# Assignment 10.a for **STATGR6103**

submitted to Professor Andrew Gelman

Advait Rajagopal

13 November 2016

## 1 Question 1

Consider the model, $y_j \sim \text{Binomial}(n_j, \theta_j)$, where $\theta_j = \text{logit}^{-1}(\alpha + \beta x_j)$, for $j = 1, ..., J$, and with independent prior distributions, $\alpha \sim t_4(0, 2^2)$ and $\beta \sim t_4(0, 1^2)$. Suppose J = 10, the $x_j$ values are randomly drawn from a U(0, 1) distribution, and $n_j \sim \text{Poisson}^+(5)$, where $\text{Poisson}^+$ is the Poisson distribution restricted to positive values.

### 1.1 Part A

**Sample a dataset at random from the model.**
We are given the following information in the question and this helps to build the model so we can sample from it.

$$n_j \sim \text{Poisson}^+(5) \tag{1}$$
$$x_j \sim \text{U}(0, 1) \tag{2}$$
$$\alpha \sim t_4(0, 2^2) \tag{3}$$
$$\beta \sim t_4(0, 1^2) \tag{4}$$
$$\theta_j = \text{logit}^{-1}(\alpha + \beta x_j) \tag{5}$$
$$y_j \sim \text{Binomial}(n_j, \theta_j) \tag{6}$$

I generate one $\alpha$ and one $\beta$ from their respective prior distributions and the values I get are $\alpha$ = -0.2908 and $\beta$ = -0.6904. Using these values I generate $n$ and $y$ using the distributions described above. The resultant values are;

$$n = \{5, 3, 7, 6, 4, 6, 4, 9, 7, 6\}$$
$$y = \{2, 0, 3, 2, 1, 3, 3, 2, 3, 2\}$$

### 1.2 Part B

**Use rejection sampling to get 1000 independent posterior draws from** $(\alpha, \beta)$**.**
It is useful to write out the posterior distribution of $(\alpha, \beta)$. The independent prior distributions of $\alpha$ and $\beta$ are given by equations 3 and 4.

$$\begin{aligned} p(\alpha, \beta | y, n, x) &\propto p(\alpha, \beta) p(y | n, x, \alpha, \beta) \\ &= p(\alpha) p(\beta) p(y | n, x, \alpha, \beta) \end{aligned}$$

$$p(\alpha) = \frac{\Gamma(\frac{5}{2})}{\Gamma(2)2\sqrt{4\pi}}\left(1 + \frac{1}{4}\left(\frac{\alpha}{2}\right)^2\right)^{-\frac{5}{2}}$$
$$p(\beta) = \frac{\Gamma(\frac{5}{2})}{\Gamma(2)2\sqrt{4\pi}}\left(1 + \frac{\beta^2}{4}\right)^{-\frac{5}{2}}$$

The likelihood function is binomial with the probability parameter $\theta_j$ as described in equation 5. Therefore the joint likelihood is given by;

$$\begin{aligned} p(y|n, x, \alpha, \beta) &\propto \prod_{j=1}^{10} (\theta_j)^{y_j}(1-\theta_j)^{n_j - y_j} \\ &= \prod_{j=1}^{10} (\text{logit}^{-1}(\alpha + \beta x_j))^{y_j}(1 - \text{logit}^{-1}(\alpha + \beta x_j))^{n_j - y_j} \end{aligned}$$

We can compute the log posterior density[1] by adding the logs of the prior and the likelihood.

$$\log(p(\alpha, \beta|y, n, x)) \propto \log(p(\alpha)) + \log(p(\beta)) + \log(p(y|n, x, \alpha, \beta))$$
$$\begin{aligned} \log(p(\alpha, \beta|y, n, x) = &-2.5\log\left(1 + \frac{1}{4}\left(\frac{\alpha}{2}\right)^2\right) - 2.5\log\left(1 + \frac{\beta^2}{4}\right) \\ &+ \sum_{j=1}^{10} y_j \log(\text{logit}^{-1}(\alpha + \beta x_j)) + (n_j - y_j)\log(1 - \text{logit}^{-1}(\alpha + \beta x_j)) + C \end{aligned}$$

In order to perform rejection sampling we need to define a proposal distribution that is ideally proportional to the posterior and make sure that the ratio of the posterior density to the proposal is bounded by some constant $M$. The rejection sampling algorithm proceeds in two steps;

1. Sample $\alpha$ and $\beta$ at random from the distribution proportional to $g$.
2. With probability $\frac{p(\alpha,\beta|y,n,x)}{Mg(\alpha,\beta)}$ , accept $\alpha$ and $\beta$ as a draw from p. If the drawn $\alpha$ and $\beta$ are rejected return to step 1.

For this case I use the prior distribution of $(\alpha, \beta)$ as the proposal distribution and with some simplification I infer that $M$ is the binomial constant because the joint likelihood function $p(y|n, x, \alpha, \beta)$ has to be bounded by the binomial constant since the part that is a function of $\theta_j$ is actually less than or equal to 1. Figure one shows a histogram of 1000 draws from the posterior and the red line shows the true distribution. It is clear that the rejection sampling algorithm is able to capture the marginal posterior densities fairly well.

[1] I use the log posterior for computational convenience. For the rejection sampling I use $\exp(\log(p(\alpha, \beta|y, n, x))$.

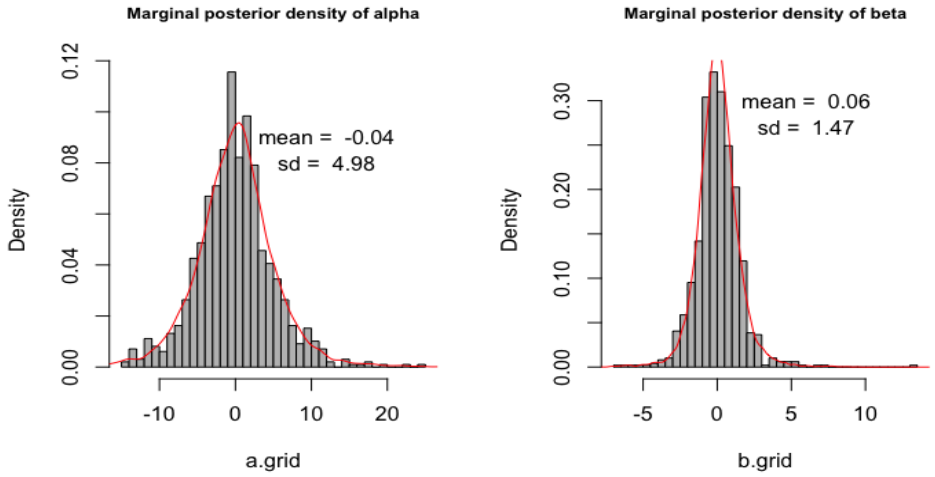


Figure 1: The histogram shows the 1000 draws of $\alpha$ and $\beta$ while the red line is the true $t$ distribution with scale parameters 4 and 1 respectively.

## 1.3 Part C

**Approximate the posterior density for $(\alpha, \beta)$ by a normal centered at the posterior mode with covariance matrix fit to the curvature at the mode.**

This task is accomplished by approximating the posterior density $p(\alpha, \beta|y, n, x)$ by using a normal distribution as follows;

$$p(\alpha, \beta|y, n, x) \approx \mathcal{N}((\hat{\alpha}, \hat{\beta}), [I(\hat{\alpha}, \hat{\beta})]^{-1})$$

Where $(\hat{\alpha}, \hat{\beta})$ is the posterior mode and $[I(\hat{\alpha}, \hat{\beta})]$ is the observed information corresponding to the second order derivative of the log of the posterior density evaluated at the mode and this the 'curvature' at the mode.

In order to compute the posterior mode, I use the *LearnBayes* package which uses a Nelder-Mead method to compute the mode and covariance matrix. The resulting joint mode and covariance matrix are given by;

$$(\hat{\alpha}, \hat{\beta}) = (0.151, 0.195)$$

$$I((\hat{\alpha}, \hat{\beta}))^{-1} = \begin{bmatrix} 0.0435 & -0.0192 \\ -0.0192 & 0.0667 \end{bmatrix}$$

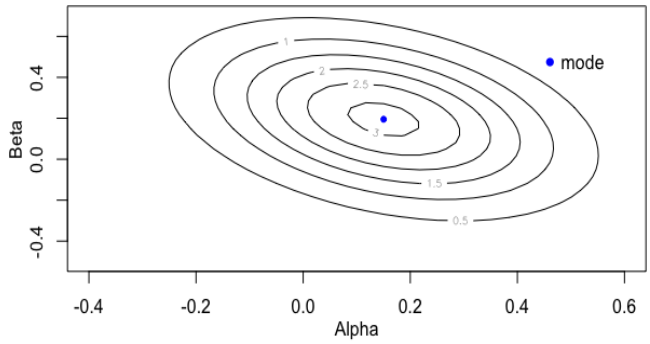


Figure 2: Contour plot displaying the normal approximation of the posterior density with the modal value shown in blue.

## 1.4 Part D

**Take 1000 draws from the two-dimensional $t_4$ distribution with that center and scale matrix and use importance sampling to estimate $\mathbf{E}(\alpha|y)$ and $\mathbf{E}(\beta|y)$.**

The output of the *LearnBayes* package clearly gives a location and scale matrix for the normal approximation of a posterior density. We use this (explicitly mentioned in section 1.3) to draw 1000 samples $(\alpha^S, \beta^S)$ from a two dimensional $t$ distribution with 4 degrees of freedom and the center and scale mentioned in section 1.3. I further sample $(\alpha, \beta)$ from it with the probability of sampling it equal to the weight function given below;

$$w(\alpha, \beta) = \frac{p(\alpha, \beta|y, n, x)}{g(\alpha, \beta)}$$

Then, I multiply the sampled $\alpha^S$ by the weights $w(\theta^S)$ and divide it by the sum of all weighing values $\sum_{s=1}^{S} w(\theta^S)$ (in this case 1000 values, since S = 1000). I repeat this 1000 times without replacement and divide the sum of $\alpha^S w(\theta^S)$ by 1000 to get $E(\alpha|y)$. I repeat the process for $E(\beta|y)$ to obtain the following values for the expectations;

$$E(\alpha|y, n, x) = 0.1344$$
$$E(\beta|y, n, x) = 0.1874$$

# 2 Code

## 2.1 R Code

**Listing 1: R Code**

```
rm(list = ls())
install.packages("arm", dependencies = T)
install.packages("numDeriv", dependencies = T)
library("arm")
library(LearnBayes)
library(mvtnorm)
library(numDeriv)
library(stats)
set.seed(8)

N <- 10
rtpois <- function(N1, lambda){
        qpois(runif(N1, dpois(0, lambda), 1), lambda)
}
n <- rtpois(N, 5)

alpha <- 4*rt(1,4)
beta <- rt(1, 4)
x <- runif(N,0,1)
theta <- invlogit(alpha + beta*x)
#Part A
y <- rbinom(N, n,theta)

#Part B
M <- choose(n,y)
M <- prod(M)
ab.mat <- NULL
ab.mat <- data.frame(
  a.grid = 4*rt(10^4, 4),
  b.grid = rt(10^4, 4)
)
str(ab.mat)
prior <- function(ab){
  exp(dt(ab[1],4, log = T) + dt(ab[2],4,log = T) + log(4))
}
```

**Listing 2: R code contd.**

```
lik <- function(ab){
  exp(sum(y*log(invlogit(ab[1] + ab[2]*x))
      + (n - y)*log(1 - invlogit(ab[1] + ab[2]*x))))
}
post <- function(ab){
  prior(ab)*lik(ab)
}
accept <- function(ab){
  ifelse(
   10^30 post(ab)/(prior(ab)) >= M,
    1, 0
  )
}

attempt <- apply(cbind(ab.mat$a.grid, ab.mat$b.grid), 1, accept)
summary(attempt)
ab.mat$attempt <- attempt
thing1 <- sample(ab.mat$a.grid[ab.mat$attempt==1],1000 )
mean(thing1, na.rm = T)
thing2 <- sample(ab.mat$b.grid[ab.mat$attempt==1],1000 )
mean(thing2, na.rm = 2)

par(mfcol = c(1,2))
hist(thing1, breaks = 50, freq = F,
     main = "Marginal␣posterior␣density␣of␣alpha",
     cex.main = 0.8, xlab = "a.grid", col = "gray")
lines(density(4*rt(10^4, 4)), col = "red")
text(x = 12, y = 0.09,
     labels = paste("mean␣=␣",
                    round(mean(mean(thing1, na.rm = T)),2)))
text(x = 12, y = 0.08,
     labels = paste("sd␣=␣",
                    round(sd(thing1, na.rm = T),2)))

hist(sample(ab.mat$b.grid[ab.mat$attempt==1],1000),
breaks = 50, freq = F,
     main = "Marginal␣posterior␣density␣of␣beta",
     cex.main = 0.8, xlab = "b.grid", col = "gray")
lines(density(rt(10^4, 4)), col = "red")
```

**Listing 3: R code contd.**

```
text(x = 6, y = 0.3,
     labels = paste("mean = ",
                    round(mean(mean(thing2, na.rm = T)),2)))
text(x = 6, y = 0.27,
     labels = paste("sd = ",
                    round(sd(thing2, na.rm = T),2)))
######
##Part C
log.post <- function(ab){
  sum(y*log(invlogit(ab[1] + ab[2]*x))
      + (n - y)*log(1- invlogit(ab[1] + ab[2]*x))
      + log(dt(ab[1], 4))
      + log(4)
      + log(dt(ab[2], 4)))
}

la_t <- laplace(log.post, c(0, -1))
alpha.sim <- seq(-5, 5, length.out = 300)
beta.sim <- seq(-5, 5, length.out = 300)
la <- length(alpha.sim)
lb <- length(beta.sim)
mode1 <- c(la_t$mode[1], la_t$mode[2])
sigma <- la_t$var
approx <- matrix(0, la, lb)

for(i in 1:la){
  for (j in 1:lb){
    approx[i,j] <- dmvnorm(c(alpha.sim[i],beta.sim[j]), mode1, sigma)
  }
}

#contour
par(mfrow = c(1, 1),
    mar = c(3, 3, 1, 1),
    oma = c(.5, .5, .5, .5),
    mgp=c(2,1,0))
contour(alpha.sim, beta.sim, approx, nlevels = 5,
        xlim=c(-0.4, 0.6),
        ylim=c(-0.5, .7),
        xlab="Alpha",
        ylab="Beta",
        main=NULL)
```

**Listing 4: R code contd.**

```
points(la_t$mode[1],
       la_t$mode[2],
       pch=20,
       cex=1,
       col="blue")
legend(0.44, 0.55,
       legend=c("mode"),
       col=c("blue"),
       cex=1,
       pch=c(16),
       bty="n")
##
#Part D
#Sample from the multi-t
draws <- rmvt(1000,
              delta = mode1,
              sigma = sigma,
              df = 4)
alphas <- draws[,1]
betas <- draws[,2]

#Importance sampling algorithm
nsims <- 1000
post.dens <- NULL
for(i in 1:nsims){
  post.dens[i] <- exp(log.post(c(alphas[i], betas[i])))
}
g <- NULL
for(i in 1:nsims){
  g[i] <- dmvt(c(alphas[i], betas[i]),
               delta = mode1,
               sigma = sigma,
               df = 4,
               log = FALSE)
}
e_alpha <- sum(alphas*(post.dens/sum(post.dens))/
                 (g/sum(g)))/nsims
e_alpha
e_beta <- sum(betas*(post.dens/sum(post.dens))/
                (g/sum(g)))/nsims
e_beta
```