Assignment 11.b for **STATGR6103**

submitted to Professor Andrew Gelman

Advait Rajagopal

23 November 2016

1 Question 1

Program HMC in R for the bioassay logistic regression example from Chapter 3.

1.1 Part A

Code the gradients analytically and numerically and check that the two programs give the same result.

It is important to write the full exposition of the model before computing the analytical and numerical gradients of the posterior distribution. 20 animals are divided 4 equally sized groups and each group is administered with a certain dosage, where the log of the dosage is denoted by x_i and the number of deaths recorded in that group are denoted by y_i. The number of animals, in each group out of which y_i die are denoted by n_i. If the outcomes of the five animals ($n_i = 5$) in each group are treated as exchangeable then y_i is binomially distributed as below;

$$y_i|\theta_i \sim \text{Bin}(n_i, \theta_i)$$

Where θ_i is the probability parameter of the binomial distribution that denotes the probability of death. We use a logistic transform of θ such that

$$\text{logit}(\theta_i) = \alpha + \beta x_i$$

Given the above information, the joint likelihood function for the data becomes;

$$p(y|\alpha, \beta, n, x) \propto \prod_{i=1}^{k} \text{logit}^{-1}(\alpha + \beta x_i)^{y_i}[1 - \text{logit}^{-1}(\alpha + \beta x_i)]^{n_i - y_i} \tag{1}$$

I assume noninformative priors on the parameters α and β which can be summarized as below;

$$p(\alpha, \beta) \propto 1 \tag{2}$$

From equations 1 and 2 we infer that the joint posterior density of the parameters α and β is given by the equation below,

$$\begin{aligned} p(\alpha,\beta|y,n,x) &\propto p(\alpha,\beta|n,x)p(y|\alpha,\beta,n,x) \\ &\propto p(\alpha,\beta)\prod_{i=1}^{k} p(y_i|\alpha,\beta,n_i,x_i) \\ &\propto \prod_{i=1}^{5} \text{logit}^{-1}(\alpha+\beta x_i)^{y_i}[1-\text{logit}^{-1}(\alpha+\beta x_i)]^{n_i-y_i} \end{aligned}$$

The log of the posterior then becomes;

$$\log(p(\alpha,\beta|y,n,x)) \propto \sum_{i=1}^{4} y_i \log(\text{logit}^{-1}(\alpha+\beta x_i)) + (n_i-y_i)\log([1-\text{logit}^{-1}(\alpha+\beta x_i)]) \tag{3}$$

The first derivatives[1] of equation 3 with respect to α and β are given below;

$$\frac{\partial \log(p(\alpha,\beta|y,n,x))}{\partial \alpha} = \sum_{i=1}^{4} y_i - \frac{n_i(e^{\alpha+\beta x_i})}{1+e^{\alpha+\beta x_i}} \tag{4}$$

$$\frac{\partial \log(p(\alpha,\beta|y,n,x))}{\partial \beta} = \sum_{i=1}^{4} x_i y_i - \frac{n_i x_i(e^{\alpha+\beta x_i})}{1+e^{\alpha+\beta x_i}} \tag{5}$$

Equations 4 and 5 represent the analytical gradient. The numerical gradient on the other hand is computed by second order divided differences using the following formula;

$$\frac{\Delta \log(p(\alpha,\beta|y,n,x))}{\Delta \alpha} = \frac{\log(p(\alpha+\epsilon,\beta|y,n,x)) - \log(p(\alpha-\epsilon,\beta|y,n,x))}{2\epsilon} \tag{6}$$

$$\frac{\Delta \log(p(\alpha,\beta|y,n,x))}{\Delta \beta} = \frac{\log(p(\alpha,\beta+\epsilon|y,n,x)) - \log(p(\alpha,\beta-\epsilon|y,n,x))}{2\epsilon} \tag{7}$$

Equations 6 and 7 represent the numerical gradient as computed by the second differences formula. I calculate the gradients in equations 4,5,6 and 7 a 1000 times for values of α and β generated by a uniform distribution and plot the histogram of the differences between equations 6 and 4 and 5 and 7 respectively, below. Intuitively I expect these differences to be equal to or very close to 0.

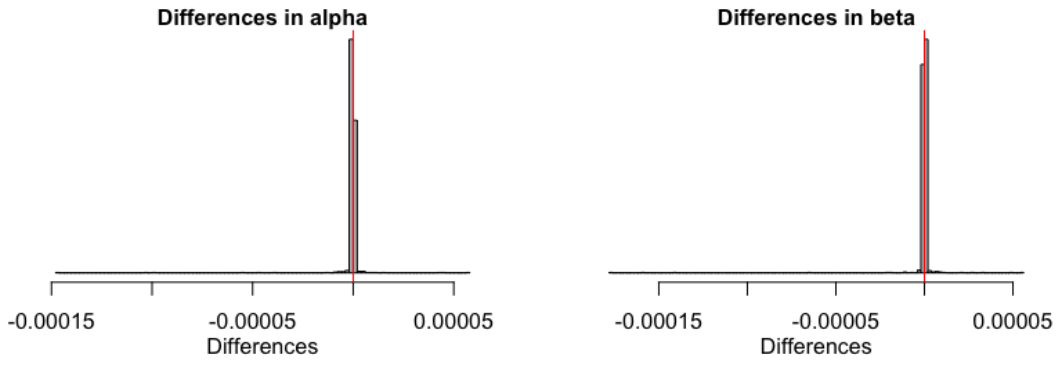

Figure 1: We observe that the histograms are centered at zero or around the red vertical line

It is clear from Figure 1 that there is a negligible difference between the gradients of α and β as computed by the analytical and numerical methods.

[1] See Mathematical Appendix.

1.2 Part B

Pick reasonable starting values for the mass matrix, step size, and number of steps.
The mass matrix is called M, the step size is denoted by ϵ and the number of steps is L. According to Hamiltonian dynamics I need a momentum distribution which is denoted by $p(\phi)$ where ϕ_j is the momentum variable and the parameter interest in the target density and ϕ get updated simultaneously. It is common to give ϕ a multivariate normal distribution with mean 0 and covariance equal to the diagonal mass matrix M. Therefore $\phi_j \sim \mathrm{N}(0, M_{jj})$ where $j = 1, .., d$ and d is the number of dimensions of the parameter of the target density. As advised by Gelman et. al (2013) I allow M to scale with the inverse covariance matrix of the posterior distribution. So I obtain the covariance matrix using the Nelder-Mead algorithm with the *laplace* function in the *LearnBayes* package. Then I compute the inverse of this covariance matrix of the posterior using the *MASS* package and the *ginv* function obtaining the Fisher information matrix. I set the mass matrix M equal to this value obtained, ignoring the off diagonal elements for simplicity.

$$M = \begin{bmatrix} 1.964 & 0 \\ 0 & 0.086 \end{bmatrix}$$

Again as recommended by Gelman et. al(2013) I set $\epsilon = 0.1$ and $L = 10$. I use starting values from the normal distribution with mean 0 and standard deviation 15. I also use 2000 iterations. The posterior estimates of the parameters are summarized in Table 1.

Table 1: Posterior Distributions of Parameters

	mean	se.mean	sd	2.5%	25%	50%	75%	97.5%	n.eff	Rhat
α	1.4	0.1	1.2	-0.5	0.6	1.3	2.0	4.0	367	1
β	11.9	0.3	6.2	3.4	7.4	10.9	15.3	27.2	315	1

The acceptance probability for each of the 4 chains with the (ϵ, L, M) specification described in this section is given in Table 2.

Table 2: Acceptance Rates

Chain 1	Chain 2	Chain 3	Chain 4
1	1	1	1

The algorithm therefore performs quite well.

1.3 Part C

Tune the algorithm to an approximate 65% acceptance rate.
I use the same mass vector M but tweak $\epsilon = 1$ and $L = 3$ run for 50 iterations to give an approximate

65% acceptance rate[2]. Posterior estimates from this configuration are given in Table 3 and the acceptance probability of the chains is given in Table 4.

Table 3: Posterior Distributions of Parameters

	mean	se.mean	sd	2.5%	25%	50%	75%	97.5%	n.eff	Rhat
α	1.3	0.2	1.2	-0.6	0.4	1.2	1.9	3.6	26	1.2
β	12.6	1.2	6.4	2.4	7.1	11.9	16.0	26.2	27	1.1

Table 4: Acceptance Rates

Chain 1	Chain 2	Chain 3	Chain 4
0.75	0.57	0.65	0.67

The average acceptance probability of all 4 chains is 66%.

1.4 Part D

Run 4 chains long enough so that each has an effective sample size of at least 100. How many iterations did you need?

I use the (ϵ, L, M) used in section 1.2. I run 4 chains with 500 iterations and get an effective sample size of 114 and 103 for each parameter. The results of this are summarized in Table 5.

Table 5: Posterior Distributions of Parameters

	mean	se.mean	sd	2.5%	25%	50%	75%	97.5%	n.eff	Rhat
α	1.4	0.1	1.1	-0.5	0.6	1.3	2.0	4.2	114	1
β	11.9	0.5	5.5	3.5	7.8	10.8	15.2	24.5	103	1

1.5 Part E

Check that your inferences are consistent with those from the direct approach in Chapter 3.

The direct approach used in Chapter 3 involves drawing 1000 draws of α and β from a grid at which the posterior density is computed. The direct approach and the HMC are compared in Figure 2. I use the model specified in Table 5 from section 1.4.

[2]See code attached.

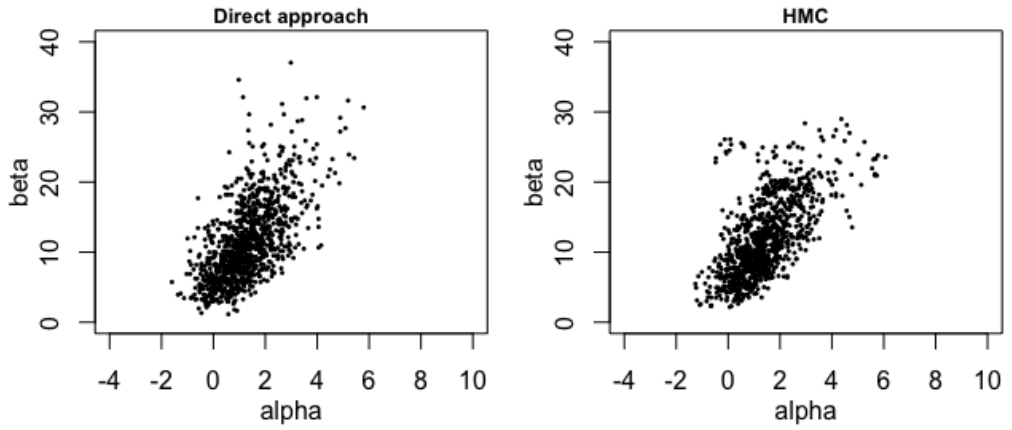

Figure 2: (a) Shows the direct approach through grid approximation and (b) shows draws from the joint posterior estimated by HMC.

The two approaches seem quite similar. I also compare the LD50 which is the probability of death that is 50%. We draw α and β from the estimated posterior distribution and calculate LD50 which is the dose level at which probability of death is 50%. Thus for us a 50% survival rate corresponds to;

$$LD50 : E(\frac{y_i}{n_i}) = \text{logit}^{-1}(\alpha + \beta x_i) = 0.5$$

simplifying this yields LD50 $= -\alpha/\beta$. This calculation is done for the values drawn from the posterior distribution coming from the HMC algorithm as well as the direct approach and the corresponding histograms are shown in Figure 3.

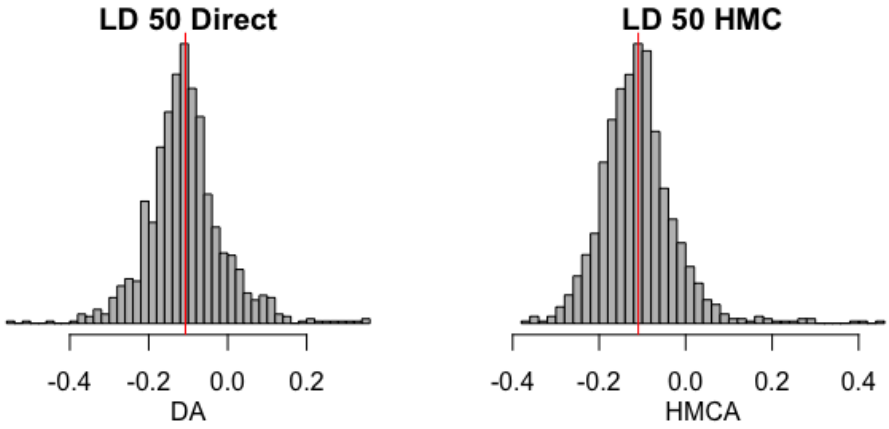

Figure 3: (a) Histogram of LD50 with the direct approach and (b) Histogram of LD50 calculated from HMC.

The two approaches therefore appear to yield very similar results and thus they are consistent with inferences from Chapter 3.

2 Mathematical Appendix

Derivation of the partial differentials of the posterior density :

Consider equation 3 which shows the log posterior density;

$$\begin{aligned}
\log(p(\alpha,\beta|y,n,x)) &\propto \sum_{i=1}^{4} y_i \log(\text{logit}^{-1}(\alpha+\beta x_i)) + (n_i - y_i)\log([1-\text{logit}^{-1}(\alpha+\beta x_i)]) \\
&= \sum_{i=1}^{4} y_i \log\Big(\frac{e^{\alpha+\beta x_i}}{1+e^{\alpha+\beta x_i}}\Big) + n_i \log\Big(1-\frac{e^{\alpha+\beta x_i}}{1+e^{\alpha+\beta x_i}}\Big) - y_i\Big(1-\frac{e^{\alpha+\beta x_i}}{1+e^{\alpha+\beta x_i}}\Big) \\
&= \sum_{i=1}^{4} y_i[\alpha+\beta x_i - \log(1+e^{\alpha+\beta x_i})] - n_i \log(1+e^{\alpha+\beta x_i}) + y_i \log(1+e^{\alpha+\beta x_i}) \\
&= \sum_{i=1}^{4} \alpha y_i + \beta x_i y_i - n_i \log(1+e^{\alpha+\beta x_i})
\end{aligned}$$

Now I analytically compute the partial derivatives of the simplified log posterior density;

$$\frac{\partial \log(p(\alpha,\beta|y,n,x))}{\partial \alpha} = \sum_{i=1}^{4} y_i - \frac{n_i(e^{\alpha+\beta x_i})}{1+e^{\alpha+\beta x_i}}$$

$$\frac{\partial \log(p(\alpha,\beta|y,n,x))}{\partial \beta} = \sum_{i=1}^{4} x_i y_i - \frac{n_i x_i(e^{\alpha+\beta x_i})}{1+e^{\alpha+\beta x_i}}$$

3 Code

3.1 R Code

Listing 1: R Code

```
rm(list = ls())
setwd("/Users/Advait/Desktop/New␣School/Fall16/BDA/Class21")
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
library("arm")
install.packages("LearnBayes")
library("LearnBayes")
install.packages("MASS")
library("MASS")
#####################
x <- c(-0.86, -0.3, -0.05, 0.73)
n <- rep(5,4)
y <- c(0,1,3,5)
##1Posterior
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, invlogit(param[1]
  + param[2]*x),log = T))
  return(log_prior + log_likelihood)
}
##PART A
#Analytical gradient
gradient_an <- function(param, x,y){
  d_alpha <- sum(y - n*exp(param[1] + param[2]*x)
  /(1 + exp(param[1] + param[2]*x)))
  d_beta <- sum(y*x - n*x*exp(param[1] + param[2]*x)
  /(1 + exp(param[1] + param[2]*x)))
  return(c(d_alpha,d_beta))
}
#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
```

Listing 2: R code contd.

```
    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)
}
#Histogram
checking <-matrix(NA ,ncol=2,nrow =1000)
for(i in 1:1000){
  random_unif <- runif(2, -10,10)
  checking[i,]<-gradient_num(random_unif ,x,y)
  -gradient_an(random_unif ,x,y)
}
options(scipen =999)
checking <- checking[complete.cases(checking),]
as.matrix(checking)
par(mfrow = c(1, 2),
    mar = c(3, 3, 1, 1),
    oma = c(.5, .5, .5, .5),
    mgp = c(2,1,0))
hist(checking[,1], breaks = 100, col = "gray",
     main = "Differences in alpha",
     xlab = "Differences",
     yaxt = "n",
     ylab = NA,
     cex.main = 1)
abline(v = 0, col = "red")
#
hist(checking[,2], breaks = 100, col = "gray",
     main = "Differences in beta",
     xlab = "Differences",
     yaxt = "n",
     ylab = NA,
     cex.main = 1)
abline(v = 0, col = "red")
############################################################
##PART B
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)
}
```

Listing 3: R code contd.

```
lat <- laplace(log_post_interim, c(1,15))
M <- ginv(lat$var)
#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)
  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)
```

Listing 4: R code contd.

```
    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))
}

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

#
mass_vector <- c(1.9648937, 0.08595404)

#Starts
starts <- array (NA,c(chains,d),dimnames=list(NULL,parameter_names))
for (j in 1:chains){
  starts[j,1] <- rnorm (1,0,15)
  starts[j,2] <- rnorm (1,0,15)
}
###PART C
#65\% accuracy
M2 <- hmc_run (starting_values=starts, iter=100,
                 epsilon_0=1, L_0=3, M=mass_vector)
##PART D
#100 n_eff
M1 <- hmc_run (starting_values=starts, iter=500,
               epsilon_0=.1, L_0=10, M=mass_vector)
#PART E
grid_sim <- function(alpha, beta){
  x <- x
  y <- y
  n <- n
  grid_length = length(alpha)
  grid = matrix(NA, grid_length, grid_length)
```

Listing 5: R code contd.

```
  for(i in 1:grid_length){
    for(j in 1:grid_length){
      grid[i,j] <- exp(log_post(c(alpha[i], beta[j]), x, y))
    }
  }
  return(grid/sum(grid)) #normalizing
}

sampling <- function(S=1000, grid, alpha, beta){
  marginal_alpha <- apply(grid,1,sum)
  cdf_alpha <- cumsum(marginal_alpha)
  alpha_sim = rep(0,S)
  beta_sim = rep(0,S)
  for(s in 1:S){
    random_unif <- runif(1,0,1)
    f.alpha <- max(cdf_alpha[cdf_alpha <= random_unif])
    alpha_sim[s] <- alpha[cdf_alpha == f.alpha]
    beta_p <- length(alpha[alpha <= alpha_sim[s]])
    grid[beta_p, ] <- grid[beta_p,]/sum(grid[beta_p,])
    cdf_beta_con <- cumsum(grid[beta_p,])
    random_unif<-runif(1,0,1)
    f.beta <- max(cdf_beta_con[cdf_beta_con <= random_unif ])
    beta_sim[s] <- beta[cdf_beta_con == f.beta] }
  return(list(alpha_sim = alpha_sim,
              beta_sim = beta_sim)) }

alpha_dir <- seq(-2,8,length.out = 300)
beta_dir <- seq(-10,39,length.out = 300)

grid <- grid_sim(alpha_dir, beta_dir)
draws <- sampling(S=1000, grid, alpha_dir, beta_dir)
alpha_draws<-draws[[1]]
beta_draws<-draws[[2]]
mean(alpha_draws)
mean(beta_draws)
sd(alpha_draws)
sd(beta_draws)
```

Listing 6: R code contd.

```
###
par(mfcol = c(1,2), mar = c(3, 3, 1, 1),
    oma = c(.5, .5, .5, .5),
    mgp = c(2,1,0))
#alpha grid scatter
plot(alpha_draws, beta_draws, pch = 16, cex = 0.4,
main = "Direct␣approach", cex.main = 0.8,
xlim = c(-4,10), ylim = c(0,40),
     xlab = "alpha", ylab = "beta")
plot(M1$sims[250:500,,1], M1$sims[250:500,,2],
pch = 16, cex = 0.4, main = "HMC", cex.main = 0.8,
xlim = c(-4,10), ylim = c(0,40),
     xlab = "alpha", ylab = "beta")
plot(density(alpha_draws))
abline(v = mean(alpha_draws))
plot(density(M1$sims[,,1]))
str(M1$sims)
###
#LD50
DA <- -(alpha_draws/beta_draws)
hist(DA, main = "LD␣50␣Direct",
     ylab = NA, yaxt = "n", breaks = 50,
     col = "gray")
abline(v = mean(DA), col = "red")

HMCA <- -(M1$sims[250:500,,1]/M1$sims[250:500,,2])
hist(HMCA, main = "LD␣50␣HMC",
     ylab = NA, yaxt = "n", breaks = 50,
     col = "gray")
abline(v = mean(HMCA), col = "red")
```