```
##
#
# 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(mvtnorm)
library(coda)

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

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



bayes.logreg <- function(n,y,X,beta.0,Sigma.
0.inv,niter=10000,burnin=1000,
              print.every=1000,retune=100,verbose=TRUE)
{
    beta = matrix(ncol = length(beta.0), nrow = niter+1)
 beta[1,] = beta.0
 v = rep(1, ncol(beta))
 rej = rep(0, ncol(beta))
 for (i in 2:nrow(beta)){
  beta[i,] = beta[i-1,]
  for (j in 1:ncol(beta)){
   beta[i,j] = rnorm(1, beta[i-1,j], v[j])
   post1 = post(n, y, X, as.numeric(beta[i,]),
as.numeric(beta[i-1,]), diag(v))
   post2 = post(n, y, X, as.numeric(beta[i-1,]),
as.numeric(beta[i,]), diag(v))
   alpha = post1 - post2
   if ((alpha < 0) & (runif(1, 0, 1) > exp(alpha))) {
    beta[i, j] = beta[i-1, j]
    rej[j] = rej[j] + 1
   }
  }
  if (verbose & ((i-1)%%print.every == 0) & i >= burnin) {
   cat(paste("Interations:\n",
        i-1, "\nSample:\n",
        paste(beta[i,], collapse=","),"\n\n"))
        #"\nAcceptance Rate:\n",
        #paste(1-rej/(i-1), collapse=",")
  }
  if (((i-1)%%retune == 0) & (i < burnin)) {
   v[(1-rej/(i-1)) > 0.6] = v[(1-rej/(i-1)) > 0.6] * 1.1
   v[(1-rej/(i-1)) < 0.2] = v[(1-rej/(i-1)) < 0.3] / 1.1
  }
 }
 return(beta[(burnin+2):nrow(beta),])
}

################################################
# Set up the specifications:
p = 2
beta.0 <- matrix(c(0,0))
Sigma.0.inv <- diag(rep(1.0,p))
```

```
niter <- 10000
post = function(n, y, X, beta, mu, sig.inv) {
  p1 = t(X %*% beta) %*% y - 1/2 * t(beta - mu) %*% sig.inv %*% (beta
- mu)
  p2 = - t(n) %*% log(1+exp(X %*% beta))
  poster = p1 + p2
  poster
}
# etc... (more needed here)
#################################################

# Read data corresponding to appropriate sim_num:
dat.df = read.csv(sprintf("~/STA250/Stuff/HW1/BayesLogit/data/
blr_data_%d.csv", sim_num), header = TRUE)
# Extract X and y:
y = dat.df$y
X = cbind(dat.df$X1, dat.df$X2)
# Fit the Bayesian model:
beta.res = bayes.logreg(n = dat.df$n, y = y, X = X, beta.0, Sigma.
0.inv, verbose=FALSE)
# Extract posterior quantiles...
cred.int = apply(beta.res, 2, function(x)quantile(x, probs = c(0.025,
0.975)))
percentiles = apply(beta.res, 2, function(x)quantile(x, probs =
seq(0.01, 0.99, 0.01)))
# Write results to a (99 x p) csv file...
write.table(percentiles, sprintf("~/STA250/Stuff/HW1/BayesLogit/
results/blr_res_%d.csv", sim_num), sep=",", row.names = FALSE,
col.names = FALSE)
# Go celebrate.

cat("done. :)\n")
```