```
##
#
# 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)


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)) < 0.2] / 1.1
    }
  }
  return(beta[(burnin+2):nrow(beta),])
```

```
}

################################################
# Set up the specifications:
p = 11
beta.0 <- matrix(rep(0, p))
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.table("~/STA250/Stuff/HW1/BayesLogit/breast_cancer.txt",
header = TRUE)
#standardize covariate
dat.df[,-11] = sapply(dat.df[,-11], function(x)(x-mean(x))/sd(x))
# Extract X and y:
y = as.numeric(dat.df$diagnosis)-1
X = cbind(rep(1, nrow(dat.df)), as.matrix(dat.df[,-11]))
# Fit the Bayesian model:
beta.res = bayes.logreg(n = rep(1, nrow(dat.df)), y = y, X = X, beta.
0, Sigma.0.inv,
                        niter = 100000, burnin = 10000, verbose=FALSE)
# Extract posterior quantiles...
cred.int = apply(beta.res, 2, function(x)quantile(x, probs = c(0.025,
0.975)))
percentiles = apply(beta.res, 2, function(x)quantile(x, probs =
seq(0.01, 0.99, 0.01)))
# Write results to a (99 x p) csv file...
write.table(percentiles,"~/STA250/Stuff/HW1/BayesLogit/pb3_res.csv",
            sep=",", row.names = FALSE, col.names = FALSE)
save(beta.res, file="~/STA250/Stuff/HW1/BayesLogit/pb3_beta_res.rda")
#compute lag-1 autocorrelations for each component of beta
acf.beta = apply(beta.res, 2, function(x)acf(x, lag.max = 1)$acf[2])
pdf("~/STA250/Stuff/HW1/BayesLogit/pb3_acf_plot.pdf")
plot(acf.beta, axes=FALSE, main = "Lag-1 autocorrelation for each
component of beta",
     xlab = "beta", ylab = "autocorrelation")
box()
axis(2)
axis(1, at=1:11, labels=1:11)
dev.off()
```

```
# Go celebrate.

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