HW6

(a) Assuming a noninformative uniform prior on the parameters in question (α, β), the posterior is $\propto p(y_i|\alpha, \beta, n_i, x_i) \propto [\text{logit}^{-1}(\alpha + \beta x_i)]^{y_i}[1 - \text{logit}^{-1}(\alpha + \beta x_i)]^{n_i - y_i}$. For the purposes of HMC, we'll use the log posterior, which is given by $y_i((\alpha + \beta x_i) - \log(1 + e^{\alpha+\beta x_i})) - (n_i - y_i)\log(1 + e^{\alpha+\beta x_i}) = y_i(\alpha + \beta x_i) - n_i \log(1 + e^{\alpha+\beta x_i})$. In code:

```
inv.logit <- function(x){
  exp(x)/(1+exp(x))
}

log_p_th <- function(th, y, n, x){
  alpha <- th[1]
  beta <- th[2]
  log_prior <- 0
  log_likelihood <- sum(log((inv.logit(alpha + beta * x) ^ y)
                            * (1 - inv.logit(alpha + beta * x)) ^ (n-y)))
  return (log_likelihood + log_prior)
}
```

We can now get the gradients $\frac{d \log p(\alpha,\beta|y_i,n_i,x_i)}{d\alpha} = y_i - n_i \frac{e^{\alpha+\beta x_i}}{1+e^{\alpha+\beta x_i}}$ and $\frac{d \log p(\alpha,\beta|y_i,n_i,x_i)}{d\beta} = y_i x_i - n_i x_i \frac{e^{\alpha+\beta x_i}}{1+e^{\alpha+\beta x_i}} = x_i(y_i - n_i \frac{e^{\alpha+\beta x_i}}{1+e^{\alpha+\beta x_i}}$. In code:

```
gradient_th <- function(th, y, n, x){
  alpha <- th[1]
  beta <- th[2]
  d_alpha <- sum(y - n * inv.logit(alpha + beta * x))
  d_beta <- sum(x * (y - n * inv.logit(alpha + beta * x)))
  return (c(d_alpha, d_beta))
}
```

Now we want to check the gradient numerically:

```
gradient_th_numerical <- function(th, y, n, x){
  d <- length(th)
  e <- .0001
  diff <- rep(NA, d)
  for (k in 1:d){
    th_hi <- th
    th_lo <- th
    th_hi[k] <- th[k] + e
    th_lo[k] <- th[k] - e
    diff[k] <- (log_p_th(th_hi, y, n, x) - log_p_th(th_lo, y, n, x))/(2 * e)
  }
  return (diff)
}

setwd("~/Documents/BDA/Homework 6")
bioassay <- read.table("bioassay_data.txt", header=TRUE)
x <- bioassay$x
y <- bioassay$y
```

```
n <- bioassay$n
gradient_th(c(1, 1), y, n, x)
```

```
[1] -4.868292  3.582460
```

```
gradient_th_numerical(c(1, 1), y, n, x)
```

```
[1] -4.868292  3.582460
```

HMC Code:

```
library(rstan)
fround <- function (x, digits) {
  format (round (x, digits), nsmall=digits)
}

hmc_iteration <- function(th, y, n, x, epsilon, L, M) {
  M_inv <- 1/M
  d <- length(th)
  phi <- rnorm (d, 0, sqrt(M))
  th_old <- th
  log_p_old <- log_p_th (th, y, n, x) - 0.5 * sum(M_inv * phi ^ 2)
  phi <- phi + 0.5 * epsilon * gradient_th(th, y, n, x)
  for (l in 1:L) {
    th <- th + epsilon * M_inv * phi
    phi <- phi + (if (l == L) 0.5 else 1) * epsilon * gradient_th(th, y, n, x)
  }
  phi <- -phi
  log_p_star <- log_p_th(th, y, n, x) - 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)
  th_new <- if (runif(1) < p_jump) th else th_old
  return (list (th = th_new, p_jump = p_jump))
}

hmc_run <- function(starting_values, iter, epsilon_0, L_0, M) {
  chains <- nrow(starting_values)
  d <- ncol (starting_values)
  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) {
    th <- 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(th, y, n, x, epsilon, L, M)
      p_jump[t, j] <- temp$p_jump
      sims[t, j, ] <- temp$th
      th <- temp$th
    }
```

```
  }
  print(cat("Avg acceptance probs:", fround(colMeans(p_jump[(warmup + 1):iter,]), 2), "\n"))
  return(monitor(sims, warmup, print=FALSE))
}
```

We know from fitting the model in Homework 2 that the standard deviation of alpha is around 1 and the standard deviation around beta is around 5. We want to make M roughly scale with the inverse of the covariance matrix. However, making it equal to the covariance matrix turned out to be too conservative. Instead I set it to 1/3 * covariance matrix:

```
mass_vector <- c(1 / (3 ^ 2), 1 / (15 ^ 2))
```

Then we set the other parameter values to the recommended values and run the sampler. After playing around with the parameters it seems like you need 250 iterations to get 100 effective samples and $\epsilon = .2$ and $L = 5$ to get the optimal acceptance probabilities:

```
parameter_names <- c("alpha", "beta")
d <- length(parameter_names)
chains <- 4
starts <- array(NA, c(chains, d), dimnames = list(NULL, parameter_names))
for (j in 1:chains) {
  starts[j,] <- rnorm(d, 0, 15)
}

M1 <- hmc_run(starting_values = starts, iter = 250, epsilon_0 = .2, L_0 = 5, M = mass_vector)
```

```
Avg acceptance probs: 0.69 0.66 0.68 0.62
NULL
```

```
library(xtable)
m1table <- xtable(as.data.frame(M1))
print(m1table, type="latex")
```

% latex table generated in R 3.1.1 by xtable 1.7-4 package % Fri Oct 23 08:41:08 2015

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
alpha	1.39	0.10	1.15	-0.41	0.64	1.26	2.06	3.81	145.60	1.01
beta	12.16	0.66	6.06	3.37	7.99	11.13	15.12	27.55	84.63	1.05