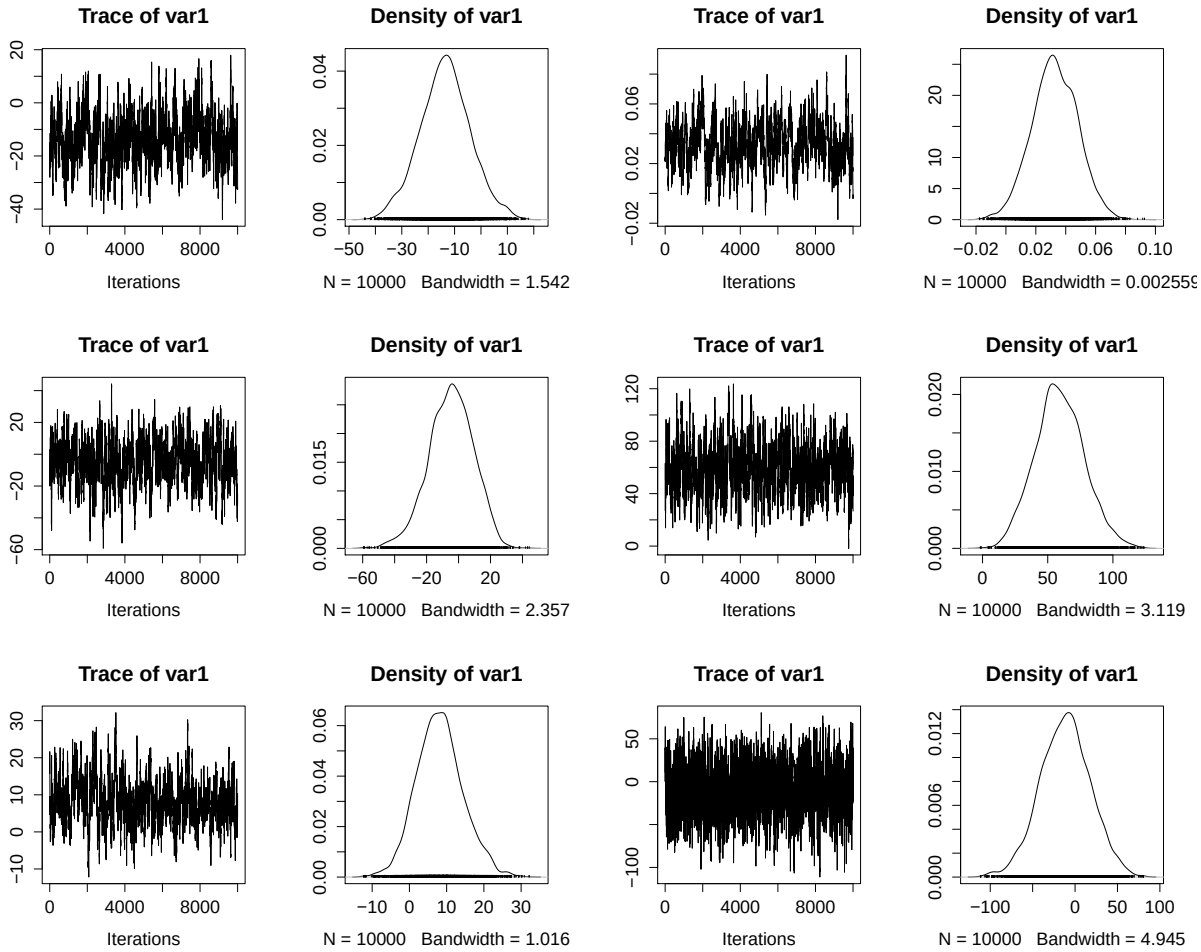

Figure 4: Traceplots of $\beta_1 \sim \beta_6$

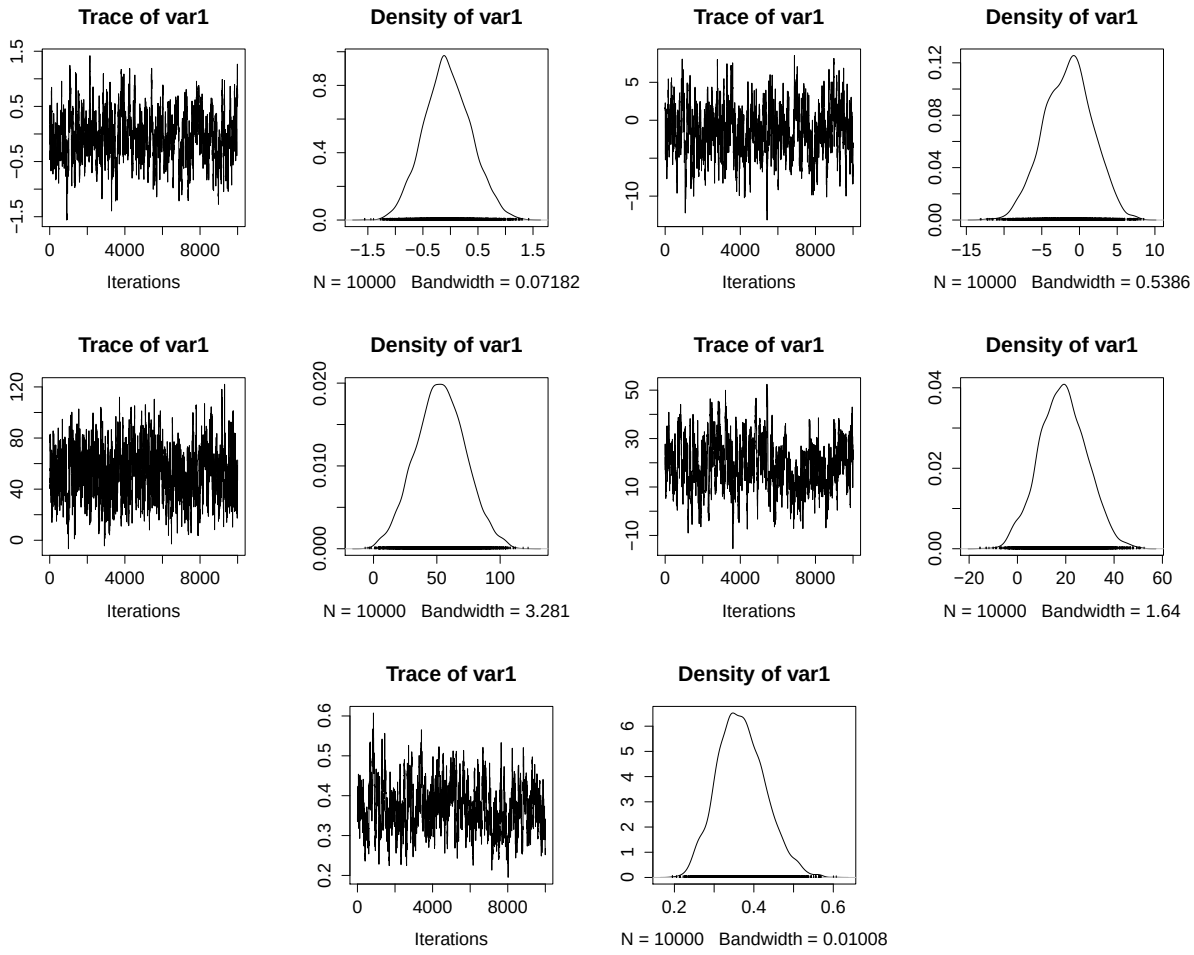

Figure 5: Traceplots of $\beta_7 \sim \beta_{11}$

B Program Codes

R Code

```
########################
#Q2 part d MCMC routine#
########################

library(MASS)
library(coda)

data<-read.table(file=paste("data/blr_data_",as.character(sim_num),".csv",
sep=""), sep=",", header=T)
m<-data$n
y<-data$y
x<-cbind(data$X1, data$X2)
beta.0<-c(0,0)
Sigma.0.inv<-diag(2)

"log.target.density" <- function(m,y,x,beta,beta.0,Sigma.0.inv){
  log.p <- t(y)%*%x%*%beta-t(m)%*%log(1+exp(x%*%beta)) -
  t(beta-beta.0)%*%Sigma.0.inv%*%(beta-beta.0)/2
  return(log.p)
}

"bayres.logreg"<-function(m,y,x,beta.0,Sigma.0.inv,
                          niter=10000,burnin=1000,
                          retune=100,verbose=TRUE) {

  ## Tune the covariance matrix of the proposal distribution

  # Proposal distribution: beta.prop ~ N(beta.curr, V)
  # Set the initial value of V as the identity matrix
  V <- diag(2)
  beta.curr <- c(0,0)
  accept.rate=0

  while (accept.rate <= 0.3 | accept.rate >= 0.6) {
    n.accept <- 0
    for (j in 1:retune) {
      beta.prop <- mvrnorm(n=1,beta.curr,V)
```

```
    log.alpha <- log.target.density(m,y,x,beta=beta.prop,beta.0,Sigma.0.inv) -
    log.target.density(m,y,x,beta=beta.curr,beta.0,Sigma.0.inv)
    log.u <- log(runif(1))
    if (log.u < log.alpha){
      beta.curr <- beta.prop
      n.accept <- n.accept+1
    } else {
    }
  }
  accept.rate<-round(n.accept/retune,2)
  if (accept.rate<=0.3) {
    V<-V/exp(1)
    } else
    if (accept.rate>=0.6) {
      V<-V*exp(1)
      } else {
      }
}

cat(paste("Acceptance rate after tuning V was ",100*accept.rate,"%\n",sep=""))
cat(paste("The covariance matrix after tuning was \n", sep=""))
print(V)

## Begin the sampling procedure

beta.curr <- c(0,0)
# Store the samples
beta.draws<-matrix(rep(NA, 2*(niter+burnin)), ncol=2)
# Track tha acceptance rate:
n.accept <- rep(0,(niter+burnin))

for (i in 1:(niter+burnin)){
  beta.prop <- mvrnorm(n=1,beta.curr,V)
  log.alpha <- log.target.density(m,y,x,beta=beta.prop,beta.0,Sigma.0.inv) -
  log.target.density(m,y,x,beta=beta.curr,beta.0,Sigma.0.inv)
  log.u <- log(runif(1))
  if (log.u < log.alpha){
    beta.curr <- beta.prop
    n.accept[i] <- 1
  } else {
```

```
    }
    # Store the current state:
    beta.draws[i,] <- beta.curr
  }
  # Throw away burnin period
  beta.samples<-beta.draws[(burnin+1):(niter+burnin),]
  # Report the acceptance rate:
  cat(paste("Acceptance rate was ",
  100*round(sum(n.accept[(burnin+1):(niter+burnin)])/niter,2), "%\n",sep=""))

  ## Check the samples I generated
  beta1<-mcmc(beta.samples[,1])
  plot(beta1)
  beta2<-mcmc(beta.samples[,2])
  plot(beta2)
  cat(paste("ESS for beta1 = ",round(effectiveSize(beta1),4), sep=""),"\n")
  cat(paste("ESS for beta2 = ",round(effectiveSize(beta2),4), sep=""),"\n")
  cat(paste("The 95% HPD interval for beta1 = (",
  round(HPDinterval(beta1,0.95)[1,1],4),",",
  round(HPDinterval(beta1,0.95)[1,2],4),")", sep=""),"\n")
  cat(paste("The 95% HPD interval for beta2 = (",
  round(HPDinterval(beta2,0.95)[1,1],4),",",
  round(HPDinterval(beta2,0.95)[1,2],4),")", sep=""),"\n")

  ## Output the percentiles to a csv file
  percentile<-matrix(rep(0, 2*99), ncol=2)
  for (i in 1:99) {
    percentile[i,1]=quantile(beta.samples[,1],probs=i/100)
    percentile[i,2]=quantile(beta.samples[,2],probs=i/100)
  }
  write.table(percentile,
  file=paste("results/blr_res_",as.character(sim_num),".csv", sep=""),
  sep=",", row.names=F, col.names=F)

}

bayres.logreg(m,y,x,beta.0,Sigma.0.inv,
              niter=10000,burnin=1000,
              retune=100,verbose=TRUE)
```

```
###################################
#Q3 breast cancer dataset fitting#
###################################

library(MASS)
library(coda)

"log.target.density" <- function(y,x,beta,beta.0,Sigma.0.inv){
  log.p <- t(y)%*%x%*%beta-sum(log(1+exp(x%*%beta))) -
  t(beta-beta.0)%*%Sigma.0.inv%*%(beta-beta.0)/2
  return(log.p)
}

"logit.inv" <- function(x){
  v<-exp(x)/(1+exp(x))
  return(v)
}

data<-read.table("breast_cancer.txt", header=T)
n<-length(data$diagnosis)
p<-11
intercept<-rep(1,n)
response<-rep(0,n)
data2<-cbind(intercept, data, response)
data2[which(data2[,12]=="M"),13]<-1
cancer<-data2[,-12]

# Set the initial value of V as the covariance estimates from glm fit
fit<-glm(response~area+compactness+concavepts+concavity+fracdim+
           perimeter+radius+smoothness+symmetry+texture,
           family=binomial(),data=cancer)
V<-as.matrix(summary(fit)$cov.unscaled)

y<-as.matrix(cancer$response)
x<-as.matrix(cancer[1:p])

beta.0<-rep(0,p)
Sigma.0.inv<-diag(p)*0.001

"bayres.logreg"<-function(y,x,beta.0,Sigma.0.inv,
```

```
                        niter=10000,burnin=5000,
                        retune=200,verbose=TRUE) {

## Tune the covariance matrix of the proposal distribution

# Proposal distribution: beta.prop ~ N(beta.curr, V)
beta.curr <- rep(0,p)
accept.rate=0

while (accept.rate <= 0.3 | accept.rate >= 0.6) {
  n.accept <- 0
  for (j in 1:retune) {
    beta.prop <- mvrnorm(n=1,beta.curr,V)
    log.alpha <- log.target.density(y,x,beta=beta.prop,beta.0,Sigma.0.inv) -
    log.target.density(y,x,beta=beta.curr,beta.0,Sigma.0.inv)
    log.u <- log(runif(1))
    if (log.u < log.alpha){
      beta.curr <- beta.prop
      n.accept <- n.accept+1
    } else {
    }
  }
  accept.rate<-round(n.accept/retune,2)
  if (accept.rate<=0.3) {
    V<-V/exp(1)
  } else
    if (accept.rate>=0.6) {
      V<-V*exp(1)
    } else {
    }
}

cat(paste("Acceptance rate after tuning V was ",100*accept.rate,"%\n",sep=""))
cat(paste("The covariance matrix after tuning was \n", sep=""))
print(V)

## Begin the sampling procedure
beta.draws<-matrix(rep(NA, p*(niter+burnin)), ncol=p)
n.accept <- rep(0,(niter+burnin))
```

```
  for (i in 1:(niter+burnin)){
    beta.prop <- mvrnorm(n=1,beta.curr,V)
    log.alpha <- log.target.density(y,x,beta=beta.prop,beta.0,Sigma.0.inv) -
    log.target.density(y,x,beta=beta.curr,beta.0,Sigma.0.inv)
    log.u <- log(runif(1))
    if (log.u < log.alpha){
      beta.curr <- beta.prop
      n.accept[i] <- 1
    } else {
    }
    beta.draws[i,] <- beta.curr
  }

  # Throw away burnin period
  beta.samples<-beta.draws[(burnin+1):(niter+burnin),]
  # Report the acceptance rate:
  cat(paste("Acceptance rate was ",
  100*round(sum(n.accept[(burnin+1):(niter+burnin)])/niter,2),"%\n",sep=""))

  return(beta.samples)
}

beta.samples<-bayres.logreg(y,x,beta.0,Sigma.0.inv,
                            niter=10000,burnin=5000,
                            retune=200,verbose=TRUE)

## Check if the markov chain converges

for (j in 1:p) {
  plot(mcmc(beta.samples[,j]))
  cat(paste("ESS for ",j,"-th beta = ",
  round(effectiveSize(mcmc(beta.samples[,j])),4), sep=""),"\n")
}

## Compute the lag-1 autocorrelation for each component of beta
for (j in 1:p) {
  cat(paste("lag-1 autocorrelation for ",j,"-th beta = ",
  round(acf(beta.samples[,j],
  type="correlation",plot=F)$acf[2] , 4), sep=""),"\n")
}
```

```
## Create posterior credible intervals
for (j in 1:p) {
  cat(paste("The 95% posterior credible interval for ",j,"-th beta =
  (",round(HPDinterval(mcmc(beta.samples[,j]),0.95)[1,1],4),",",
  round(HPDinterval(mcmc(beta.samples[,j]),0.95)[1,2],4),")", sep=""),"\n")
}

## Posterior predictive checking
n.new<-1000
y.new<-matrix(rep(NA, n.new*n), ncol=n)
index<-sample(1:niter, n.new)
for (i in 1:n.new) {
  beta<-beta.samples[index[i], ]
  for (j in 1:n) {
    y.new[i,j]<-rbinom(1, 1, logit.inv(x[j,]%*%beta))
  }
}
y.mean<-apply(y.new, 1, mean)
hist(y.mean, xlab="mean of the simulated datasets")
abline(v=mean(y),col="red")
legend("topright","true mean",lty=1,col="red")
```

R Code