Assignment 13.a for **STATGR6103**

submitted to Professor Andrew Gelman

Advait Rajagopal

2 December 2016

1 Question 1

Millions of people in rural Bangladesh are exposed to dangerous levels of arsenic in their drinking water which they get from home wells. Several years ago a survey was conducted in small area of Bangladesh to see if people with high arsenic levels would be willing to switch to a neighbor's well. File `http://www.stat.columbia.edu/ gelman/bda.course/wells.dat` has the data; all you need here are the variables switch (1 if the respondent said he or she would switch, 0 otherwise) and arsenic (the concentration in the respondent's home well, with anything over 0.5 considered dangerous). Apply the probit model described in exercise 1 above to predict the probability of switching given arsenic level. The goal is inference for the coefficients a and b.

Given that the goal of the survey is inference about the parameters 'a' and 'b', I first examine the data in order to understand the predictor and the dependent variable. able 1 displays summary statistics of these two variables.

Table 1: Data summary

stat	switch	arsenic
min	0	0.51
mean	0.57	1.66
max	1	9.65

The dependent variable (switch) is a binary response variable, where people indicate their willingness to switch to a neighbor's well. The predictor (arsenic) is the level of arsenic in the drinking water. We are asked to implement a probit model to predict probability of switching given arsenic levels. We are asked to motivate the probit model using a latent variable formulation with $z_i \sim \mathrm{N}(\alpha + \beta x_i, 1)$ such that;

$$y_i = \begin{cases} 1, z_i > 0 \\ 0, \text{otherwise} \end{cases}$$

The probability that $y_i = 1$ is given by the probit regression below;

$$\begin{aligned} Pr(y_i = 1|x_i) &= \Phi(z_i) \\ &= \Phi(\alpha + \beta x_i) \end{aligned}$$

Where Φ is the cumulative distribution function (CDF) of the standard normal distribution and our goal is to estimate coefficients or parameters α and β using a probit regression, with three techniques. I program an HMC and allow it to estimate the parameters of the probit model, then I run the same model in Stan and finally estimate the parameters using a variational inference algortihm and compare the results.

Since the outcome variable is binary, I use a binomial likelihood function with the following analytical form;

$$p(y|\alpha, \beta, x) \propto \prod_{i=1}^{N} (\Phi(\alpha + \beta x_i))^{y_i} (1 - \Phi(\alpha + \beta x_i))^{1-y_i}$$

I also consider noninformative priors on α andβ such that;

$$p(\alpha, \beta) \propto 1$$

Thus the posterior density is given by;

$$\begin{aligned} p(\alpha, \beta|y, x) &\propto p(\alpha, \beta) p(y|\alpha, \beta, x) \\ &= \prod_{i=1}^{N} (\Phi(\alpha + \beta x_i))^{y_i} (1 - \Phi(\alpha + \beta x_i))^{1-y_i} \end{aligned}$$

1.1 Part A

Program Hamiltonian Monte Carlo for the probit model described in the previous exercise, again using the latent-variable formulation (so you are jumping in a space of n+2 dimensions). Tune the algorithm and run to approximate convergence.

I program an HMC and run 4 chains with 1000 iterations. I set the mass matrix to scale with the variance of the posterior density and tune the ϵ and L parameters in order to get approximately 65% convergence. Table 2 shows the posterior intervals as estimated by the HMC algorithm

Table 2: Posterior Distributions: HMC in R

	mean	se.mean	sd	2.5%	25%	50%	75%	97.5%	n.eff	Rhat
α	-0.2	0	0	-0.3	-0.2	-0.2	-0.2	-0.1	572	1
β	0.2	0	0	0.2	0.2	0.2	0.2	0.3	714	1

1.2 Part B

Check your results by running Stan.

I fit a probit model in Stan and get the results summarized in Table 3.

Table 3: Posterior Distributions: Stan

	mean	se.mean	sd	2.5%	25%	50%	75%	97.5%	n.eff	Rhat
α	-0.18	0.00	0.04	-0.26	-0.21	-0.18	-0.15	-0.11	839	1
β	0.23	0.00	0.02	0.19	0.21	0.23	0.24	0.27	869	1

1.3 Part C

Compare to the variational Bayes inferences for a and b from the previous above.
I program a variational Bayes algorithm to estimate the parameters α and β. The posterior intervals of the parameters are displayed in Table 4. I make some plots to compare the inferences from the

Table 4: Posterior Distributions: Stan

	mean	se.mean	sd	2.5%	25%	50%	75%	97.5%
α	-0.178	0.00	0.03	-0.29	-0.21	-0.18	-0.16	-0.11
β	0.227	0.00	0.03	0.20	0.21	0.23	0.25	0.27

3 approaches, namely HMC, Stan and variational inference. Figure 1 shows a scatterplot of samples drawn from the posterior densities for each of the 3 approaches.

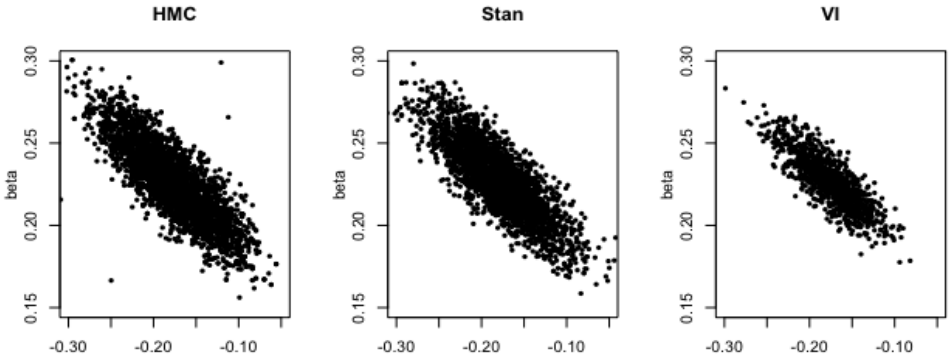

Figure 1: *The results from all the approaches are quite similar.*

Figure 2 and 3 show the marginal posterior density of the parameters α and β as estimated by the 3 approaches, respectively.

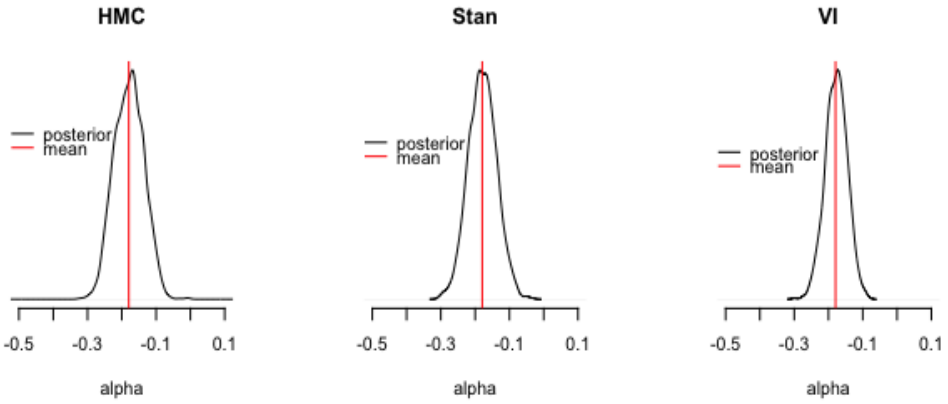

Figure 2: *All 3 approaches estimate a similar posterior density for α, the mean of the marginal posterior distribution is marked by the red line.*

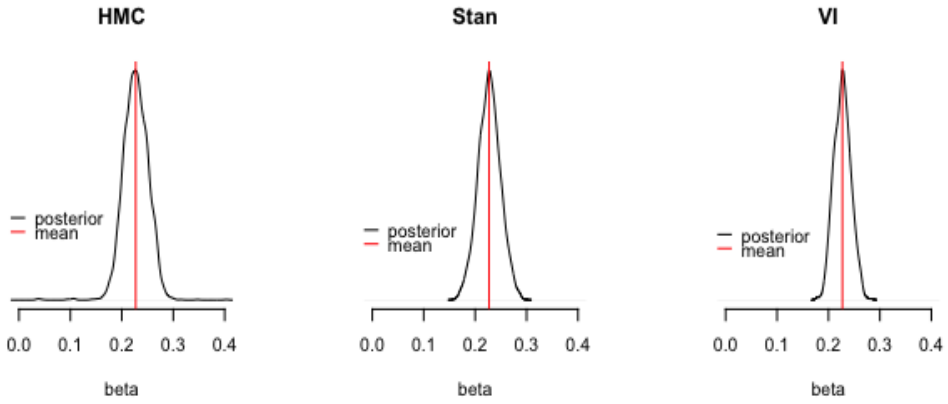

Figure 3: *All 3 approaches estimate a similar posterior density for β, the mean of the marginal posterior distribution is marked by the red line.*

We see that the results from HMC, Stan and variational Bayes algorithm are very similar and comparable.

2 Code

2.1 R code

Listing 1: R code

```
rm(list = ls())
setwd("/Users/Advait/Desktop/New␣School/Fall16/BDA/Class24")
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
install.packages("data.table", dependencies = T)
library(data.table)
library ( plyr )
library (arm)
library (msm)
library (mvtnorm)
library(LearnBayes)
####################################################
df <- fread("http://www.stat.columbia.edu/
~gelman/bda.course/wells.dat")
str(df)
names(df) <- c("s.no","switch", "arsenic", "dist", "assoc", "educ")
summary(df)
###
df_final <- data.frame(df$switch, df$arsenic)
class(df_final)
str(df_final)
##
#Stan
y <- df_final$df.switch
x <- df_final$df.arsenic
N <- length(df_final$df.switch)
N
stanc("13a.stan")
fit_stan <- stan("13a.stan", data = list("N", "y", "x"),
                 iter = 2000, chains = 3)
print(fit_stan)

# plot(ext_stan$y_rep)
# points(ext_stan$y_rep, col = "red")
# plot(colMeans(ext_stan$y_rep), y, pch = 16, cex = 0.7)
# abline(0,1)
```

Listing 2: R code contd.

```
##
##PartA
#HMC
#log posterior function
n <- 1
log_post <- function(param, x , y){
  param_prior <- dunif(param,-100,100, log = T)
  log_prior <- sum(param_prior)
  log_likelihood <- sum(dbinom(y, n,
  prob = pnorm(param[1] + param[2]*x),log = T))
  return(log_prior + log_likelihood)
}
log_post(c(0.1,-0.3),x,y)

#Numerical gradient
gradient_num <- function(param, x, y){
  d <- length(param)
  e <- 0.0001
  diff <- rep(NA,d)
  for(k in 1:d){
    th_hi <- param
    th_lo <- param
    th_hi[k] <- param[k] + e
    th_lo[k] <- param[k] - e
    diff[k] <- (log_post(th_hi,x,y) - log_post(th_lo,x,y))/(2*e)
  }
  return(diff)
}
gradient_num(c(-0.001,-.04),x,y)
###
#HMC iter
hmc_iteration <- function(param,x,y, epsilon,L,M){
  M_inv <- 1/M
  d <- length(param)
  # Sample 10 points randomly from a normal
  #distribution with mean = 0 and standard deviation = sqrt(M)
  phi <- rnorm(d , 0, sqrt(M))
  param_old <- param
  log_p_old <- log_post(param,x,y) - 0.5*sum(M_inv*phi^2)
  phi <- phi + 0.5*epsilon*gradient_num(c(param), x, y)
```

Listing 3: R code contd.

```
  for (l in 1:L){
    param <- param + epsilon*M_inv*phi
    phi <- phi + (if (l==L)0.5 else 1)*epsilon*
    gradient_num(c(param),x,y)
  }
  phi <- -phi
  log_p_star <- log_post(c(param),x,y) - 0.5*sum(M_inv*phi^2)
  r <- exp(log_p_star - log_p_old)
  if(is.nan(r)) r <- 0
  p_jump <- min(r,1)
  param_new <- if(runif(1) < p_jump) param else param_old
  return (list (param = param_new, p_jump = p_jump))
}
##
##HMC run
hmc_run <- function (starting_values, iter, epsilon_0, L_0, M) {
  # Get the number of rows and store in chains
  chains <- nrow (starting_values)
  # The number of parameters that you have in the starting values
  d <- ncol (starting_values)
  # Create space to store iterations of the parameters
  sims <- array (NA, c(iter, chains, d),
                 dimnames=list (NULL, NULL,
                 colnames (starting_values)))
  warmup <- 0.5*iter
  p_jump <- array (NA, c(iter, chains))
  for (j in 1:chains){
    param <- starting_values[j,]
    for (t in 1:iter){
      epsilon <- runif (1, 0, 2*epsilon_0)
      L <- ceiling (2*L_0*runif(1))
      temp <- hmc_iteration (param,x,y, epsilon,L,M)
      p_jump[t,j] <- temp$p_jump
      sims[t,j,] <- temp$param
      param <- temp$param
    } }
  monitor (sims, warmup)
  cat ("Avg␣acceptance␣probs:",
       fround(colMeans(p_jump[(warmup+1):iter,]),2),"\n")
  return (list (sims=sims, p_jump=p_jump))
}
```

Listing 4: R code contd.

```
##RUN it
parameter_names <- c (paste ("beta[",1:2,"]",sep=""))
d <- 2
chains <- 4



#
log_post_interim <- function(param){
  log_prior <- 0
  log_likelihood <- sum(dbinom(y, n,
  invlogit(param[1] + param[2]*x),log = T))
  return(log_prior + log_likelihood)
}
lat <- laplace(log_post_interim, c(0.5,-0.5))
M <- ginv(lat$var)
mass_vector <- c(0.005, 0.001)

#Starts
starts <- array (NA,c(chains,d),
           dimnames=list(NULL,parameter_names))
for (j in 1:chains){
  starts[j,1] <- rnorm (1,0,0.5)
  starts[j,2] <- rnorm (1,0,0.5)
}
fit_hmc1 <- hmc_run(starting_values = starts,
                    iter = 1000,
                    epsilon_0 = 0.0005,
                    L_0 = 10,
                    M = mass_vector)
```

Listing 5: R code contd.

```
###############
##Variational Bayes

N <- 3020
X <- matrix(c(rep(1, N), x), nrow = N, ncol = 2)

#Lower bound function
lower.bound2 <- function(param, X, stars, sigs, draws){
  ab <- t(X)\%*\%X\%*\%(param\%*\%t(param) + sigs)
  part1 <- sum(diag(ab))/2
  part2 <- t(param)\%*\%param + sum(diag(sigs))
  part2 <- part2*(beta.mat.prior[1,1])
  part2 <- part2/2
  + (1/2)*log(det(solve(beta.mat.prior)))
  + (length(param)/2)*log(2*pi)
  part3 <- t(stars)\%*\%stars
  part4 <- length(param)/2
  + (1/2)*log(det(sigs))
  + (length(param)/2)*log(2*pi)
  bounds <-  part1  +  part2 + part3/2 + part4
  parts <- c(-part1, -part2, part3/2, part4)
  bounds<- list(bounds, parts)
  names(bounds)<- c("bounds", "parts")
  return(bounds)
}
##we will use the lower-bound to monitor convergence,
##stopping the model when the lower-bound drops below 1e-8
func.reg<- function(X, draws){
  beta.mat.prior <<- diag(1/10, ncol(X))
  beta.VA <- matrix(NA, nrow=1000, ncol=ncol(X))
  ystars <- rep(0, nrow(X))
  for(j in 1:nrow(X)){
    ystars[j] <- ifelse(draws[j]==1,
                        rtnorm(1, mean=0.5, sd=1, lower=0, upper=Inf),
                        rtnorm(1, mean=-0.5, sd=1, lower=-Inf, upper=0))
  }
  ##we will store the progress of the lower bound on the model
  bounds<- c()
  zz <- 0
```

Listing 6: R code contd.

```
  ##this stores the parts of the lower bound
  parts<- matrix(NA, nrow=1000, ncol=4)
  j <- 0
  ##creating a while loop
  while(zz == 0){
    j <- j + 1
    ##updating the beta parameters
    beta.VA[j,] <- solve(t(X)\%*\%X + beta.mat.prior)
    \%*\%t(X)\%*\%ystars
    sigs <- solve(t(X)\%*\%X + beta.mat.prior)
    ##computing the inner product of the
    ##covariates current estimates of the coefficients
    stars <- X\%*\%beta.VA[j,]
    denom1 <- pnorm(-stars)
    num1 <- dnorm(-stars)
    ##now, computing the expected value for each
    ##individual's augmented data, given
    ##current estimates of the approximating
    ##distribution on the coefficients
    ystars[which(draws==0)] <- stars[draws==0] +
      -num1[which(draws==0)]/denom1[which(draws==0)]
    ystars[which(draws==1)] <- stars[draws==1] +
      num1[which(draws==1)]/(1 - denom1[which(draws==1)])
    ##calculating the lower bound
    trial <- lower.bound2(beta.VA[j,], X, ystars, sigs, draws)
    bounds[j] <- trial$bounds
    parts[j,] <- trial$parts
    if(j>1){
      ##monitor convergence
      ab <- abs(bounds[j] - bounds[j-1])
      if(ab < 1e-8){
        zz <- 1}
    }
  }
  ##the information to be returned, after convergence
  stuff<- list(bounds, beta.VA[j,], sigs)
  names(stuff)<- c("bound","param", "sigma")
  return(stuff)
  }
```

Listing 7: R code contd.

```
example.run<- func.reg(X, y)
example.run
betas <- rmvnorm(10^3,
                 mean = example.run$param,
                 sigma = example.run$sigma)
###############

par(mfrow = c(1, 3),
    mgp = c(2,1,0),
    mar = c(2,3,4,2))
plot(fit_hmc1$sims[,,1],fit_hmc1$sims[,,2], ylim = c(0.15,0.30),
     xlim = c(-0.3,-0.05), pch = 16, cex = 0.7,
     main = "HMC", xlab = "alpha", ylab = "beta")
plot(ext_stan$alpha, ext_stan$beta, pch = 16, cex = 0.7,
     main = "Stan",xlab = "alpha", ylab = "beta", ylim = c(0.15,0.30),
     xlim = c(-0.3,-0.05))
plot(betas[,1], betas[,2], pch = 16, cex = 0.7,
     main = "VI",xlab = "alpha", ylab = "beta",ylim = c(0.15,0.30),
     xlim = c(-0.3,-0.05))
#############
par(mfrow = c(1, 3))
plot(density(fit_hmc1$sims[,,1]),main = "HMC",
     xlab = "alpha", ylab = NA,
     bty = "n", yaxt = "n", xlim = c(-0.5, 0.1))
abline(v = mean(fit_hmc1$sims[,,1]), col = "red")
legend(-0.55,7, legend = c("posterior", "mean"),lty = c(1,1),
       col = c("black", "red"), bty = "n")
##
plot(density(ext_stan$alpha),main = "Stan",
     xlab = "alpha", ylab = NA,
     bty = "n", yaxt = "n", xlim = c(-0.5, 0.1))
abline(v = mean(fit_hmc1$sims[,,1]), col = "red")
legend(-0.55,7, legend = c("posterior", "mean"),lty = c(1,1),
       col = c("black", "red"), bty = "n")
#
plot(density(betas[,1]),main = "VI",
     xlab = "alpha", ylab = NA,
     bty = "n", yaxt = "n", xlim = c(-0.5, 0.1))
abline(v = mean(fit_hmc1$sims[,,1]), col = "red")
legend(-0.55,8.5, legend = c("posterior", "mean"),lty = c(1,1),
       col = c("black", "red"), bty = "n")
```

Listing 8: R code contd.

```
###################
par(mfrow = c(1, 3))
plot(density(fit_hmc1$sims[,,2]),main = "HMC",
     xlab = "beta", ylab = NA,
     bty = "n", yaxt = "n", xlim = c(0, 0.4))
abline(v = mean(fit_hmc1$sims[,,2]), col = "red")
legend(-0.05,7, legend = c("posterior", "mean"),lty = c(1,1),
       col = c("black", "red"), bty = "n")
##
plot(density(ext_stan$beta),main = "Stan",
     xlab = "beta", ylab = NA,
     bty = "n", yaxt = "n", xlim = c(0, 0.4))
abline(v = mean(fit_hmc1$sims[,,2]), col = "red")
legend(-0.05,7, legend = c("posterior", "mean"),lty = c(1,1),
       col = c("black", "red"), bty = "n")
#
plot(density(betas[,2]),main = "VI",
     xlab = "beta", ylab = NA,
     bty = "n", yaxt = "n", xlim = c(0, 0.4) )
abline(v = mean(fit_hmc1$sims[,,2]), col = "red")
legend(-0.05,8.5, legend = c("posterior", "mean"),lty = c(1,1),
       col = c("black", "red"), bty = "n")
```

2.2 Stan Code

Listing 9: Stan Code

```
data{
int N;
int y[N];
real x[N];
}
parameters{
  real alpha;
  real beta;
}
model{
  for(i in 1:N)
  y[i] ~ bernoulli(Phi(alpha + beta * x[i]));
// }
// generated quantities{
//   real y_rep[N];
//   for(i in 1:N)
//   y_rep[i] = bernoulli_rng(Phi(alpha + beta * x[i]));
}
```