Homework 4
Bayesian Statistical Methods

Michael Cotner
Department of Biomedical Engineering
The University of Texas at Austin
Email: mcotner@utexas.edu

Akshay Umashankar
Department of Educational Psychology
The University of Texas at Austin
Email: akshayumashankar38@utexas.edu

March 23, 2023

Introduction

The data in 'mosquitofish.txt' represent the number of surviving mosquitofish y_i (for i = 1, . . . , n) from n = 32 groups out of N_i initial fish in each group. For this homework assignment, we seek to construct a Binomial regression model to learn about covariates that may predict survival probability (θ_i). These covariates are standardized final mean length of surviving fish in group (`L1`), standardized duration in days of study (`d`), and also the product of the standardized L1 and d.

1 Using JAGS, develop the R code to fit the following Bayesian binomial regression model:

The model is as follows:

$$y_i \sim \text{Binom}(N_i, \theta_i)$$

$$\text{probit}(\theta_i) = \boldsymbol{x_i'}\boldsymbol{\beta}$$

$$\boldsymbol{\beta} \sim \text{N}(\boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_\beta)$$

The full JAGS implementation can be found in the code block in the appendix, but the relevant JAGS snippet is:

```
mosquito_jags <-"
    model{
        for(i in 1:n){
            y[i] ~ dbin (theta[i] , N[i])
            probit(theta[i]) = b0 + b1*X[i,2] + b2*X[i,3] + b3*(X[i,2]*X[i,3])
        }
        b0 ~ dnorm(mu, tau)
        b1 ~ dnorm(mu, tau)
        b2 ~ dnorm(mu, tau)
        b3 ~ dnorm(mu, tau)
    }
"
```

2 Use JAGS to fit the binomial regression model above to the mosquitofish data set using y =N1 as the response variable, N=N0, and the following combinations of the three covariates: (a) intercept, standardized L1, standardized d, and a product of the two covariates (as an interaction term) (b) intercept, standardized L1, standardized d (c) intercept, standardized L1 (d) intercept, standardized d (e) intercept only

Each implementation of the JAGS model ran successfully with good mixing and convergence to posteriors. The trace plots for each condition are shown in Figure 1.

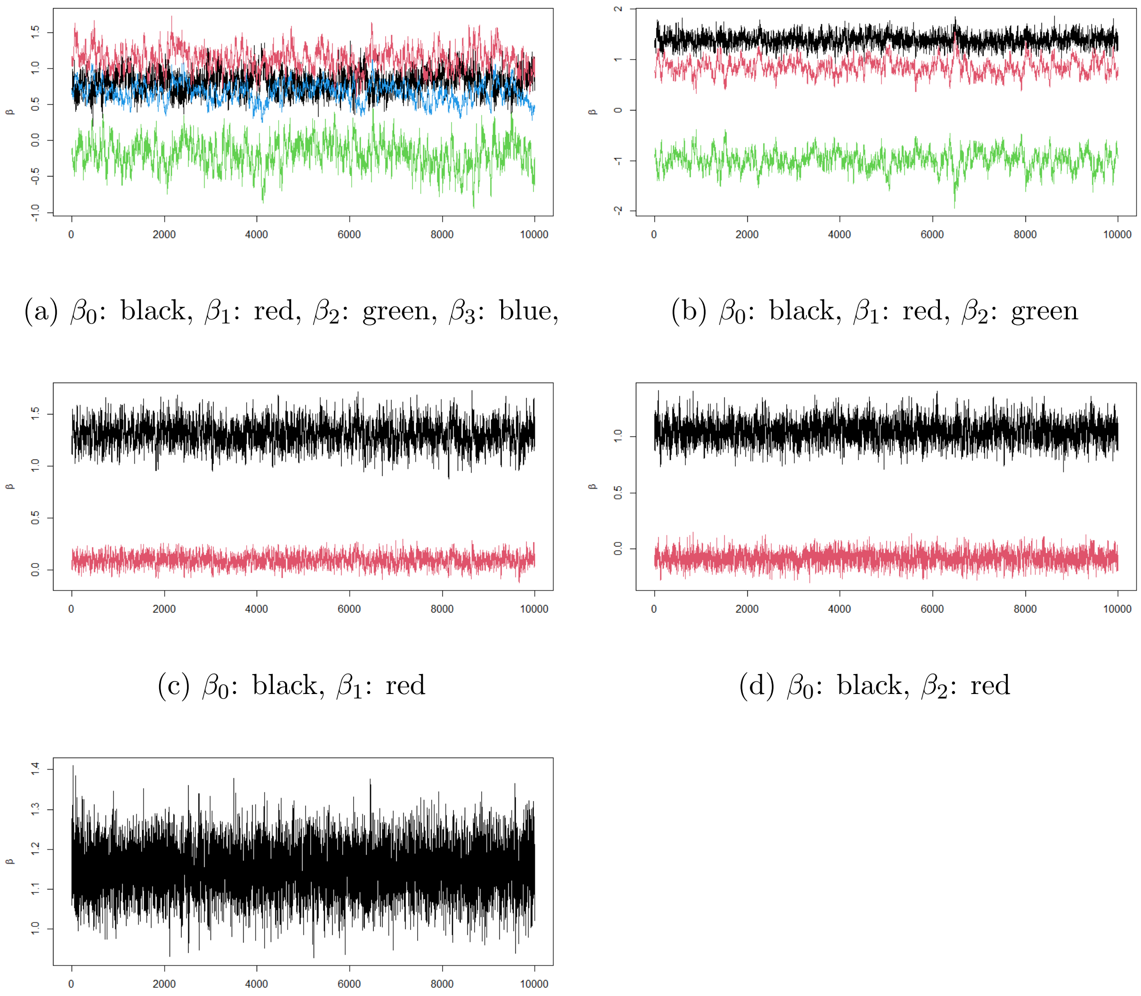
A table of the posterior parameters and their DIC values for each part are shown in Table 1

Model	DIC	β_0	β_1	β_2	β_3
a	77.448	1.351	-0.420	-2.423	2.687
b	95.881	1.387	0.847	-0.983	
c	134.916	1.310	0.095		
d	134.944	1.049		-0.085	
e	138.830	1.151			

Table 1: Results of DIC and regression condition

3 Compare the DIC across model fits and make inference using the best scoring model according to DIC; include a violin plot for the regression coefficients.

As shown in Table 1, the DIC values for c, d, and e are very close, suggesting that surviving fish L_1 in group or days d of study alone are not enough to predict y_i better



(a) β_0: black, β_1: red, β_2: green, β_3: blue,

(b) β_0: black, β_1: red, β_2: green

(c) β_0: black, β_1: red

(d) β_0: black, β_2: red

(e)

Figure 1: Trace plot

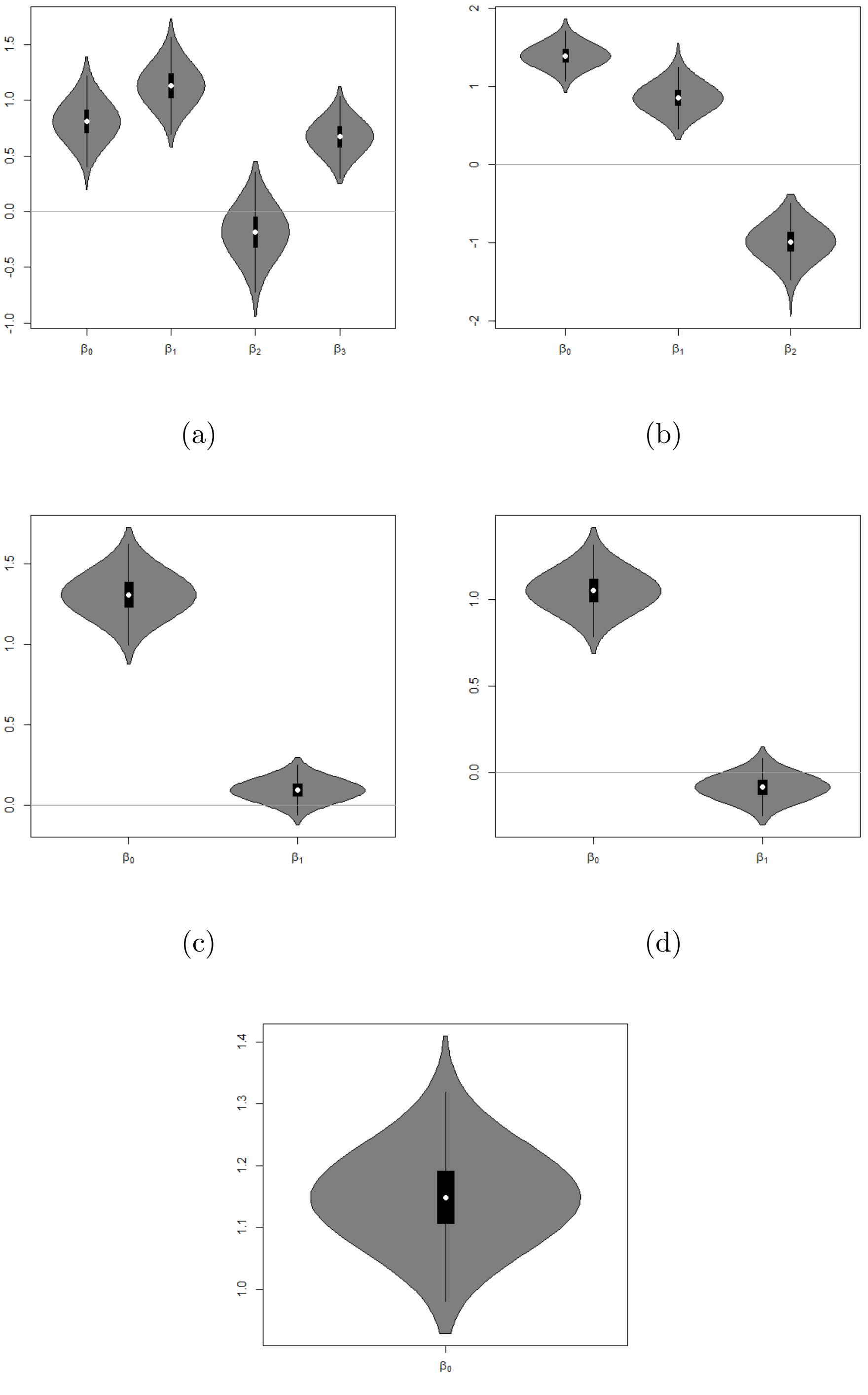
than a constant value (β_0) alone. This observation is further supported by the violin plots for c and d (Figure 2), where the slope values β_1 cross 0 and cannot be said to be statistically significant. However, together, surviving fish L_1 and days d together produce a better model, with some valid prediction of y_i. The values of β_1 and β_2 do not cross zero for part b, and the DIC for this implementation is much better. The predictive model further improves with the inclusion of an interaction term $L_1 * d$. The DIC is the lowest for this implementation, and the β_1 (L_1) and β_3 parameters ($L_1 * d$) show statistical significance to the predictive ability of the model.

References

Soubeyand, S., R. Beaudouin, N. Desassis, and G. Monod. (2007). Model-based estimation of the link between the daily survival probability and a time-varying covariate, application to mosquitofish survival data. Mathematical Biosciences, 210: 508-522.

Author Contributions

- AU and MC individually wrote a model implementation in R and analyzed the data set. Both authors compared answers and contributed to the homework solution document.



(a)

(b)

(c)

(d)

(e)

Figure 2: Violin plots for MCMC values for β parameters

APPENDIX A: AU's R Script

```
###   Homework 4

###   Libraries
library(rjags)
library(vioplot)
###
###   Read the Data
###



setwd("~/UT Austin/Spring 2023/Bayesian Stats/hw4/hw4")
mosquito=read.table("mosquitofish.txt",header=TRUE) # read in mosquitofish data
head(mosquito)


n  = 32 # those groups of fish that were only counted twice
y  = mosquito$N1[1:n]
N  = mosquito$N0[1:n]
l1 = mosquito$L1[1:n]
d  = mosquito$d[1:n]

l1_stand = scale(l1)[,1] #standardized L1 - mean length of surviving fish
d_stand = scale(d)[,1] #standardized Days of study

l1_d_product = l1_stand * d_stand


## Jags Regression
# (a)

# Covariates - Intercept, stand l1, stand d, product of l1 and d
p = 4
X = matrix(1,n,p)
X[,2] = l1_stand
X[,3] = d_stand
X[,4] = l1_d_product

mosquito_jags <-"
  model{
    for(i in 1:n){
      y[i] ~ dbin(theta[i],N[i])
      probit(theta[i]) = b0 + b1*X[i,2] + b2*X[i,3] + b3*(X[i,2]*X[i,3])
    }
    b0 ~ dnorm(mu,tau)
    b1 ~ dnorm(mu,tau)
```

```r
    b2 ~ dnorm(mu,tau)
    b3 ~ dnorm(mu,tau)
}
"
mod <- textConnection(mosquito_jags)
#
# mu.beta=rep(0,p)
# Sig.beta=100*diag(p)
# Tau.beta=solve(Sig.beta)

mu=0
tau=1/2.25 #Precision. 1 / sigma^2
m.out<-jags.model(mod,data=list('y'=y,'n'=n, 'N'=N, 'X'=X,'mu'=mu,'tau'=tau),n.chains=1,n.adapt=0) # build model and algorithm

n.mcmc=10000
n.burn=round(.2*n.mcmc)
update(m.out,n.burn) # perform burn-in
m.samples=jags.samples(m.out,c('b0','b1','b2','b3'),n.mcmc) # fit model post-burn-in

#Trace Plot for (a)
beta.post.mat=cbind(m.samples$b0[1,,1],m.samples$b1[1,,1],m.samples$b2[1,,1],m.samples$b3[1,,1])
matplot(beta.post.mat,type="l",lty=1,ylab=bquote(beta))

#Violin plot for (a)
vioplot(data.frame(beta.post.mat),names=expression(beta[0],beta[1],beta[2],beta[3]))
abline(h=0,col=8)

apply(beta.post.mat,2,mean) # marginal posterior means for beta
apply(beta.post.mat,2,sd) # marginal posterior sd for beta
apply(beta.post.mat,2,quantile,c(0.025,.975)) # marginal posterior 95% CI for beta


# Covariates - Intercept, stand l1, stand d
# (b)

mosquito_jags_b <-"
  model{
    for(i in 1:n){
      y[i] ~ dbin(theta[i],N[i])
      probit(theta[i]) = b0 + b1*X[i,2] + b2*X[i,3]
    }
    b0 ~ dnorm(mu,tau)
    b1 ~ dnorm(mu,tau)
    b2 ~ dnorm(mu,tau)
}
"
```

```
mod_b <- textConnection(mosquito_jags_b)
#
# mu.beta=rep(0,p)
# Sig.beta=100*diag(p)
# Tau.beta=solve(Sig.beta)

mu=0
tau=1/2.25 #Precision. 1 / sigma^2
m.out_b<-jags.model(mod_b,data=list(’y’=y,’n’=n, ’N’=N, ’X’=X,’mu’=mu,’tau’=tau),n.chains=1,n.adapt=0) # build model and algorithm

n.mcmc=10000
n.burn=round(.2*n.mcmc)
update(m.out_b,n.burn) # perform burn-in
m.samples_b=jags.samples(m.out_b,c(’b0’,’b1’,’b2’),n.mcmc) # fit model post-burn-in

#Trace Plot for (b)
beta.post.mat_b=cbind(m.samples_b$b0[1,,1],m.samples_b$b1[1,,1],m.samples_b$b2[1,,1])
matplot(beta.post.mat_b,type="l",lty=1,ylab=bquote(beta))

#Violin plot for (b)
vioplot(data.frame(beta.post.mat_b),names=expression(beta[0],beta[1],beta[2]))
abline(h=0,col=8)

apply(beta.post.mat_b,2,mean) # marginal posterior means for beta
apply(beta.post.mat_b,2,sd) # marginal posterior sd for beta
apply(beta.post.mat_b,2,quantile,c(0.025,.975)) # marginal posterior 95% CI for beta

# Covariates - Intercept, stand l1
# (c)

mosquito_jags_c <-"
  model{
    for(i in 1:n){
      y[i] ~ dbin(theta[i],N[i])
      probit(theta[i]) = b0 + b1*X[i,2]
    }
    b0 ~ dnorm(mu,tau)
    b1 ~ dnorm(mu,tau)
}
"
mod_c <- textConnection(mosquito_jags_c)
#
# mu.beta=rep(0,p)
# Sig.beta=100*diag(p)
# Tau.beta=solve(Sig.beta)

```

```r
mu=0
tau=1/2.25 #Precision. 1 / sigma^2
m.out_c<-jags.model(mod_c,data=list('y'=y,'n'=n, 'N'=N, 'X'=X,'mu'=mu,'tau'=tau),n.chains=1,n.adapt=0) # build model and algorithm

n.mcmc=10000
n.burn=round(.2*n.mcmc)
update(m.out_c,n.burn) # perform burn-in
m.samples_c=jags.samples(m.out_c,c('b0','b1'),n.mcmc) # fit model post-burn-in

#Trace Plot for (c)
beta.post.mat_c=cbind(m.samples_c$b0[1,,1],m.samples_c$b1[1,,1],m.samples_c$b2[1,,1])
matplot(beta.post.mat_c,type="l",lty=1,ylab=bquote(beta))

#Violin plot for (c)
vioplot(data.frame(beta.post.mat_c),names=expression(beta[0],beta[1]))
abline(h=0,col=8)

apply(beta.post.mat_c,2,mean) # marginal posterior means for beta
apply(beta.post.mat_c,2,sd) # marginal posterior sd for beta
apply(beta.post.mat_c,2,quantile,c(0.025,.975)) # marginal posterior 95% CI for beta


# Covariates - Intercept, stand d
# (d)

mosquito_jags_d <-"
  model{
    for(i in 1:n){
      y[i] ~ dbin(theta[i],N[i])
      probit(theta[i]) = b0 + b1*X[i,3]
    }
    b0 ~ dnorm(mu,tau)
    b1 ~ dnorm(mu,tau)
}
"
mod_d <- textConnection(mosquito_jags_d)
#
# mu.beta=rep(0,p)
# Sig.beta=100*diag(p)
# Tau.beta=solve(Sig.beta)

mu=0
tau=1/2.25 #Precision. 1 / sigma^2
m.out_d<-jags.model(mod_d,data=list('y'=y,'n'=n, 'N'=N, 'X'=X,'mu'=mu,'tau'=tau),n.chains=1,n.adapt=0) # build model and algorithm

```

```
n.mcmc=10000
n.burn=round(.2*n.mcmc)
update(m.out_d,n.burn) # perform burn-in
m.samples_d=jags.samples(m.out_d,c('b0','b1'),n.mcmc) # fit model post-burn-in

#Trace Plot for (d)
beta.post.mat_d=cbind(m.samples_d$b0[1,,1],m.samples_d$b1[1,,1])
matplot(beta.post.mat_d,type="l",lty=1,ylab=bquote(beta))

#Violin plot for (d)
vioplot(data.frame(beta.post.mat_d),names=expression(beta[0],beta[1]))
abline(h=0,col=8)

apply(beta.post.mat_d,2,mean) # marginal posterior means for beta
apply(beta.post.mat_d,2,sd) # marginal posterior sd for beta
apply(beta.post.mat_d,2,quantile,c(0.025,.975)) # marginal posterior 95% CI for beta

# Covariates - Intercept ONLY
# (e)

mosquito_jags_e <-"
  model{
    for(i in 1:n){
      y[i] ~ dbin(theta[i],N[i])
      probit(theta[i]) = b0
    }
    b0 ~ dnorm(mu,tau)
}
"
mod_e <- textConnection(mosquito_jags_e)
#
# mu.beta=rep(0,p)
# Sig.beta=100*diag(p)
# Tau.beta=solve(Sig.beta)

mu=0
tau=1/2.25 #Precision. 1 / sigma^2
m.out_e<-jags.model(mod_e,data=list('y'=y,'n'=n, 'N'=N, 'X'=X,'mu'=mu,'tau'=tau),n.chains=1,n.adapt=0) # build model and algorithm

n.mcmc=10000
n.burn=round(.2*n.mcmc)
update(m.out_e,n.burn) # perform burn-in
m.samples_e=jags.samples(m.out_e,c('b0'),n.mcmc) # fit model post-burn-in

#Trace Plot for (e)
beta.post.mat_e=cbind(m.samples_e$b0[1,,1])
```

```
matplot(beta.post.mat_e,type="l",lty=1,ylab=bquote(beta))

#Violin plot for (e)
vioplot(data.frame(beta.post.mat_e),names=expression(beta[0]))
abline(h=0,col=8)

apply(beta.post.mat_e,2,mean) # marginal posterior means for beta
apply(beta.post.mat_e,2,sd) # marginal posterior sd for beta
apply(beta.post.mat_e,2,quantile,c(0.025,.975)) # marginal posterior 95% CI for beta

# --------------------------------------------------------------------------------
### PLOTS and Quantiles

## All Trace Plots
matplot(beta.post.mat,type="l",lty=1,ylab=bquote(beta))   #(a)
matplot(beta.post.mat_b,type="l",lty=1,ylab=bquote(beta)) #(b)
matplot(beta.post.mat_c,type="l",lty=1,ylab=bquote(beta)) #(c)
matplot(beta.post.mat_d,type="l",lty=1,ylab=bquote(beta)) #(d)
matplot(beta.post.mat_e,type="l",lty=1,ylab=bquote(beta)) #(e)

## All Violin Plots

vioplot(data.frame(beta.post.mat),names=expression(beta[0],beta[1],beta[2],beta[3]))  #(a)
abline(h=0,col=8)

vioplot(data.frame(beta.post.mat_b),names=expression(beta[0],beta[1],beta[2]))         #(b)
abline(h=0,col=8)

vioplot(data.frame(beta.post.mat_c),names=expression(beta[0],beta[1]))                 #(c)
abline(h=0,col=8)

vioplot(data.frame(beta.post.mat_d),names=expression(beta[0],beta[1]))                 #(d)
abline(h=0,col=8)

vioplot(data.frame(beta.post.mat_e),names=expression(beta[0]))                         #(e)
abline(h=0,col=8)

## All Quantiles
apply(beta.post.mat,2,quantile,c(0.025,.975))   #(a)
apply(beta.post.mat_b,2,quantile,c(0.025,.975)) #(b)
apply(beta.post.mat_c,2,quantile,c(0.025,.975)) #(c)
apply(beta.post.mat_d,2,quantile,c(0.025,.975)) #(d)
apply(beta.post.mat_e,2,quantile,c(0.025,.975)) #(e)
```

```


## DIC Calculations

# (a) DIC = 74.6349860620726
# (b) DIC = 97.1322909927127
# (c) DIC = 136.613174557372
# (d) DIC = 137.52021356181
# (e) DIC = 139.469354229737

#(a)

postbeta = apply(beta.post.mat, 2, mean)
posttheta = pnorm(X%*%postbeta)
Dhat = mean(-2*sum(log(dbinom(y,N,posttheta))))
pD = 2 * sum(log(mean(exp(beta.post.mat), 2)))
DIC=Dhat+2*pD
print(paste("DIC =",DIC))
# "DIC = 74.6349860620726"

#(b)
postbeta = apply(beta.post.mat_b, 2, mean)
posttheta = pnorm(X[,-4]%*%postbeta)
Dhat = mean(-2*sum(log(dbinom(y,N,posttheta))))
pD = 2 * sum(log(mean(exp(beta.post.mat), 2)))
DIC=Dhat+2*pD
print(paste("DIC =",DIC))
# "DIC = 97.1322909927127"

#(c)
postbeta = apply(beta.post.mat_c, 2, mean)
posttheta = pnorm(X[,1:2]%*%postbeta)
Dhat = mean(-2*sum(log(dbinom(y,N,posttheta))))
pD = 2 * sum(log(mean(exp(beta.post.mat), 2)))
DIC=Dhat+2*pD
print(paste("DIC =",DIC))
# "DIC = 136.613174557372"

#(d)
postbeta = apply(beta.post.mat_d, 2, mean)
posttheta = pnorm(X[,c(1,3)]%*%postbeta)
Dhat = mean(-2*sum(log(dbinom(y,N,posttheta))))
pD = 2 * sum(log(mean(exp(beta.post.mat), 2)))
DIC=Dhat+2*pD
print(paste("DIC =",DIC))
# "DIC = 137.52021356181"

#(e)
postbeta = apply(beta.post.mat_e, 2, mean)
posttheta = pnorm(X[,1]*postbeta)
Dhat = mean(-2*sum(log(dbinom(y,N,posttheta))))
```

```
pD = 2 * sum(log(mean(exp(beta.post.mat), 2)))
DIC=Dhat+2*pD
print(paste("DIC =",DIC))
# "DIC = 139.469354229737"
```

APPENDIX B: MC’s R Script

```r
library(rjags)
mosquitofish = read.table(file = 'RStudio/bayesian/mosquitofish.txt', header=TRUE)
set.seed(1234)

### a
X = data.matrix(mosquitofish[c('L1', 'd')])
X = cbind(X, X[,1]*X[,2])
xavg = c(0, apply(X, 2, mean))
xsd = c(1, apply(X, 2, sd))
X = scale(X)
X = cbind(1, X)

### b
X = data.matrix(mosquitofish[c('L1', 'd')])
X = scale(X)
X = cbind(1, X)

### c
X = data.matrix(mosquitofish[c('L1')])
X = scale(X)
X = cbind(1, X)

### d
X = data.matrix(mosquitofish[c('d')])
X = scale(X)
X = cbind(1, X)

### e
X = matrix(1, n)

m.jags <-"
  model{
    for(i in 1:n)
    {
      y[i,] ~ dbin(theta[i], N[i,])
      probit(theta[i]) = inprod(X[i,], b)
    }
    b ~ dmnorm(mu_b[],tau_b[,])
}
"

mod<-textConnection(m.jags)

y = data.matrix(mosquitofish['N1'])
N = data.matrix(mosquitofish['N0'])
n = dim(y)[1]

nx = dim(X)[2]
```

```
mu_b = apply(X, 2, mean)
sig_b = diag(nx)*((apply(X, 2, sd)^2)+1)
tau_b = solve(sig_b)
m.out<-jags.model(mod,data=list('y'=y,'X'=X,'n'=n,'N'=N,'mu_b'=mu_b,
    'tau_b'=tau_b),n.chains=1,n.adapt=0)
n.mcmc=100000
n.burn=round(.1*n.mcmc)
update(m.out,n.burn) # perform burn-in
m.samples=jags.samples(m.out,c('theta','b'),n.mcmc)

par(mfrow=c(nx,1))
for(i in 1:nx)
{
  plot(m.samples$b[i,,],type="l", ylab=paste("b_",i,sep=""))
}
#plot(theta,type="l", ylab="tau")

postbeta = apply(m.samples$b, 1, mean)
posttheta = pnorm(X%*%postbeta)
Dhat = mean(-2*sum(log(dbinom(y,N,posttheta))))
pD = 2 * sum(log(mean(exp(m.samples$b), 2)))
DIC=Dhat+2*pD
print(paste("DIC =",DIC))


#post_beta_unscale = postbeta*xsd

library(vioplot)
vioplot(t(m.samples$b[,,]), names = list("b_0", "b_1", "b_2", "b_3")
    )
abline(h=0,col=8)
```